Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *CASE-технологія для системи проектування технологічних процесів виготовлення деталей машин*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент 4 курсу, група КН-19-1* _____ *В.Ю. Медведчук*
Курс, група виконавця — Підпис — Ініціали, прізвище

Керівник: *к.т.н., доцент кафедри КН* _____ *Р.О. Багрій*
Науковий ступінь, посада — Підпис — Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КН* _____ *Р.О. Багрій*
Науковий ступінь, посада — Підпис — Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КН, д.т.н., професор _____ *О.В. Бармак*
Підпис — Ініціали, прізвище

«01» 06 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*

Кафедра *комп'ютерних наук*

Освітній ступінь *бакалавр*

Галузь знань *12 – Інформаційні технології*

Спеціальність *122 – Комп'ютерні науки*

Освітня програма *освітньо-професійна програма підготовки бакалавра*

ЗАТВЕРДЖУЮ

Завідувач кафедри *комп'ютерних наук*

(підпис)

д.т.н., професор О.В. Бармак

«*06*» *03* *2023* року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«CASE-технологія для системи проектування технологічних процесів виготовлення деталей машин»*

2. Завдання видано студенту *Медведчуку Віталію Юрійовичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *доцент кафедри КН Багрій Руслан Олександрович*
(посада, прізвище, ім'я, по батькові)

4. Затверджено наказом університету від «*01*» *03* *2023* р. № *5*

5. Дата видачі студенту: «*03*» *03* *2023* р.

6. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Провести аналіз предметної області, огляд методів та засобів CASE-технології, існуючих програмних рішень та сформувати постановку задачі. Спроектувати функціональну структуру та структуру інформаційної системи проектування технологічних процесів з використанням CASE-технології. Створити програмну реалізацію інформаційної системи проектування технологічних процесів виготовлення деталей машин з використанням сучасних засобів створення програмного забезпечення. Вихідними даними є довідкова інформація про технологічні операції виготовлення деталі, пристрої та інструменти та режими їх роботи.

7. Календарний план виконання кваліфікаційної роботи бакалавра:

№	Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Примітка
1	*Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником*	грудень 2022	*виконано*
2	*Ознайомлення з предметною областю, формулювання мети та задач дослідження, визначення об'єкта та предмета дослідження*	січень 2023	*виконано*
3	*Робота над розділом 1 – Характеристика предметної області та постановка задачі*	січень 2023	*виконано*
4	*Робота над розділом 2 – Інформаційна система проектування технологічних процесів*	березень 2023	*виконано*
5	*Робота над розділом 3 – Програмна реалізація інформаційної системи проектування технологічних процесів*	квітень 2023	*виконано*
6	*Оформлення пояснювальної записки згідно вимог*	травень 2023	*виконано*
7	*Підготовка статті до журналу, попередній захист кваліфікаційної роботи бакалавра*	травень 2023	*виконано*
8	*Захист кваліфікаційної роботи бакалавра на засіданні Екзаменаційної комісії*	червень 2023	*виконано*

Виконавець: *студент 4 курсу, група КН-19-1* (Курс, група виконавця) _____ (Підпис) *В.Ю. Медведчук* (Ініціали, прізвище)

Керівник: *к.т.н., доцент кафедри КН* (Науковий ступінь, посада) _____ (Підпис) *Р.О. Багрій* (Ініціали, прізвище)

Анотація

Тема кваліфікаційної роботи бакалавра: *CASE-технологія для системи проектування технологічних процесів виготовлення деталей машин*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-19-1 Медведчук Віталій Юрійович*

Керівник кваліфікаційної роботи бакалавра: *к.т.н., доцент кафедри КН Багрій Руслан Олександрович*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
60	38	10	25	10

Метою кваліфікаційної роботи бакалавра є розробка системи проектування технологічних процесів виготовлення деталей машин з використанням CASE-технології.

Результатом виконання кваліфікаційної роботи бакалавра є створення інформаційної системи проектування технологічних процесів виготовлення деталей машин з використанням CASE-технології.

Ключові слова: CASE-технології, проектування технологічних процесів.

Виконавець: *студент 4 курсу, група КН-19-1* (Курс, група виконавця) _____ (Підпис) *В.Ю. Медведчук* (Ініціали, прізвище)

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
ПЗ	Програмне забезпечення
ПП	Програмний продукт
КРБ	Кваліфікаційна робота бакалавра
КН	Комп’ютерні науки
ІС	Інформаційна система
ЖЦ	Життєвий цикл
БД	База даних
СУБД	Система управління базами даних
КД	Конструкторська документація
САПР ТП	Система автоматизованого проєктування технологічних процесів

Вступ

Кваліфікаційна робота бакалавра присвячена розробці системи проектування технологічних процесів виготовлення деталей машин з використанням CASE-технології.

Актуальність теми.

Система автоматизованого проектування технологічних процесів (САПР ТП) вирішує більшість завдань у рамках технологічної підготовки виробництва та дозволяє спростити формування та супровід техпроцесів. Завдання САПР ТП полягає у складанні плану виробництва виробу (маршрут виготовлення), в який входить послідовність технологічних операцій виготовлення деталі, пристрої та інструменти та режими їх роботи. Створення програмних продуктів для САПР ТП є не простою задачею через використання значного об'єму довідникових даних та функціональних вимог.

CASE-технології (англ. computer-aided software engineering) допомагають забезпечити високу якість програм, відсутність помилок та простоту в обслуговуванні складних систем програмного забезпечення. Це забезпечується сукупністю методів і засобів проектування інформаційних систем з інтегрованими автоматизованими інструментами.

Мета кваліфікаційної роботи бакалавра полягає у розробці системи проектування технологічних процесів виготовлення деталей машин з використанням CASE-технології.

Об’єкт дослідження – процес проектування технологічних процесів з використанням CASE-технології.

Предмет дослідження – методи збору та аналізу інформації, технології та методи проектування інформаційних систем проектування технологічних процесів виготовлення деталей машин.

Завдання кваліфікаційної роботи бакалавра – проаналізувати задачу проектування технологічних процесів виготовлення деталей машин; розробити функціональну структуру інформаційної системи проектування технологічних

процесів з використанням CASE-технології; розробити структуру інформаційної системи проектування технологічних процесів з використанням CASE-технології; розробити програмну реалізацію інформаційної системи проектування технологічних процесів виготовлення деталей машин.

Розділ 1 Характеристика предметної області та постановка задачі

1.1 Аналіз предметної області

Для будь-якого виду виробництва характерною є зміна характеристик ресурсів та праці з метою отримання готової продукції. Цей процес відомий як виробничий процес і може включати різноманітні основні, допоміжні та обслуговуючі процеси[1]. Основні процеси - це технологічні процеси, під час яких відбуваються зміни у формі та розмірах продукції.

Технологічний процес є частиною виробничого процесу і складається з послідовно виконуваних технологічних операцій над однорідними або подібними об'єктами праці[2]. Ці операції змінюють агрегатний стан, розташування або властивості об'єкта праці, який має певну виробничу призначеність. Для виконання технологічного процесу розробляється схема або карта, в якій описуються всі технологічні операції, які включають робочі дії однакового технологічного змісту. Технологічна операція включає в себе послідовні робочі дії, які виконуються з використанням однакових інструментів і пристосувань[3].

Технологічні процеси можна класифікувати за декількома ознаками:

1) зміна фізичних, механічних та хімічних властивостей сировини під час її переробки;
2) організаційний спосіб процесу;
3) напрямок руху теплових та сировинних потоків;
4) агрегатний стан складових сировини;
5) тепловий ефект процесу;
6) основні чинники, що спричиняють та прискорюють процеси.

Залежно від умов виробництва і призначення, технологічні процеси поділяються на одиничні та уніфіковані. Також вони можуть бути класифіковані за рівнем уніфікації, рівнем науково-технічних досягнень, змістом операцій переміщення та рівнем деталізації опису[4].

З впровадженням автоматизації люди почали зменшувати кількість непотрібних дій у своїй праці, особливо в сфері виробництва товарів. Автоматизація технологічних процесів полягає у використанні енергії неживої природи для безпосереднього виконання або керування цими процесами без участі людини. Це дозволяє зменшити трудові витрати, поліпшити умови виробництва, збільшити обсяги та якість продукції[5].

Для реалізації технологічного процесу необхідні засоби технологічного оснащення та робоче місце, а також предмет праці. Технологічна операція складається з технологічних та допоміжних переходів, установок, позицій та робочих ходів. Допоміжний перехід включає в себе дії людини та/або устаткування, які не змінюють властивостей об'єктів праці, але необхідні для виконання технологічного переходу[6].

Для розробки програмної реалізації інформаційної системи використовуються методи та інструменти комп'ютерних наук та інформаційних технологій. Це дозволяє використовувати готові алгоритми технологічного процесу або створювати нові. Автоматизація роботи спеціалістів та виробництва товарів може зменшити кількість працівників, трудові витрати та помилки, а також покращити якість продукції.

Загалом, виробничий процес і технологічні процеси є важливими компонентами у будь-якому виробництві, і їх автоматизація може принести багато переваг у вигляді ефективності, якості та економії ресурсів.

1.2 Огляд методів та засобів CASE-технології

CASE (Computer-Aided Software Engineering) - це набір інструментів та методів інформаційних технологій, які використовуються для проектування програмного забезпечення. Вони сприяють автоматизації процесу проектування технологічних процесів, допомагають зменшити кількість помилок, спрощують обслуговування та забезпечують високу якість програм[7].

CASE-технологія може бути розглянута як методологія проектування інформаційних систем або як набір інструментів для моделювання предметної області та її аналізу. Вона ґрунтується на методологіях структурного або об'єктно-орієнтованого аналізу та проектування[8]. Основна мета CASE-технології полягає в тому, щоб відокремити процес проектування автоматизованих інформаційних систем від процесу їх кодування та наступних етапів розробки, а також максимально автоматизувати процеси розробки та функціонування систем[9].

На сьогоднішній день CASE-технології займають важливе місце серед технологій проектування різних типів та складності інформаційних систем. Вони включають засоби, що підтримують повний життєвий цикл програмного продукту. CASE-засоби можна класифікувати за типами та категоріями. Класифікація за типами відображає функціональну орієнтацію на різні процеси життєвого циклу програмного забезпечення. Класифікація за категоріями визначає ступінь інтегрованості за функціональними можливостями, включаючи окремі засоби для вирішення конкретних завдань, частково інтегровані засоби, що охоплюють багато етапів життєвого циклу інформаційних систем, та повністю інтегровані засоби, що підтримують весь життєвий цикл і пов'язані спільним репозиторієм.

CASE-засоби також можна класифікувати за іншими ознаками, такими як:

1) за застосовуваними методологіями, моделями систем і БД;
2) за ступенем інтегрованості з СУБД;
3) за доступними платформами.

CASE-технології, що базуються на методології об'єктно-орієнтованого аналізу та проектування, мають чотири основні блоки:

1) аналіз;
2) проектування;
3) розробка;
4) інфраструктура[10].

CASE-технології знайшли широке застосування у галузі побудови різних типів автоматизованих інформаційних систем. Вони особливо популярні у галузі

розробки ділових та автоматизованих інформаційних систем. CASE-технології допомагають покращити якість створюваних систем, дозволяють швидко створювати прототипи, прискорюють процес проектування та розробки, сприяють творчому підходу розробників, підтримують повторне використання компонентів розробки, а також сприяють розвитку та супроводженню автоматизованих інформаційних систем.

З цього можна зробити висновок, що CASE-технології не можна вважати самостійними методологіями, вони є лише споміжними для структурних методологій, роблячи їх ефективнішими за рахунок автоматизації. Окрім даної переваги, CASE-технології також вирізняються тим, що:

1) покращують якість створюваних автоматизованих ІС;

2) дозволяють за короткий час створювати прототип майбутньої автоматизованої інформаційної системи;

3) прискорюють процес проектування і розробки системи;

4) дають можливість робітникам більше зосередитися на творчій частині розробки;

5) підтримують технології повторного використання компонентів розробки;

6) підтримують розвиток і супроводження розробки автоматизованої інформаційної системи.

1.3 Аналіз існуючих рішень для подібних систем

"ТехноПро" є системою, яка призначена для автоматизації процесу технологічної підготовки виробництва. Вона охоплює такі етапи, як проектування технологій, технологічні та економічні розрахунки, а також отримання документації та звітів. Однією з можливостей програми є використання бази конструкторської документації для створення бази даних зі складу виробів та специфікацій підприємства, а також внесення змін до цієї документації. "ТехноПро" автоматизує процес проектування маршрутів та операційної

технології, включаючи одиничні, типові та групові технологічні процеси. Вона також розраховує технологічні та економічні параметри та генерує відповідну технологічну документацію[11].

Програма "ТехноПро" надає комплексний набір інструментів та можливостей для створення та керування технологічними процесами. Це дозволяє користувачам відстежувати зміни в технічних процесах, захищати доступ до даних і технологій, формувати групи користувачів та керувати ними, а також визначати статус процесів. Крім того, користувачі можуть проектувати та створювати різноманітні технологічні процеси, налаштовувати доступ до даних і технологій, а також створювати наскрізні технології та/або уздовж технологічних маршрутів. Програма також забезпечує можливість одночасної роботи користувача в технологічному процесі, розрахунок технологічних та економічних параметрів і показників, а також формування необхідної технологічної документації у відповідних форматах. Завдяки програмі користувачі можуть також створювати технології на основі параметрів для максимального спрощення роботи.

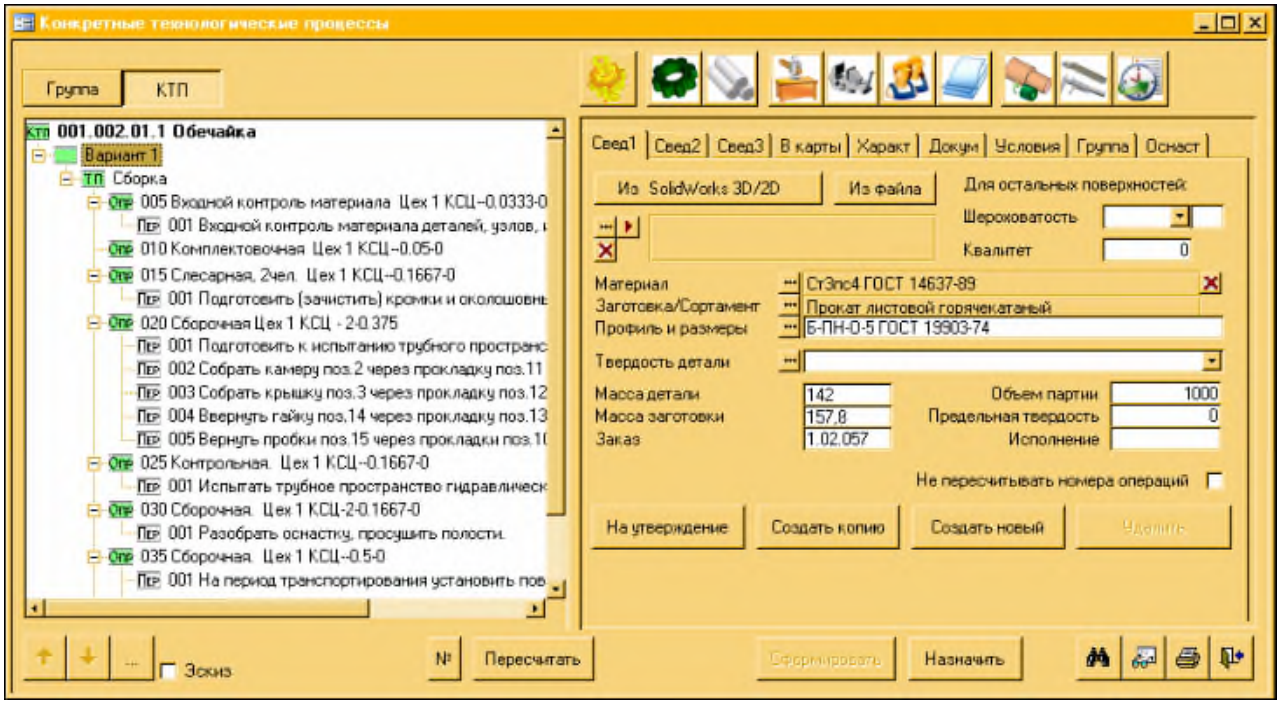


Рисунок 1.1 – Інтерфейс програми «ТехноПро»[11]

"ТехноПро" допомагає у забезпеченні технологічного процесу створення різних типів документів за допомогою Microsoft Word. У базі даних системи зібрані попередньо розроблені та типові технологічні процеси, які можуть бути змінені технологом шляхом додавання або видалення операцій враховуючи певні умови, наприклад, розмір заготівлі. Крім того, можна вибирати інструменти, орієнтуючись на результати певних розрахунків. Технологічні процеси можуть бути переглянуті та, у разі потреби, відредаговані в режимі діалогу. Після натискання кнопки друку користувач може отримати технологічні процеси у формі технологічних карток у вигляді файлів Microsoft Word, як показано на наведеному нижче зображенні.

ГОСТ 3.1118-82 форма 1 САПР

Дубл.			
Взам.			
Подл.			
		3	1

Разраб.	Лихачев А.А.	Вектор	2345-4789
Проверил	Иванов И.И.		
Нормиров.	Рабинович Р.Р.		
Метролог	Сидоров С.С.	0сь	
Н.контр.	Никонов Н.Н.		

М 01	Ст4кп ГОСТ 4543-71									
	Код	ЕВ	МД	ЕН	Н.расх.	КИМ	Код загот.	Профиль и размеры	КД	МЗ
М 02		кг	0.02	1	0.028	0.71	Круг	10 ГОСТ 2590-88	5	0.138

А	Цех	Уч.	РМ	Опер.	Код, наименование операции	Обозначение документа												
Б	Код, наименование оборудования					СМ	Проф.	Р	УТ	КР	КОИД	ЕН	ОП	Кшт.	Тпз	Тшт.		
А03	06	05	13	005	Заготовительная	ИОТ 59-85												
Б04	Ленточно-пильный														4,6	3,5		
Т05	Тиски 7200-0008 160 ГОСТ 14904-80																	
О06	1 Отрезать заготовку ⌀10 мм L=39 js14 (±0,31) мм (1 заготовка на 1 деталь).																	
Т07	Пила В230-s1,6-t50 3405-0002 ГОСТ 10670-77																	
Т08	Штангенциркуль ШЦ-I-125-0,1-1 ГОСТ 166-89																	
09																		
А10	3	2	12	010	4110 Токарная													
Б11	Токарно-центровой 16К20														8	12,8		
Т12	Патрон D160 7108-0005 Н ГОСТ 24568-81																	
О13	А. Установить деталь на станке, закрепить и снять после обработки																	
О14	1 Торцевать диаметр с 10 мм, как чисто																	
Т15	Резец 16х10 2112-0031 ГОСТ 18871-73																	
О16	2 Точить диаметр до Ф8 h12 (-0,15) мм на проход																	
МК/КТП																		

Рисунок 1.2 – Приклад технологічної карти[11]

Програма пропонує численні переваги для виробничого середовища, зокрема значне прискорення процесу підготовки, зниження витрат, поліпшення планування та контролю виробництва, спрощену роботу в технологічних та інших

підрозділах, повну автоматизацію технологічних, технологічних та економічних розрахунків. З іншого боку, програма має деякі недоліки, такі як обмежена можливість ефективної роботи з графічним матеріалом технологічного процесу, застарілий інтерфейс та висока складність роботи з програмним продуктом, що може відлякати користувачів.

"Вертикаль" - це система автоматизованого проектування технологічних процесів, яка допомагає вирішувати більшість завдань у рамках технологічної підготовки виробництва. Вона спрощує формування та супровід технологічних процесів, підвищує якість технологічної документації та досягає оптимальних показників використання ресурсів підприємства. "Вертикаль" може бути використана як самостійний інструмент для технологічної підготовки виробництва або інтегруватися з іншими продуктами.

Програмне забезпечення "Вертикаль" пропонує різноманітні функції, які допомагають у інженерному та виробничому процесах. Це включає проектування, формування команд для програмування обладнання з ЧПУ, технологічні розрахунки (трудовитрати, норми матеріалів, режими різання та зварювання), формування технологічної документації відповідно до стандартів компанії та підтримку єдиного інформаційного простору для управління життєвим циклом продукції.

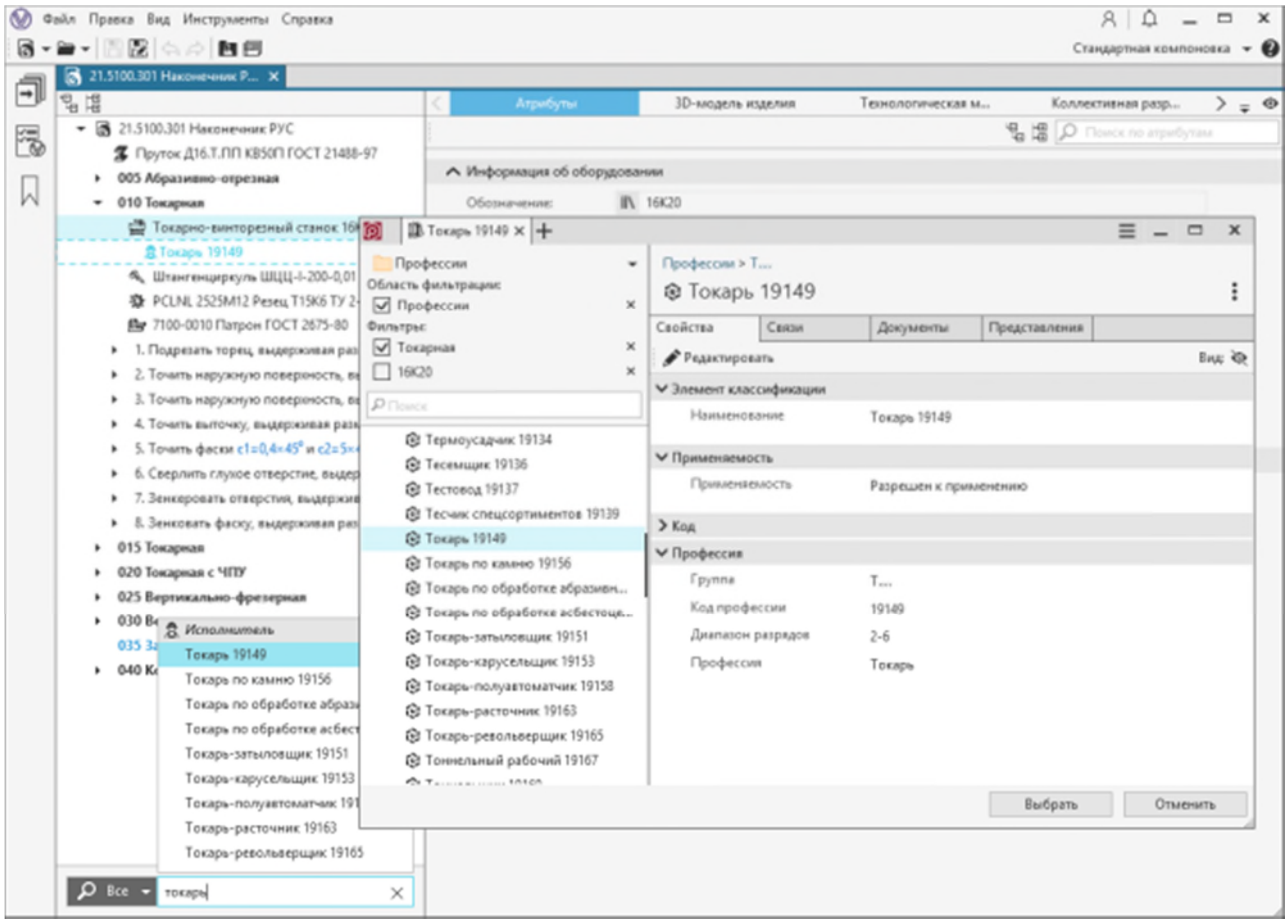


Рисунок 1.3 – Інтерфейс програми «Вертикаль»

Система "Вертикаль" дозволяє проектувати складні технологічні процеси в автоматизованому ієрархічному порядку. Технологи можуть проектувати такі процеси в режимі реального часу паралельно. Крім того, забезпечується зв'язок між параметрами технологічного процесу та графічними документами КОМПАС-3D, такими як креслення, ескізи та 3D-моделі. Зміна параметрів графічних документів автоматично відображається в параметрах технологічного процесу і навпаки. Це дозволяє користувачам оновлювати значення параметрів у графічному документі відповідно до змін у технологічному процесі.

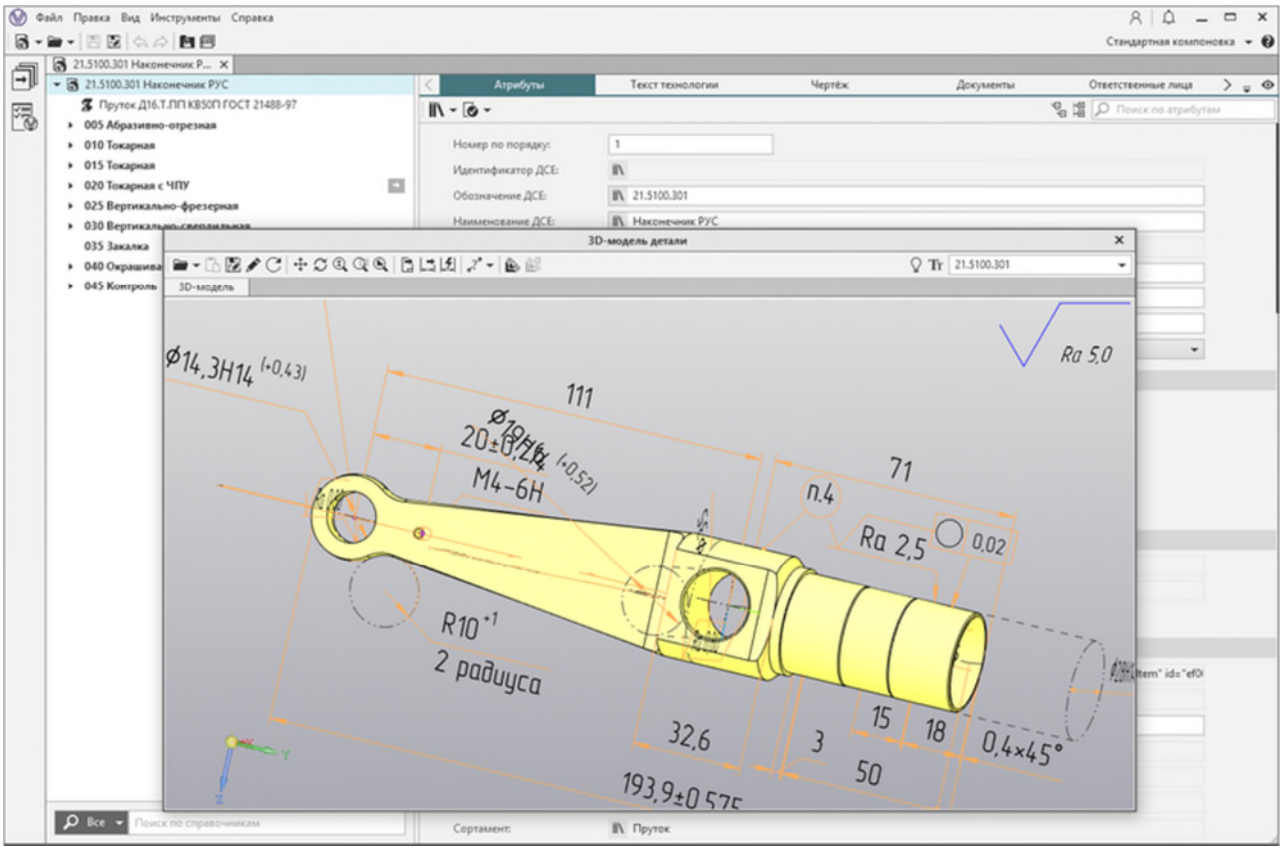


Рисунок 1.4 – 3D модель деталі

Переваги програми "Вертикаль" включають можливість проектування технологічних процесів у різних автоматизованих режимах, підтримку всіх видів електронного інженерного документообігу та інноваційні рішення в галузі семантичної інтеграції даних. Серед недоліків програми можна відзначити відсутність зручного механізму перегляду планів обробки окремих компонентів та необхідність придбання додаткових розрахункових застосунків.

Після аналізу існуючих рішень можна зробити висновки, що програмні реалізації інформаційної системи, які були описані раніше, мають певні недоліки. Одним з основних недоліків є нестабільність, що може призвести до непередбачуваних помилок і незручностей для користувачів. Крім того, високий поріг входження для користування застосунком може відлякати потенційних користувачів і обмежувати його популярність. Деякі з цих продуктів також можуть мати обмежену функціональність, що вимагає додаткових покупок інших продуктів для повного вирішення проблеми.

Також варто враховувати, що на вітчизняному ринку використовуються переважно іноземні програмні продукти. У зв'язку з цим, розробка власного вітчизняного застосунку для вирішення подібних проблем може бути доцільною. Це дозволить забезпечити національну підтримку та адаптацію до потреб вітчизняних користувачів.

Отже, висновком є те, що розробка українського програмного продукту для вирішення цих проблем може бути перспективною, з урахуванням недоліків, виявлених у наявних рішеннях.

1.4 Мета, завдання та вимоги до реалізації інформаційної системи

Метою кваліфікаційної роботи бакалавра є розробка системи проектування технологічних процесів виготовлення деталей машин з використанням CASE-технології.

Для досягнення поставленої мети необхідно реалізувати виконання наступних задач:

– проаналізувати задачу проектування технологічних процесів виготовлення деталей машин;

– розробити функціональну структуру інформаційної системи проектування технологічних процесів з використанням CASE-технології;

– розробити структуру інформаційної системи проектування технологічних процесів з використанням CASE-технології;

– розробити програмну реалізацію інформаційної системи проектування технологічних процесів виготовлення деталей машин.

При впровадженні програмного продукту важливо використовувати методи та інструменти комп'ютерних наук та інформаційних технологій. У цьому контексті, використання CASE-технологій (Computer-Aided Software Engineering) може бути ефективним підходом для створення автоматизованих інформаційних систем. Вони допомагають покращити якість продукту, прискорити процес

розробки та зосередитися на творчій складовій роботи. Такі технології також сприяють розвитку і підтримці розробки.

Проте, при виборі програмного продукту для вирішення проблеми варто ретельно аналізувати всі його недоліки та переконатися, що він відповідає вимогам. Крім того, врахуйте, що на ринку існують програмні продукти з різними локалізаціями, не обов'язково виключно українськими. Розгляд можливості розробки власного вітчизняного продукту може бути доцільним, особливо якщо це забезпечить національну підтримку та адаптацію до потреб користувачів українського ринку.

Розділ 2 Інформаційна система проектування технологічних процесів

2.1 Моделювання інформаційної системи проектування технологічних процесів з використанням CASE-технології

CASE (Computer-Aided Software Engineering) є методологією, яка надає підтримку в процесі розробки програмного забезпечення шляхом використання спеціалізованих комп'ютерних інструментів. За допомогою CASE-технології можна створити модель інформаційної системи для проектування технологічних процесів. Описана технологія дозволяє виконувати наступні кроки в процесі моделювання інформаційної системи для проектування технологічних процесів:

1) визначення вимог до системи: збір вимог від користувачів і визначення функціональної та нефункціональної специфікації системи;

2) розробка моделей: створення моделей, які відображають процеси та компоненти системи. Моделі можуть бути створені за допомогою діаграм, наприклад, діаграм вимог, діаграм взаємодії, діаграм компонентів;

3) розробка коду: написання коду системи на основі моделей, що були розроблені на попередньому етапі;

4) тестування: перевірка правильності роботи системи та виявлення та виправлення помилок;

5) впровадження та супровід: встановлення системи та підтримка її роботи в майбутньому.

Застосування CASE-технології для моделювання інформаційної системи проектування технологічних процесів дає численні переваги. Воно дозволяє зменшити витрати на розробку та підтримку системи, забезпечити високу якість програмного забезпечення та ефективне управління процесом розробки.

CASE-інструменти дозволяють використовувати стандартизовані методики розробки програмного забезпечення, що сприяє єдності підходів та поліпшує співпрацю між учасниками проекту. Завдяки CASE-технології можна створювати різні моделі системи на різних етапах проекту. Наприклад, на початкових етапах можна розробити моделі вимог та взаємодії, а наступні етапи

можуть бути присвячені моделюванню компонентів та внесенню змін до коду системи.

Узагальнюючи, моделювання інформаційної системи проектування технологічних процесів з використанням CASE-технології дозволяє забезпечити високу якість програмного забезпечення, ефективне управління процесом розробки та зменшення витрат на проект.

Крім того, UML (Unified Modeling Language) - це стандартна мова моделювання, яка дозволяє описувати інформаційні системи, їх компоненти, процеси взаємодії та інші аспекти їх функціонування[12]. UML прийнята міжнародним консорціумом з розробки програмного забезпечення (OMG). При моделюванні інформаційної системи проектування технологічних процесів UML може бути використана для створення різних видів моделей, включаючи:

1) модель вимог – описує функціональні та нефункціональні вимоги до системи;

2) модель взаємодії – показує взаємодію між різними компонентами системи та їх поведінку в різних сценаріях взаємодії;

3) модель класів – описує структуру системи та відношення між її компонентами;

4) модель послідовності – описує послідовність виконання різних етапів процесу та взаємодію між компонентами системи на кожному етапі;

5) модель діяльності – описує бізнес-процеси та дії, які повинні виконуватися в рамках цих процесів.

UML-моделювання відіграє важливу роль у візуалізації та оптимізації роботи інформаційної системи проектування технологічних процесів. Воно сприяє забезпеченню якості та ефективності процесів розробки та підтримки системи, а також полегшує спілкування між учасниками проекту.

UML-моделювання допомагає уникнути розбіжностей між вимогами до системи та її фактичним функціонуванням, спрощує процес комунікації між різними учасниками проекту, забезпечує якісне проектування системи та підвищує її стійкість та ефективність. Однією з ключових переваг UML-

моделювання є той факт, що UML є стандартом, яким користуються багато фахівців у галузі розробки програмного забезпечення. Це сприяє стандартизації та встановленню спільної мови спілкування між учасниками проекту, що допомагає уникнути недорозумінь та помилок під час проектування та розробки системи.

Для моделювання інформаційної системи проектування технологічних процесів можна використовувати різні типи UML-діаграм, зокрема:

1) діаграма варіантів використання (Use Case Diagram) – показує взаємодію між акторами та системою, тобто як система повинна взаємодіяти з користувачами та іншими системами;

2) діаграма класів (Class Diagram) – показує структуру системи та відношення між її компонентами;

3) діаграма послідовності (Sequence Diagram) – показує послідовність взаємодії між компонентами системи на різних етапах проектування та реалізації;

4) діаграма діяльності (Activity Diagram) – показує послідовність дій та процесів, які повинні виконуватися в системі;

5) діаграма станів (State Diagram) – показує стани системи та переходи між ними;

6) діаграма компонентів (Component Diagram) – показує компоненти системи та їх залежності[13].

На рисунку 2.1 зображено діаграму взаємодії користувача та інформаційної системи.

Рисунок 2.1 – Діаграма варіантів використання застосунку

Наведена вище діаграма демонструє ситуацію, яка описує інформаційну систему та користувача, який може користуватись функціоналом цієї системи. Він може переглядати технологічні процеси, переглядати інформацію про елементи технологічного процесу, редагувати інформацію про елементи технологічного процесу, видаляти елементи технологічного процесу, формувати звіти, додавати елементи технологічного процесу.

На рисунку 2.2 зображено діаграму послідовності.

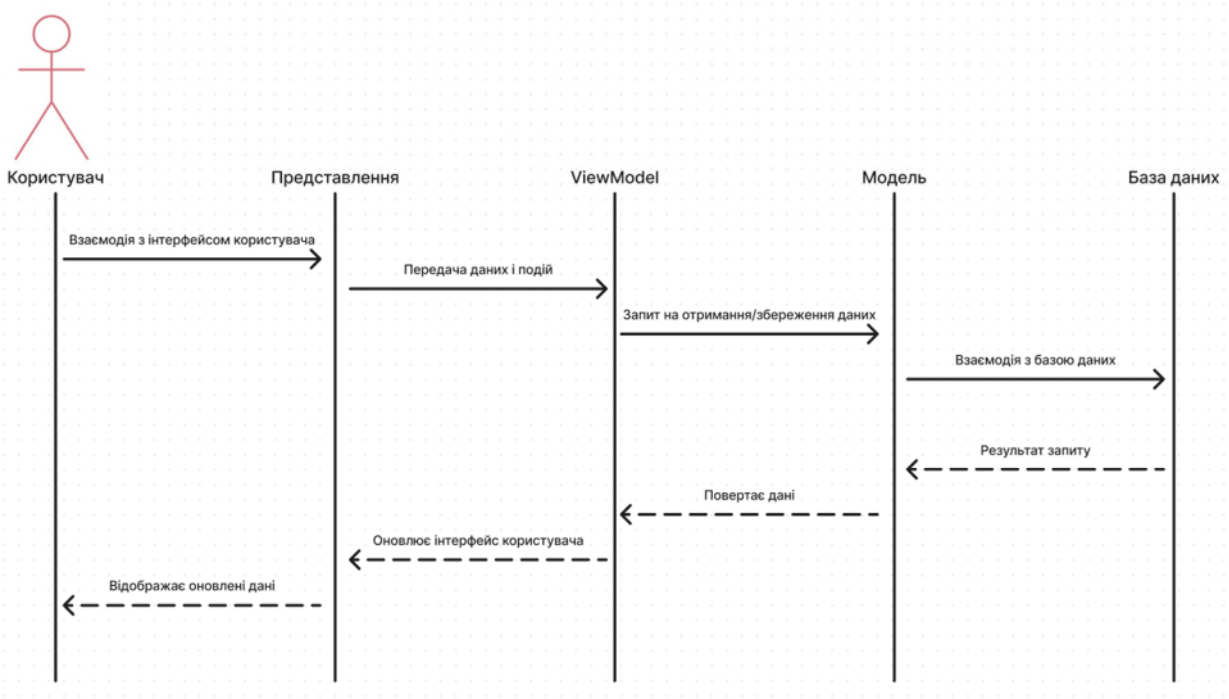


Рисунок 2.2 – Діаграма послідовності

В наведеній вгорі діаграмі послідовності демонструється послідовність дій, що відбуватиметься при ініціації будь-якої дії користувачем інформаційної системи. Якщо розглянути її детальніше, то діаграма показує, коли користувач ініціює взаємодію з інтерфейсом користувача, відклик переходить до представлення, після чого передає дані і події у ViewModel, зв’язаний з представленням. Після чого формується запит на отримання чи збереження даних у моделі, яка в свою чергу взаємодіє з базою даних і починається зворотня взаємодія від бази даних до користувача, де база даних повертає результат запиту моделі, яка повертає дані у ViewModel, який в свою чергу оновлює інтерфейс користувача у представленні, яке відображає оновлені дані користувачу. Ці дії та відповідні повідомлення у самій програмі описують дії програми при взаємодії з нею.

2.2 Функціональна структура інформаційної системи

При створенні системи проектування технологічних процесів виготовлення деталей машин за допомогою CASE-технології необхідно розробити функціональну структуру інформаційної системи проектування технологічних процесів з використанням CASE-технології. Функціональну структуру можна представити у вигляді наступних груп функцій: «Робота із складанням», «Робота із деталлю», «Робота із технологічним процесом», «Робота із операцією», «Робота із переходом» та «Загальні функції».

Група функцій «Робота із складанням» містить наступні функції:

- перегляд складань;
- перегляд інформації про вибране складання;
- редагування інформації про складання.

Група функцій «Робота із деталлю» містить наступні функції:

- перегляд деталей;
- перегляд інформації про вибрану деталь;
- додавання нової деталі;
- редагування інформації про деталь;
- видалення деталі.

Група функцій «Робота із технологічним процесом» містить наступні функції:

- перегляд технологічного процесу;
- перегляд інформації про вибраний технологічний процес;
- редагування інформації про технологічний процес;
- додавання нових технологічних процесів;
- видалення технологічних процесів.

Група функцій «Робота із операцією» містить наступні функції:

- перегляд операцій;
- перегляд інформації про вибрану операцію;
- додавання нової операції;

– редагування інформації про операцію;
– видалення операції.

Група функцій «Робота із переходом» містить наступні функції:

– перегляд переходу;
– перегляд інформації про вибраний перехід;
– додавання нового переходу;
– редагування інформації про перехід;
– видалення переходу.

На рисунку 2.3 зображено діаграму активності при користувача з інформаційною системою.

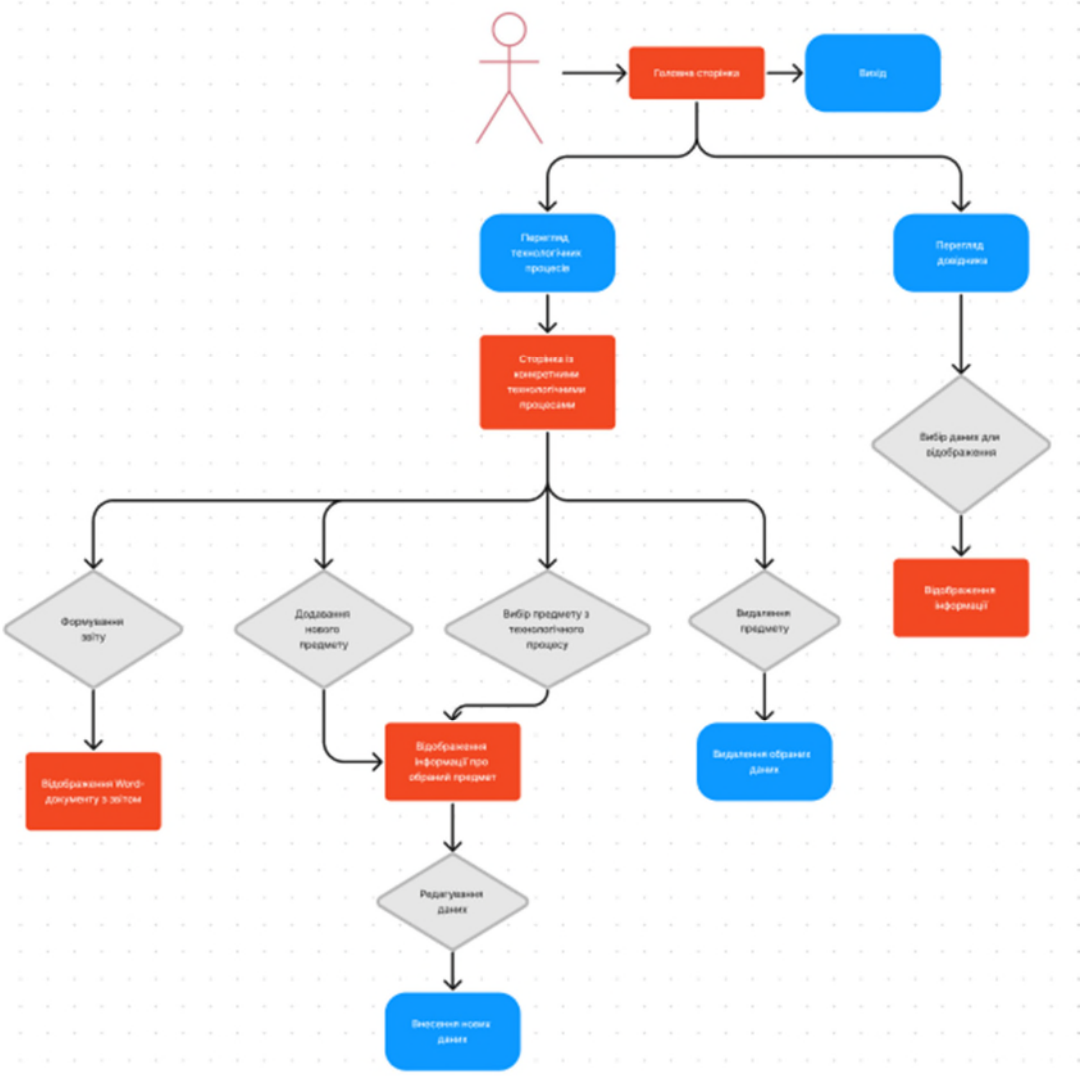


Рисунок 2.3 – Діаграма активності користувача

Діаграма активності ілюструє послідовність дій, які виконуються в процесі використання інформаційної системи користувачем. В наведеній вгорі діаграмі активності користувача прослідковуються можливі дії при користуванні інформаційною системою. При запуску застосунку користувач потраплятиме на головну сторінку, звідки він зможе закрити систему або ж переглянути технологічні процеси чи довідник. При перегляді довідника користувач повинен обрати дані для подальшого відображення, після чого перед ним з’явиться вікно з інформацією. При перегляді технологічних процесів користувач потрапить на сторінку із конкретними технологічними процесами, звідки зможе ініціювати вибір предмету з технологічного процесу, додавання чи видалення елементу технологічного процесу, формування звіту. При формуванні звіту перед користувачем відкриється відображення Word-документу з звітом. При видаленні елементу технологічного процесу станеться видалення обраних даних. При виборі елементу з технологічного процесу чи додаванні елементу технологічного процесу відбудеться відображення інформації про обраний предмет. Після чого буде доступне редагування даних в результаті чого буде занесено нові дані.

На рисунку 2.4 зображено діаграму компонентів інформаційної системи.

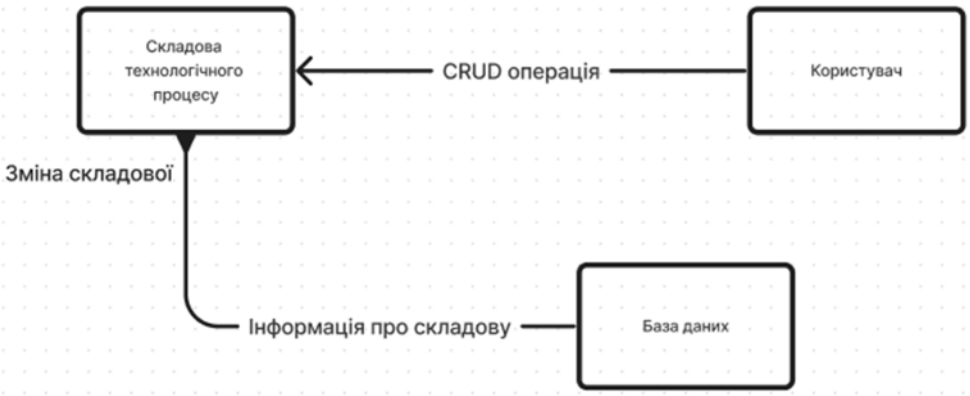


Рисунок 2.4 – Діаграма компонентів

Діаграма компонентів – діаграма, на якій відображаються компоненти, залежності та зв'язки між ними. На діаграмі компонентів, що розміщена вгорі продемонстровано зв’язок при якому користувач виконує CRUD операцію відносно елементу технологічного процесу, яка в свою чергу надає базі даних інформацію про елемент та її зміни. CRUD – («Create», «Read», «Update», «Delete») 4 основні функції управління даними «створення, читання, оновлення і вилучення»[14].

2.3 Проектування структури інформаційної системи

2.3.1 Інфологічна модель даних

Інфологічна модель даних є важливим інструментом у сфері баз даних і системного проектування. Вона дозволяє описати структуру та взаємозв'язки між різними сутностями домену знань або предметної області. Основна мета інфологічної моделі даних - відобразити ключові концепції, залежності та взаємозв'язки між об'єктами, що існують в предметній області[15].

Розроблено інфологічну модель даних з необхідними сутностями для подальшої програмної реалізації інформаційної системи проектування технологічних процесів за CASE-технологією.

Для проектування технологічних процесів виготовлення деталей машин з використанням CASE-технології згідно функціональної структури розроблено інфологічну модель даних (Рисунок 2.5). Вона включає в себе наступні сутності: DbAssemblies(складання), DbDetails(деталі), DbTechProcesses(технологічні процеси), DbOperations(операції), DbOperationTools(інструменти для виконання операцій), DbTransitions(переходи), DbTPViews(види технологічних процесів), DbTPTypes(типи технологічних процесів) та розвідні таблиці DbOperationOperationTools(операції - інструменти для виконання операцій), DbTechProcessTransitions(технологічні процеси - переходи).

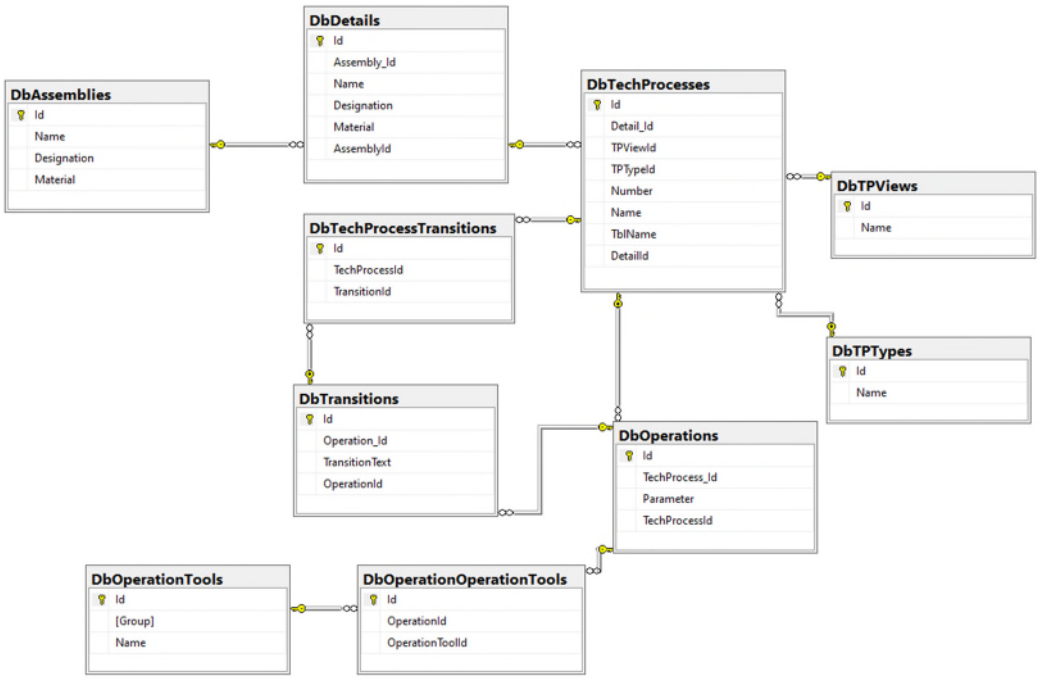


Рисунок 2.5 – Інфологічна модель даних

Сутність «DbAssemblies» призначена для збереження даних про складання.

Таблиця 2.1 – Атрибути сутності «DbAssemblies»

№ п/п	**Назва атрибуту**	**Тип даних**	**Метод**	**Опис**
1	Id	INT	-	Первинний ключ для однозначної ідентифікації записів у таблиці
2	Name	STRING	Nullable	Назва складання
3	Designation	STRING	Nullable	Позначення складання
4	Material	STRING	Nullable	Матеріал складання
5	Details	OBSERVABLE COLLECTION	Nullable	Колекція деталей, що відносяться до складання

Сутність «DbDetails» призначена для збереження даних про деталі.

Таблиця 2.2 – Атрибути сутності «DbDetails»

№ п/п	Назва атрибуту	Тип даних	Метод	Опис
1	Id	INT	-	Первинний ключ для однозначної ідентифікації записів у таблиці
2	Assembly_Id	INT	-	Зовнішній ключ, що вказує до якого складання відноситься деталь
3	Name	STRING	Nullable	Назва деталі
4	Designation	STRING	Nullable	Позначення деталі
5	Material	STRING	Nullable	Матеріал деталі
6	TechProcesses	OBSERVABLE COLLECTION	Nullable	Колекція технологічних процесів, що відносяться до деталі

Сутність «DbTechProcesses» призначена для збереження даних про технологічні процеси.

Таблиця 2.3 – Атрибути сутності «DbTechProcesses»

№ п/п	Назва атрибуту	Тип даних	Метод	Опис
1	Id	INT	-	Первинний ключ для однозначної ідентифікації записів у таблиці

Продовження таблиці 2.3

2	Detail_Id	INT	-	Зовнішній ключ, що вказує до якої деталі відноситься технологічний процес
3	TPViewId	INT	Nullable	Зовнішій ключ, що вказує на вид, для опису деталі технологічного опису
4	TPViews	TPView	Nullable	Обраний вид деталі технологічного процесу
5	TPTypeId	INT	Nullable	Зовнішій ключ, що вказує на тип, для опису деталі технологічного опису
6	TPTypes	TPType	Nullable	Обраний тип деталі технологічного процесу
7	Number	INT	Nullable	Номер елементу
8	Name	STRING	Nullable	Назва елементу
9	TblName	STRING	Nullable	Назва для відображення елементу технологічного процесу у дереві збірки
10	Operations	OBSERVABLE COLLECTION	Nullable	Колекція операцій, що відносяться до технологічного процесу

Сутність «DbTPViews» призначена для збереження даних про вид деталі технологічного процесу.

Таблиця 2.4 – Атрибути сутності «DbTPViews»

№ п/п	Назва атрибуту	Тип даних	Метод	Опис
1	Id	INT	-	Первинний ключ для однозначної ідентифікації записів у таблиці
2	Name	STRING	Nullable	Назва елементу

Сутність «DbTPTypes» призначена для збереження даних про тип деталі технологічного процесу.

Таблиця 2.5 – Атрибути сутності «DbTPTypes»

№ п/п	Назва атрибуту	Тип даних	Метод	Опис
1	Id	INT	-	Первинний ключ для однозначної ідентифікації записів у таблиці
2	Name	STRING	Nullable	Назва елементу

Сутність «DbOperations» призначена для збереження даних про операції.

Таблиця 2.6 – Атрибути сутності «DbOperations»

№ п/п	Назва атрибуту	Тип даних	Метод	Опис
1	Id	INT	-	Первинний ключ для однозначної ідентифікації записів у таблиці

Продовження таблиці 2.6

2	TechProcess_Id	INT	-	Зовнішній ключ, що вказує до якого технологічного процесу відноситься операція
3	Parameter	STRING	Nullable	Текст операції
4	Transitions	OBSERVABLE COLLECTION	Nullable	Колекція переходів, що відносяться до операції

Сутність «DbOperationTools» призначена для збереження даних про інструменти, що використовуються для операцій.

Таблиця 2.7 – Атрибути сутності «DbOperationTools»

№ п/п	**Назва атрибуту**	**Тип даних**	**Метод**	**Опис**
1	Id	INT	-	Первинний ключ для однозначної ідентифікації записів у таблиці
2	Group	STRING	Nullable	Назва групи елементу
3	Name	STRING	Nullable	Назва елементу

Сутність «DbOperationOperationTools» представлено розвідною таблицею для зв’язку між сутностями «DbOperations» та «DbOperationTools».

Таблиця 2.8 – Атрибути сутності «DbOperationOperationTools»

№ п/п	Назва атрибуту	Тип даних	Метод	Опис
1	Id	INT	-	Первинний ключ для однозначної ідентифікації записів у таблиці
2	OperationId	INT	-	Зовнішій ключ, що вказує на операцію для створення зв’язку
3	Operations	Operation	Nullable	Обрана операція для створення зв’язку
4	OperationToolId	INT	-	Зовнішій ключ, що вказує на інструмент операції для створення зв’язку
5	OperationTools	OperationTool	Nullable	Обраний інструмент операції для створення зв’язку

Сутність «DbTransitions» призначена для збереження даних про переходи.

Таблиця 2.9 – Атрибути сутності «DbTransitions»

№ п/п	Назва атрибуту	Тип даних	Метод	Опис
1	Id	INT	-	Первинний ключ для однозначної ідентифікації записів у таблиці

Продовження таблиці 2.9

2	OperationId	INT	-	Зовнішній ключ, що вказує до якої операції відноситься перехід
3	TransitionText	STRING	Nullable	Текст переходу

Сутність «DbTechProcessTransitions» представлено розвідною таблицею для зв’язку між сутностями «DbTechProcesses» та «DbTransitions».

Таблиця 2.10 – Атрибути сутності «DbTechProcessTransitions»

№ п/п	**Назва атрибуту**	**Тип даних**	**Метод**	**Опис**
1	Id	INT	-	Первинний ключ для однозначної ідентифікації записів у таблиці
2	TechProcessId	INT	Nullable	Зовнішій ключ, що вказує на технологічний процес для створення зв’язку
3	TechProcesses	TechProcess	Nullable	Обраний технологічний процес для створення зв’язку
4	TransitionId	INT	-	Зовнішій ключ, що вказує на перехід для створення зв’язку
5	Transitions	Transition	Nullable	Обраний перехід для створення зв’язку

2.3.2 Проектування інтерфейсу інформаційної системи

Проектування інтерфейсу інформаційної системи для кожного програмного продукту включає дві головні складові: UI (user interface - інтерфейс користувача) та UX (user experience - користувацький досвід).

UI описує дизайн елементів, з якими користувач взаємодіє, таких як кнопки, іконки, текстові поля, меню і форми. Це забезпечує можливість користувачеві ефективно взаємодіяти з програмою або сайтом.

З іншого боку, фокус UX полягає у сприйнятті та взаємодії користувачів з продуктом. UX дизайнери намагаються зробити продукт легким у використанні та забезпечити задоволення користувачів[16].

Щоб забезпечити успіх застосунку, дизайнери використовують різні стратегії і техніки, такі як аналіз поведінки користувачів, ітераційне тестування та оптимізацію процесу взаємодії. Гарний дизайн UI та UX допомагає створити зручний, ефективний та привабливий продукт, який задовольняє потреби та очікування користувачів. Це поліпшує користувацький досвід, сприяє збільшенню лояльності та успішності продукту.

У контексті реалізації інтерфейсу, який бачить користувач при запуску застосунку, особливу увагу приділено представленню (View), згаданому вище, під час розгляду користувацького досвіду роботи з програмним продуктом. Нижче наведено зображення стандартного вигляду інтерфейсу.

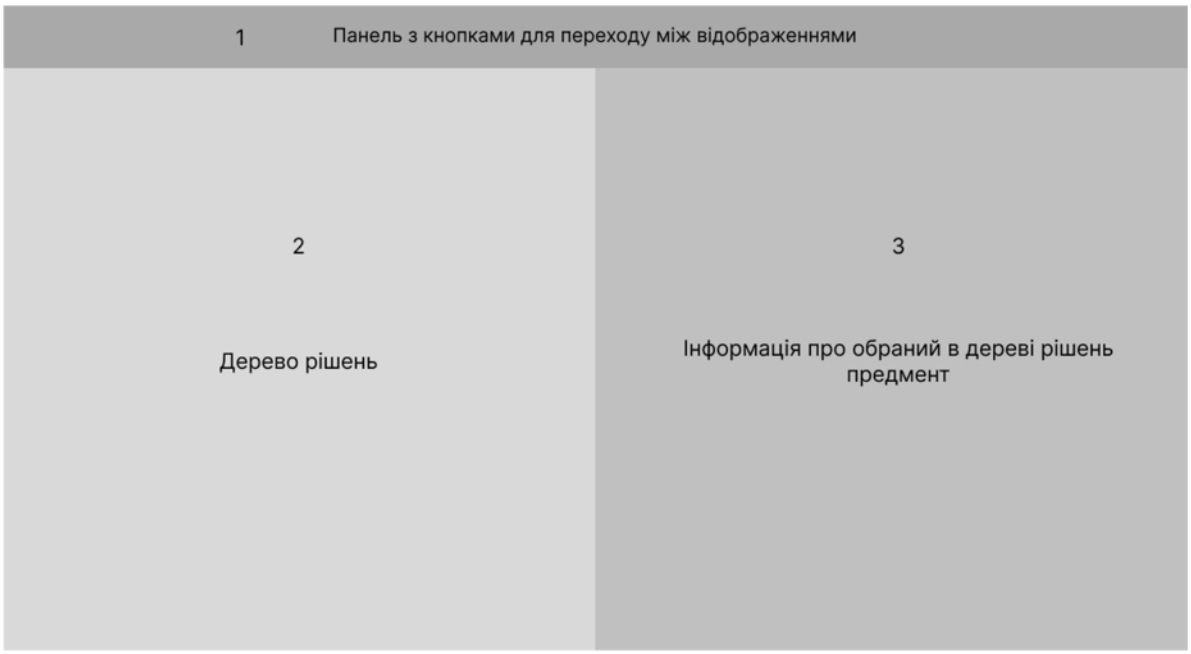


Рисунок 2.6 – Вигляд інтерфейсу за замовчуванням

2.3.3 Проектування архітектури інформаційної системи

Основний підхід, що буде використаний для досягнення зручності користувача і реалізації UX, є використання патерну проектування з шаруванням (Layered architecture). Цей підхід дозволяє розбити код програми на окремі шари, кожен з яких виконує визначену функцію і має свою відповідальність. Кожен шар може використовувати функціональність нижче розташованих шарів, але не може звертатися до функціональності вище розташованих шарів. Цей підхід забезпечує гнучкість та модульність архітектури, що дозволяє змінювати окремі компоненти програми, не впливаючи на решту системи. Він також полегшує тестування та підтримку програми.

Застосування патерну з шаруванням є одним із кроків для створення ефективного інтерфейсу та досягнення високої якості UX для даного продукту[17].

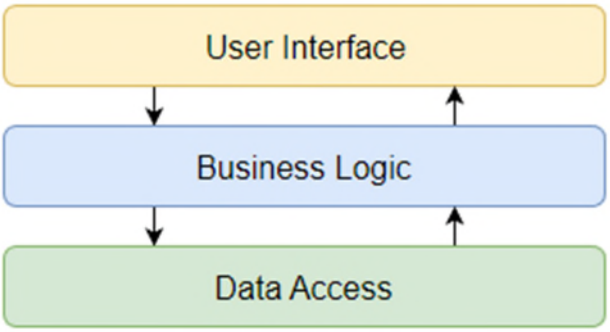


Рисунок 2.7 – Потік даних у Layered architecture

Розглянемо приклад простої ситуації, коли користувач бажає отримати інформацію про певну деталь. Щоб це зробити, він обирає деталь на інтерфейсі, що спричиняє перенаправлення запиту до відповідної функції. Ця функція перевіряє дані про деталь в базі даних і передає їх обробнику, який відображає обрану деталь на інтерфейсі для перегляду користувачем. Цей процес є прикладом використання MVVM моделі.

MVVM (Model-View-ViewModel) - це паттерн проектування, використовуваний в розробці програмного забезпечення, зокрема в галузі UI/UX, з метою розділення логіки програми від способу її відображення. MVVM включає в себе три основні компоненти:модель (Model) - це частина, що представляє дані та бізнес-логіку застосунку;

1) представлення (View) - відповідає за візуалізацію даних на екрані;

2) ViewModel виконує роль посередника між моделлю та представленням, обробляючи події користувацького інтерфейсу та оновлюючи дані в моделі [18].

MVVM в контексті UI/UX дозволяє відокремити логіку застосунку від його представлення, що робить код зрозумілішим, легше підтримуваним та розширюваним. ViewModel служить посередником між моделлю та представленням, що дозволяє обробляти події користувацького інтерфейсу та оновлювати дані в моделі. Це дозволяє краще контролювати та відстежувати стан застосунку, що в свою чергу підвищує його продуктивність та надійність.

2.4 Висновки до розділу 2

У другому розділі розглянуто застосування CASE-технології для моделювання інформаційних систем та проектування технологічних процесів, а також використання UML-моделювання для візуалізації та оптимізації роботи інформаційної системи. Описано різні групи функцій та їх функціональність, а також були побудовані схеми навігації та UML-діаграма активності користувача. Детально описано загальну структуру інфологічної моделі даних та кожен стовбець таблиць. Крім того, розглянуто питання UI (User Interface) та UX (User Experience) та їх можливе застосування.

Використання CASE-технологій та UML-моделювання може сприяти зниженню витрат на розробку та підтримку інформаційної системи та проектуванні технологічних процесів. Вони полегшують комунікацію між учасниками проекту, покращують якість програмного забезпечення і забезпечують ефективне управління процесом розробки. Описані інструменти допомагають побудувати діаграми, таблиці та архітектуру бази даних, а також розробляти UI/UX дизайн для створення застосунків.

Розділ 3 Програмна реалізація інформаційної системи проектування технологічних процесів

3.1 Структура модулів системи, їх взаємозв'язок

Для реалізації інформаційної системи, що включає проектування технологічних процесів з використанням CASE-технології, був використаний патерн проектування MVVM (Model-View-ViewModel), що використовує наступні компоненти: моделі (Models), представлення (Views) та представлення-моделі (ViewModels). Представлення відповідають за відображення даних на екрані. Моделі представляють дані та бізнес-логіку застосунку. ViewModel в свою чергу служить для зв'язку моделі та представлення, обробляє події користувацького інтерфейсу та оновлює дані в моделі.

…

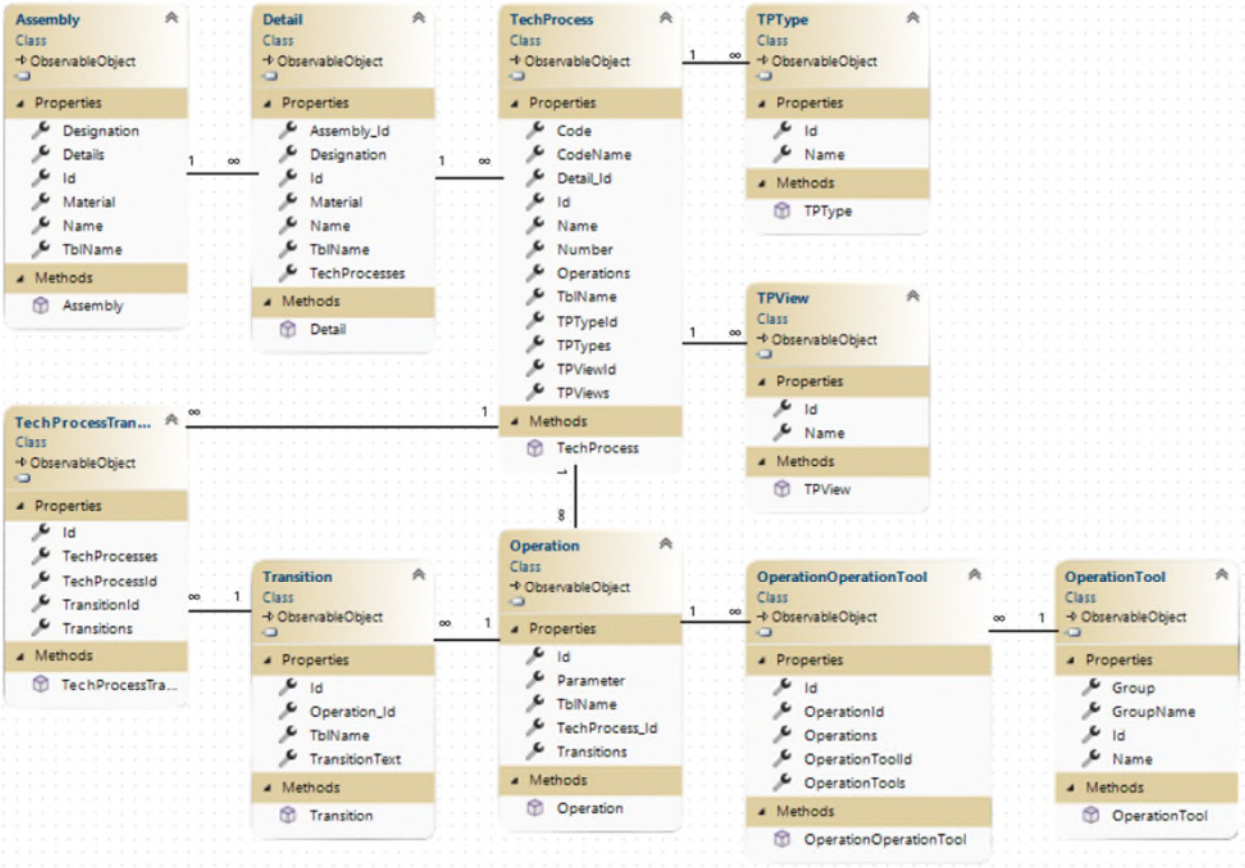


Рисунок 3.1 – Діаграма класів моделей

Діаграма класів включає такі моделі, як:

– Assembly – модель, що відповідає за бізнес-логіку, пов’язана з інформацією про складання;

– Detail – модель, що відповідає за бізнес-логіку, пов’язана з інформацією про деталі;

– TechProcess – модель, що відповідає за бізнес-логіку, пов’язана з інформацією про технологічні процеси;

– TPView – модель, що відповідає за бізнес-логіку, пов’язана з інформацією про види технологічного процесу;

– TPType – модель, що відповідає за бізнес-логіку, пов’язана з інформацією про типи технологічного процесу;

– Operation – модель, що відповідає за бізнес-логіку, пов’язана з інформацією про операції;

– OperationTool – модель, що відповідає за бізнес-логіку, пов’язана з інформацією про інструменти для виконання операцій;

– Transition – модель, що відповідає за бізнес-логіку, пов’язана з інформацією про переходи;

– OperationOperationTool – модель, що відповідає за бізнес-логіку, пов’язана з інформацією про зв’язок двох моеделей Operation та OperationTool;

– TechProcessTransition – модель, що відповідає за бізнес-логіку, пов’язана з інформацією про зв’язок двох моеделей TechProcess та Transition.

Компонент паттерну MVVM, який відповідає за представлення складається з:

– MainWindow – відповідає за відображення початкової сторінки;

– SpecificTP – відповідає за відображення сторінки з деревом збірки;

– Assembly_Info – відповідає за відображення сторінки з інформацією про обране складання;

– Detail_Info – відповідає за відображення сторінки з інформацією про обрану деталь;

– TP_Info – відповідає за відображення сторінки з інформацією про обраний технологічний процес;

– Operation_Info – відповідає за відображення сторінки з інформацією про обрану операцію;

– Transition_Info – відповідає за відображення сторінки з інформацією про обраний перехід;

– NewDetail – відповідає за відображення сторінки для заповнення інформації для створення нової деталі;

– NewOperation – відповідає за відображення сторінки для заповнення інформації для створення нової операції;

– NewTransition – відповідає за відображення сторінки для заповнення інформації для створення нового переходу;

– Directory_OperationTool – відповідає за відображення сторінки з інформацією про доступні інструменти для виконання операції;

– Directory_Views – відповідає за відображення сторінки з інформацією про доступні види технологічного процесу;

– Directory_Type – відповідає за відображення сторінки з інформацією про доступні типи технологічного процесу;

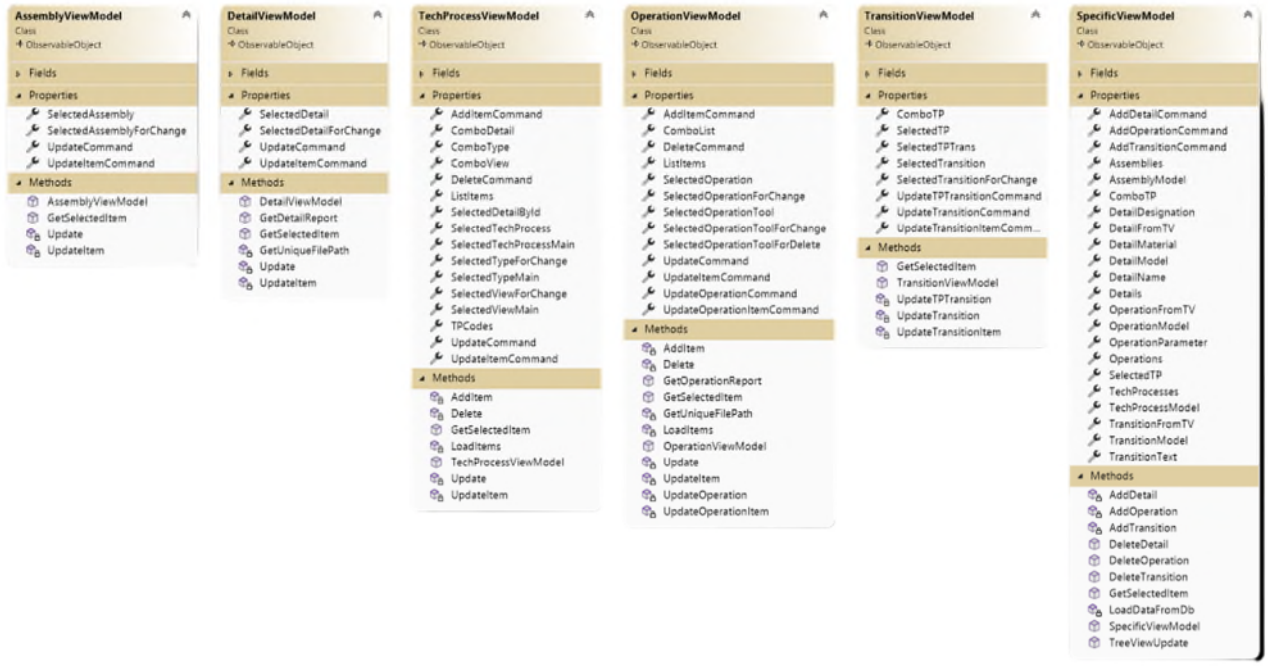


Рисунок 3.2 – Діаграма класів ViewModels

Діаграма класів ViewModels включає наступні класи:

– SpecificViewModel – відповідає роботу з даними та базою даних, створення нових деталей, операцій, переходів та їх видалення;

– AssemblyViewModel – відповідає за роботу з складаннями та базою даних;

– DetailViewModel – відповідає за роботу з деталями та базою даних;

– TechProcessViewModel – відповідає за роботу з технологічними процесами та базою даних;

– OperationViewModel – відповідає за роботу з операціями та базою даних;

– TransitionViewModel – відповідає за роботу з переходами та базою даних.

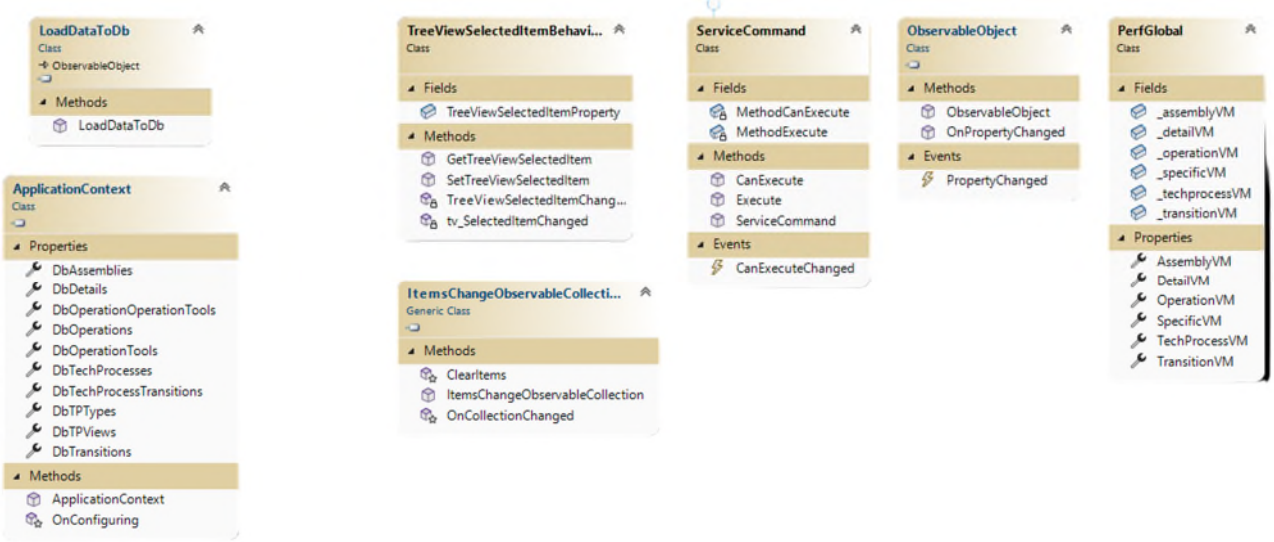


Рисунок 3.3 – Діаграма класів роботи з базою даних та допоміжних класів

Діаграма класів включає класи для роботи з базою даних, та допоміжні класи:

– ApplicationContext – відповідає за створення таблиць та зв’язків між ними;

– LoadDataToDb – відповідає за завантаження початкової бази даних ;

– ObservableObject – відповідає за наслідування події OnPropertyChanged();

– ItemsChangeObservableCollection – відповідає за зміну у дереві збірки;

– TreeViewSelectedItemBehaviour – відповідає за обраний елемент у дереві збірки;

– ServiceCommand – відповідає за роботу команд;

– PerfGlobal – відповідає за можливість використання ViewModels.

3.2 Особливості реалізації програмних складових системи

Для програмної реалізації інформаційної системи проектування технологічних процесів за CASE-технологією обрано мову програмування C# на базі платформи WPF(Windows Presentation Foundation). C# є мовою програмування, розробленою компанією Microsoft. Вона є однією з основних мов для розробки програмного забезпечення під платформу .NET[19].

Для реалізації дизайну у інформаційній системі використано XAML. XAML (eXtensible Application Markup Language) - це мова розмітки, що використовується для створення користувацьких інтерфейсів у технології Windows Presentation Foundation (WPF). XAML дозволяє описувати елементи користувацького інтерфейсу, їх властивості, стилі та анімації у вигляді розмітки, спрощуючи процес дизайну та розробки застосунків [20].

Задля реалізації XAML дизайну платформою розробки обрано WPF. WPF (Windows Presentation Foundation) - технологія розробки програмного забезпечення, що дозволяє створювати різноманітні, привабливі та інтерактивні інтерфейси користувача для Windows-застосунків. WPF має кілька переваг порівняно з традиційним WinForms, включаючи вбудовану підтримку графіки, 3D-рендеринг, анімацію, стилі та шаблони, відокремлення логіки та представлення. Обрано платформою розробки, так як надає потужні засоби для створення модерних. Загалом дана платформа розробки є потужним інструментом для розробки графічних інтерфейсів користувача в операційній системі Windows, адже надає можливість візуально описувати структуру та вигляд елементів інтерфейсу і взаємодіяти з ними різними методами[21].

Для зручності розділення логіки продукту на підсистеми, що зменшує складність, підвищує перевикористовування коду і полегшує модифікацію системи, вирішено використовувати Layered architecture. Переваги шарової архітектури включають зменшення залежності між компонентами, покращення масштабованості і підтримку різних платформ.

Підхід Code First використано при побудові бази даних, де таблиці створюються на основі моделей проекту[22]. Цей підхід використовує фреймворк "Entity Framework", який є технологією доступу до даних для платформи .NET. Entity Framework спрощує взаємодію програмістів з базою даних, надаючи об'єктно-орієнтовану парадигму замість складних SQL-запитів[23].

Entity Framework можна розглядати як структуру об'єктно-реляційного відображення, що забезпечує зв'язок між базами даних і об'єктно-орієнтованими мовами програмування. Використання об'єктно-реляційного відображення дозволяє ефективно працювати з базами даних, скорочує кількість написаного коду і покращує безпеку[24].

Основні переваги використання Entity Framework включають спрощену взаємодію з базою даних, автоматичне створення таблиць, міграції бази даних, кешування та оптимізацію, а також підтримку багатьох провайдерів баз даних.

Методологія Code First відноситься до підходу розробки програмного забезпечення, де спочатку створюються моделі даних, а потім на їх основі автоматично генерується база даних та код для доступу до неї. Використання Code First спрощує розробку та підтримку бази даних, дозволяє працювати з даними у вигляді об'єктів і забезпечує розширення та підтримку різних баз даних.

Отже, використання Entity Framework у поєднанні з підходом Code First дозволяє спростити розробку та підтримку бази даних, забезпечує автоматичне створення таблиць, міграції бази даних, кешування та оптимізацію, а також підтримку багатьох провайдерів баз даних. Це робить процес розробки більш ефективним і швидким, оскільки програмістам не потрібно вручну створювати таблиці та виконувати складні SQL-запити.

Крім того, підхід Code First надає гнучкість управління структурою бази даних. Починаючи з моделей даних, розробники можуть визначати відношення між таблицями, налаштовувати обмеження цілісності даних і виконувати інші настройки бази даних. При цьому є можливість використовувати міграції бази даних для збереження змін структури бази даних та оновлення існуючих даних без втрати інформації.

Загалом, використання підходу Code First з Entity Framework є потужним інструментом для розробки та керування базами даних. Він дозволяє зосередитись на розробці моделей даних і бізнес-логіки, спрощує взаємодію з базою даних і забезпечує гнучкість управління структурою бази даних[25].

Розглядаючи програмну реалізацію інформаційної системи проектування технологічних процесів за CASE-технологією часто зустрічаються операції CRUD, а саме Create(Створення), Read(Читання), Update(Оновлення), Delete(Видалення). Ці операції є основними операціями для управління даними в багатьох типах програмних додатків і зазвичай використовуються для взаємодії з базою даних. CRUD дозволяє додавати, зчитувати, оновлювати та видаляти дані, що дозволяє забезпечити повну функціональність системи керування даними.

3.3 Опис функціональних можливостей інформаційної системи

На рисунку 3.4 продемонстровано початкове вікно застосунку.

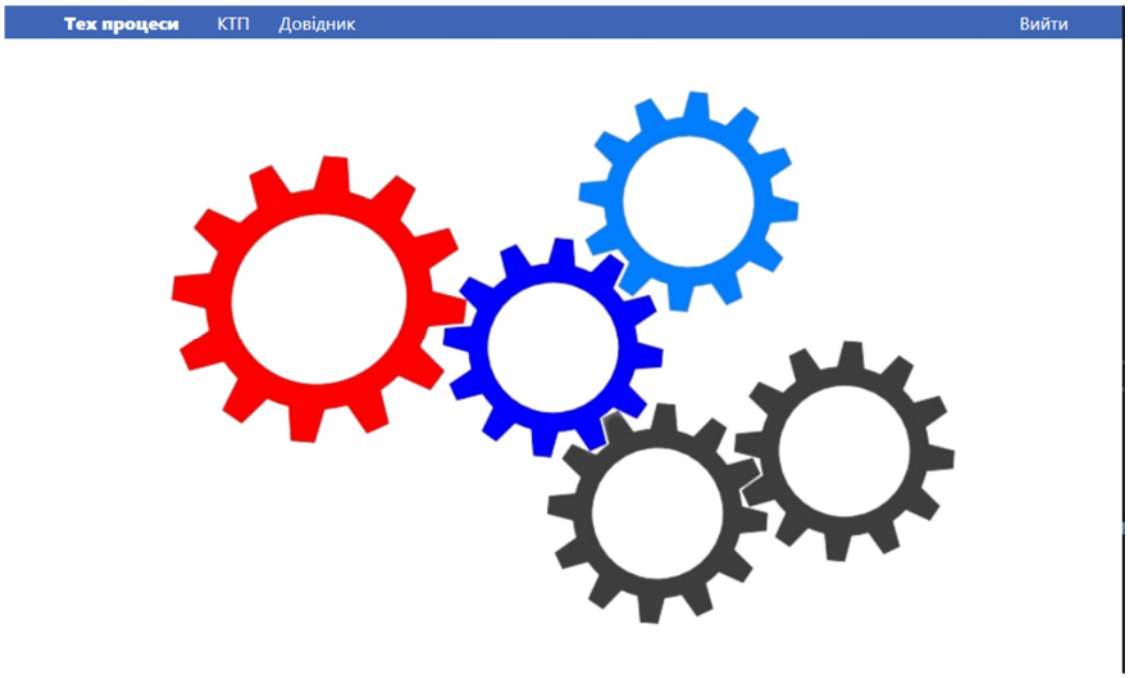


Рисунок 3.4 – Головне вікно застосунку

Для відображення дерева збірки необхідно натиснути кнопку «КТП» на верхній панелі вікна.

Рисунок 3.5 – Вікно з деревом збірки

Для відображення інформації про обраний елемент дерева збірки в правій частині застосунку необхідно натиснути на будь-який елемент з дерева збірки, інформацію про який необхідно переглянути.

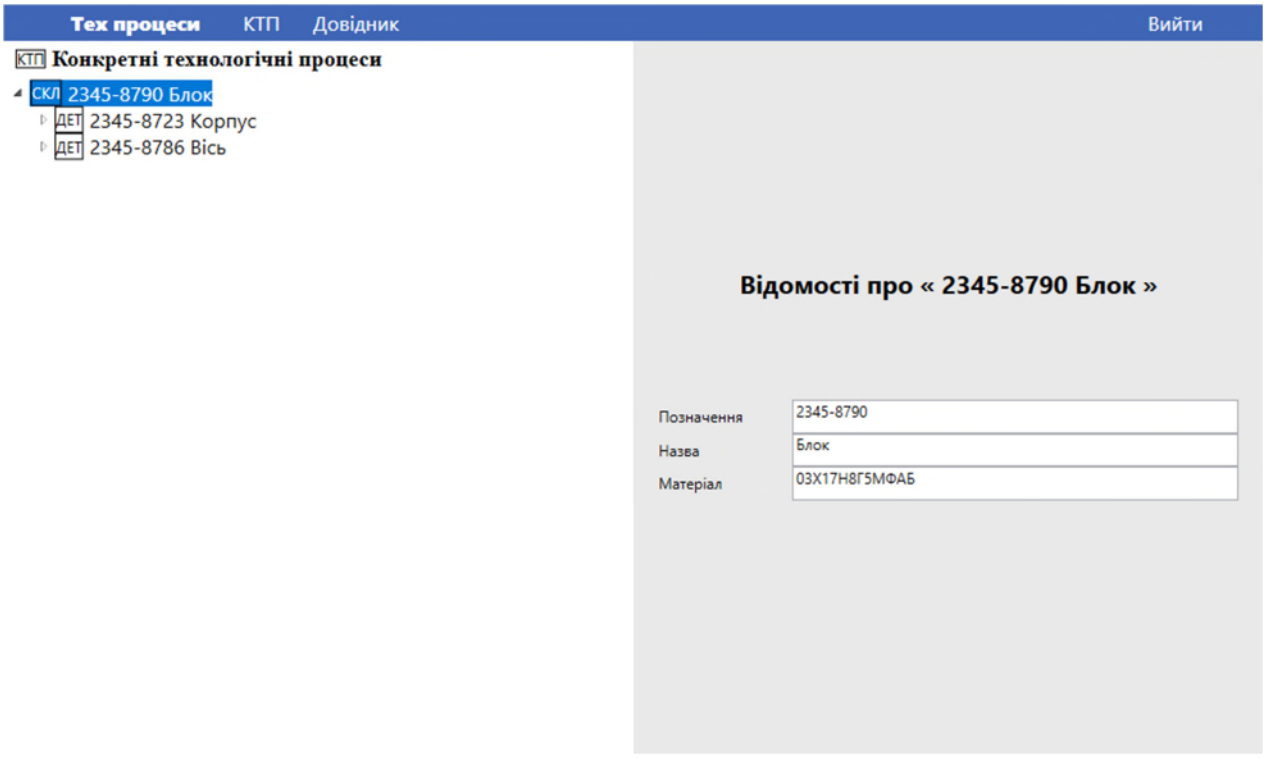


Рисунок 3.6 – Відображення інформацій про елемент дерева збірки

Для зміни інформації як про складання, так і про деталь, необхідно натиснути правою клавішою миші на текст, де вказано, про який елемент надаються відомості.

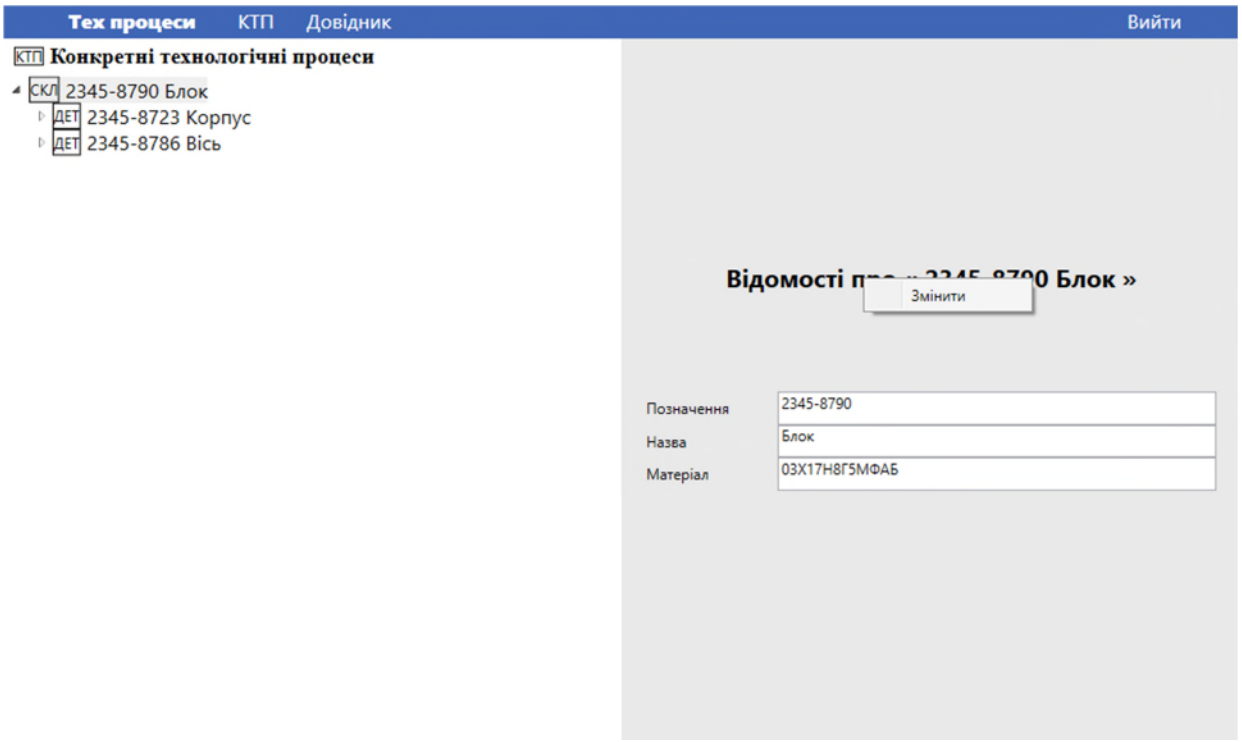


Рисунок 3.7 – Вибір кнопки «Змінити» для обраного елементу

Для зміни інформації про елемент необхідно натиснути кнопку «Змінити». Для підтвердження нової інформації необхідно натиснути кнопку, що з’явиться під полями з інформацією.

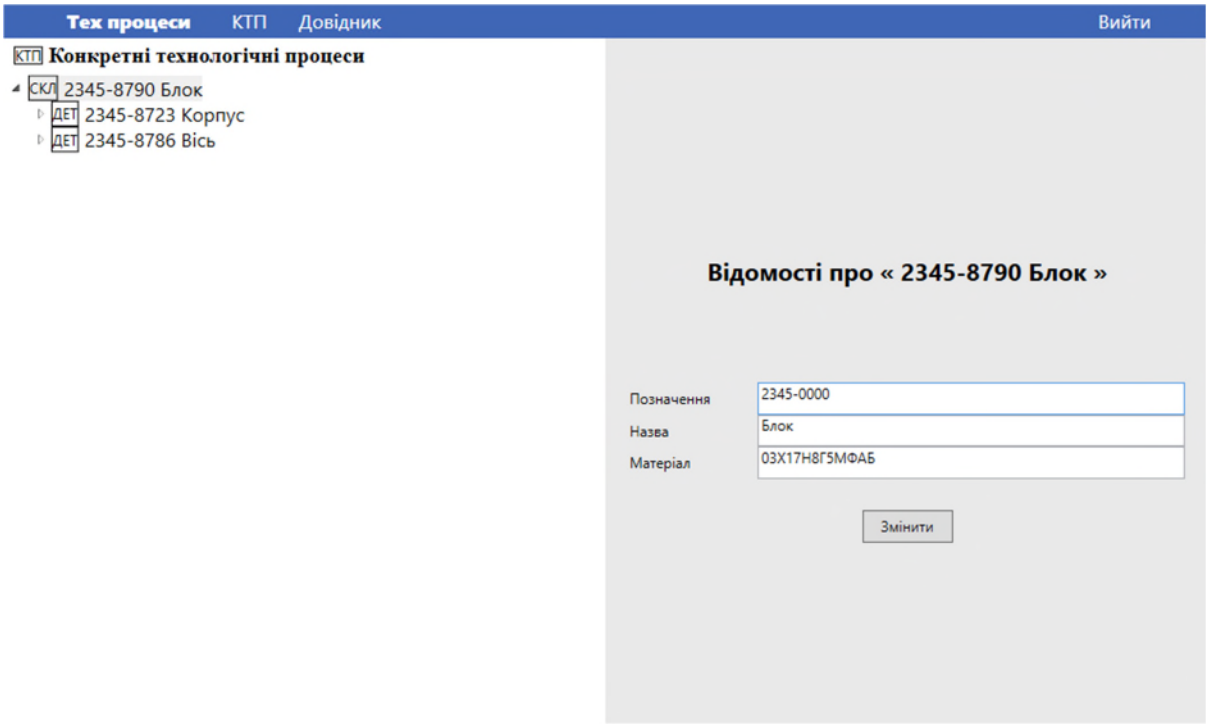


Рисунок 3.8 – Зміна інформації про елемент

Після зміни інформації і натискання кнопки «Змінити», кнопка зникне, а інформація оновиться, як в застосунку, так і в базі даних.

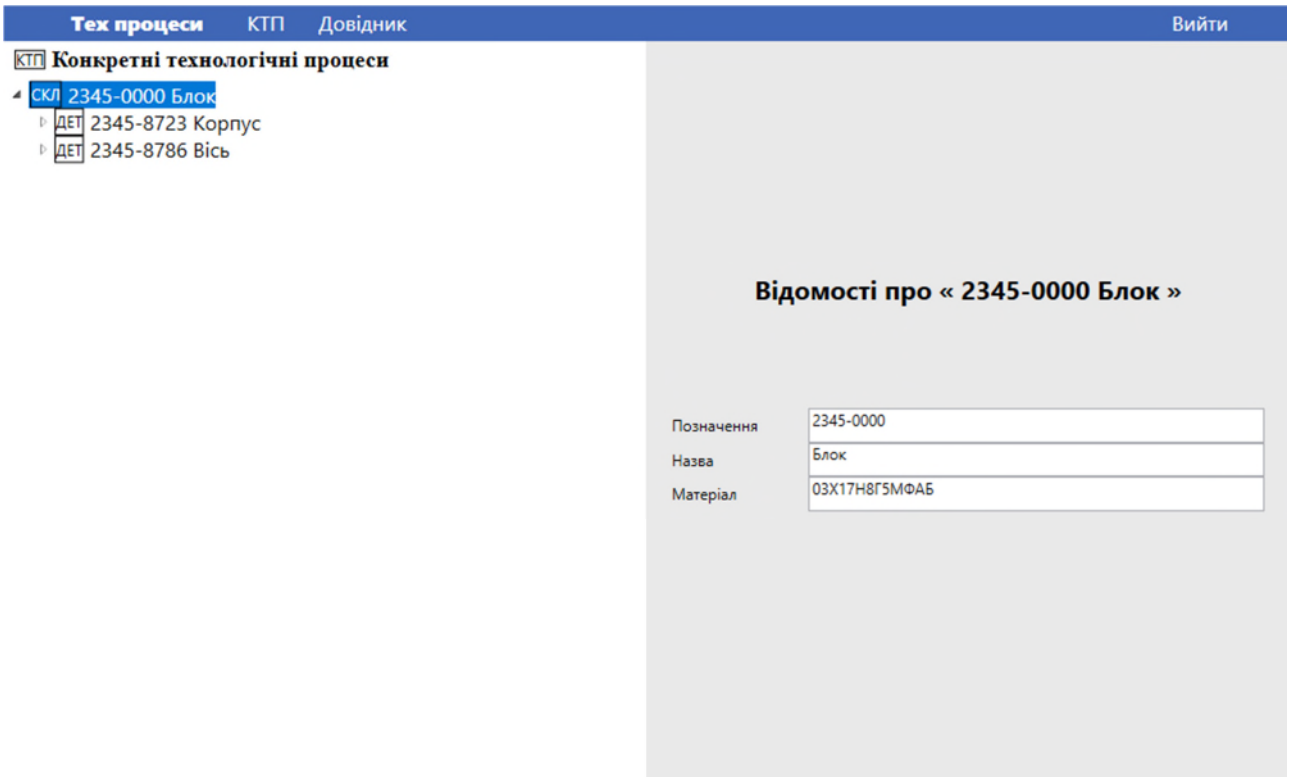


Рисунок 3.9 – Оновлена інформація про елемент

Для додавання чи видалення деталі необхідно натиснути правою кнопкою миші по обраній деталі.

Для додавання нової деталі необхідно натиснути кнопку «Додати», після чого з'явиться можливість ввести інформацію про нову деталь.

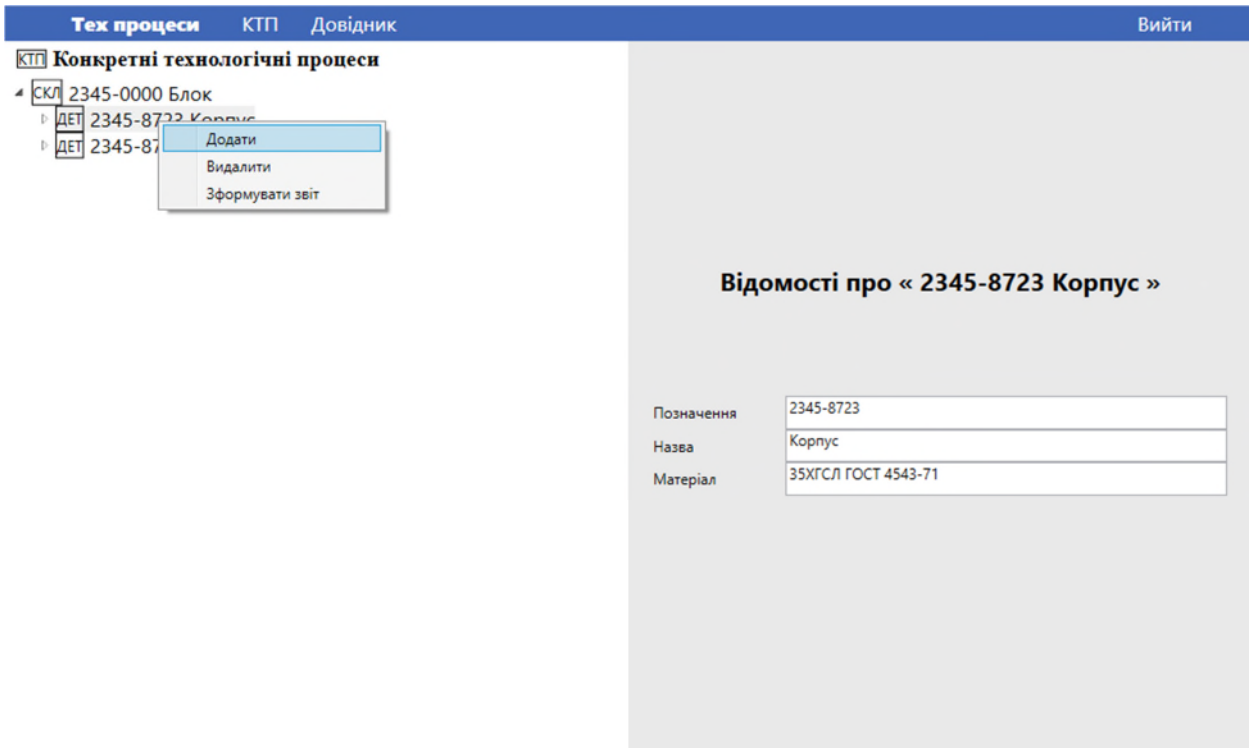


Рисунок 3.10 – Додавання нової деталі

Для додавання нової деталі, необхідно ввести інформацію про неї у відповідні поля.

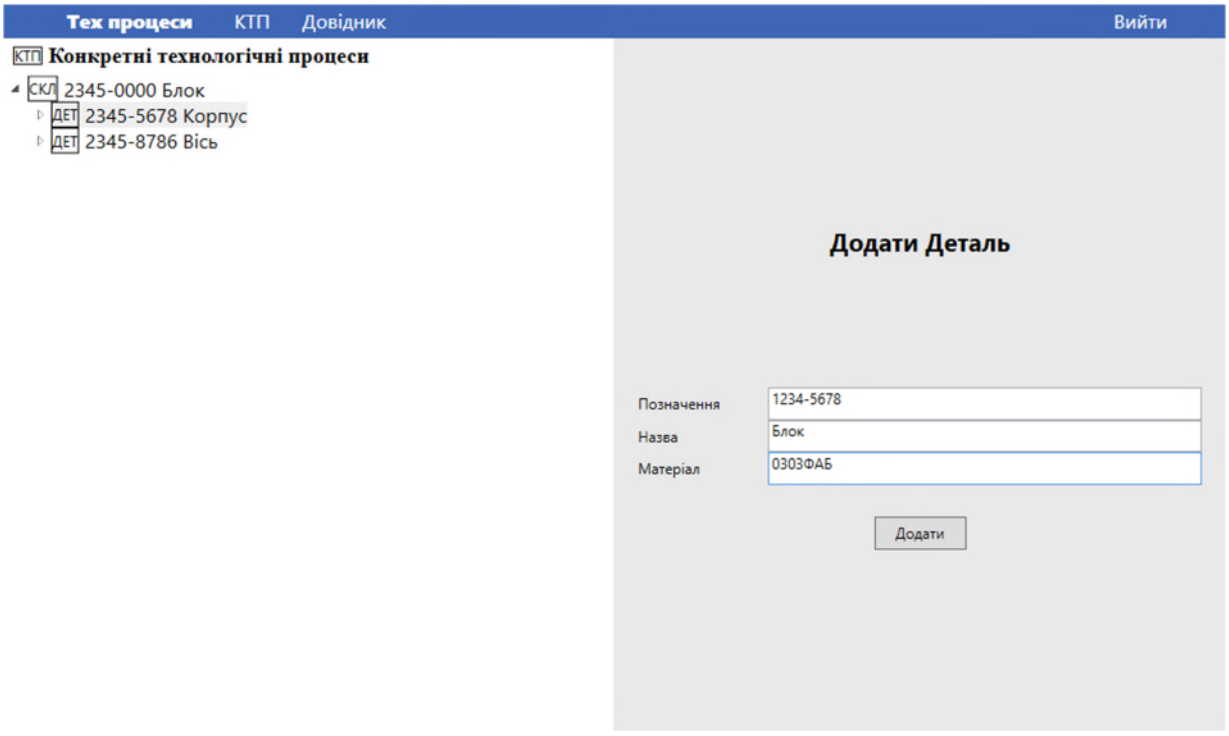


Рисунок 3.11 – Заповнення інформації

Після натискання кнопки «Додати», в дереві збірки з’явиться нова деталь з внесеною інформацією, як в застосунку, так і в базі даних.

Також, після додавання деталі, автоматично створено пустий технологічний процес.

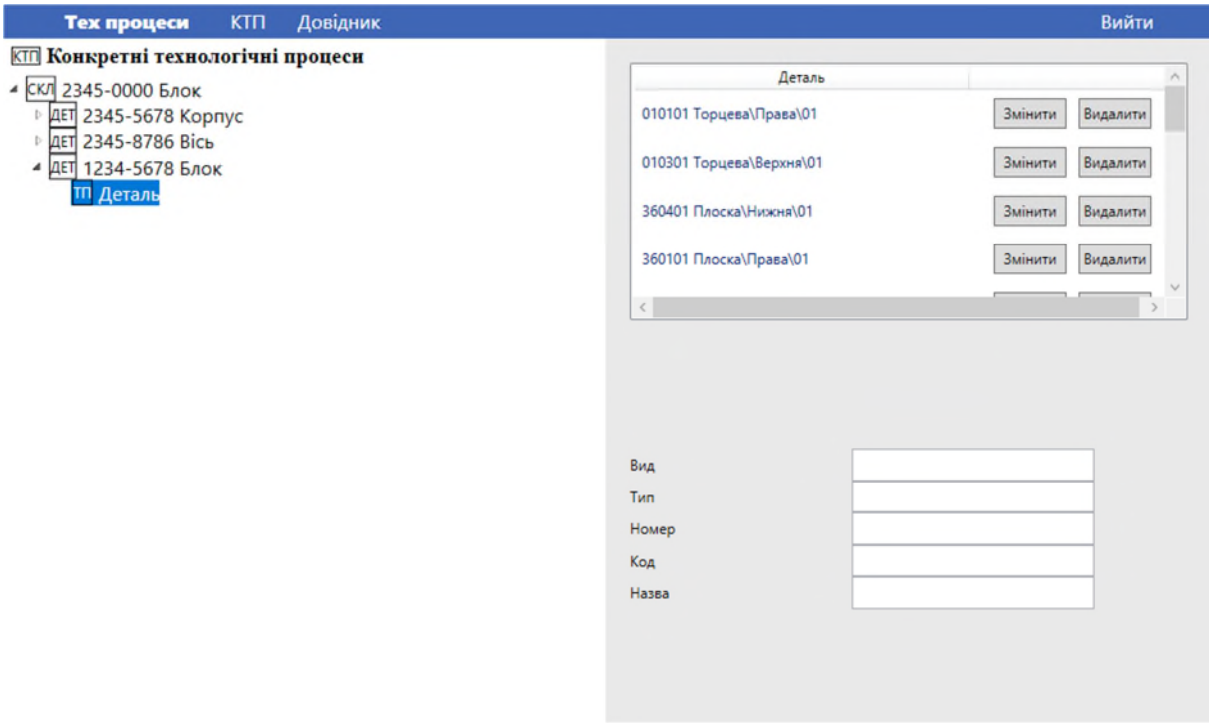


Рисунок 3.12 – Автоматично згенерований технологічний процес

Для видалення деталі необхідно натиснути правою кнопку миші на додану раніше деталь та обрати з випадаючого контекстного меню кнопку «Видалити». Після чого інформація про наявні деталі в дереві збірки оновиться і видалену деталь, як і її технологічний процес видалено, так як і в базі даних.

Для формування звіту для деталі необхідно натиснути правою кнопку миші на деталь та обрати з випадаючого контекстного меню кнопку «Зформувати звіт». Після чого створено новий Word-документ, що автоматично відкрито.

ЗВІТ ПРО ФОРМУВАННЯ ДЕТАЛІ

Назва деталі	2345-5678 Корпус	**Матеріал деталі**	35ХГСЛ ГОСТ 4543-71

ОПЕРАЦІЇ ДЛЯ СТВОРЕННЯ ДЕТАЛІ

Операція №1	Лиття по виплавлюваним моделям	**Інструменти, що були використані для виконання операції**	Б Піч для розігріву розплаву
Операція №2	Слюсарна	**Інструменти, що були використані для виконання операції**	Б Верстак
Операція №3	Контрольна	**Інструменти, що були використані для виконання операції**	
Операція №4	Очищення	**Інструменти, що були використані для виконання операції**	Б Апарат дробоструминний
Операція №5	Травлення ел.хімічних	**Інструменти, що були використані для виконання операції**	Б Ванна електрохімічна
Операція №6	Контрольна	**Інструменти, що були використані для виконання операції**	Б Стіл контрольний
Операція №7	Розмітка	**Інструменти, що були використані для виконання операції**	Б Прес гідравлічний
Операція №8	Фрезерна	**Інструменти, що були використані для виконання операції**	Б Вертикально-фрезерний 654РФ3, ПР Прихват 80х30 7011-0057 ГОСТ 12937-67
Операція №9	Слюсарна	**Інструменти, що були використані для виконання операції**	Б Верстак
Операція №10	Токарна	**Інструменти, що були використані для виконання операції**	Б Токарно-центровий 16К20, ПР Планшайба кутова
Операція №11	Розмітка	**Інструменти, що були використані для виконання операції**	Б Плита розмічувальна
Операція №12	Фрезерна	**Інструменти, що були використані для виконання операції**	Б Ножиці гільйотинні, Б Універсально-фрезерний 67К16В, ПР Прихват 80х30 7011-0057 ГОСТ 12937-67, ПР Головка ділильна D160 7036-0051 ГОСТ 8615-89
Операція №13	Слюсарна	**Інструменти, що були використані для виконання операції**	Б Верстак
Операція №14	Розточувальна	**Інструменти, що були використані для виконання операції**	Б Координатно-розточувальний, ПР Прихват 80х30 7011-0057 ГОСТ 12937-67
Операція №15	Розмітка	**Інструменти, що були використані для виконання операції**	Б Плита розмічувальна
Операція №16	Свердлильна	**Інструменти, що були використані для виконання операції**	Б Вертикально-свердлильний 2С132Ф2І
Операція №17	Слюсарна	**Інструменти, що були використані для виконання операції**	Б Верстак

Рисунок 3.13 – Вигляд Word-документу звіту деталі

У сформованому звіті описано всі операції та інструменти необхідні для виконання кожної із операцій для створення обраної деталі.

На шар нижче у дереві збірки описано технологічні процеси, вони представлені у виді ListView. Для зміни чи видалення технологічного процесу необхідно натиснути на відповідну кнопку, що знаходиться у рядку з найменуванням технологічного процесу у ListView. Для додавання нового технологічного процесу необхідно натиснути правою кнопкою миші в будь-якому місці списку, після чого з’явиться контекстне меню з кнопкою «Додати».

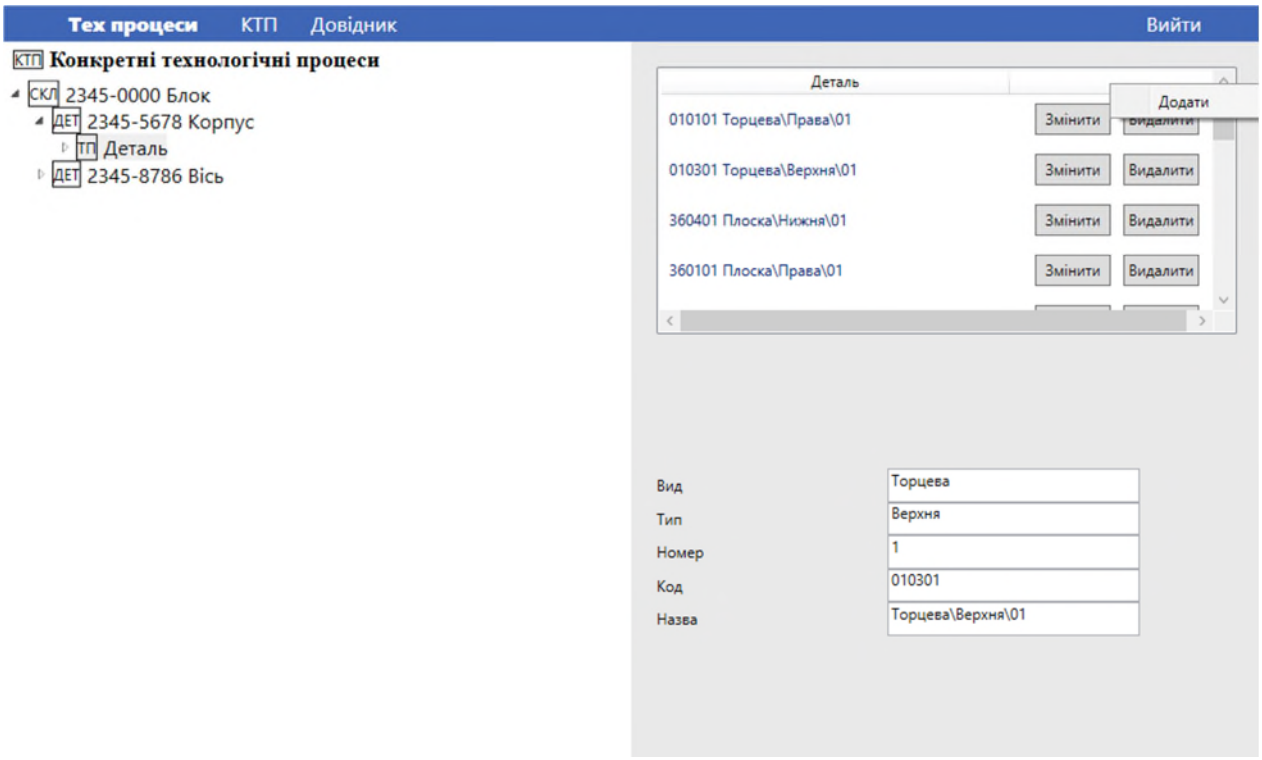


Рисунок 3.14 – Відображення сторінки з технологічними процесами

Для створення технологічного процесу необхідно натиснути на кнопку «Додати», після чого наповнення сторінки зміниться. З ComboBox необхідно обрати до якої деталі відноситься даний технологічний процес, а також його вид та тип. Після чого необхідно натиснути на кнопку «Додати» під полями з інформацією або кнопку «Відмінити», якщо потреби у додаванні технологічного процесу немає.

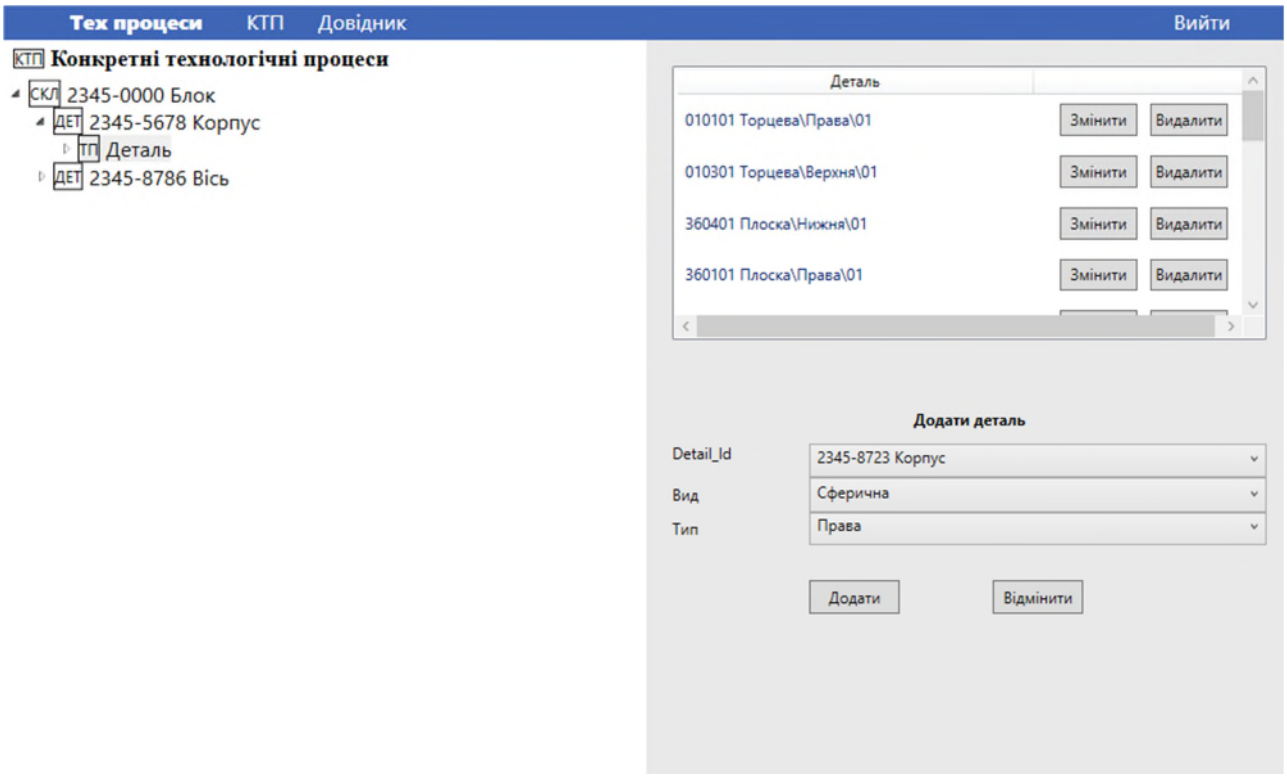


Рисунок 3.15 – Створення нового технологічного процесу

Після додавання нового технологічного процесу, його записано у ListView, також з’явиться можливість переглянути інформацію про нього.

Для зміни інформації про технологічний процес необхідно натиснути кнопку «Змінити» біля назви технологічного процесу.

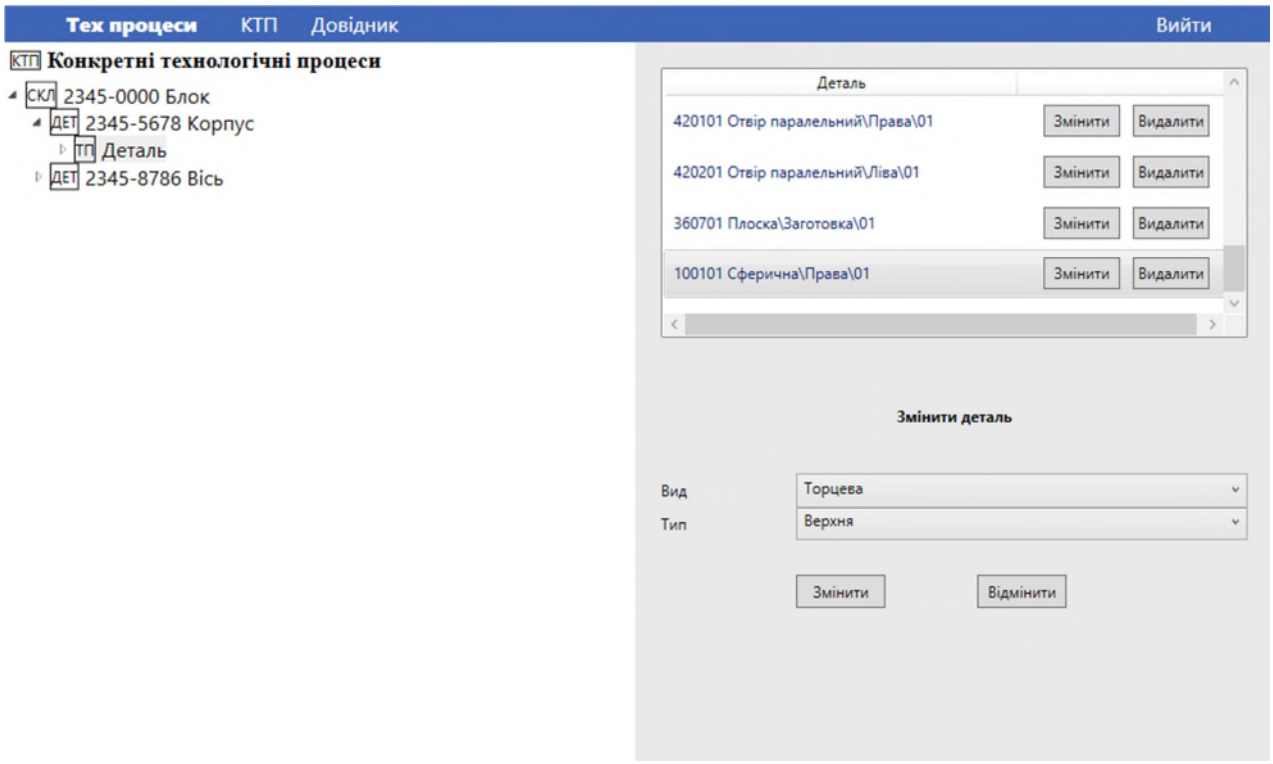


Рисунок 3.16 – Зміна інформації технологічного процесу

Під час зміни інформації обрано такі ж вид та тип технологічного процесу, як у технологічного процесу вище, після зміни інформації номер автоматично змінено, щоб не зустрічалось повторень.

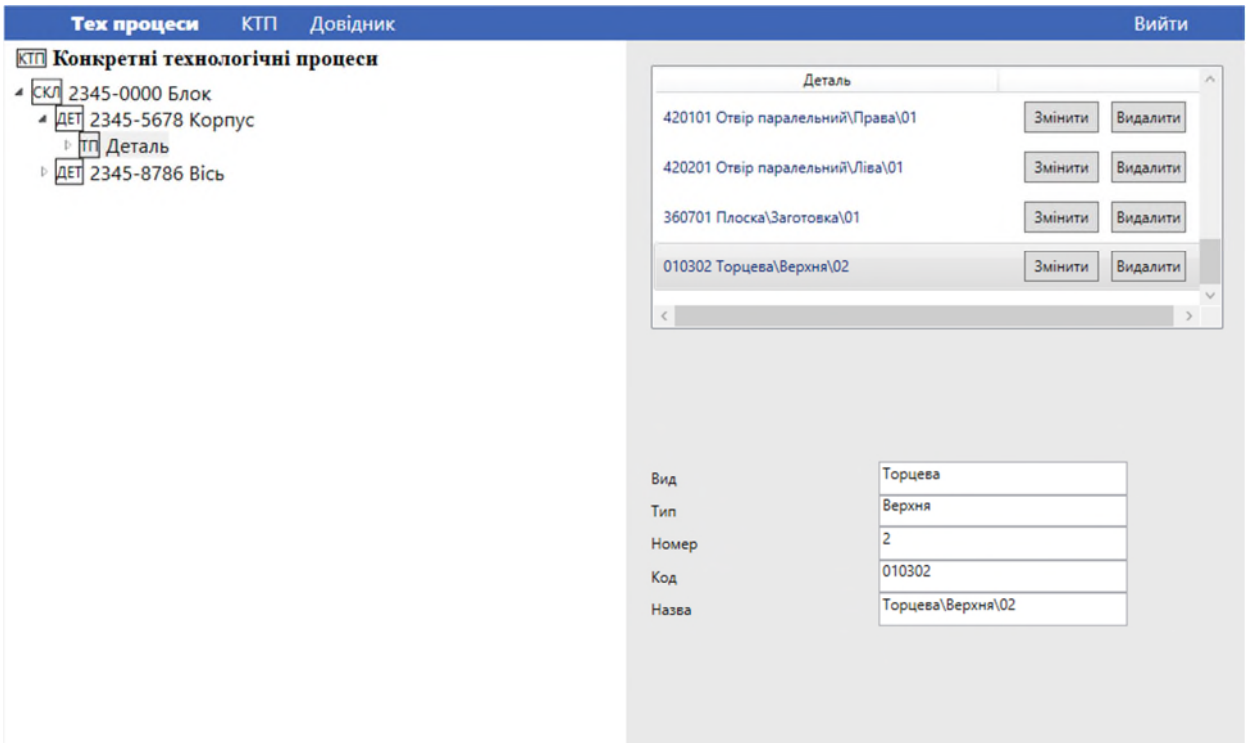


Рисунок 3.17 – Змінена інформація про технологічний процес

Для видалення технологічного процесу необхідно натиснути кнопку «Видалити», це призведе до видалення інформації про нього, як з списку, так і з бази даних.

На шар нижче у дереві збірки описано операції, для перегляду інформації про які, так само необхідно натиснути на них у дереві збірки.

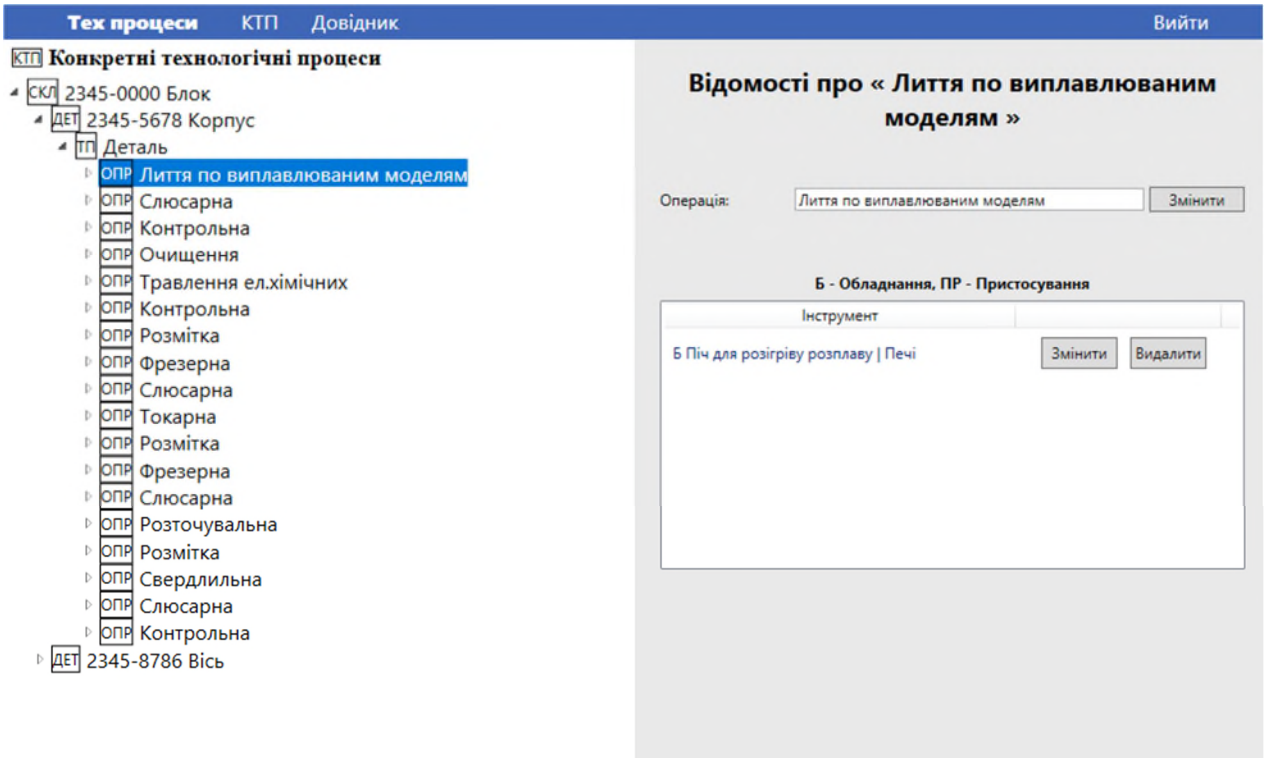


Рисунок 3.18 – Відображення інформацію про операцію

При натисканні на будь-яку з операцій в дереві збірки правою кнопкою миші з’явиться контекстне меню з кнопками «Додати», «Видалити» та «Зформувати звіт».

Для зміни інформації про операцію необхідно натиснути кнопку «Змінити», після чого змінити інформацію та натиснути кнопку «Підтвердити».

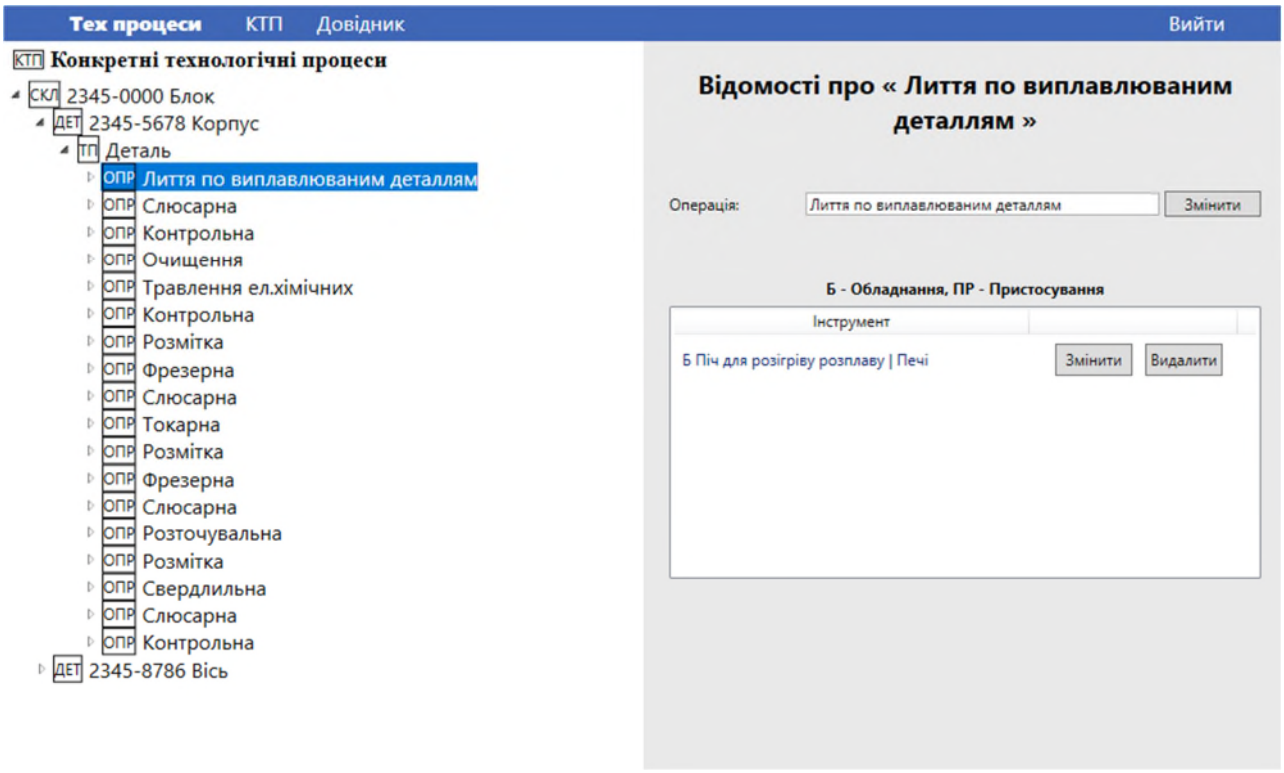


Рисунок 3.19 – Результат зміни інформації про операцію

При відображенні інформації про обрану операцію, за допомогою ListView описано інструменти, що необхідні для виконання даної операції.

Для додавання нового інструменту до операції необхідно правою кнопкою миші натиснути по тексту над ListView, після чого натиснути кнопку «Додати», що з’явиться. Наповнення сторінки зміниться, для додавання необхідно обрати з ComboBox потрібний інструмент та натиснути кнопку «Додати» або кнопку «Відмінити», якщо потреби у додаванні інструменту немає.

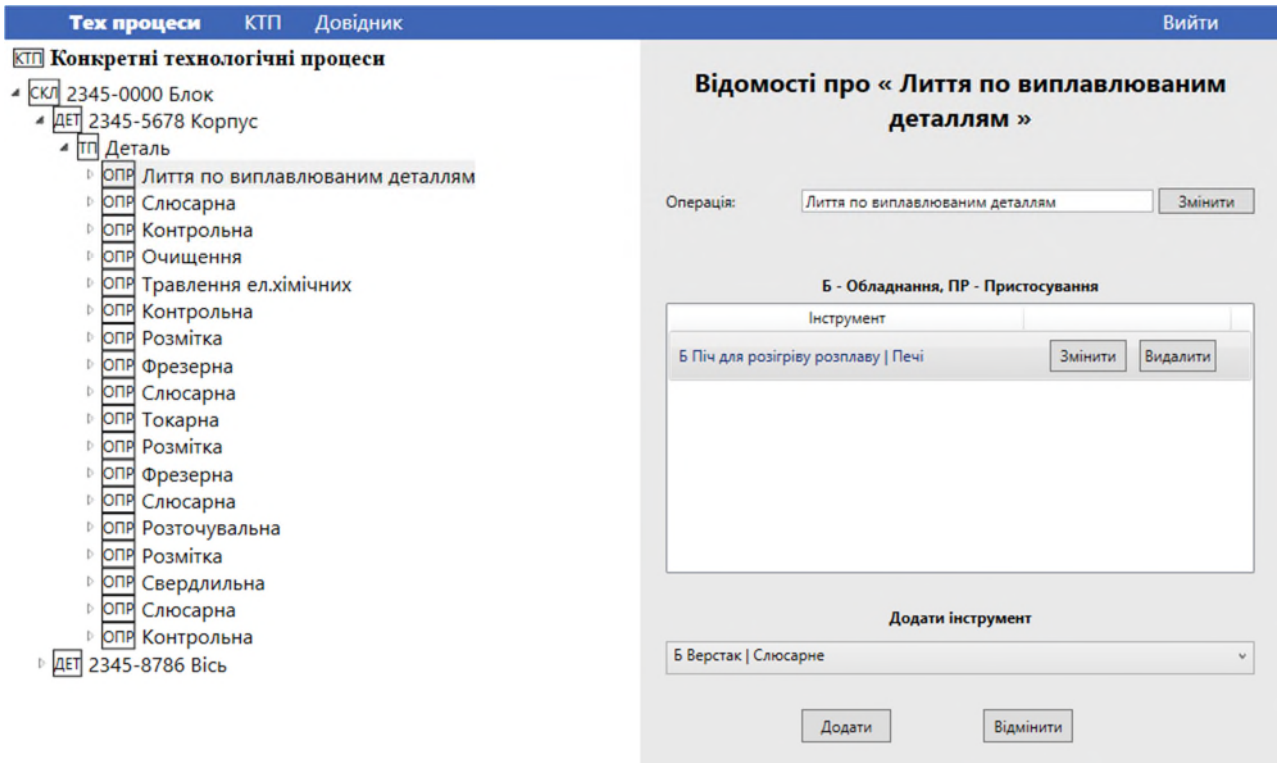


Рисунок 3.20 – Вибір потрібного інструменту

Після натискання кнопки «Додати», обраний інструмент додано в список інструментів даної операції, а в базі даних створено новий запис в розвідній таблиці, що зв’язує операцію та інструмент для виконання операції.

Для зміни інструменту необхідно натиснути кнопку «Змінити» біля інструменту і обрати інший інструмент в ComboBox, що з’явиться, після чого натиснути кнопку «Змінити».

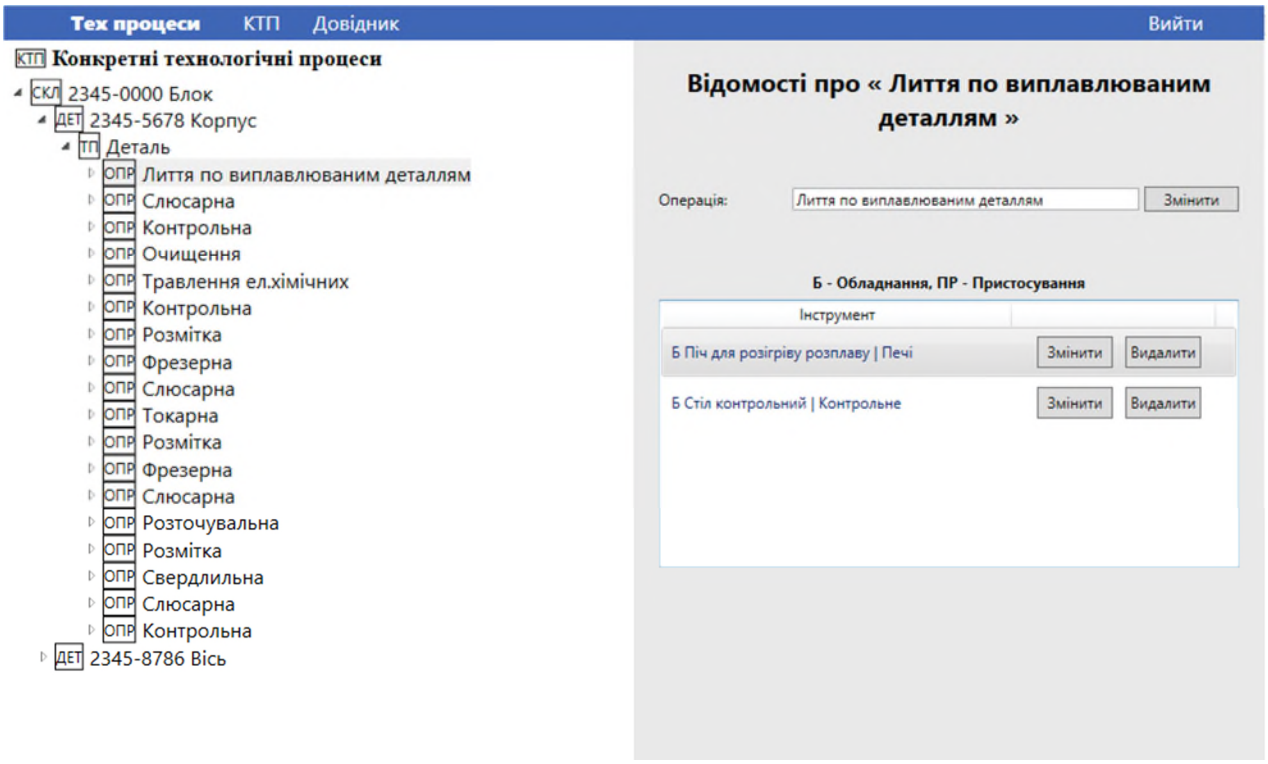


Рисунок 3.21 – Результат зміни інструменту

Для видалення інструменту необхідно натиснути кнопку «Видалити» біля інструменту.

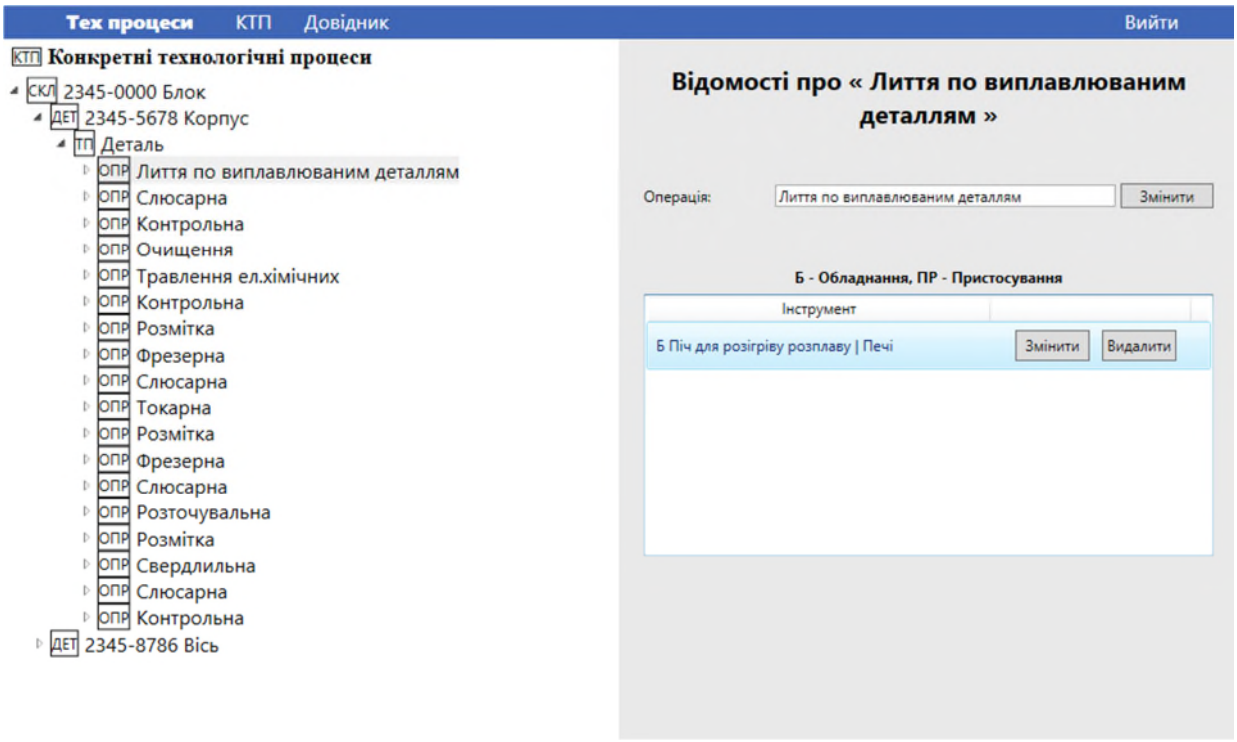


Рисунок 3.22 – Результат після видалення інструменту

В результаті інструмент видалено, і він зник з списку та бази даних.

На шар нижче у дереві збірки описано переходи.

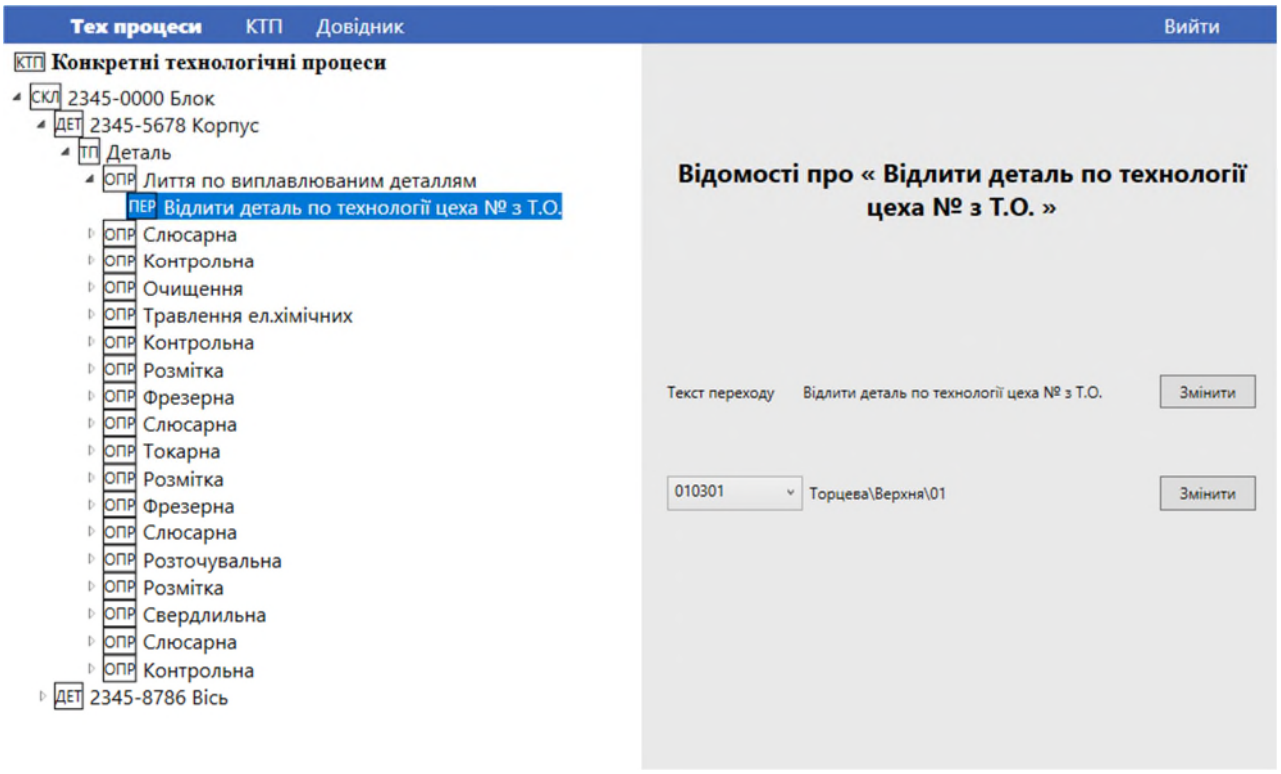


Рисунок 3.23 – Відображення інформації про обраний перехід

При натисканні на будь-який з переходів в дереві збірки правою кнопкою миші з’явиться контекстне меню з кнопками «Додати» і «Видалити». Інформація про перехід зміниться так, як і в операції.

Для зміни технологічного процесу, до якого відноситься перехід необідно обрати інший код технологічного процесу після чого натиснути кнопку «Змінити».

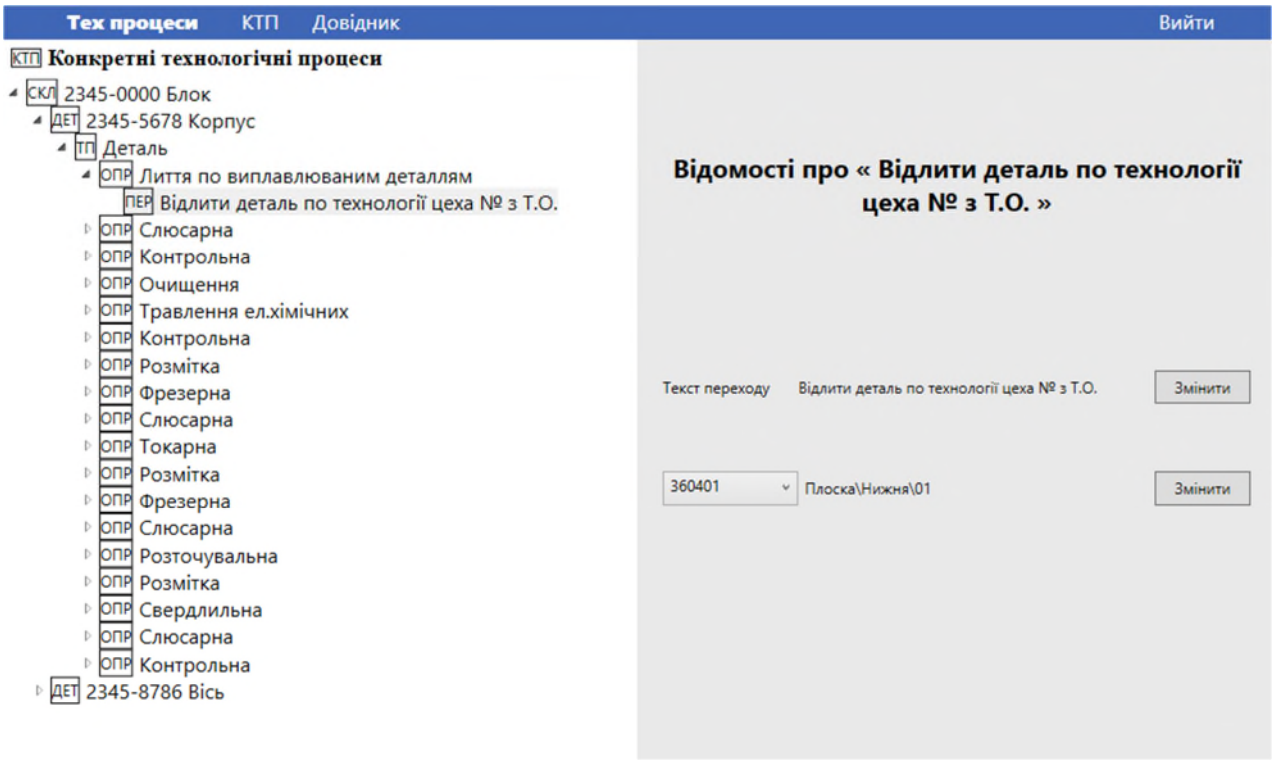


Рисунок 3.24 – Результат зміни технологічного процесу

Для взаємодії із вкладкою «Довідник» необхідно натиснути правою кнопкою миші на «Довідник», після чого з’явиться контекстне меню з вибором між «Вид елемента», «Тип елемента» та «Інструменти для виконання операцій». Для того, щоб відкрити будь-яке вікно необхідно натиснути на кнпоку з відповідною назвою.

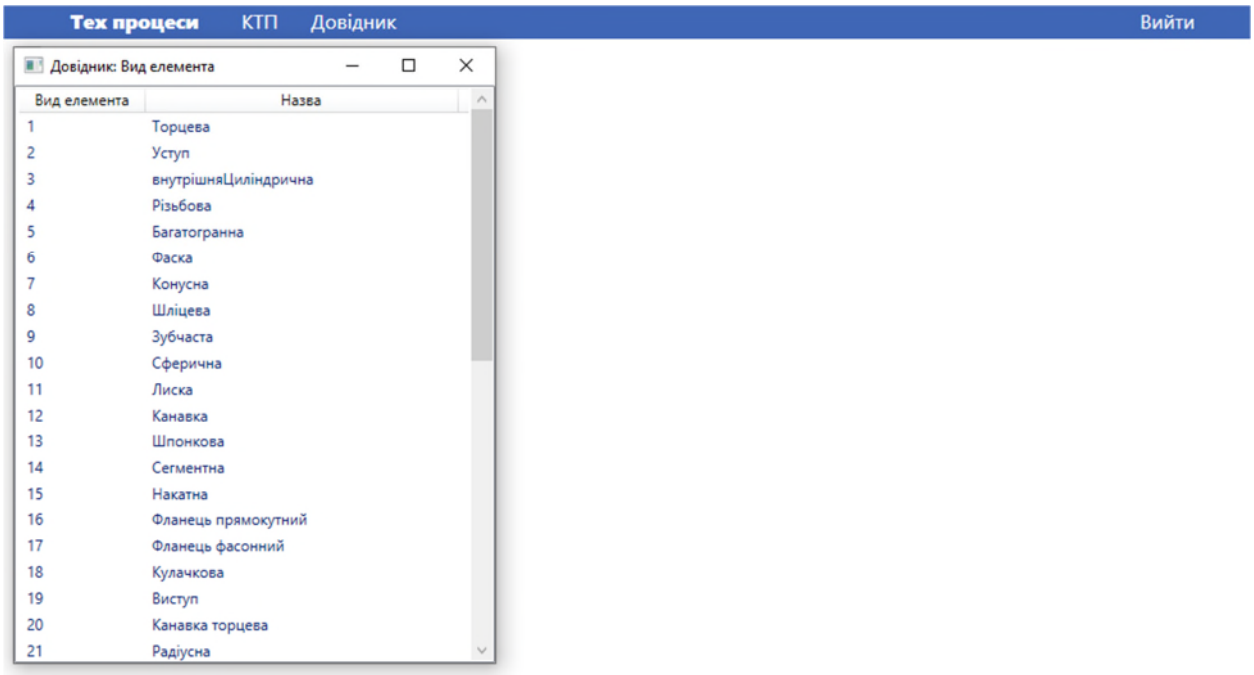


Рисунок 3.25 – Вікно «Вид елемента»

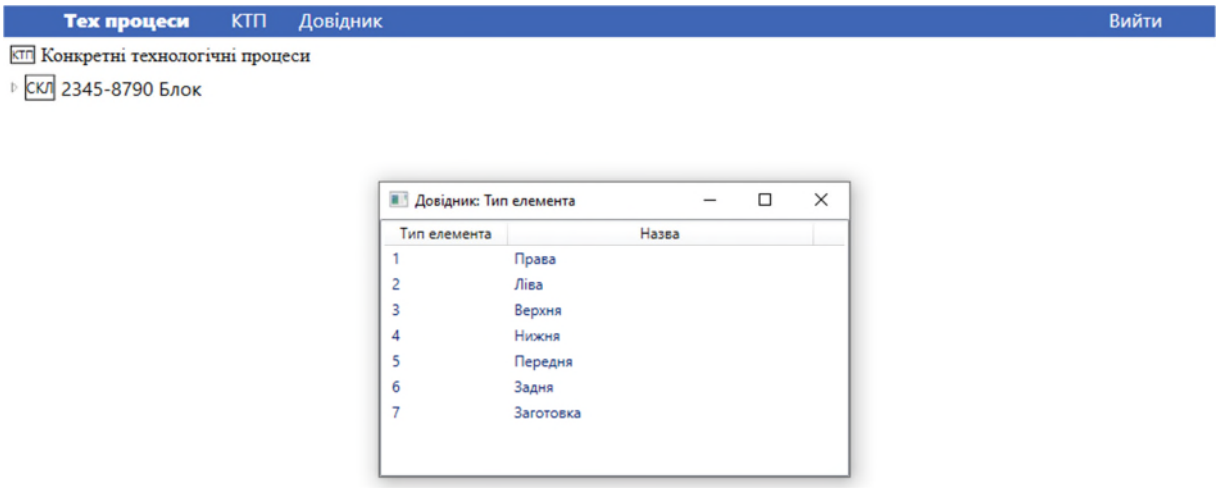


Рисунок 3.26 – Вікно «Тип елемента»

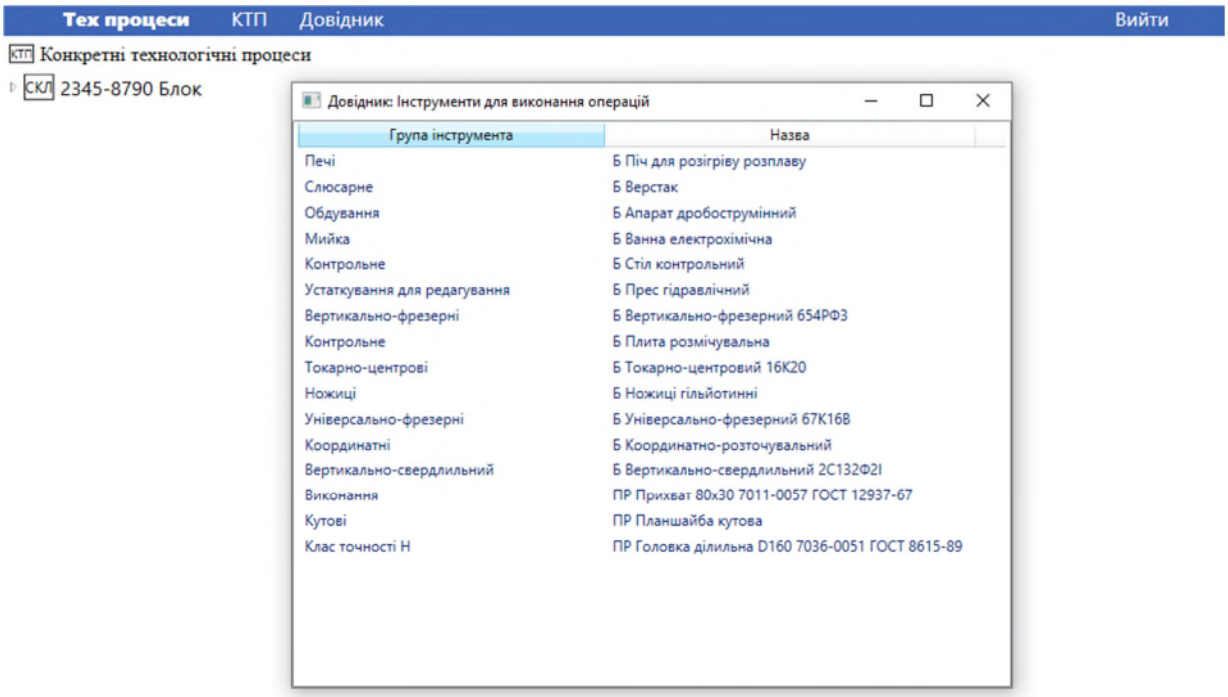


Рисунок 3.27 – Вікно «Інструменти для виконання операцій»

Для закриття застосунку необхідно натиснути кнопку «Вийти» на верхній панелі вікна, після чого застосунок закриється, так як і всі дочірні вікна, без різниці скільки їх відчинено.

3.4 Вимоги до розгортання системи

Для коректної та якісної роботи системи потрібно, щоб комп’ютер відповідав необхідним характеристикам:

– процесор – x86, 800GH;

– об’єм оперативної пам’яті – не менше, ніж 1024 Мб;

– мінімальний об’єм відео пам’яті – 512 Мб;

– операційна система – версії не нижчої за Windows 7;

– Microsoft Visual Studio 2015 або вище;

– дисплей з розширенням екрану – не менше, ніж 1280 x 720;

– периферія (клавіатура, миша).

Також для коректної роботи потрібна наявність Entity Framework.

3.5 Висновки до розділу 3

В третьому розділі розглянуто структуру модулів системи та їх взаємозв’язок, наведено діаграми класів для віповідних модулів. Також у третьому розділі розглянуто особливості реалізації програмних складових системи, підсумовуючи можна сказати те, що програмна реалізація інформаційної системи проектування технологічних процесів використовує мову програмування C# та платформу WPF. Шарова архітектура дозволяє розділити логіку на підсистеми, спрощуючи розробку і полегшуючи модифікацію. Використання патерну MVVM допомагає розподілити обов'язки між моделлю, представленням і ViewModel. Підхід Code First з використанням Entity Framework спрощує розробку бази даних, забезпечуючи автоматичне створення таблиць та спрощену взаємодію з базою даних. Це робить процес розробки ефективним і швидким, забезпечуючи гнучкість управління структурою бази даних. Описано про операції CRUD, що використовуються у програмній реалізації інформаційної системи, забезпечуючи повну функціональність системи керування даними.

Також детально описано функціональні можливості інформаційної системи, наведено скріншоти для наглядності майбутнього користувача, та описано мінімальні вимоги до розгортання системи, для її коректної роботи.

Висновки

Результатом виконання кваліфікаційної роботи бакалавра є створення інформаційної системи проектування технологічних процесів виготовлення деталей машин з використанням CASE-технології.

Використання CASE-технологій (Computer-Aided Software Engineering) можуть забезпечити автоматизацію розробки інформаційних систем, поліпшити їх якість та прискорити процес розробки. Разом з CASE-технологіями, UML-моделювання дозволяє нам візуалізувати та оптимізувати роботу інформаційної системи, покращує комунікацію між учасниками проекту та сприяє зниженню витрат на розробку та підтримку системи.

Подальше використання підходу Code First та фреймворку Entity Framework допомагає спростити розробку програмного забезпечення, зменшуючи кількість потрібного коду, але вимагає розуміння роботи з базами даних та міграцій. Також варто зазначити, що шарова архітектура, патерн MVVM та використання мови програмування C# та платформи WPF сприяють розділенню логіки на підсистеми, полегшуючи розробку та модифікацію.

Враховуючи функціональні можливості та мінімальні вимоги до розгортання системи, зроблено висновок, що застосування методів та інструментів комп'ютерних наук та інформаційних технологій в розробці програмного продукту сприяє покращенню якості продукту, прискорює розробку та забезпечує ефективне управління процесом розробки.

В ході виконання кваліфікаційної роботи бакалавра поставлено завдання реалізувати виконання наступних задач:

– проаналізувати задачу проектування технологічних процесів виготовлення деталей машин;

– розробити функціональну структуру інформаційної системи проектування технологічних процесів з використанням CASE-технології;

– розробити структуру інформаційної системи проектування технологічних процесів з використанням CASE-технології;

– розробити програмну реалізацію інформаційної системи проектування технологічних процесів виготовлення деталей машин.

В майбутньому розроблена система може використовуватись для проектування технологічних процесів виготовлення деталей машин.

Аналізуючи отриманий результат можна зробити висновок, що програмна реалізація інформаційної системи працює коректно та надає можливість для виконання функцій визначених в технічному завданні кваліфікаційної роботи бакалавра.

Перелік посилань

1. Вікіпедія. Виробничий процес. URL: https://uk.wikipedia.org/wiki/Виробничий_процес

2. Вікіпедія. Технологічний процес. URL: https://uk.wikipedia.org/wiki/Технологічний_процес

3. Вікіпедія. Технологічна операція. URL: https://uk.wikipedia.org/wiki/Технологічна_операція

4. Buklib. Види технологічних процесів. URL: https://buklib.net/books/36251/

5. Вікіпедія. Автоматизація технологічних процесів. URL: https://uk.wikipedia.org/wiki/Автоматизація_технологічних_процесів

6. Studfile. Технологічний процес у машинобудуванні, його складові частини і структура. URL: https://studfile.net/preview/5398871/page:3/

7. Вікіпедія. CASE. URL: https://uk.wikipedia.org/wiki/CASE

8. Pidru4niki. CASE-технології та CASE-засоби проектування. URL: https://pidru4niki.com/10760623/informatika/case-tehnologiyi_case-zasobi_proektuvannya

9. Allref. Використання case-технологій при розробці програмного забезпечення та баз даних. URL: https://allref.com.ua/uk/skachaty/Vikoristannya_case-tehnologiiy_pri_rozrobci_programnogo_zabezpechennya_ta_baz_danih

10. Studfile. CASE-технології. URL: https://stud.com.ua/59738/informatika/case_tehnologiyi

11. Tehnopro. ТехноПро. URL: https://www.tehnopro.com/4-produkty/abouttexnopro/

12. Вікіпедія. UML. URL: https://uk.wikipedia.org/wiki/Unified_Modeling_Language

13. Naurok. UML-діаграми. URL: https://naurok.com.ua/prezentaciya-diagrami-uml-diagrami-precedentiv-238715.html

14. Вікіпедія. CRUD. URL: https://uk.wikipedia.org/wiki/CRUD

15. Elib. Інфологічна модель даних. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/BD_2016_3/page5.html

16. Itstep. UI/UX. URL: https://te.itstep.org/blog/ui-and-ux-design

17. Baeldung. Layered architecture. URL: https://www.baeldung.com/cs/layered-architecture

18. Вікіпедія. MVVM. URL: https://uk.wikipedia.org/wiki/Model-View-ViewModel

19. Вікіпедія. C#. URL: https://uk.wikipedia.org/wiki/C_Sharp

20. Вікіпедія. XAML. URL: https://uk.wikipedia.org/wiki/XAML

21. Вікіпедія. WPF. URL: https://uk.wikipedia.org/wiki/Windows_Presentation_Foundation

22. Microsoft. Code First. URL: https://learn.microsoft.com/en-us/ef/ef6/modeling/code-first/workflows/new-database

23. Вікіпедія. Entity Framework. URL: https://uk.wikipedia.org/wiki/Entity_Framework

24. Вікіпедія. ORM. URL: https://uk.wikipedia.org/wiki/Об%27єктно-реляційне_відображення

25. Entityframeworktutorial. What is Code-First?. URL: https://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx

ДОДАТКИ

Додаток А

Програмні коди

Лістинг файлу Assembly.cs

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

namespace DataModel.Models;

public class Assembly : ObservableObject
{
	public int Id { get; set; }
	public string Name { get; set; }
	public string Designation { get; set; }
	public string Material { get; set; }
	public string TblName
	{
		get
		{
			return $"{Designation} {Name}";
		}
	}
	private ItemsChangeObservableCollection<Detail> _details;
	public ItemsChangeObservableCollection<Detail> Details
	{
		get { return _details; }
		set { _details = value; OnPropertyChanged("Details"); }
	}
}
```

Лістинг файлу Detail.cs

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

namespace DataModel.Models
{
	public class Detail : ObservableObject
	{
		public int Id { get; set; }
		public int Assembly_Id { get; set; }
		public string Name { get; set; }
		public string Designation { get; set; }
		public string Material { get; set; }
		public string TblName
		{
			get
			{
				return $"{Designation} {Name}";
			}
		}
		private ItemsChangeObservableCollection<TechProcess> _techProcesses;
		public ItemsChangeObservableCollection<TechProcess> TechProcesses
```

```
		{
			get { return _techProcesses; }
			set { _techProcesses = value; OnPropertyChanged(); }
		}
	}
}
```

Лістинг файлу Operation.cs

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

namespace DataModel.Models
{
	public class Operation : ObservableObject
	{
		public int Id { get; set; }
		public int TechProcess_Id { get; set; }
		public string Parameter { get; set; }
		public string TblName
		{
			get
			{
				return $"{Parameter}";
			}
		}

		private ItemsChangeObservableCollection<Transition> _transitions;
		public ItemsChangeObservableCollection<Transition> Transitions
		{
			get { return _transitions; }
			set { _transitions = value; OnPropertyChanged("Transitions"); }
		}
	}
}
```

Лістинг файлу OperationOperationTool.cs

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

namespace DataModel.Models
{
    public class OperationOperationTool : ObservableObject
    {
        public int Id { get; set; }
        public int OperationId { get; set; }
        public int OperationToolId { get; set; }
        public Operation Operations { get; set; }
        public OperationTool OperationTools { get; set; }
    }
}
```

Лістинг файлу OperationTool.cs

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

```
namespace DataModel.Models
{
	public class OperationTool : ObservableObject
	{
		public int Id { get; set; }
		public string Group { get; set; }
		public string Name { get; set; }
		public string GroupName
		{
			get
			{
				return $"{Name} | {Group}";
			}
		}
	}
}
```

Лістинг файлу TechProcess.cs

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

namespace DataModel.Models
{
	public class TechProcess : ObservableObject
	{
		public int Id { get; set; }
		public int Detail_Id { get; set; }
		public int? TPViewId { get; set; }
		public TPView? TPViews { get; set; }
		public int? TPTypeId { get; set; }
		public TPType? TPTypes { get; set; }
		public int? Number { get; set; }
		public string Code
		{
			get
			{
				if (TPViewId != null && TPTypeId !=null && Number !=null)
				{
					if (TPViewId < 10)
					{ return $"0{TPViewId}0{TPTypeId}0{Number}"; }
					else { return $"{TPViewId}0{TPTypeId}0{Number}"; }
				}
				else { return ""; }
			}
		}
		public string? Name { get; set; }
		public string CodeName
		{
			get
			{
				return $"{Code} {Name}";
			}
		}

		public string? TblName { get; set; }

		private ItemsChangeObservableCollection<Operation> _operations;
		public ItemsChangeObservableCollection<Operation> Operations
		{
```

```
                get { return _operations; }
                set { _operations = value; OnPropertyChanged("Operations"); }
            }

            public TechProcess()
            {
                TblName = "Деталь";
            }
        }
}
```

Лістинг файлу TechProcessTransition.cs

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

namespace DataModel.Models
{
    public class TechProcessTransition : ObservableObject
    {
        public int Id { get; set; }
        public int? TechProcessId { get; set; }
        public TechProcess? TechProcesses { get; set; }
        public int TransitionId { get; set; }
        public Transition Transitions { get; set; }
    }
}
```

Лістинг файлу TPType.cs

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

namespace DataModel.Models
{
    public class TPType : ObservableObject
    {
        public int Id { get; set; }
        public string Name { get; set; }
    }
}
```

Лістинг файлу TPView.cs

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

namespace DataModel.Models
{
    public class TPView : ObservableObject
    {
        public int Id { get; set; }
        public string Name { get; set; }
    }
}
```

Лістинг файлу Transition.cs

```
using System;
```

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

namespace DataModel.Models
{
	public class Transition : ObservableObject
	{
		public int Id { get; set; }
		public int Operation_Id { get; set; }
		public string TransitionText { get; set; }
		public string TblName
		{
			get
			{
				return $"{TransitionText}";
			}
		}
	}
}
```

Лістинг файлу ApplicationContext.cs

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using DataModel.Models;

namespace DataModel
{
		public class ApplicationContext : DbContext
		{
			public DbSet<Assembly> DbAssemblies { get; set; }
			public DbSet<Detail> DbDetails { get; set; }
			public DbSet<TechProcess> DbTechProcesses { get; set; }
			public DbSet<TPView> DbTPViews { get; set; }
			public DbSet<TPType> DbTPTypes { get; set; }
			public DbSet<Operation> DbOperations { get; set; }
			public DbSet<Transition> DbTransitions { get; set; }
			public DbSet<OperationTool> DbOperationTools { get; set; }
			public DbSet<OperationOperationTool> DbOperationOperationTools { get;
set; }
			public DbSet<TechProcessTransition> DbTechProcessTransitions { get;
set; }

			protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
			{

	optionsBuilder.UseSqlServer(@"Server=(localdb)\MSSQLLocalDB;Database=dbTP;Trusted_
Connection=True;");
			}

		}
}
```

Лістинг файлу AssemblyViewModel.cs

```
using DataModel;
using DataModel.Helpers;
using DataModel.Models;
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;
using static Microsoft.EntityFrameworkCore.DbLoggerCategory;
using wpf_technological.Utility;

namespace wpf_technological.ViewModels
{
	public class AssemblyViewModel : ObservableObject
	{
		private DataModel.Models.Assembly _selectedAssembly;
		public DataModel.Models.Assembly SelectedAssembly
		{
			get => _selectedAssembly;
			set
			{
				_selectedAssembly = value;
				OnPropertyChanged();
			}
		}

		private DataModel.Models.Assembly _selectedAssemblyForChange;
		public DataModel.Models.Assembly SelectedAssemblyForChange
		{
			get => _selectedAssemblyForChange;
			set
			{
				_selectedAssemblyForChange = value;
				OnPropertyChanged();
			}
		}

		public ICommand UpdateCommand { get; set; }
		public ICommand UpdateItemCommand { get; set; }

		ApplicationContext context;

		public AssemblyViewModel()
		{
			context = new ApplicationContext();

			UpdateCommand = new ServiceCommand((x) => true, Update);
			UpdateItemCommand = new ServiceCommand((x) => true, UpdateItem);
		}

		private void Update(object obj)
		{
			if (SelectedAssembly != null)
			{
				SelectedAssemblyForChange = SelectedAssembly;
			}
		}

		private void UpdateItem(object obj)
		{
			if (SelectedAssemblyForChange != null)
			{
				var selectedAssembly = context.DbAssemblies.
					Where(x => x.Id ==
SelectedAssemblyForChange.Id).First();
```

```
                        if (selectedAssembly != null)
                        {
                                selectedAssembly.Designation =
SelectedAssemblyForChange.Designation;
                                selectedAssembly.Name = SelectedAssemblyForChange.Name;
                                selectedAssembly.Material =
SelectedAssemblyForChange.Material;

                                SelectedAssembly = selectedAssembly;

                                context.Update(selectedAssembly);
                                context.SaveChanges();

                                SpecificViewModel specificVM;
                                PerfGlobal objGlobal;

                                objGlobal = new PerfGlobal();
                                specificVM = objGlobal.SpecificVM;

                                specificVM.TreeViewUpdate();
                        }
                }
        }

        public void GetSelectedItem(object TreeViewSelectedItem)
        {
                var selected = TreeViewSelectedItem as DataModel.Models.Assembly;

                if (selected != null)
                {
                        SelectedAssembly = selected;
                }
        }
    }
}
```

Лістинг файлу DetailViewModel.cs

```
using DataModel;
using DataModel.Helpers;
using DataModel.Models;
using Microsoft.Office.Interop.Word;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Input;
using wpf_technological.Utility;
using wpf_technological.Views;
using File = System.IO.File;
using Path = System.IO.Path;
using Word = Microsoft.Office.Interop.Word;

namespace wpf_technological.ViewModels
{
    public class DetailViewModel : ObservableObject
    {
            private Detail _selectedDetail;
            public Detail SelectedDetail
            {
                    get => _selectedDetail;
                    set
```

```
                {
                        _selectedDetail = value;
                        OnPropertyChanged();
                }
        }

        private Detail _selectedDetailForChange;
        public Detail SelectedDetailForChange
        {
                get => _selectedDetailForChange;
                set
                {
                        _selectedDetailForChange = value;
                        OnPropertyChanged();
                }
        }

        public ICommand UpdateCommand { get; set; }
        public ICommand UpdateItemCommand { get; set; }

        ApplicationContext context;
        public DetailViewModel()
    {
                context = new ApplicationContext();

                UpdateCommand = new ServiceCommand((x) => true, Update);
                UpdateItemCommand = new ServiceCommand((x) => true, UpdateItem);
        }

        private void Update(object obj)
        {
                if (SelectedDetail != null)
                {
                        SelectedDetailForChange = SelectedDetail;
                }
        }

        private void UpdateItem(object obj)
        {
                if (SelectedDetailForChange != null)
                {
                        var selectedDetail = context.DbDetails.
                                Where(x => x.Id == SelectedDetailForChange.Id).First();

                        if (selectedDetail != null)
                        {
                                selectedDetail.Designation =
SelectedDetailForChange.Designation;
                                selectedDetail.Name = SelectedDetailForChange.Name;
                                selectedDetail.Material =
SelectedDetailForChange.Material;

                                SelectedDetail = selectedDetail;

                                context.Update(selectedDetail);
                                context.SaveChanges();

                                SpecificViewModel specificVM;
                                PerfGlobal objGlobal;

                                objGlobal = new PerfGlobal();
                                specificVM = objGlobal.SpecificVM;
```

```
                                specificVM.TreeViewUpdate();
                        }
                }
        }

        public void GetDetailReport()
        {
                var wordApp = new Word.Application();
                wordApp.Visible = false;

                try
                {
                        var wordDoc = wordApp.Documents.Add();

                        var tp = context.DbTechProcesses.Where(x => x.Detail_Id ==
SelectedDetail.Id).First();
                        var operations =
context.DbOperations.Where(x=>x.TechProcess_Id==tp.Id).ToList();

                        var table = wordDoc.Tables.Add(wordDoc.Range(),
operations.Count + 3, 4);

                        table.Borders.Enable = 1;
                        table.Borders.OutsideLineStyle = WdLineStyle.wdLineStyleSingle;
                        table.Borders.InsideLineStyle = WdLineStyle.wdLineStyleSingle;

                        foreach (Cell cell in table.Range.Cells)
                        {
                                cell.Range.ParagraphFormat.Alignment =
WdParagraphAlignment.wdAlignParagraphCenter;
                                cell.VerticalAlignment =
WdCellVerticalAlignment.wdCellAlignVerticalCenter;
                                cell.Range.Font.Size = 12;
                        }

                        foreach (Row row in table.Rows)
                        {
                                row.Height = 35;
                        }

                        table.Cell(1, 1).Merge(table.Cell(1, 4));
                        table.Cell(1, 1).Range.Text = "ЗВІТ ПРО ФОРМУВАННЯ ДЕТАЛІ";
                        table.Cell(1, 1).Range.Font.Size = 20;
                        table.Cell(1, 1).Range.Font.Bold = 1;
                        table.Cell(1, 1).Height = 50;

                        table.Cell(2, 1).Range.Text = "Назва деталі";
                        table.Cell(2, 1).Range.Font.Bold = 1;
                        table.Cell(2, 2).Range.Text = SelectedDetail.TblName;
                        table.Cell(2, 3).Range.Text = "Матеріал деталі";
                        table.Cell(2, 3).Range.Font.Bold = 1;
                        table.Cell(2, 4).Range.Text = SelectedDetail.Material;

                        table.Cell(3, 1).Merge(table.Cell(3, 4));
                        table.Cell(3, 1).Range.Text = "ОПЕРАЦІЇ ДЛЯ СТВОРЕННЯ ДЕТАЛІ";
                        table.Cell(3, 1).Range.Font.Size = 20;
                        table.Cell(3, 1).Range.Font.Bold = 1;
                        table.Cell(3, 1).Height = 50;

                        for (int i = 0; i < operations.Count; i++)
                        {
                                table.Cell(i + 4, 1).Range.Text = "Операція №" + (i+1);
                                table.Cell(i + 4, 1).Range.Font.Bold = 1;
```

```
                                table.Cell(i + 4, 2).Range.Text = operations[i].TblName;
                                table.Cell(i + 4, 3).Range.Text = "Інструменти, що були
використані для виконання операції";
                                table.Cell(i + 4, 3).Range.Font.Bold = 1;

                                var oots = context.DbOperationOperationTools.
                                      Where(x => x.OperationId ==
operations[i].Id).ToList();

                                var list_ot = new List<string>();

                                for (int j = 0; j < oots.Count; j++)
                                {
                                      var ot = context.DbOperationTools.
                                            Where(x => x.Id ==
oots[j].OperationToolId).First();
                                      list_ot.Add(ot.Name);
                                }

                                string list_string_ot = string.Join(", ", list_ot);
                                table.Cell(i + 4, 4).Range.Text = list_string_ot;
                          }

                          string fileName = $"Звіт - " + SelectedDetail.TblName +
".docx";
                          string filePath = GetUniqueFilePath(fileName);

                          wordDoc.SaveAs2(filePath);
                          wordApp.Visible = true;
                    }
                    catch
                    {
                          MessageBox.Show("EXCEPTION ERROR");
                    }
              }

              static string GetUniqueFilePath(string fileName)
              {
                    string directory = "D:\\visual my
files\\wpf_technological\\SystemReports\\";
                    string filePath = Path.Combine(directory, fileName);

                    int counter = 1;
                    while (File.Exists(filePath))
                    {
                          string newFileName = Path.GetFileNameWithoutExtension(fileName)
+ counter.ToString() + Path.GetExtension(fileName);
                          filePath = Path.Combine(directory, newFileName);
                          counter++;
                    }

                    return filePath;
              }

              public void GetSelectedItem(object TreeViewSelectedItem)
              {
                    var selected = TreeViewSelectedItem as Detail;

                    if (selected != null)
                    {
                          SelectedDetail = selected;
                    }
              }
```

```
	}
}
```

Лістинг файлу OperationViewModel.cs

```
using DataModel;
using DataModel.Helpers;
using DataModel.Models;
using System;
using Microsoft.Office.Interop.Word;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Documents;
using System.Windows.Input;
using wpf_technological.Views;
using System.Windows;
using wpf_technological.Utility;
using System.Windows.Controls;
using File = System.IO.File;
using Path = System.IO.Path;
using Word = Microsoft.Office.Interop.Word;

namespace wpf_technological.ViewModels
{
	public class OperationViewModel : ObservableObject
	{
		public ObservableCollection<OperationTool> ComboList { get; set; }

		private ObservableCollection<OperationTool> _listItems;

		public ObservableCollection<OperationTool> ListItems
		{
			get { return _listItems; }
			set { _listItems = value; OnPropertyChanged(); }
		}

		private Operation _selectedOperation;
		public Operation SelectedOperation
		{
			get => _selectedOperation;
			set
			{
				_selectedOperation = value;
				OnPropertyChanged();
			}
		}

		private Operation _selectedOperationForChange;
		public Operation SelectedOperationForChange
		{
			get => _selectedOperationForChange;
			set
			{
				_selectedOperationForChange = value;
				OnPropertyChanged();
			}
		}

		private OperationTool _selectedOperationTool;
		public OperationTool SelectedOperationTool
		{
			get { return _selectedOperationTool; }
```

```
                set { _selectedOperationTool = value; OnPropertyChanged(); }
            }

            private OperationTool _selectedOperationToolForChange;
            public OperationTool SelectedOperationToolForChange
            {
                get { return _selectedOperationToolForChange; }
                set { _selectedOperationToolForChange = value; OnPropertyChanged(); }
            }

            private OperationOperationTool _selectedOperationToolForDelete;
            public OperationOperationTool SelectedOperationToolForDelete
            {
                get { return _selectedOperationToolForDelete; }
                set { _selectedOperationToolForDelete = value; OnPropertyChanged(); }
            }

            public ICommand AddItemCommand { get; set; }
            public ICommand UpdateCommand { get; set; }
            public ICommand UpdateItemCommand { get; set; }
            public ICommand UpdateOperationCommand { get; set; }
            public ICommand UpdateOperationItemCommand { get; set; }
            public ICommand DeleteCommand { get; set; }

            ApplicationContext context;

            public OperationViewModel()
            {
                context = new ApplicationContext();

                ComboList = new
ObservableCollection<OperationTool>(context.DbOperationTools);

                AddItemCommand = new ServiceCommand((x) => true, AddItem);
                UpdateCommand = new ServiceCommand((x) => true, Update);
                UpdateItemCommand = new ServiceCommand((x) => true, UpdateItem);
                UpdateOperationCommand = new ServiceCommand((x) => true,
UpdateOperation);
                UpdateOperationItemCommand = new ServiceCommand((x) => true,
UpdateOperationItem);
                DeleteCommand = new ServiceCommand((x) => true, Delete);
            }

            private void LoadItems()
            {
                ListItems = new
ObservableCollection<OperationTool>(context.DbOperationOperationTools.
                            Where(x => x.OperationId ==
SelectedOperation.Id).Select(x => x.OperationTools));
            }

            private void AddItem(object obj)
            {
                if (SelectedOperation != null && SelectedOperationToolForChange !=
null)
                {
                    var newOOTSelectedOperation = context.DbOperations.
                        Where(x => x.Id == SelectedOperation.Id).First();

                    var newOOTSelectedOperationTool = context.DbOperationTools.
                        Where(x => x.Id ==
SelectedOperationToolForChange.Id).First();
```

```
                        var newOOT = new OperationOperationTool
                        {
                                OperationId = newOOTSelectedOperation.Id,
                                Operations = newOOTSelectedOperation,
                                OperationToolId = newOOTSelectedOperationTool.Id,
                                OperationTools = newOOTSelectedOperationTool
                        };

                        bool isThisOperationExist = false;
                        bool isThisOperationToolExist = false;

                        foreach (int operationId in context.DbOperationOperationTools.
                                Select(x=>x.OperationId))
                        {
                                if (newOOT.OperationId == operationId)
                                {
                                        isThisOperationExist = true;
                                }
                                else
                                {
                                        isThisOperationExist = false;
                                }
                        }

                        if (isThisOperationExist == true)
                        {
                                foreach (int toolId in
context.DbOperationOperationTools.
                                                        Where(x => x.OperationId ==
newOOT.OperationId).
                                                        Select(x => x.OperationToolId))
                                {
                                        if (newOOT.OperationToolId == toolId)
                                        {
                                                isThisOperationToolExist = true;
                                        }
                                        else
                                        {
                                                isThisOperationToolExist = false;
                                        }
                                }
                        }

                        if (isThisOperationExist == true && isThisOperationToolExist ==
true)
                        {
                                MessageBox.Show("Оберіть інший інструмент, цей вже
додано");
                        }
                        else
                        {
                                context.DbOperationOperationTools.Add(newOOT);
                                context.SaveChanges();
                        }
                }
                else
                {
                        MessageBox.Show("Оберіть інструмент");
                }

                LoadItems();
        }
```

```
		private void Update(object obj)
		{
			SelectedOperationTool = obj as OperationTool;
		}

		private void UpdateItem(object obj)
		{
			if (SelectedOperation != null && SelectedOperationTool != null)
			{
				var selectedOOT = context.DbOperationOperationTools.
					Where(x=>x.OperationId==SelectedOperation.Id).
					Where(x => x.OperationToolId ==
SelectedOperationTool.Id).First();

				if (selectedOOT != null && SelectedOperationToolForChange !=
null)
				{
					bool isThisOperationToolExist = false;

					if (SelectedOperationToolForChange.Id ==
selectedOOT.OperationToolId)
					{
						isThisOperationToolExist = true;
					}
					else
					{
						isThisOperationToolExist = false;
					}

					if (isThisOperationToolExist == true)
					{
						MessageBox.Show("Оберіть інший інструмент, цей вже
додано");
					}
					else
					{
						selectedOOT.OperationToolId =
SelectedOperationToolForChange.Id;
						selectedOOT.OperationTools =
SelectedOperationToolForChange;
						context.Update(selectedOOT);
						context.SaveChanges();
					}
				}
			}

			LoadItems();
		}
		private void UpdateOperation(object obj)
		{
			if (SelectedOperation != null)
			{
				SelectedOperationForChange = SelectedOperation;
			}
		}

		private void UpdateOperationItem(object obj)
		{
			if (SelectedOperationForChange !=null)
			{
				var selectedOperation = context.DbOperations.
					Where(x => x.Id ==
SelectedOperationForChange.Id).First();
```

```
                        if (selectedOperation != null)
                        {
                                selectedOperation.Parameter =
SelectedOperationForChange.Parameter;

                                SelectedOperation = selectedOperation;

                                context.Update(selectedOperation);
                                context.SaveChanges();

                                SpecificViewModel specificVM;
                                PerfGlobal objGlobal;

                                objGlobal = new PerfGlobal();
                                specificVM = objGlobal.SpecificVM;

                                specificVM.TreeViewUpdate();
                        }
                }
        }

        private void Delete(object obj)
        {
                SelectedOperationTool = obj as OperationTool;

                if (SelectedOperationTool != null)
                {
                        SelectedOperationToolForDelete =
context.DbOperationOperationTools.
                                Where(x => x.OperationTools ==
SelectedOperationTool).First();

      context.DbOperationOperationTools.Remove(SelectedOperationToolForDelete);
                        context.SaveChanges();
                        ListItems.Remove(SelectedOperationTool);
                }
        }

        public void GetOperationReport()
        {
                var wordApp = new Word.Application();
                wordApp.Visible = false;

                try
                {
                        var wordDoc = wordApp.Documents.Add();

                        var oots = context.DbOperationOperationTools.
                                Where(x => x.OperationId ==
SelectedOperation.Id).ToList();

                        var list_ot = new List<string>();

                        for (int i = 0; i < oots.Count; i++)
                        {
                                var ot = context.DbOperationTools.
                                        Where(x => x.Id ==
oots[i].OperationToolId).First();
                                list_ot.Add(ot.Name);
                        }

                        string list_string_ot = string.Join(", ", list_ot);
```

```
                        var transitions = context.DbTransitions.
                              Where(x => x.Operation_Id ==
SelectedOperation.Id).ToList();

                        var table = wordDoc.Tables.Add(wordDoc.Range(),
transitions.Count + 3, 4);

                        table.Borders.Enable = 1;
                        table.Borders.OutsideLineStyle = WdLineStyle.wdLineStyleSingle;
                        table.Borders.InsideLineStyle = WdLineStyle.wdLineStyleSingle;

                        foreach (Cell cell in table.Range.Cells)
                        {
                              cell.Range.ParagraphFormat.Alignment =
WdParagraphAlignment.wdAlignParagraphCenter;
                              cell.VerticalAlignment =
WdCellVerticalAlignment.wdCellAlignVerticalCenter;
                              cell.Range.Font.Size = 12;
                        }

                        foreach (Row row in table.Rows)
                        {
                              row.Height = 35;
                        }

                        table.Cell(1, 1).Merge(table.Cell(1, 4));
                        table.Cell(1, 1).Range.Text = "ЗВІТ ПРО ФОРМУВАННЯ ОПЕРАЦІЇ";
                        table.Cell(1, 1).Range.Font.Size = 20;
                        table.Cell(1, 1).Range.Font.Bold = 1;
                        table.Cell(1, 1).Height = 50;

                        table.Cell(2, 1).Range.Text = "Назва операції";
                        table.Cell(2, 1).Range.Font.Bold = 1;
                        table.Cell(2, 2).Range.Text = SelectedOperation.TblName;
                        table.Cell(2, 3).Range.Text = "Інструменти, що були використані
для виконання операції";
                        table.Cell(2, 3).Range.Font.Bold = 1;
                        table.Cell(2, 4).Range.Text = list_string_ot;

                        table.Cell(3, 1).Merge(table.Cell(3, 4));
                        table.Cell(3, 1).Range.Text = "ПЕРЕХОДИ ДЛЯ ВИКОНАННЯ
ОПЕРАЦІЇ";
                        table.Cell(3, 1).Range.Font.Size = 20;
                        table.Cell(3, 1).Range.Font.Bold = 1;
                        table.Cell(3, 1).Height = 50;

                        for (int i = 0; i < transitions.Count; i++)
                        {
                              table.Cell(i + 4, 1).Range.Text = "Перехід №" + (i + 1);
                              table.Cell(i + 4, 1).Range.Font.Bold = 1;
                              table.Cell(i + 4, 2).Range.Text =
transitions[i].TblName;
                              table.Cell(i + 4, 3).Range.Text = "Перехід виконується
по наступному тех процесу";
                              table.Cell(i + 4, 3).Range.Font.Bold = 1;

                              var tptrans = context.DbTechProcessTransitions.
                                    Where(x => x.TransitionId ==
transitions[i].Id).First();

                              var tp = context.DbTechProcesses.Where(x => x.Id ==
tptrans.TechProcessId).FirstOrDefault();
```

```
                                if (tp != null)
                                {
                                        table.Cell(i + 4, 4).Range.Text = tp.Name + "\n" +
tp.Code;
                                }
                                else
                                {
                                        table.Cell(i + 4, 4).Range.Text = string.Empty;
                                }
                        }

                        string fileName = $"Звіт - " + SelectedOperation.TblName +
".docx";
                        string filePath = GetUniqueFilePath(fileName);

                        wordDoc.SaveAs2(filePath);
                        wordApp.Visible = true;
                }
                catch
                {
                        MessageBox.Show("EXCEPTION ERROR");
                }
        }

        static string GetUniqueFilePath(string fileName)
        {
                string directory = "D:\\visual my
files\\wpf_technological\\SystemReports\\";
                string filePath = Path.Combine(directory, fileName);

                int counter = 1;
                while (File.Exists(filePath))
                {
                        string newFileName = Path.GetFileNameWithoutExtension(fileName)
+ counter.ToString() + Path.GetExtension(fileName);
                        filePath = Path.Combine(directory, newFileName);
                        counter++;
                }

                return filePath;
        }

        public void GetSelectedItem(object TreeViewSelectedItem)
        {
                var selected = TreeViewSelectedItem as Operation;
                if (selected != null)
                {
                        SelectedOperation = selected;

                        LoadItems();
                }
        }
    }
}
```

Лістинг файлу SpecificViewModel.cs

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

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using DataModel;
using DataModel.Helpers;
using DataModel.Models;
using wpf_technological.Utility;
using wpf_technological.Views;

namespace wpf_technological.ViewModels
{
	public class SpecificViewModel : ObservableObject
	{
		public Transition TransitionModel { get; set; }
		public Operation OperationModel { get; set; }

		private ItemsChangeObservableCollection<Operation> _operations;
		public ItemsChangeObservableCollection<Operation> Operations
		{
			get { return _operations; }
			set { _operations = value; OnPropertyChanged(); }
		}
		public TechProcess TechProcessModel { get; set; }

		private ItemsChangeObservableCollection<TechProcess> _techProcesses;
		public ItemsChangeObservableCollection<TechProcess> TechProcesses
		{
			get { return _techProcesses; }
			set { _techProcesses = value; OnPropertyChanged(); }
		}
		public Detail DetailModel { get; set; }

		private ItemsChangeObservableCollection<Detail> _details;
		public ItemsChangeObservableCollection<Detail> Details
		{
			get { return _details; }
			set { _details = value; OnPropertyChanged(); }
		}

		public DataModel.Models.Assembly AssemblyModel { get; set; }

		private ItemsChangeObservableCollection<DataModel.Models.Assembly>
_assemblies;
		public ItemsChangeObservableCollection<DataModel.Models.Assembly> Assemblies
		{
			get { return _assemblies; }
			set { _assemblies = value; OnPropertyChanged(); }
		}

		#region NewDetail Properties

		private Detail _detailFromTV;
		public Detail DetailFromTV
		{
			get => _detailFromTV;
			set
			{
				_detailFromTV = value;
				OnPropertyChanged();
			}
		}

		public string DetailDesignation
```

```
{
	get
	{
		return DetailModel.Designation;
	}
	set
	{
		DetailModel.Designation = value;
		OnPropertyChanged();
	}
}

public string DetailName
{
	get
	{
		return DetailModel.Name;
	}
	set
	{
		DetailModel.Name = value;
		OnPropertyChanged();
	}
}

public string DetailMaterial
{
	get
	{
		return DetailModel.Material;
	}
	set
	{
		DetailModel.Material = value;
		OnPropertyChanged();
	}
}
#endregion

#region NewOperation Properties

private Operation _operationFromTV;
public Operation OperationFromTV
{
	get => _operationFromTV;
	set
	{
		_operationFromTV = value;
		OnPropertyChanged();
	}
}

public string OperationParameter
{
	get
	{
		return OperationModel.Parameter;
	}
	set
	{
		OperationModel.Parameter = value;
		OnPropertyChanged();
	}
```

```
        }

        #endregion

        #region NewTransition Properties

        public ObservableCollection<TechProcess> ComboTP { get; set; }

        private Transition _transitionFromTV;
        public Transition TransitionFromTV
        {
            get => _transitionFromTV;
            set
            {
                _transitionFromTV = value;
                OnPropertyChanged();
            }
        }

        public string TransitionText
        {
            get
            {
                return TransitionModel.TransitionText;
            }
            set
            {
                TransitionModel.TransitionText = value;
                OnPropertyChanged();
            }
        }

        private TechProcess _selectedTP;
        public TechProcess SelectedTP
        {
            get => _selectedTP;
            set
            {
                _selectedTP = value;
                OnPropertyChanged();
            }
        }

        #endregion

        public ICommand AddDetailCommand { get; set; }
        public ICommand AddOperationCommand { get; set; }
        public ICommand AddTransitionCommand { get; set; }

        ApplicationContext context;

        public SpecificViewModel()
        {
            context = new ApplicationContext();

            TransitionModel = new Transition();
            OperationModel = new Operation();
            TechProcessModel = new TechProcess();
            DetailModel = new Detail();
            AssemblyModel = new DataModel.Models.Assembly();

            _assemblies = new
ItemsChangeObservableCollection<DataModel.Models.Assembly>();
```

```
                var LoadData = new LoadDataToDb();

                ComboTP = new
ObservableCollection<TechProcess>(context.DbTechProcesses);

                AddDetailCommand = new ServiceCommand((x) => true, AddDetail);
                AddOperationCommand = new ServiceCommand((x) => true, AddOperation);
                AddTransitionCommand = new ServiceCommand((x) => true,
AddTransition);

                LoadDataFromDb();
            }

            private void AddDetail(object obj)
            {
                if (DetailDesignation != null && DetailName != null &&
                    DetailMaterial != null && DetailFromTV != null)
                {
                    var assembly = context.DbAssemblies.
                                        Where(x => x.Id ==
DetailFromTV.Assembly_Id).First();

                    var new_detail = new Detail
                    {
                        Assembly_Id = DetailFromTV.Assembly_Id,
                        Designation = DetailDesignation,
                        Name = DetailName,
                        Material = DetailMaterial,
                        TechProcesses = new
ItemsChangeObservableCollection<TechProcess>()
                    };

                    context.DbDetails.Add(new_detail);
                    assembly.Details.Add(new_detail);
                    context.SaveChanges();

                    var new_tp = new TechProcess
                    {
                        Detail_Id = new_detail.Id,
                        Operations = new
ItemsChangeObservableCollection<Operation>()
                    };

                    context.DbTechProcesses.Add(new_tp);
                    new_detail.TechProcesses.Add(new_tp);
                    context.SaveChanges();
                }
            }

            public void DeleteDetail(object TreeViewSelectedItem)
            {
                var selected = TreeViewSelectedItem as Detail;

                if (selected != null)
                {
                    DetailFromTV = selected;

                    var assembly = context.DbAssemblies.
                        Where(x => x.Id == DetailFromTV.Assembly_Id).First();

                    var detail = context.DbDetails.
                        Where(x=>x.Id== DetailFromTV.Id).First();
```

```
                        if (assembly != null && detail != null)
                        {
                              bool isThisDetailGotTP = false;

                              foreach (int detailId in context.DbTechProcesses.
                                    Select(x => x.Detail_Id))
                              {
                                    if (detail.Id == detailId)
                                    {
                                          isThisDetailGotTP = true;
                                    }
                                    else
                                    {
                                          isThisDetailGotTP = false;
                                    }
                              }

                              if (isThisDetailGotTP == true)
                              {
                                    var tp = context.DbTechProcesses.
                                          Where(x => x.Detail_Id ==
DetailFromTV.Id).First();

                                    if (tp != null)
                                    {
                                          detail.TechProcesses.Remove(tp);
                                          context.DbTechProcesses.Remove(tp);
                                    }
                              }
                        }

                        assembly.Details.Remove(detail);
                        context.DbDetails.Remove(detail);
                        context.Update(assembly);
                        context.SaveChanges();
                  }
            }

            private void AddOperation(object obj)
            {
                  if (OperationFromTV != null && OperationParameter != null)
                  {
                        var tp = context.DbTechProcesses.
                              Where(x => x.Id ==
OperationFromTV.TechProcess_Id).First();

                        var new_operation = new Operation
                        {
                              TechProcess_Id = OperationFromTV.TechProcess_Id,
                              Parameter = OperationParameter,
                              Transitions = new
ItemsChangeObservableCollection<Transition>()
                        };

                        context.DbOperations.Add(new_operation);
                        tp.Operations.Add(new_operation);
                        context.SaveChanges();
                  }
            }

            public void DeleteOperation(object TreeViewSelectedItem)
            {
                  var selected = TreeViewSelectedItem as Operation;
```

```
                    if (selected != null)
                    {
                          OperationFromTV = selected;

                          var tp = context.DbTechProcesses.
                                Where(x => x.Id ==
OperationFromTV.TechProcess_Id).First();

                          var operation = context.DbOperations.
                                Where(x => x.Id == OperationFromTV.Id).First();

                          if (tp != null && operation != null)
                          {
                                bool isThisOperationGotTools = false;

                                foreach (int operationId in
context.DbOperationOperationTools.
                                      Select(x => x.OperationId))
                                {
                                      if (operation.Id == operationId)
                                      {
                                            isThisOperationGotTools = true;
                                      }
                                      else
                                      {
                                            isThisOperationGotTools = false;
                                      }
                                }

                                if (isThisOperationGotTools == true)
                                {
                                      var oot = context.DbOperationOperationTools.
                                            Where(x => x.OperationId ==
OperationFromTV.Id).First();

                                      if (oot != null)
                                      {

      context.DbOperationOperationTools.Remove(oot);
                                      }
                                }

                                tp.Operations.Remove(operation);
                                context.DbOperations.Remove(operation);
                                context.Update(tp);
                                context.SaveChanges();
                          }
                    }
              }

              private void AddTransition(object obj)
              {
                    if (TransitionFromTV != null && TransitionText != null)
                    {
                          var operation = context.DbOperations.
                                Where(x => x.Id ==
TransitionFromTV.Operation_Id).First();

                          var new_transition = new Transition
                          {
                                Operation_Id = TransitionFromTV.Operation_Id,
                                TransitionText = TransitionText
```

```
                        };

                        if (SelectedTP != null)
                        {
                              var new_TPTransition = new TechProcessTransition
                              {
                                    TechProcessId = SelectedTP.Id,
                                    TechProcesses = SelectedTP,
                                    TransitionId = new_transition.Id,
                                    Transitions = new_transition
                              };

                              context.DbTechProcessTransitions.Add(new_TPTransition);
                        }
                        else
                        {
                              var new_TPTransition = new TechProcessTransition
                              {
                                    TransitionId = new_transition.Id,
                                    Transitions = new_transition
                              };

                              context.DbTechProcessTransitions.Add(new_TPTransition);
                        }


                        context.DbTransitions.Add(new_transition);
                        operation.Transitions.Add(new_transition);
                        context.SaveChanges();
                  }
            }

            public void DeleteTransition(object TreeViewSelectedItem)
            {
                  var selected = TreeViewSelectedItem as Transition;

                  if (selected != null)
                  {
                        TransitionFromTV = selected;

                        var operation = context.DbOperations.
                              Where(x=>x.Id==TransitionFromTV.Operation_Id).First();

                        var transition = context.DbTransitions.
                              Where(x => x.Id == TransitionFromTV.Id).First();

                        if (operation != null && transition != null)
                        {
                              bool isThisTransitionGotTP = false;

                              foreach (int transitionId in
context.DbTechProcessTransitions.
                                    Select(x => x.TransitionId))
                              {
                                    if (transition.Id == transitionId)
                                    {
                                          isThisTransitionGotTP = true;
                                    }
                                    else
                                    {
                                          isThisTransitionGotTP = false;
                                    }
                              }
```

```
                                if (isThisTransitionGotTP == true)
                                {
                                      var TPTrans = context.DbTechProcessTransitions.
                                            Where(x => x.TransitionId ==
TransitionFromTV.Id).First();

                                      if (TPTrans != null)
                                      {

      context.DbTechProcessTransitions.Remove(TPTrans);
                                      }
                                }
                          }

                          operation.Transitions.Remove(transition);
                          context.DbTransitions.Remove(transition);
                          context.Update(operation);
                          context.SaveChanges();
                    }
              }

              private void LoadDataFromDb()
              {
                    var transitions = context.DbTransitions.ToList();
                    var operations = context.DbOperations.ToList();
                    var techprocesses = context.DbTechProcesses.ToList();
                    var details = context.DbDetails.ToList();
                    var assemblies = context.DbAssemblies.ToList();

                    foreach (var assembly in assemblies)
                    {
                          Assemblies.Add(assembly);
                    }
              }

              public void TreeViewUpdate()
              {
                    Assemblies.Clear();

                    LoadDataFromDb();
              }

              public void GetSelectedItem(object TreeViewSelectedItem)
              {
                    if (TreeViewSelectedItem is Detail)
                    {
                          var selected = TreeViewSelectedItem as Detail;

                          if (selected != null)
                          {
                                DetailFromTV = selected;
                          }
                    }
                    else if (TreeViewSelectedItem is Operation)
                    {
                          var selected = TreeViewSelectedItem as Operation;

                          if (selected != null)
                          {
                                OperationFromTV = selected;
                          }
                    }
```

```
                else if (TreeViewSelectedItem is Transition)
                {
                        var selected = TreeViewSelectedItem as Transition;

                        if (selected != null)
                        {
                                TransitionFromTV = selected;
                        }
                }
        }
    }
}
```

Лістинг файлу TechProcessViewModel.cs

```
using DataModel;
using DataModel.Helpers;
using DataModel.Models;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;
using static Microsoft.EntityFrameworkCore.DbLoggerCategory;
using wpf_technological.Utility;
using System.Xml.Linq;
using System.Windows;

namespace wpf_technological.ViewModels
{
        public class TechProcessViewModel : ObservableObject
        {
                public ObservableCollection<TPView> ComboView { get; set; }
                public ObservableCollection<TPType> ComboType { get; set; }
                public ObservableCollection<Detail> ComboDetail { get; set; }
                public List<string> TPCodes { get; set; }

                private ObservableCollection<TechProcess> _listItems;
                public ObservableCollection<TechProcess> ListItems
                {
                        get { return _listItems; }
                        set { _listItems = value; OnPropertyChanged(); }
                }

                private TechProcess _selectedTechProcessMain;
                public TechProcess SelectedTechProcessMain
                {
                        get => _selectedTechProcessMain;
                        set
                        {
                                _selectedTechProcessMain = value;
                                OnPropertyChanged();
                        }
                }

                private TPView _selectedViewMain;
                public TPView SelectedViewMain
                {
                        get => _selectedViewMain;
                        set
                        {
                                _selectedViewMain = value;
                                OnPropertyChanged();
                        }
                }
```

```
private TPType _selectedTypeMain;
public TPType SelectedTypeMain
{
	get => _selectedTypeMain;
	set
	{
		_selectedTypeMain = value;
		OnPropertyChanged();
	}
}

private TechProcess _selectedTechProcess;
public TechProcess SelectedTechProcess
{
	get => _selectedTechProcess;
	set
	{
		_selectedTechProcess = value;
		OnPropertyChanged();
	}
}

private TPView _selectedViewForChange;
public TPView SelectedViewForChange
{
	get => _selectedViewForChange;
	set
	{
		_selectedViewForChange = value;
		OnPropertyChanged();
	}
}

private TPType _selectedTypeForChange;
public TPType SelectedTypeForChange
{
	get => _selectedTypeForChange;
	set
	{
		_selectedTypeForChange = value;
		OnPropertyChanged();
	}
}

private Detail _selectedDetailById;
public Detail SelectedDetailById
{
	get => _selectedDetailById;
	set
	{
		_selectedDetailById = value;
		OnPropertyChanged();
	}
}

public ICommand AddItemCommand { get; set; }
public ICommand UpdateCommand { get; set; }
public ICommand UpdateItemCommand { get; set; }
public ICommand DeleteCommand { get; set; }

ApplicationContext context;

public TechProcessViewModel()
{
	context = new ApplicationContext();
```

```
                ComboView = new ObservableCollection<TPView>(context.DbTPViews);
                ComboType = new ObservableCollection<TPType>(context.DbTPTypes);
                ComboDetail = new ObservableCollection<Detail>(context.DbDetails);

                TPCodes = new
List<string>(context.DbTechProcesses.Select(x=>x.Code));

                AddItemCommand = new ServiceCommand((x) => true, AddItem);
                UpdateCommand = new ServiceCommand((x) => true, Update);
                UpdateItemCommand = new ServiceCommand((x) => true, UpdateItem);
                DeleteCommand = new ServiceCommand((x) => true, Delete);
        }

        private void LoadItems()
        {
                ListItems = new
ObservableCollection<TechProcess>(context.DbTechProcesses);
        }

        private void AddItem(object obj)
        {
                if (SelectedViewForChange !=null && SelectedTypeForChange !=null
                        && SelectedDetailById !=null)
                {
                        var selectedDetail = context.DbDetails.
                                Where(x => x.Id == SelectedDetailById.Id).First();

                        var newTP = new TechProcess
                        {
                                Detail_Id = SelectedDetailById.Id,
                                TPViewId = SelectedViewForChange.Id,
                                TPViews = SelectedViewForChange,
                                TPTypeId = SelectedTypeForChange.Id,
                                TPTypes = SelectedTypeForChange,
                                Number = 1,
                                Name =
$"{SelectedViewForChange.Name}\\{SelectedTypeForChange.Name}\\01",
                        };

                        foreach (string code in TPCodes)
                        {
                                if (newTP.Code == code)
                                {
                                        newTP.Number++;
                                        newTP.Name =
$"{SelectedViewForChange.Name}\\{SelectedTypeForChange.Name}\\0{newTP.Number}";
                                }
                        }
                        context.DbTechProcesses.Add(newTP);
                        selectedDetail.TechProcesses.Add(newTP);
                        context.SaveChanges();
                }
                else
                {
                        MessageBox.Show("Оберіть параметри деталі");
                }

                LoadItems();
        }

        private void Update(object obj)
        {
                SelectedTechProcess = obj as TechProcess;
        }

        private void UpdateItem(object obj)
```

```
		{
			if (SelectedTechProcess != null)
			{
				var selectedTP = context.DbTechProcesses.
					Where(x => x.Id == SelectedTechProcess.Id).First();

				if (selectedTP !=null && SelectedViewForChange !=null &&
SelectedTypeForChange !=null)
				{
					selectedTP.TPViewId = SelectedViewForChange.Id;
					selectedTP.TPViews = SelectedViewForChange;
					selectedTP.TPTypeId = SelectedTypeForChange.Id;
					selectedTP.TPTypes = SelectedTypeForChange;
					selectedTP.Number = SelectedTechProcess.Number;

					foreach (string code in TPCodes)
					{
						if (selectedTP.Code == code)
						{
							selectedTP.Number++;
						}
					}

					selectedTP.Name =
$"{SelectedViewForChange.Name}\\{SelectedTypeForChange.Name}\\0{selectedTP.Number}";
					selectedTP.TblName = SelectedTechProcess.TblName;
					context.Update(selectedTP);
					context.SaveChanges();

					SelectedTechProcessMain = selectedTP;
					SelectedViewMain = SelectedViewForChange;
					SelectedTypeMain = SelectedTypeForChange;
				}
			}

			LoadItems();
		}

		private void Delete(object obj)
		{
			SelectedTechProcess = obj as TechProcess;

			if (SelectedTechProcess != null)
			{
				context.DbTechProcesses.Remove(SelectedTechProcess);
				context.SaveChanges();
				ListItems.Remove(SelectedTechProcess);
			}
		}

		public void GetSelectedItem(object TreeViewSelectedItem)
		{
			var selectedTP = TreeViewSelectedItem as TechProcess;
			if (selectedTP != null)
			{
				SelectedViewMain = context.DbTechProcesses.
					Where(x => x.Id == selectedTP.Id).
					Select(x => x.TPViews).First();

				SelectedTypeMain = context.DbTechProcesses.
					Where(x => x.Id == selectedTP.Id).
					Select(x => x.TPTypes).First();

				selectedTP.TPViews = SelectedViewMain;
				selectedTP.TPTypes = SelectedTypeMain;
```

```
                        SelectedTechProcessMain = selectedTP;
                    }

                    LoadItems();
                }
            }
        }
```

Лістинг файлу TransitionViewModel.cs

```
using DataModel;
using DataModel.Helpers;
using DataModel.Models;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;
using static Microsoft.EntityFrameworkCore.DbLoggerCategory;
using wpf_technological.Utility;
using System.Windows;

namespace wpf_technological.ViewModels
{
    public class TransitionViewModel : ObservableObject
    {
        public ObservableCollection<TechProcess> ComboTP { get; set; }

        private Transition _selectedTransition;
        public Transition SelectedTransition
        {
            get => _selectedTransition;
            set
            {
                _selectedTransition = value;
                OnPropertyChanged();
            }
        }

        private Transition _selectedTransitionForChange;
        public Transition SelectedTransitionForChange
        {
            get => _selectedTransitionForChange;
            set
            {
                _selectedTransitionForChange = value;
                OnPropertyChanged();
            }
        }

        private TechProcess _selectedTP;
        public TechProcess SelectedTP
        {
            get => _selectedTP;
            set
            {
                _selectedTP = value;
                OnPropertyChanged();
            }
        }

        private TechProcessTransition _selectedTPTrans;
        public TechProcessTransition SelectedTPTrans
        {
            get => _selectedTPTrans;
            set
```

```
                {
                        _selectedTPTrans = value;
                        OnPropertyChanged();
                }
        }

        public ICommand UpdateTransitionCommand { get; set; }
        public ICommand UpdateTransitionItemCommand { get; set; }
        public ICommand UpdateTPTransitionCommand { get; set; }

        ApplicationContext context;

        public TransitionViewModel()
        {
                context = new ApplicationContext();

                ComboTP = new
ObservableCollection<TechProcess>(context.DbTechProcesses);

                UpdateTransitionCommand = new ServiceCommand((x) => true,
UpdateTransition);
                UpdateTransitionItemCommand = new ServiceCommand((x) => true,
UpdateTransitionItem);
                UpdateTPTransitionCommand = new ServiceCommand((x) => true,
UpdateTPTransition);
        }

        private void UpdateTransition(object obj)
        {
                if (SelectedTransition != null)
                {
                        SelectedTransitionForChange = SelectedTransition;
                }
        }

        private void UpdateTransitionItem(object obj)
        {
                if (SelectedTransitionForChange != null)
                {
                        var selectedTransition = context.DbTransitions.
                                Where(x => x.Id ==
SelectedTransitionForChange.Id).First();

                        if (selectedTransition != null)
                        {
                                selectedTransition.TransitionText =
SelectedTransitionForChange.TransitionText;

                                SelectedTransition = selectedTransition;

                                context.Update(selectedTransition);
                                context.SaveChanges();

                                SpecificViewModel specificVM;
                                PerfGlobal objGlobal;

                                objGlobal = new PerfGlobal();
                                specificVM = objGlobal.SpecificVM;

                                specificVM.TreeViewUpdate();
                        }
                }
        }

        private void UpdateTPTransition(object obj)
        {
```

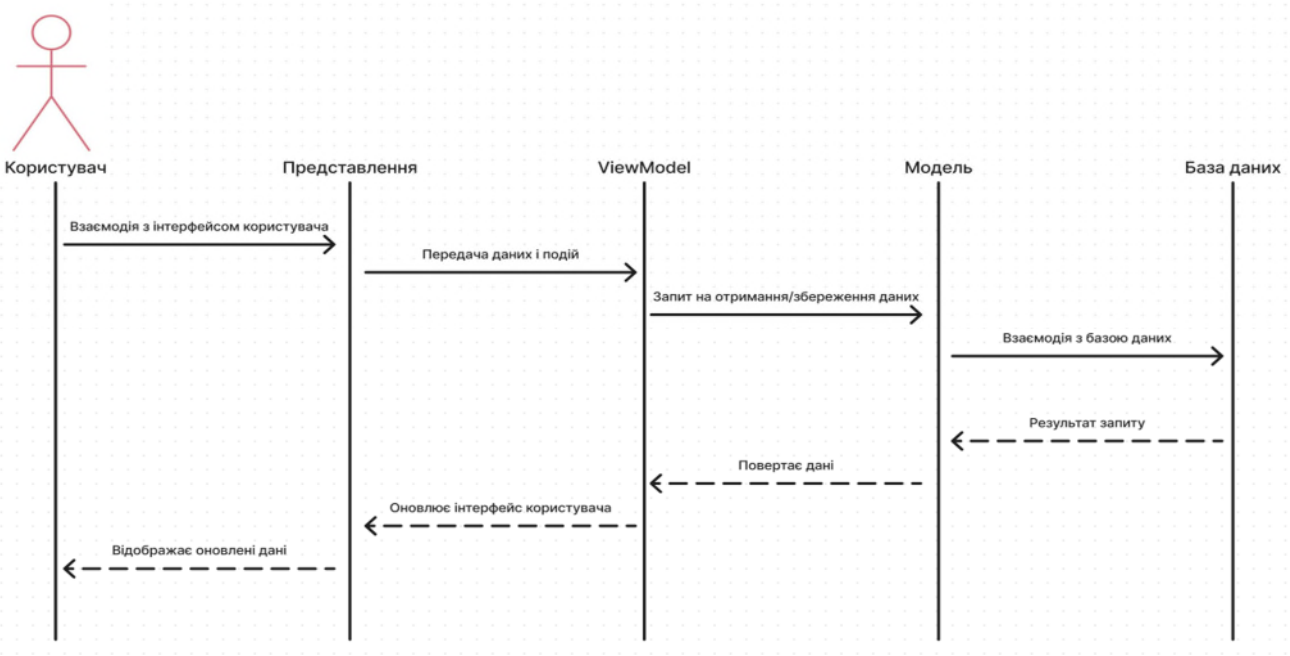
```
                  if (SelectedTransition !=null && SelectedTP != null &&
SelectedTPTrans != null)
                  {
                        var selectedTPTrans = context.DbTechProcessTransitions.
                              Where(x => x.TransitionId ==
SelectedTransition.Id).First();

                        var selectedTP = context.DbTechProcesses.
                              Where(x => x.Id == SelectedTP.Id).First();

                        if (selectedTPTrans != null && selectedTP!=null)
                        {
                              if (selectedTP.Id == selectedTPTrans.TechProcessId)
                              {
                                    MessageBox.Show("Оберіть інший технологічний
процес!");
                              }
                              else
                              {
                                    selectedTPTrans.TechProcessId = selectedTP.Id;
                                    selectedTPTrans.TechProcesses = selectedTP;

                                    context.Update(selectedTPTrans);
                                    context.SaveChanges();
                              }
                        }
                  }
            }

            public void GetSelectedItem(object TreeViewSelectedItem)
            {
                  var selected = TreeViewSelectedItem as Transition;

                  if (selected != null)
                  {
                        SelectedTransition = selected;

                        var selectedTPTrans = context.DbTechProcessTransitions.
                              Where(x => x.TransitionId ==
SelectedTransition.Id).First();

                        if (selectedTPTrans != null)
                        {
                              SelectedTP = selectedTPTrans.TechProcesses;

                              SelectedTPTrans = selectedTPTrans;
                        }
                  }
            }
      }
}
```

Додаток Б

Діаграма варіантів використання застосунку

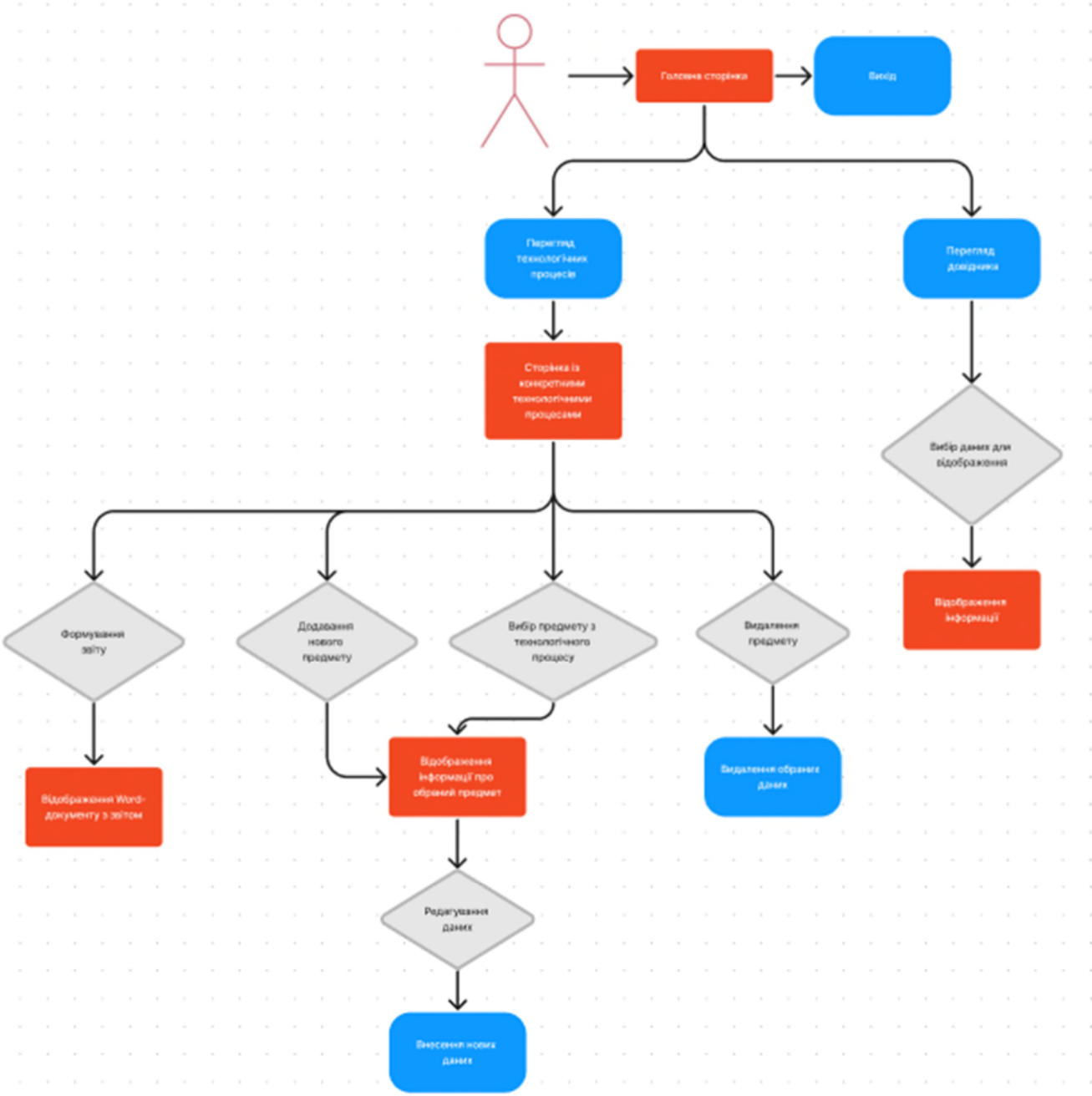
Додаток В

Діаграма послідовності



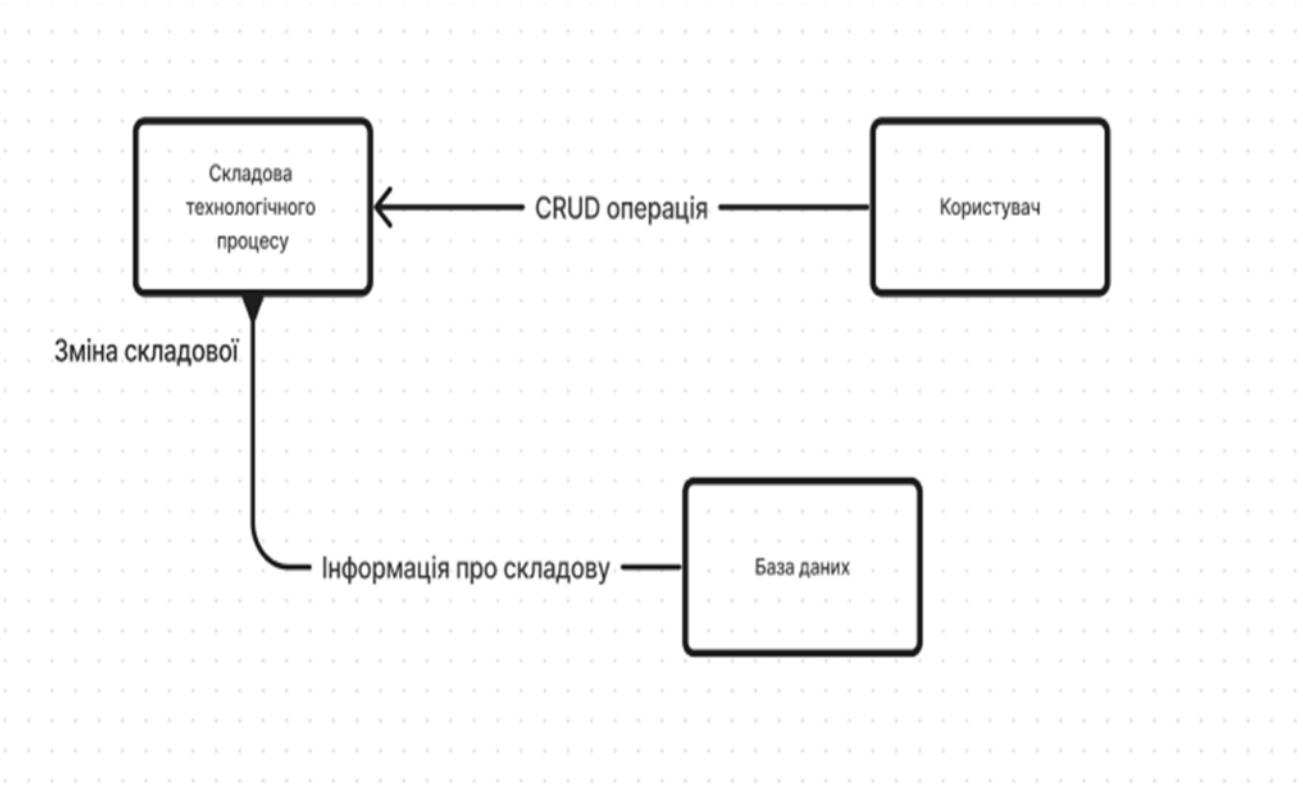
Додаток Г

Діаграма активності користувача



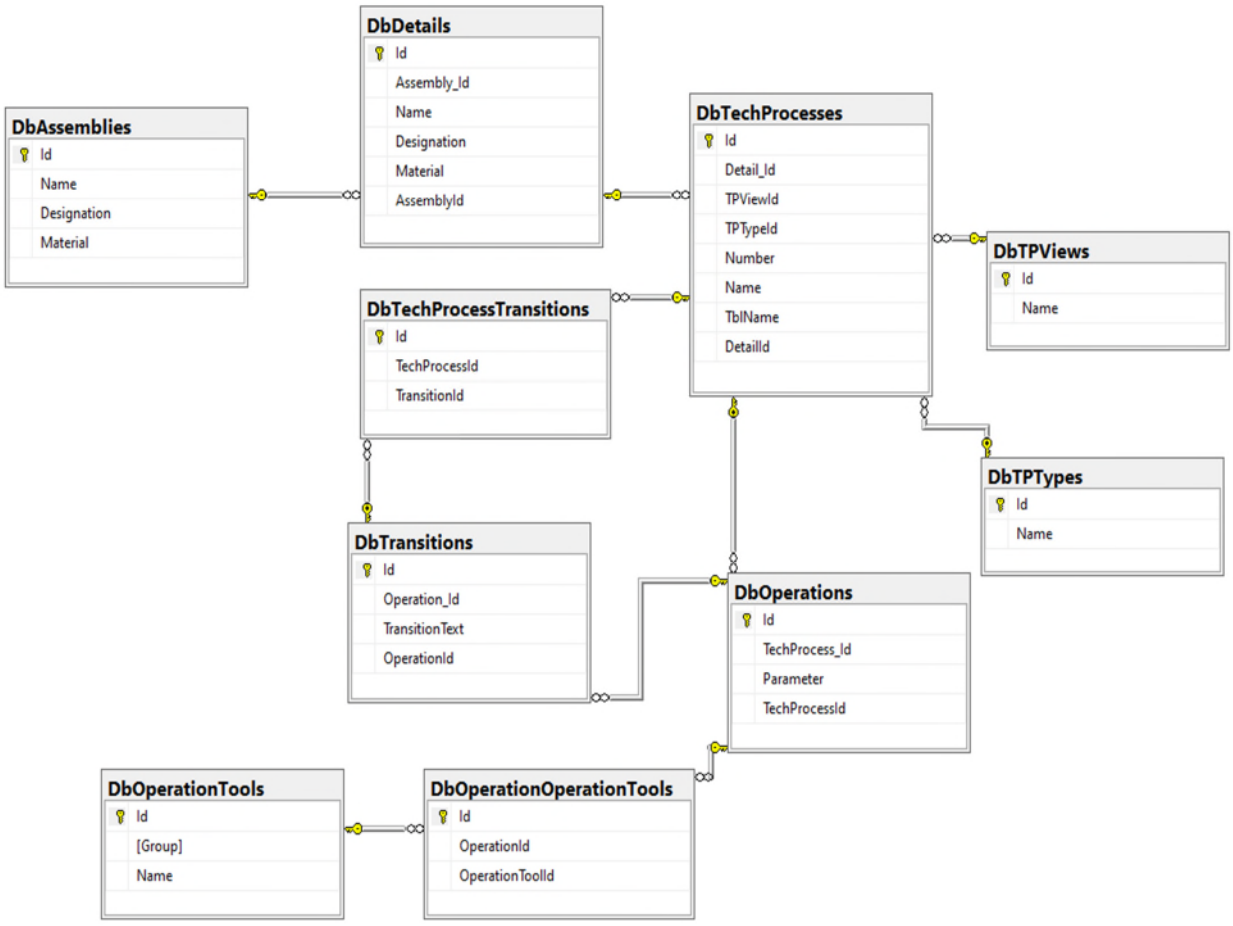
Додаток Д

Діаграма компонентів



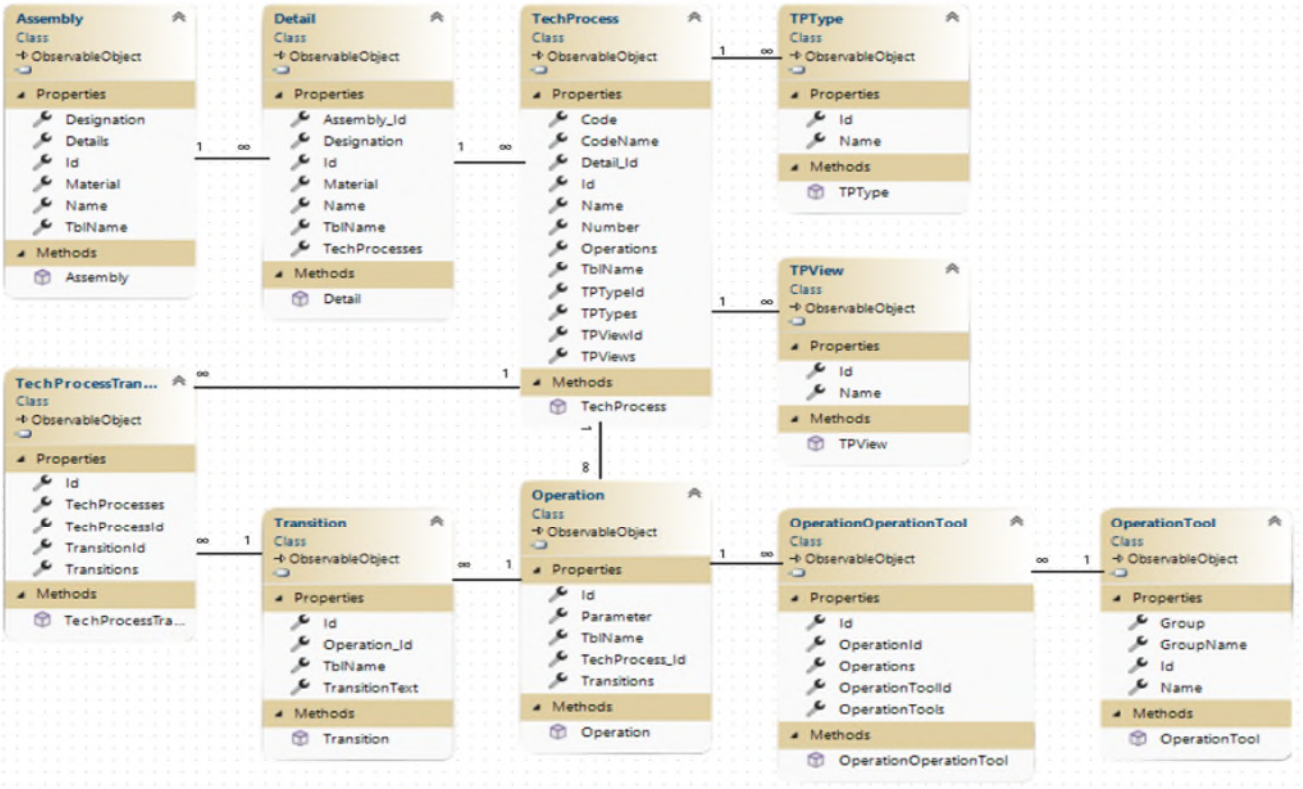
Додаток Е

Інфологічна модель даних



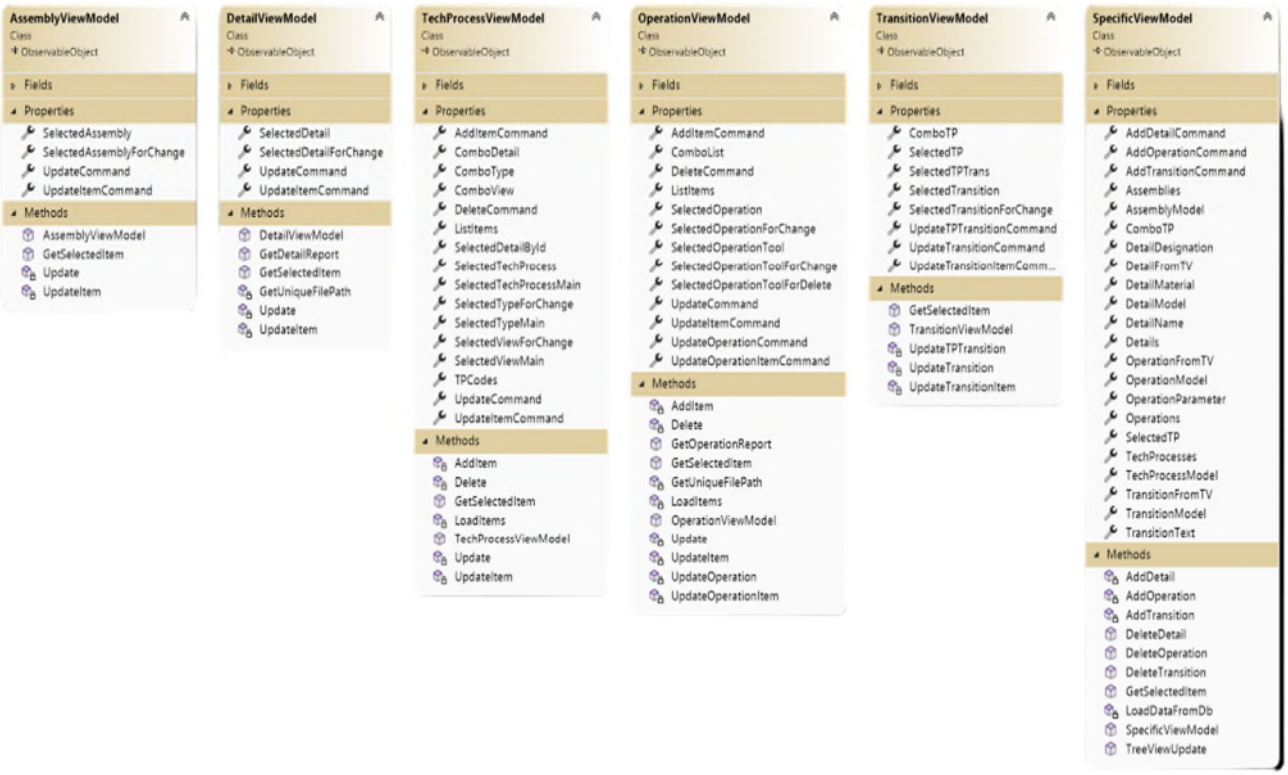
Додаток Є

Діаграма класів моделей



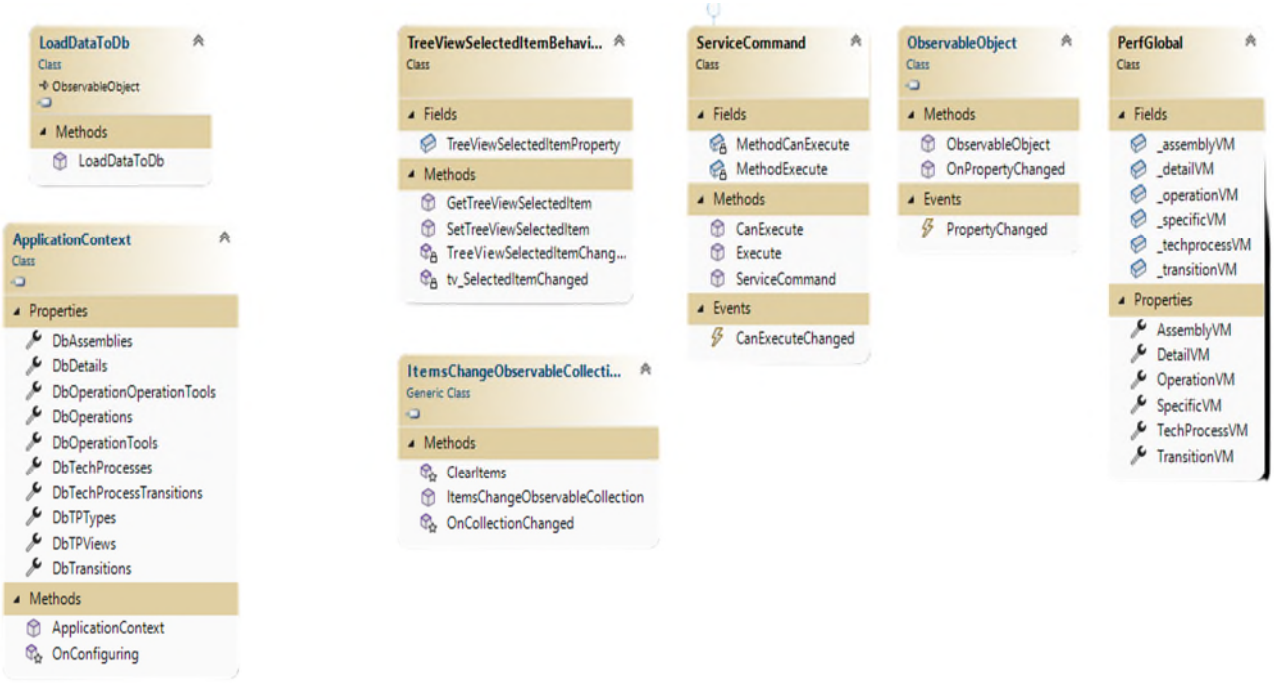
Додаток Ж

Діаграма класів ViewModels



Додаток З

Діаграма класів роботи з базою даних та допоміжних класів



Додаток И

Презентаційний матеріал

Кваліфікаційна Робота Бакалавра

CASE-технологія для системи проектування технологічних процесів виготовлення деталей машин

ВИКОНАВ:
Студент 4 курсу , групи КН - 19- 1
Медведчук Віталій Юрійови ч

КЕРІВНИК
к.т.н., доцент кафедри КН
Багрій Руслан Олександрович

Актуальність

САПР ТП спрощує технологічну підготовку виробництва, включаючи планування та послідовність операцій. CASE-технології забезпечують якість та простоту обслуговування програмного забезпечення. Вони використовують інтегровані автоматизовані інструменти для проектування інформаційних систем.Враховуючи те, що на вітчизняному ринку переважають іноземні програмні продукти, розробка власного вітчизняного застосунку для вирішення подібних проблем може бути доцільною. Це дозволить забезпечити національну підтримку та адаптацію до потреб вітчизняних користувачів.

Мета та постановка задачі

Метою кваліфікаційної роботи бакалавра є розробка системи проектування технологічних процесів виготовлення деталей машин з використанням CASE-технології.

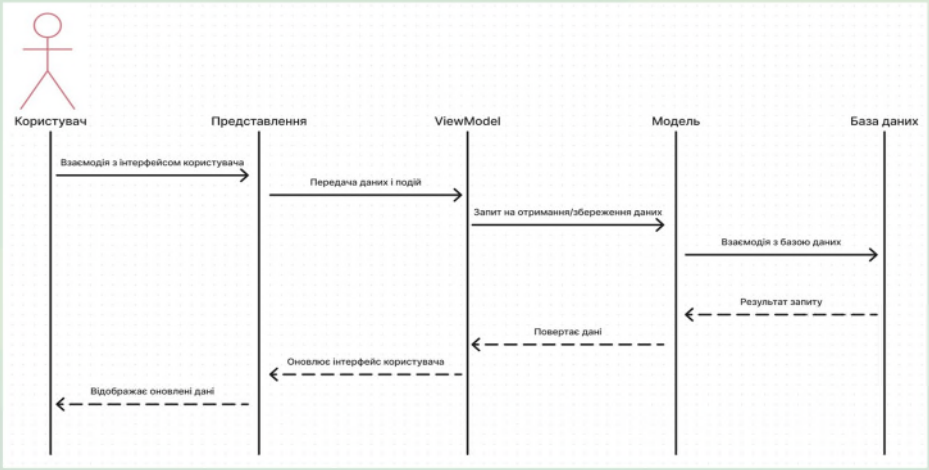
Для досягнення поставленої мети необхідно реалізувати виконання наступних задач:

- Проаналізувати задачу проектування технологічних процесів виготовлення деталей машин.
- Розробити функціональну структуру інформаційної системи проектування технологічних процесів з використанням CASE-технології.
- Розробити структуру інформаційної системи проектування технологічних процесів з використанням CASE-технології.
- Розробити програмну реалізацію інформаційної системи проектування технологічних процесів виготовлення деталей машин.

Діаграма варіантів використання застосунку

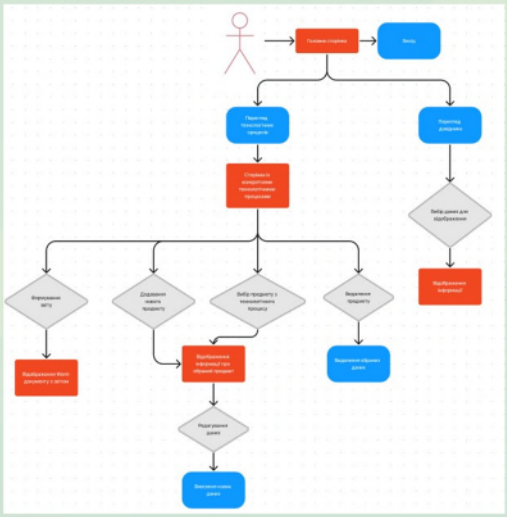
Наведена вище діаграма демонструє ситуацію, яка описує інформаційну систему та користувача, який може користуватись функціоналом цієї системи.

Діаграма послідовності



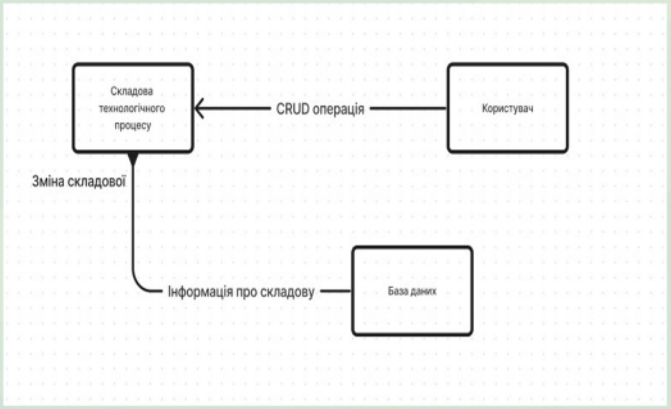
В наведеній вгорі діаграмі послідовності демонструється послідовність дій, що відбуватиметься при ініціації будь-якої дії користувачем інформаційної системи.

Діаграма активності користувача



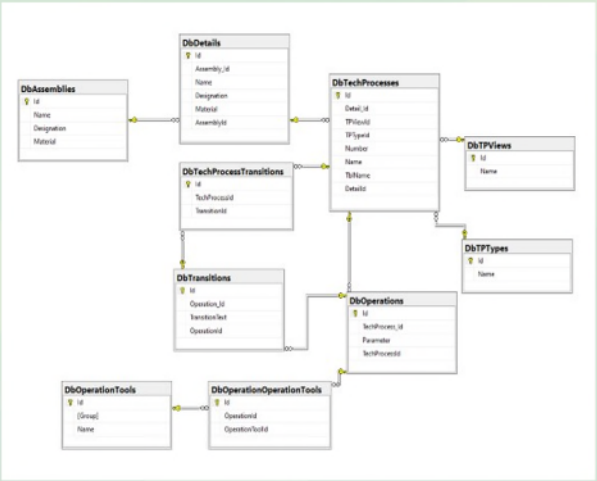
Діаграма активності ілюструє послідовність дій, які виконуються в процесі використання інформаційної системи користувачем.

Діаграма компонентів



Діаграма компонентів – діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Інфологічна модель даних

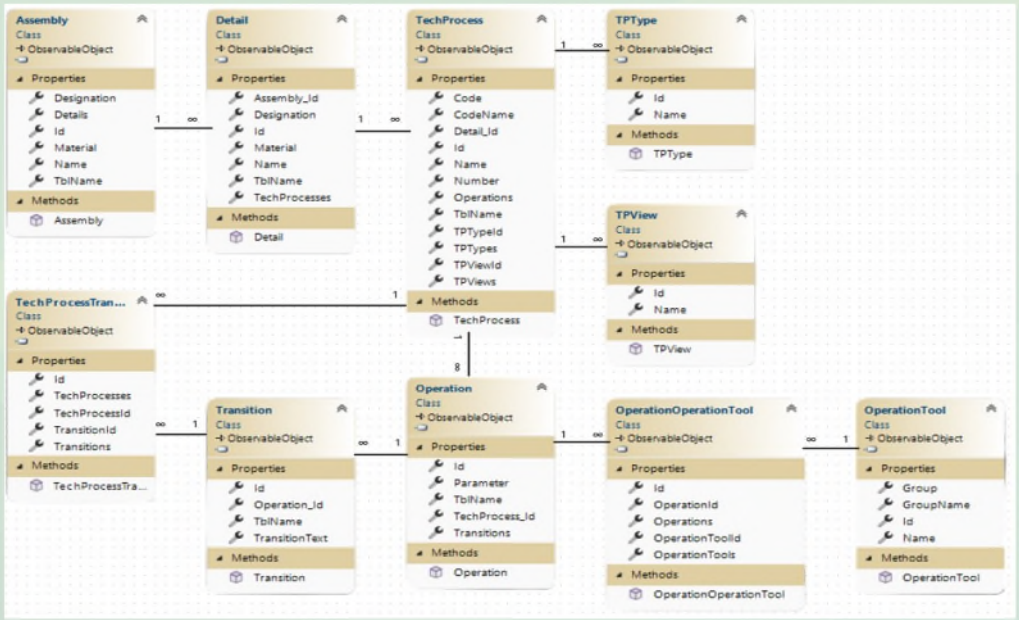


Основна мета інфологічної моделі даних - відобразити ключові концепції, залежності та взаємозв'язки між об'єктами, що існують в предметній області.

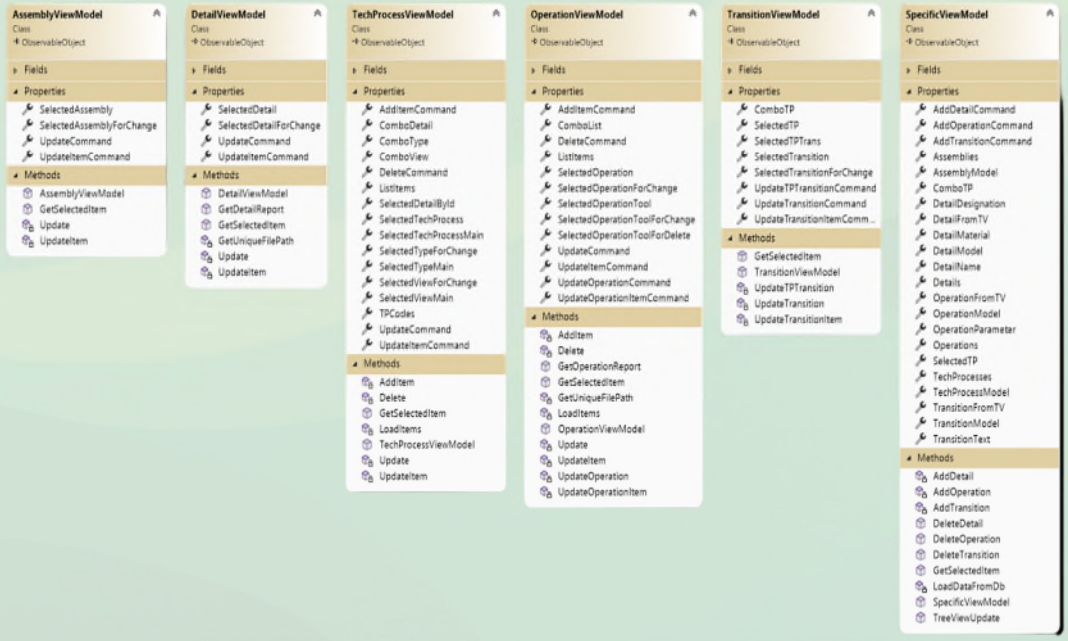
Засоби розробки

- Основною мовою програмування було використано C#
- Для створення застосунку платформою було обраноWPF
- Для реалізації дизайну використано XAML
- Для розділення логіки продукту на підсистеми використано Layered Architecture
- Для розділення логіки застосунку від способу її відображення використано паттерн MVVM
- Для зв'язку із базою даних було використано бібліотеку EntityFramework Core
- Для побудови баз даних використано підхід Code First

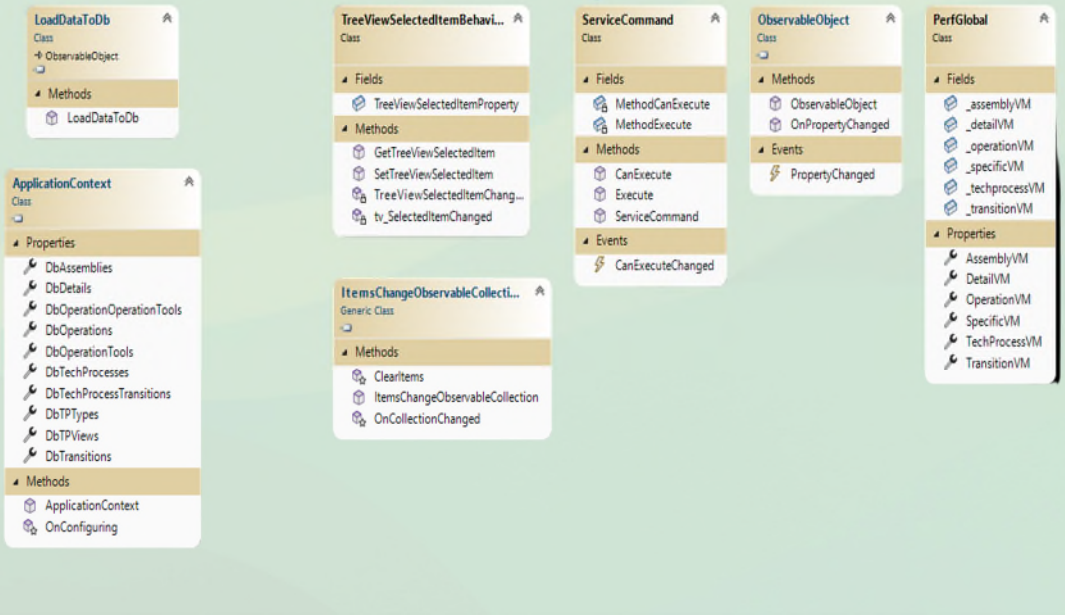
Діаграма класів моделей



Діаграма класів ViewModels



Діаграма класів роботи з базою даних та допоміжних класів



Висновки

Результатом виконання кваліфікаційної роботи бакалавра є створення інформаційної системи проектування технологічних процесів виготовлення деталей машин з використанням CASE-технології. Використання CASE-технологій (Computer-Aided Software Engineering) можуть забезпечити автоматизацію розробки інформаційних систем, поліпшити їх якість та прискорити процес розробки. Разом з CASE-технологіями, UML-моделювання дозволяє нам візуалізувати та оптимізувати роботу інформаційної системи, покращує комунікацію між учасниками проекту та сприяє зниженню витрат на розробку та підтримку системи. Подальше використання підходу Code First та фреймворку Entity Framework допомагає спростити розробку програмного забезпечення, зменшуючи кількість потрібного коду. Також варто зазначити, що шарова архітектура, патерн MVVM та використання мови програмування C# та платформи WPF сприяють розділенню логіки на підсистеми, полегшуючи розробку та модифікацію.

В ході виконання кваліфікаційної роботи бакалавра поставлено завдання реалізувати виконання наступних задач:

- Проаналізувати задачу проектування технологічних процесів виготовлення деталей машин.
- Розробити функціональну структуру інформаційної системи проектування технологічних процесів з використанням CASE-технології.
- Розробити структуру інформаційної системи проектування технологічних процесів з використанням CASE-технології.
- Розробити програмну реалізацію інформаційної системи проектування технологічних процесів виготовлення деталей машин.

В майбутньому розроблена система може використовуватись для проектування технологічних процесів виготовлення деталей машин. Аналізуючи отриманий результат можна зробити висновок, що програмна реалізація інформаційної системи працює коректно та надає можливість для виконання функцій визначених в технічному завданні кваліфікаційної роботи бакалавра.

Дякую за увагу!

Wed May 31 08:40:37 EEST 2023, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 4.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 10%**

	Документ		Сумарний збіг по Базі Даних	
ID: 114353 Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА Додано в БД: 2023-05-31 Автора: В.Ю. Медведчук Керівники: Р.О. Багрій Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	49742	758	3466 (7%)	59 (8%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра КН

ID перевірки:
1015336018

Дата перевірки:
31.05.2023 09:17:35 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
31.05.2023 09:25:37 EEST

ID користувача:
100005671

Назва документа: **КН-19-1 Медведчук**

Кількість сторінок: **64** Кількість слів: **7589** Кількість символів: **60786** Розмір файлу: **2.09 MB** ID файлу: **1015005024**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

7.72%
Схожість

Найбільша схожість: 3.39% з джерелом з Бібліотеки (ID файлу: 1014976252)

7.08% Джерела з Інтернету	349	Сторінка 66
4.6% Джерела з Бібліотеки	103	Сторінка 69

0.11% Цитат

Цитати	2	Сторінка 70

Не знайдено жодних посилань

0%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за 8 слів та 0%)

0% Вилучення з Інтернету	2	Сторінка 71

Немає вилучених бібліотечних джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	24 сторінки

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *CASE-технологія для системи проектування технологічних процесів виготовлення деталей машин*

Автор: *студент гр. КН-19-1 Медведчук Віталій Юрійович*

Спеціальність: *122 – Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *к.т.н., доц. Багрій Р.О.*

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	***відповідає***
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) за програмою Anti-Plagiarism виявлені 4% є фрагментарними – містять поширені конструкції, загальновідомі терміни, скорочення та визначення.
2) За програмою UNICHECK виявлені 7.72%, що є запозиченнями, які розміщені в розділах аналізу існуючих технологій та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 4% і 7.72% відповідно, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Руслан Багрій

Гарант ОП _____ Олександр Мазурець

Завідувач кафедри КН _____ Олександр Бармак

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-1 Медведчука Віталія Юрійовича*

за темою *CASE-технологія для системи проектування технологічних процесів виготовлення деталей машин*

1. Актуальність теми

Актуальність теми достатньо обґрунтована, оскільки CASE-технологія для системи проектування технологічних процесів виготовлення деталей машин дозволяє автоматизувати процесу розробки програмного забезпечення. Особливістю теми є використання CASE-інструментів, що надають можливості для моделювання, візуалізації та аналізу вимог, створення діаграм та моделей для проектування архітектури програми, генерації автоматичного коду, тестування програмного забезпечення, керування процесами розробки.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

Теми кваліфікаційної роботи "CASE-технологія для системи проектування технологічних процесів виготовлення деталей машин" відповідає предметній області спеціальності 122 Комп'ютерні науки та вимогам до кваліфікаційної роботи бакалавра, оскільки результатом роботи є створення інформаційної системи проектування технологічних процесів виготовлення деталей машин з використанням CASE-технології. При вирішенні поставленої задачі використано методи збору та аналізу інформації, методи моделювання UML, методи аналізу вимог, методи генерації коду, методи тестування, технології та методи проектування інформаційних систем проектування технологічних процесів виготовлення деталей.

3. Професійні та особистісні якості бакалавра

Медведчук В. Ю. продемонстрував глибоке розуміння теоретичних аспектів та практичних навичок використання CASE-інструментів. Проявив високий рівень аналітичних здібностей та володіння комунікаційними та організаційними навичками, пропонував нестандартні рішення та підходи, що покращили якість роботи та мали вплив на кінцевий результат. Всі ці професійні та особистісні якості допомогли успішно виконати кваліфікаційну роботу бакалавра і досягти поставлених цілей.

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Робота виконана самостійно, академічного плагіату не виявлено, стосовно всіх запозичень наведено відповідні посилання на джерела.

5. Ступінь оволодіння методами дослідження

При реалізації кваліфікаційної роботи показала достатній рівень компетентностей та володіння необхідними інструментами та обладнанням, методами, методиками та технологіями предметної області комп'ютерних наук.

6. Повнота та якість розкриття теми роботи

Тема роботи повністю розкрита, виконані усі поставлені задачі та розроблено програмне забезпечення для підтвердження запропонованої case-технології для системи проектування технологічних процесів виготовлення деталей машин.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Викладення матеріалу логічне, послідовне та аргументоване. Мова і стиль викладення кваліфікаційної роботи відповідають стандартам, що забезпечує доступність сприймання матеріалу і відповідає вимогам до сучасних наукових робіт.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Розроблена у роботі інформаційна система проектування технологічних процесів за CASE-технологією може бути застосована для проектування технологічних процесів виготовлення деталей машин.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

*Враховуючи високий рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка «**відмінно**» .*

Керівник _____ *к.т.н., доц. Руслан Багрій*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-1 Медведчука Віталія Юрійовича*
за темою: *CASE-технологія для системи проектування технологічних процесів виготовлення деталей машин*

1. Актуальність обраної теми

Система автоматизованого проектування технологічних процесів (САПР ТП) вирішує більшість завдань у рамках технологічної підготовки виробництва та дозволяє спростити формування та супровід техпроцесів. Створення програмних продуктів для САПР ТП є не простою задачею через використання значного об'єму довідникових даних та функціональних вимог.

CASE-технології (англ. computer-aided software engineering) допомагають забезпечити високу якість програм, відсутність помилок та простоту в обслуговуванні складних систем програмного забезпечення. Це забезпечується сукупністю методів і засобів проектування інформаційних систем з інтегрованими автоматизованими інструментами.

2. Повнота розкриття мети та завдань роботи

Протягом виконання кваліфікаційної роботи бакалавра мету та завдання роботи було розкрито у повній мірі, оскільки було проведено детальний аналіз предметної області, описано задачі, реалізація яких являється програмною реалізацією інформаційної системи проектування технологічних процесів за CASE-технологією.

3. Зміст кожного розділу роботи

Записка кваліфікаційної роботи бакалавра містить три розділи. У першому розділі проведено аналіз предметної області та існуючих рішень, оглянуто методи та засоби CASE-технології, встановлено мету, завдання та вимоги до реалізації. У другому розділі описано моделювання інформаційної системи з використанням CASE-технології, проектування структури, інтерфейсу та архітектури інформаційної системи. У третьому розділі описано процес програмної реалізації інформаційної системи проектування технологічних процесів за CASE-технологією.

4. Оцінка розробленої інформаційної системи, її практична цінність

Розроблена інформаційна система проектування технологічних процесів за CASE-технологією може бути застосована для проектування технологічних процесів виготовлення деталей машин.

5. Якість оформлення кваліфікаційної роботи бакалавра

Кваліфікаційна робота бакалавра виконана належним чином, відповідає всім вимогам, що регламентують виконання дипломних робіт.

6. Недоліки кваліфікаційної роботи бакалавра

Рекомендовано покращити інформаційну систему, шляхом додавання додаткової інформації про деталі, що дозволить здійснювати більш точне проектування технологічних процесів, а також розширити функціонал, який вже наявний.

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

Враховуючи рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка « відмінно ».

Рецензент _____ доц., к.ф.-м.н. Наталія Орєшина