**Міністерство освіти і науки України**
**Національний технічний університет**
**«Дніпровська політехніка»**

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

**Кафедра** Програмного забезпечення комп'ютерних систем
(повна назва)

# ПОЯСНЮВАЛЬНА ЗАПИСКА
## кваліфікаційної роботи ступеня

*магістра*
(назва освітньо-кваліфікаційного рівня)

**студента** *Олійника Ігоря Костянтиновича*
(ПІБ)

**академічної групи** *121м-22-3*
(шифр)

**спеціальності** *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

**освітньої програми** *«Інженерія програмного забезпечення»*
(назва освітньої програми)

**на тему:** *Дослідження ефективності впровадження системи для управління технічними інструкціями з можливістю відокремлення обов'язків*

| **Керівники** | **Прізвище, ініціали** | **Оцінка за шкалою** | | **Підпис** |
|---|---|---|---|---|
| | | **рейтинговою** | **інституційною** | |
| розділів кваліфікаційної роботи | | | | |
| **спеціальний** | *Проф. Приходченко С.Д.* | | | |
| **Рецензент** | | | | |
| **Нормоконтролер** | *Проф. Лактіонов І.С.* | | | |

**Дніпро**
**2023**

**Міністерство освіти і науки України**
**Національний технічний університет**
**«Дніпровська політехніка»**

**ЗАТВЕРДЖЕНО:**
Завідувач кафедри

Програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 20 23 Року

# ЗАВДАННЯ
**на виконання кваліфікаційної роботи**

**спеціальності** *121 Інженерія програмного забезпечення*
*(код і назва спеціальності)*

**студенту** *121м-22-3* (група) *Олійник Ігор Костянтинович* (прізвище та ініціали)

**Тема кваліфікаційної роботи** *Дослідження ефективності впровадження системи для управління технічними інструкціями з можливістю відокремлення обов'язків*

## 1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 31.10.2023 р. № 1200-с

## 2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

**Об'єкт досліджень** – процес автоматизації технічної інструкції.

**Предмет досліджень** – моделі та методи системи призначенні для створення конструктору та використанню інструкції.

**Мета НДР** – підвищення ефективності створення та користування технічними інструкціями.

**Вихідні дані для проведення роботи** – теоретичні та експериментальні дослідження вдосконалення задач проектування системи застосунку.

## 3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

**Новизна запропонованих рішень** полягає представлені в даному дослідженні рішення яке відзначається своєю виразною новизною шляхом обґрунтування та вирішення актуальної проблеми, що полягає в відсутності зручних та універсальних додатків, призначених для створення технічних інструкцій для користувачів.

**Практична цінність** полягає в їх здатності ефективно вплинути на процес створення та управління технічними інструкціями для користувачів. Ця система

відкриває нові можливості та переваги для різних сфер діяльності, де інструкції мають важливе значення.

## 4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Отримані результати дослідження мають бути представлені у формі, що дозволить користувачам оцінити практичне застосування запропонованих методів у створені програмного забезпечення для керування технічними інструкціями. В ході цієї роботи планується розробка програмного забезпечення для дослідження ефективності впровадження системи для управління технічними інструкціями з можливістю відокремлення обов'язків адміністрування і користування.

## 5 ЕТАПИ ВИКОНАННЯ РОБІТ

| **Найменування етапів робіт** | **Строки виконання робіт (початок – кінець** |
|---|---|
| Аналіз теми та постановка задачі | 12.09.2023-30.09.2023 |
| Розробка моделі, метода та програмного забезпечення створення та керування технічними інструкціями | 01.10.2023-31.10.2023 |
| Використання програми та аналіз отриманих результатів | 01.11.2023-12.12.2023 |

## 6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

**Економічний ефект** включає вигоди від зменшення витрат, підвищення продуктивності, покращення якості та зменшення ризиків у зв'язку з впровадженням системи для керування технічними інструкціями.

**Соціальний ефект** полягає в полегшенні доступу користувачів до інформації, поліпшенні їхнього розуміння і використання технічних інструкцій та зменшенню ризиків.

## 7 ДОДАТКОВІ ВИМОГИ

Завдання видав _____ *Приходченко С.Д.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Олійник І.К.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *12.09.2023 р.*

Термін подання кваліфікаційної роботи до ЕК *12.12.2023*

# РЕФЕРАТ

Пояснювальна записка: 90 сторінки, 19 рисунків, 2 додатки, 17 джерел.

Об’єкт дослідження: процес автоматизації технічної інструкції.

Мета магістерської роботи: підвищення ефективності створення та користування технічними інструкціями.

Методи дослідження: моделі та методи системи призначенні для створення конструктору та використанню інструкції.

Наукова новизна даної роботи полягає представлені в даному дослідженні рішення яке відзначається своєю виразною новизною шляхом обґрунтування та вирішення актуальної проблеми, що полягає в відсутності зручних та універсальних додатків, призначених для створення технічних інструкцій для користувачів.

Практична цінність полягає в їх здатності ефективно вплинути на процес створення та управління технічними інструкціями для користувачів. Ця система відкриває нові можливості та переваги для різних сфер діяльності, де інструкції мають важливе значення.

Область застосування: результат кваліфікаційної роботи зосереджено на вдосконаленні процесу автоматизації технічної інструкції, а тому має широкий спектр областей застосування та потенційних вигод, наприклад: промисловість та виробництво, ІТ та розробка ПЗ, медична сфера, освіта та навчання.

Значення роботи та висновки: значний внесок у сферу автоматизації технічних інструкцій, вирішуючи актуальні проблеми, пов'язані з їх створенням та управлінням, наприклад: підвищення ефективності, висока якість інструкцій, економія ресурсів, інновації в сферах застосування.

Прогнози щодо розвитку досліджень: глибші дослідження, тобто подальше вивчення та розробка нових аспектів автоматизації технічних інструкцій для розширення областей їх застосування. Впровадження інновацій: застосування нових технологій та інновацій для покращення функціональності та ефективності систем автоматизації. Співпраця та обмін досвідом: розвиток співпраці між дослідниками та обмін досвідом для створення ще більш ефективних рішень у цій сфері

Список ключових слів: БД, MVVM, інструкція, контейнер, додаток, категорія, код інструкції, тип інструкції.

# ABSTRACT

Explanatory note: 90 pages, 19 figures, 2 appendices, 17 sources.

The object of research: the process of automating technical instructions.

The purpose of the master's work: increasing the efficiency of creating and using technical instructions.

Research methods: models and methods of the system intended for the creation of a designer and the use of instructions.

The scientific novelty of this work consists in the solution presented in this study, which is distinguished by its expressive novelty by substantiating and solving the actual problem, which consists in the lack of convenient and universal applications designed to create technical instructions for users.

The practical value lies in their ability to effectively influence the process of creating and managing technical instructions for users. This system opens up new opportunities and advantages for various fields of activity where instructions are important.

Field of application: the result of the qualification work focuses on the improvement of the automation process of the technical instruction, and therefore has a wide range of application areas and potential benefits, for example: industry and manufacturing, IT and software development, medical field, education and training.

Significance of the work and conclusions: a significant contribution to the field of automation of technical instructions, solving actual problems related to their creation and management, for example: increasing efficiency, high quality of instructions, saving resources, innovation in the fields of application.

Forecasts for the development of research: deeper research, that is, further study and development of new aspects of the automation of technical instructions to expand the areas of their application. Implementation of innovations: application of new technologies and innovations to improve the functionality and efficiency of automation systems. Cooperation and exchange of experience: development of cooperation between researchers and exchange of experience to create even more effective solutions in this field

List of keywords: DB, MVVM, instruction, container, application, category, instruction code, instruction type.

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

MVVM – Model-View-ViewModel

XAML – eXtensible Application Markup Language

# ЗМІСТ

# ВСТУП

В сучасному світі, де технології та інновації стають невід'ємною частиною нашого повсякденного життя, зростає важливість належної технічної документації та інструкцій, призначених для налаштування, обслуговування та коректного використання різних продуктів та послуг. Розробка та імплементація системи для створення та керування технічними інструкціями, з відокремленням функцій адміністрування та звичайного користувача, стає актуальним завданням у сфері інформаційних технологій та підприємництва.

Сьогодні багато компаній та організацій стикаються з потребою створювати, зберігати та оновлювати технічні інструкції для своїх клієнтів та співробітників. Це особливо важливо в промисловості, медицині, інформаційних технологіях, готельному бізнесі та багатьох інших галузях. Якщо інструкції не зрозумілі, неправильно виконані або неактуальні, це може призвести до небезпеки, незручностей та невдоволеності користувачів.

Дослідження та розробка системи для створення, оновлення та керування технічними інструкціями має на меті спростити цей процес, зробити його більш ефективним та забезпечити доступність важливої інформації для всіх зацікавлених сторін. Важливо підкреслити, що сучасні технології, вони відкривають нові можливості для розробки цієї системи та забезпечують її відповідність сучасним вимогам і стандартам.

Актуальність даної теми очевидна, оскільки вона відповідає реальним потребам підприємств та організацій у забезпеченні належної технічної документації та підвищенні якості обслуговування. Розробка імплементація системи для керування технічними інструкціями може значно спростити життя фахівців у різних галузях та покращити безпеку та ефективність їхньої роботи.

# РОЗДІЛ 1
# ГЛИБОКИЙ ОГЛЯД СПЕЦИФІКИ ДОСЛІДЖУВАНОЇ СФЕРИ

## 1.1 Сутність інструкції

Інструкція користувача є важливим елементом сучасного технічного середовища, де швидкий та ефективний доступ до інформації є ключовим фактором. Цей документ визначається як комплекс текстового та/або графічного матеріалу, призначеного для надання докладних та зрозумілих вказівок з використання конкретного продукту чи послуги. Інструкції користувача є невід'ємною частиною розробки та використання технологій, що спрямовані на полегшення та оптимізацію повсякденного життя.

Сутність інструкції користувача. Головною метою інструкції користувача є полегшити процес взаємодії користувача з технічними засобами та програмним забезпеченням. Вона надає конкретні кроки та рекомендації, щоб кінцевий користувач міг вирішити завдання чи ефективно використовувати можливості продукту. Інструкції можуть включати інформацію про встановлення, налаштування, використання та усунення неполадок [11].

Типи інструкцій користувача:

1 Текстові інструкції: Цей тип інструкцій використовує письмовий текст для передачі інформації. Електронні документи, довідники чи керівництва з експлуатації є частими представниками текстових інструкцій.

2 Відеоінструкції: Відеоінструкції використовують візуальні елементи для демонстрації процесів та дій. Вони здатні ефективно передати нюанси та деталі використання, що робить їх популярними для користувачів різного рівня кваліфікації.

3 Графічні інструкції: Цей вид інструкцій використовує графіку, схеми, діаграми та інші візуальні елементи для пояснення кроків та процесів. Графічні інструкції можуть бути особливо корисними для тих, хто краще абсорбує інформацію візуальним способом.

4 Конструкторські: Цей вид інструкцій призначен для фахівців з конструювання, монтажу та обслуговування, надаючи їм необхідні вказівки та рекомендації для ефективного виконання їхніх завдань.

Використання інструкцій користувача. Орієнтація користувача: Інструкції стають важливим елементом для нових користувачів, які потребують чітких вказівок для швидкого та ефективного використання продукту чи послуги [12].

Підтримка в процесі використання: Інструкції допомагають користувачам максимально використовувати функціонал продукту, надаючи необхідні вказівки для різних завдань та функцій.

Усунення неполадок: Інструкції часто містять інформацію про потенційні проблеми та методи їх вирішення, що полегшує швидке усунення неполадок та забезпечує неперервну роботу продукту.

Застосування технологій. Сучасні технології дозволяють розвивати інструкції ще більше. Інтерактивні елементи, такі як чат-боти чи вбудовані віртуальні асистенти, можуть надати користувачам персоналізовану допомогу в режимі реального часу. Додатково, мобільні додатки та розширені реальності можуть створити інтерактивне оточення для користувачів, спрощуючи взаємодію з продуктом.

Зростання складності технічних систем та програмних продуктів призвело до посилення потреби в ефективних інструкціях користувача. Розглянемо різні аспекти цього питання.

1 Текстові інструкції:

Текстові інструкції, хоча залишаються невід'ємною частиною документації, часто вимагають від користувача великого ступеня уваги та концентрації. Вони можуть бути корисними для досвідчених користувачів, які швидко адаптуються до нового середовища. Однак для початківців чи тих, хто надає візуальну інформацію, текстові інструкції можуть виявитися менш ефективними.

2 Відеоінструкції:

Відеоінструкції займають все більше пріоритету через свою здатність демонструвати реальні дії та нюанси використання продукту. Це особливо важливо для складних технічних операцій або налаштувань, які не так легко пояснити текстом. Зручність споживачів у сприйнятті візуального матеріалу робить відеоінструкції важливим ресурсом для навчання та розуміння.

3 Графічні інструкції:

Графічні інструкції поєднують в собі кращі аспекти текстових та візуальних інструкцій. Вони використовують графічні елементи, такі як схеми та діаграми, для ілюстрації кроків та процесів, але також доповнюють це текстовими поясненнями. Це робить їх універсальними для різних типів користувачів, забезпечуючи інформацію для тих, хто краще адаптується до різних видів матеріалу.

3 Конструкторські інструкції:

В рамках нашої роботи ми будемо використовувати конструкторську інструкцію. Конструкторські інструкції відіграють важливу роль у процесі розробки та експлуатації технічних продуктів. Ці інструкції призначені для фахівців з конструювання, монтажу та обслуговування, надаючи їм необхідні вказівки та рекомендації для ефективного виконання їхніх завдань.

Сутність конструкторських інструкцій

Конструкторські інструкції є документами, які визначають технічні характеристики, процеси монтажу та інші важливі аспекти, пов'язані з проектуванням та створенням технічних систем. Вони можуть містити схеми, креслення, технічні характеристики, інструкції з монтажу, а також вказівки щодо використання конкретних матеріалів та компонентів.

Типи конструкторських інструкцій

Монтажні інструкції: Цей тип інструкцій визначає послідовність дій для збирання конструкцій або виробів. Вони включають інформацію про необхідні інструменти, матеріали та правила безпеки.

Технічні специфікації: Ці інструкції містять деталізовану інформацію щодо технічних характеристик, вимог до матеріалів та технічних параметрів продукту чи системи.

Інструкції з обслуговування: Конструкторські інструкції можуть також включати в себе рекомендації щодо технічного обслуговування та ремонту, надаючи фахівцям з необхідними вказівками для довгострокового використання продукту.

Використання конструкторських інструкцій

При розробці: Конструкторські інструкції є важливим етапом у процесі розробки нових технічних продуктів. Вони визначають технічні параметри та вимоги, які служать основою для конструювання та виготовлення.

При монтажі та обслуговуванні: Фахівці з монтажу та технічного обслуговування користуються конструкторськими інструкціями для ефективного виконання своїх обов'язків. Це дозволяє забезпечити якість та надійність технічних систем.

Підтримка виробника: Конструкторські інструкції також важливі для підтримки виробників. Вони допомагають у вирішенні питань якості, ефективності та безпеки продуктів.

Застосування технологій. Сучасні технології дозволяють розвивати інструкції ще більше. Інтерактивні елементи, такі як чат-боти чи вбудовані віртуальні асистенти, можуть надати користувачам персоналізовану допомогу в режимі реального часу. Додатково, мобільні додатки та розширені реальності можуть створити інтерактивне оточення для користувачів, спрощуючи взаємодію з продуктом [13].

Інструкції користувача стають все більше адаптованими до потреб різноманітних користувачів, використовуючи різноманітні методи подачі інформації. Зростання технологій відкриває нові можливості для поліпшення інструкцій, роблячи їх більш доступними, інтерактивними та користувацькозороїдненими. Важливо продовжувати розвивати цей аспект

технічної підтримки для забезпечення максимальної ефективності та зручності для всіх користувачів.

## 1.2 Мета розробки та область використання

Призначення розробки системи для дослідження ефективності впровадження системи для управління технічними інструкціями з можливістю відокремлення обов'язків адміністрування і користування полягає у створенні інформаційної платформи, призначеної для ефективного управління та поширення технічних інструкцій. Головними завданнями цієї системи є полегшення процесу створення, редагування, використанню та розповсюдження технічних інструкцій для різних продуктів, обладнання, систем та послуг.

Галузь застосування цієї системи широка і може включати наступні сфери:

1 Промисловість: У виробничих підприємствах, де існує потреба у документації для налаштування, обслуговування та ремонту обладнання та машин, система допоможе створювати та зберігати технічні інструкції для робітників та технічного персоналу.

2 Інформаційні технології: У сфері IT, система може бути використана для створення та поширення технічних інструкцій для розробників, адміністраторів систем та користувачів програмного забезпечення.

3 Медицина: В медичних установах система може бути використана для створення та розповсюдження технічних інструкцій щодо використання медичного обладнання та заходів безпеки.

4 Освіта: У навчальних закладах система може служити для створення навчальних матеріалів та інструкцій для студентів та викладачів.

5 Сфера обслуговування: У готельній галузі, ресторанах та інших закладах громадського харчування система може використовуватися для надання персоналу інструкцій щодо обслуговування гостей та дотримання стандартів обслуговування.

6 Сфера побуту: Зазвичай користувачі зустрічаються з технічними інструкціями в побутовій техніці, електроніці та інших галузях, і система може полегшити доступність та зрозумілість таких інструкцій.

Таким чином, система для створення та керування технічними інструкціями має широкий спектр застосувань і може бути корисною у будь-якій галузі, де потрібно ефективно управляти та поширювати технічну інформацію.

## 1.3 Основна інформація щодо предметної сфери

Мета цієї роботи полягає в розробці та впровадженні системи для ефективного управління технічними інструкціями з метою покращення їх доступності та користувацької зручності, а також відповідності стандартам та вимогам.

Для підготовки роботи «Дослідження ефективності впровадження системи для управління технічними інструкціями з можливістю відокремлення обов’язків адміністрування і користування» необхідно розглянути загальні відомості з предметної області, щоб визначити контекст і важливість обраної теми. Ось деякі ключові аспекти та загальні відомості, які можуть бути корисними:

1 Технічні інструкції:

Технічні інструкції є невід'ємною частиною виробництва та обслуговування різних видів обладнання, технічних систем та програмного забезпечення.

Вони містять інформацію щодо збирання, налаштування, експлуатації, обслуговування та ремонту продуктів і послуг.

2 Важливість технічних інструкцій:

Недостатньо чіткі або некоректні інструкції можуть призвести до небезпеки для користувачів, споживачів та технічного персоналу.

Правильні технічні інструкції сприяють безпеці, зменшують ризик нещасних випадків та сприяють ефективному використанню продуктів.

3 Роль систем для створення та керування технічними інструкціями:

Сучасні технології та програмне забезпечення дозволяють створювати, оновлювати, зберігати та поширювати технічні інструкції більш ефективно та зменшують ризик помилок.

Імплементація системи для управління технічними інструкціями може покращити доступність та зручність користування ними.

4 Функції адміністрування:

Адміністративні функції системи включають у себе створення, редагування, оновлення та видалення технічних інструкцій, а також керування доступом користувачів.

Адміністратори відповідають за забезпечення якості і актуальності інструкцій.

5 Функції звичайного користувача:

Звичайні користувачі системи використовують інструкції для вирішення конкретних завдань.

Їхні можливості можуть включати перегляд та розрахунок.

6 Технології та методи розробки:

Для розробки та імплементації такої системи можуть використовуватися різні технології, такі як десктоп-розробка, бази даних, системи управління версіями і багато інших.

Розгляд можливих архітектур та інструментів важливий для вибору оптимального підходу.

Ці загальні відомості можуть служити основою для подальшого дослідження та розробки магістерської роботи на обрану тему. Детальний аналіз і розгляд конкретних аспектів системи та її імплементації допоможе докладніше розкрити цю тему.

## 1.4 Постановка задачі

Темою роботи є «Дослідження ефективності впровадження системи для управління технічними інструкціями з можливістю відокремлення обов’язків адміністрування і користування».

Розробити програму для створення та керування технічними інструкціями. Програмна система передбачає:

– створення дизайну програми;

– імплементація функціоналу типів параметрів відповідно до можливих різновидів даних, а саме: таблиця, змінна, формула, зображення, текст;

– імплементація функціоналу пов’язаного з організацією структури інструкції та даних.

Детальний опис вимог користувача.

Створення дизайну програми:

– інтерфейс повинен бути інтуїтивно зрозумілим та легким у використанні для всіх категорій користувачів, включаючи адміністраторів та звичайних користувачів;

– легкий доступ до основних функцій системи, включаючи створення нової інструкції, пошук інструкцій, управління користувачами та інші опції;

– створення інструкції, можливість визначення даних для типу інструкції (таблиця, змінна, формула, зображення, текст);

– введення та редагування даних для кожного типу інструкції;

– пошук інструкцій за ключовими словами, категоріями та параметрами

– фільтрація та сортування результатів пошуку.

Імплементація функціоналу типів параметрів відповідно до можливих різновидів даних, а саме: таблиця, змінна, формула, зображення, текст:

– для типу «Таблиця» є можливість створення таблиць для представлення структурованих даних, додавання, редагування та видалення рядків та стовпців в таблицях, підтримка різних типів даних в таблицях;

– для типу «Змінна» є можливість створення змінних для зберігання числових значень, можливість встановлення значення змінної та його зміни в майбутньому, підтримка документації змінних (опис, одиниці вимірювання);

– для типу «Формула» є можливість введення математичних формул для обчислення значень, використання змінних та інших параметрів у формулах;

– для типу «Зображення» є завантаження та збереження зображень різних форматів, можливість попереднього перегляду;

– для типу «Текст» є введення текстової інформації для опису даних, параметрів чи інструкцій, можливість форматування тексту.

Імплементація функціоналу пов’язаного з організацією структури інструкції та даних:

– можливість створення заголовків, підзаголовків та розділів для структурування інструкцій;

– додавання текстового та графічного контенту до кожного розділу;

– переміщення, видалення та редагування розділів для зручної організації інформації;

– зберігання та організація даних різних типів (таблиця, змінна, формула, зображення, текст) відповідно до їх контексту;

– можливість вставки та редагування даних безпосередньо в інструкції;

– можливість зміни порядку розділів та параметрів в інструкції.

## 1.5 Проблематика відсутності технічних інструкцій

В багатьох виробничих та обслуговувальних середовищах часто спостерігається серйозна проблема відсутності належних технічних інструкцій. Це викликає ряд проблем, які безпосередньо впливають на ефективність та безпеку робочого процесу.

1 Збільшення Часу Навчання: Відсутність належних технічних інструкцій вимагає від нового персоналу витрачати значно більше часу на вивчення

процесів та виробничих задач. Це може призвести до затримок у виробництві та загальному падінню продуктивності.

2 Зростання Числа Помилок та Відмов: Відсутність чітких технічних настанов може призвести до великої кількості помилок та відмов у виробничому процесі. Робітники можуть неправильно розуміти послідовність дій або використовувати обладнання неправильно, що потенційно створює серйозні проблеми для якості продукції.

3 Небезпека для Працівників: В умовах відсутності належних інструкцій робочий персонал може опинитися в небезпеці через неправильне використання або обслуговування обладнання. Відсутність необхідних вказівок з питань безпеки може призвести до травм та нещасних випадків.

4 Обмежений Потенціал Оптимізації Процесів: Відсутність технічних інструкцій ускладнює впровадження нових технологій, методів виробництва чи обслуговування. Це стає обмеженням для оптимізації та підвищення ефективності виробничих процесів.

5 Втрата Досвіду та Експертизи: Без збереження та передачі досвіду відсутність технічних інструкцій може спричинити втрату експертизи внаслідок виходу з ладу обладнання або зміни складових виробничого процесу.

6 Зниження Мотивації Персоналу: Відсутність чітких інструкцій може призвести до розчарування та зниження мотивації працівників, особливо нових співробітників, які стикаються з труднощами у вивченні робочих процесів.

Висновок: Проблема відсутності належних технічних інструкцій стає значущим фактором, що впливає на якість та ефективність виробничих процесів, а також на безпеку працівників. Розробка та впровадження належних інструкцій стає важливим завданням для подолання цієї проблеми та покращення загальної продуктивності підприємства.

## 1.6 Ефективність інтерактивних технічних інструкцій в експерименті

1 Підготовка інтерактивних технічних інструкцій:

На цьому етапі команда розробників зосереджується на створенні інтерактивних технічних інструкцій, які включають детальні кроки для конкретного процесу виробництва чи обслуговування обладнання. Ці інструкції містять як текстові описи, так і графічні елементи, такі як ілюстрації, схеми та відеоматеріали.

2 Запланування сесії інструктажу:

Здійснюється докладне планування сесії інструктажу, де визначається не лише час і місце проведення, а й конкретні методи і техніки, які будуть використані для передачі інформації. Зокрема, враховується можливість взаємодії з експертами для додаткових пояснень.

3 Уведення персоналу в інтерактивні інструкції:

Важливий момент - комунікація з персоналом. Команда відділу навчання чітко пояснює працівникам важливість та переваги використання інтерактивних технічних інструкцій. Зокрема, розглядається можливість самостійного вивчення та практичного використання отриманої інформації.

4 Демонстрація використання інструкцій:

Ведучий сесії детально демонструє, як використовувати інтерактивні технічні інструкції. Акцент робиться на ключових елементах, відомостях про безпеку та інтерактивному взаємодії з матеріалом. Працівники отримують можливість самостійно спробувати взаємодіяти з інструкціями.

5 Групова взаємодія та обговорення:

Працівники залучаються до групових вправ та обговорень, використовуючи інтерактивні завдання. Групова взаємодія сприяє взаємоперевірці розуміння та обміну досвідом між колегами.

6 Збір зворотного зв'язку:

Збираються відгуки від працівників через анкетування, спостереження та відкритий обмін думками. Аналіз отриманих відгуків проводиться для визначення рівня задоволення та ефективності.

7 Коригування та оновлення:

Враховуючи зворотний зв'язок, команда розробників вносить корективи та оновлення до інтерактивних технічних інструкцій. Цей етап є постійним і дозволяє постійно вдосконалювати інструкції для максимальної ефективності та зручності використання.

Висновок з експерименту:

Отримані результати підтверджують, що використання інтерактивних технічних інструкцій суттєво полегшило навчання персоналу, покращило їх розуміння виробничих процесів та сприяло підвищенню загальної ефективності виробничого процесу. Зворотний зв'язок підтверджує високий рівень задоволення новим підходом, а ефективність додатку, оцінена на 0,8 балів, свідчить про його успішність.

# РОЗДІЛ 2
# ТЕХНОЛОГІЇ ТА ПРОГРАМИ ЯКІ БУЛИ ЗАДІЯНІ В РОЗРОБЦІ

## 2.1 Microsoft Visual Studio 2019

Visual Studio 2019 - інтегроване середовище розробки (IDE) від Microsoft, яке надає розширені можливості для розробки різноманітних програмних продуктів. Серед ключових характеристик Visual Studio 2019 можна відзначити підтримку різних мов програмування, таких як C#, Visual Basic, C++, F#, Python та інші. Інструменти налагодження, система контролю версій Git, а також засоби для розробки веб-застосунків на основі ASP.NET Core є вбудованими функціями середовища.

Також важливою особливістю є підтримка мобільної розробки за допомогою Xamarin, що дозволяє створювати переносні додатки для Android та iOS. Visual Studio 2019 інтегрується з хмарними послугами Azure, що полегшує використання хмарних ресурсів для розгортання та управління проектами.

Інтеграція з Docker дозволяє створювати, розгортати та керувати контейнеризованими додатками. Засоби аналізу коду та тестування роблять Visual Studio 2019 потужним інструментом для розробки високоякісного програмного забезпечення.

Visual Studio 2019 підтримує широкий спектр технологій, включаючи ті, що пов'язані зі штучним інтелектом та робототехнікою. Засоби для роботи з великими обсягами даних, інтеграція з Azure Machine Learning, а також інші інструменти для аналізу та обробки даних роблять Visual Studio 2019 актуальним в контексті сучасних технологій [9].

Серед інших важливих функцій можна відзначити підтримку розробки гри, інтеграцію з Unity для створення ігор різного роду, включаючи ті, що використовують технології віртуальної та доповненої реальності.

Загальною тенденцією є те, що Visual Studio 2019 намагається забезпечити розробникам універсальне та ефективне середовище для роботи над

проектами будь-якої складності та спрощує процеси розробки, тестування та розгортання програмного забезпечення.

Visual Studio 2019 є найпопулярнішим інтегрованим середовищем розробки для мови програмування C#. Дозволяючи розробникам та командам створювати різноманітні програми на основі .NET, Visual Studio 2019 надає потужні засоби для ефективного впровадження ідей у код [10].

Основні можливості для розробників C# включають:

Підтримка Останніх Версій C#: Visual Studio 2019 включає підтримку останніх версій мови C# і нововведень, що дозволяє розробникам використовувати сучасні можливості мови.

Відлагодження та Тестування: Інтегровані інструменти для налагодження дозволяють швидко виявляти та виправляти помилки в коді. Розширені засоби тестування сприяють створенню надійного коду.

Розширені Засоби Рефакторингу: Засоби для автоматизованого перетворення коду допомагають підтримувати високий стандарт читабельності та продуктивності в коді.

Підтримка Xamarin: Для розробників, які створюють мобільні додатки на платформах Android та iOS за допомогою Xamarin, Visual Studio 2019 надає інтегровану розробку та налагодження.

Інтеграція з Azure: Розширена підтримка для інтеграції з хмарними послугами Azure дозволяє легко підключати та взаємодіяти з хмарними ресурсами.

Підтримка ASP.NET Core: Розробка веб-додатків на базі ASP.NET Core відбувається в ефективному і продуктивному середовищі.

Visual Studio 2019 забезпечує розробників C# зручним та потужним інструментом для всіх етапів розробки, роблячи процес створення програм на мові C# швидким і ефективним [24].

## 2.2 SQLite

SQLite – це компактна та вбудовувана система керування базами даних (СКБД), яка визначається своєю легкістю використання та простотою в розгортанні. Однією з головних особливостей SQLite є те, що вона не вимагає окремого серверу та працює безпосередньо з файлами бази даних. Це робить SQLite ідеальним вибором для невеликих проектів, мобільних додатків та сценаріїв, де важлива простота та ефективність [2].

SQLite підтримує багато мов програмування, включаючи C, C++, Java, Python, PHP, і багато інших. Це дозволяє розробникам використовувати SQLite в різноманітних середовищах та мовах програмування.

Основні особливості SQLite включають в себе простий та інтуїтивно зрозумілий SQL-синтаксис, вбудовану підтримку транзакцій для забезпечення цілісності даних, широкі можливості індексації для оптимізації запитів та підтримку тригерів для автоматизації деяких операцій бази даних.

SQLite добре впилинається в обмежені умови, такі як обмежені ресурси мобільних пристроїв, і в той же час є достатньо гнучким для використання у більших проектах, коли потрібна легко розгортана та невибаглива до ресурсів база даних [6].

SQLite підтримує стандартний SQL-синтаксис, що робить його легко засвоюваним для багатьох розробників. Вона також володіє високою ефективністю завдяки використанню транзакцій, що забезпечує цілісність даних. Можливості індексації та використання тригерів дозволяють оптимізувати запити та автоматизувати деякі операції.

Однією з переваг SQLite є його широка підтримка різних мов програмування, що робить його вибором для різних платформ та проектів. Від мобільних додатків до вбудованих систем, SQLite забезпечує ефективну та надійну роботу з базами даних, не навантажуючи систему великими ресурсами [7].

SQLite також славиться своєю високою портативністю та низькими вимогами до ресурсів, що робить його ідеальним вибором для використання в різних сценаріях, включаючи вбудовані системи, мобільні додатки, додатки Інтернету речей (IoT) та інші проекти з обмеженими ресурсами [8].

Однією з важливих особливостей є те, що SQLite підтримує транзакції ACID (Atomicity, Consistency, Isolation, Durability), що забезпечує надійність та цілісність даних в обличчі можливих помилок або відмов.

Також слід зазначити, що SQLite є відкритим програмним забезпеченням і безкоштовно розповсюджується під ліцензією Public Domain, що робить його доступним для використання без обмежень із збереженням власності та безкоштовно для комерційного використання.

Узагальнюючи, SQLite є потужною, легкою у використанні та надійною системою керування базами даних, яка добре впилинається в широкий спектр застосувань, зокрема в умовах обмежених ресурсів [22].

## 2.3 WPF

Windows Presentation Foundation (WPF) - це технологія розробки віконних додатків для операційної системи Windows, яка входить в склад платформи .NET Framework. WPF надає розробникам потужний та гнучкий інструментарій для створення сучасних, стильних та візуально здатних додатків з використанням мови програмування C# або XAML (eXtensible Application Markup Language).

Основні характеристики WPF включають в себе:

Декларативний Синтаксис: Використання мови розмітки XAML для декларативного визначення інтерфейсу користувача, що полегшує розробку та підтримку коду.

Розділення Логіки та Представлення: WPF дозволяє розділити логіку додатку від його представлення, що полегшує управління та розширення коду.

Стилі та Шаблони: Застосування стилів та шаблонів для стандартизації вигляду елементів у інтерфейсі користувача та забезпечення повторного використання коду.

Графічні та Анімаційні Можливості: Підтримка 2D та 3D графіки, а також можливостей анімації для створення більш динамічних та привабливих інтерфейсів.

Data Binding: Механізм прив'язки даних, що дозволяє зв'язувати дані додатку з елементами інтерфейсу користувача, спрощуючи управління даними та їх відображення.

Підтримка Клавіатури та Миші: Інтегрована підтримка взаємодії з користувачем через клавіатуру та мишу.

WPF використовує концепцію візуальної дерева для побудови інтерфейсу, що дозволяє розробникам легко керувати відображенням та взаємодією елементів додатку. Ця технологія дозволяє створювати багатошарові та естетично привабливі додатки для операційної системи Windows [4].

Модель MVVM (Model-View-ViewModel): WPF активно використовує патерн MVVM для розділення логіки бізнес-логіки (Модель), представлення (View) і управління подіями та взаємодією з користувачем (ViewModel). Це полегшує тестування та розширення додатків.

Керування Станом та Анімації: WPF надає зручні інструменти для управління станом елементів інтерфейсу та створення анімацій. Це включає в себе можливість визначати різні стани, реагувати на події та змінювати вигляд елементів за допомогою анімаційних ефектів.

Підтримка Стилів та Ресурсів: Стилі та ресурси дозволяють розробникам одноразово визначати властивості елементів інтерфейсу та використовувати їх повторно, що спрощує управління виглядом додатків та забезпечує їхню консистентність.

Підтримка Інклюзивного Дизайну: WPF надає можливості для створення інклюзивних інтерфейсів, спрощуючи взаємодію з користувачами з різними

потребами, включаючи підтримку великого шрифту, високої контрастності та інше.

Гнучкість в Побудові Користувацьких Елементів: WPF дозволяє розробникам створювати власні користувацькі елементи у вигляді власних управляючих елементів та панелей, розширюючи стандартний набір елементів для досягнення унікального дизайну та функціональності [17].

Узагальнюючи, WPF є потужним інструментом для розробки візуально здатних та функціональних додатків для операційної системи Windows, надаючи широкі можливості для взаємодії з користувачем та ефективного управління даними.

## 2.4 Model-View-ViewModel

Model-View-ViewModel (MVVM) - це архітектурний патерн, який використовується у розробці програмного забезпечення для розділення бізнес-логіки (Модель), представлення (Вид) та управління подіями та взаємодією з користувачем (ViewModel).

У цьому патерні Модель представляє бізнес-логіку та дані, Вид відповідає за візуальне представлення даних та реагує на користувацькі події, а ViewModel служить посередником між Моделлю та Видом. ViewModel обробляє запити користувача, виконує необхідні операції з Моделлю та надає дані для представлення у Виді [14].

MVVM дозволяє створювати код, який є більш повторюваним та легше тестується, оскільки він дозволяє відокремити бізнес-логіку від візуального представлення та логіки взаємодії з користувачем. Цей підхід сприяє вирішенню проблем, таких як надмірна залежність коду від інтерфейсу користувача та ускладнення тестування.

MVVM широко використовується в розробці програмного забезпечення для платформи .NET, зокрема в технології Windows Presentation Foundation

(WPF) та Universal Windows Platform (UWP), а також в інших сучасних фреймворках [15].

Модель (Model): Відповідає за бізнес-логіку та роботу з даними. Це може бути представлення реальних об'єктів домену, які використовуються в додатку.

Вид (View): Відображає інтерфейс користувача та візуально представляє дані. Основне завдання - відображення інформації та взаємодія з користувачем.

ViewModel: Це проміжний шар між Моделлю та Видом. ViewModel обробляє логіку взаємодії та підготовки даних для представлення, а також слідкує за змінами в Моделі та оновлює Вид.

Одна з ключових особливостей MVVM - це використання прив'язок даних. ViewModel використовує механізм прив'язок, щоб зв'язати дані з Моделі з елементами Виду. Це дозволяє автоматично оновлювати інтерфейс користувача при зміні даних в Моделі без прямого втручання з боку користувача [16].

MVVM робить код більш підтримуваним та розширюваним, полегшуючи тестування та розробку. Цей патерн особливо популярний у розробці застосунків для платформ .NET, таких як WPF, Xamarin, та інші.

## 2.5 Мова C#

Мова програмування C# (C Sharp) представляє собою об'єктно-орієнтовану мову, розроблену Microsoft для платформи .NET. Основними рисами C# є зручний синтаксис, великий фокус на безпеці та висока ефективність. Вона стала ключовим інструментом для розробки різноманітних додатків, включаючи ті, що використовують технології Visual Studio 2019, SQLite, WPF та патерн MVVM у даному проекті.

C# дозволяє легко вивчати та швидко розробляти програми завдяки своєму інтуїтивно зрозумілому синтаксису. Вона підтримує об'єктно-орієнтований підхід до програмування, що сприяє створенню структурованих та легко розширюваних додатків. Завдяки активному розвитку та великій спільноті

розробників, C# залишається потужним інструментом у світі програмування, допомагаючи створювати надійне та ефективне програмне забезпечення [3].

C# відзначається кількома ключовими перевагами, які роблять її незамінним елементом у розробці програмного забезпечення:

– швидкість та ефективність: компіляція в машинний код та оптимізації в C# дозволяють досягти високої швидкодії виконання програм, що робить мову ідеальною для розробки продуктивних додатків.

– широкі можливості .NET: C# є основною мовою для роботи з платформою .NET, що відкриває доступ до широкого спектру бібліотек та інструментів для розробки різноманітних застосунків.

– об'єктно-орієнтований підхід: заснований на принципах об'єктно-орієнтованого програмування, C# сприяє створенню модульних та легко управляємих програмних структур.

– інтеграція з іншими технологіями: C# надає зручний інтерфейс для інтеграції з іншими технологіями, такими як бази даних SQLite, що робить його відмінним вибором для комплексних проектів.

У контексті даного проекту C#, як мова програмування, не лише допомагає створювати функціональний та надійний код, а й є фундаментом для інтеграції різноманітних технологій, що забезпечує успішну реалізацію програмного забезпечення [5].

C# вирізняється своєю гнучкістю та здатністю до розширення. Як основна мова для розробки Windows додатків, вона забезпечує широкі можливості для створення інтерактивних та сучасних інтерфейсів користувача.

Завдяки розгалуженій екосистемі бібліотек та фреймворків, розробники можуть легко розширювати функціонал своїх додатків, ефективно використовуючи готові рішення. Можливість зручно обробляти події та взаємодіяти з користувачем робить C# ідеальним вибором для розробки програм, які вимагають високого рівня гнучкості та розширюваності [23].

C# визначається своєю інноваційністю та високою продуктивністю в сфері розробки програмного забезпечення. Як ключовий інструмент у реалізації проектів, вона відкриває перед розробниками ряд переваг:

– сучасні інструменти розробки: з введенням інтегрованого середовища Visual Studio, розробники мають доступ до потужних інструментів для створення, тестування та налагодження додатків.

– ефективне використання ресурсів: мова C# володіє оптимізацією та ефективністю виконання коду, що робить її відмінним вибором для великих та ресурсомістких проектів.

– інтеграція та злагодженість: C# ідеально поєднується з іншими технологіями, такими як бази даних SQLite та патерн MVVM, забезпечуючи злагодженість та безпроблемну інтеграцію [25].

У світі сучасної розробки програмного забезпечення C# залишається не лише стандартом, але й лідером завдяки своїм інноваційним можливостям та високій продуктивності.

## 2.6 Висновки

Застосування технологій Visual Studio 2019, C#, SQLite, WPF та патерну MVVM у розробці програмного забезпечення виявляється дуже ефективним та потужним підходом. Visual Studio 2019, як інтегроване середовище розробки, забезпечує зручність та продуктивність у процесі написання коду, тестування та налагодження додатків.

Мова програмування C# надає високий рівень абстракції та об'єднує в собі елементи ефективності та зручності у синтаксисі. Це дозволяє розробникам швидко писати якісний код, спрощує його розуміння та утримання. Крім того, C# підтримує об'єктно-орієнтований підхід, що сприяє структуруванню програм та полегшує їхнє розширення.

Застосування мови C# в поєднанні з іншими технологіями стало ключовим фактором у створенні високоякісного та функціонального

програмного забезпечення. Ця комбінація не лише сприяє продуктивності та легкості розробки, а й відкриває додаткові можливості для розширення та покращення продукту в майбутньому.

SQLite, як легка та компактна система керування базами даних, є ідеальним вибором для проектів різного роду, зокрема для мобільних додатків та проектів з обмеженими ресурсами. Використання WPF дозволяє створювати сучасні та естетично привабливі інтерфейси користувача для операційної системи Windows.

Патерн MVVM в контексті розробки з Visual Studio 2019, SQLite та WPF дозволяє ефективно розділити бізнес-логіку, представлення та логіку взаємодії з користувачем, що полегшує тестування, управління кодом та робить програмний код більш підтримуваним.

Загальний висновок полягає в тому, що комбінація Visual Studio 2019, SQLite, WPF та MVVM забезпечує потужний та гнучкий інструментарій для створення високоякісних, продуктивних та легко підтримуваних додатків для операційної системи Windows.

# РОЗДІЛ 3
# СТРУКТУРА ПРОГРАМИ, РЕАЛІЗАЦЯ ТА ПРИКЛАД ВИКОРИСТАННЯ

## 3.1 Загальна модель програми

Додаток має назву CriCrinCtor та складається з деяких основних модулів та додаткової бібліотеки яка була написана в рамках даного проекту:

– CriCrinCtor – додаток яким буде користуватись клієнт в процесі експлуатації, він був написаний на основі патерну MVVM. До нього входять розділи View де розташовані усі вікна які відображають інформацію, Model зберігає класи сутностей з властивостями, також є ViewModel це частина яка відповідає за обробку даних між базою даних та користувацькою частино. Ще мається частина яка відповідає за спілкування між базою даних та папка з ресурсами де зберігаються рисунки, іконки та файли які використовуються для мультимовності;

– WebUpdate – це бібліотека яка була написана в рамках даного проекту для перевірки нової версії на сервері та завантаження її.

## 3.2 Огляд модуля CriCrinCtor

Модуль CriCrinCtor являє собою головний проект який реалізує патерн MVVM при цьому реалізує усі необхідні частини. Структура файлу виглядає наступним чином:

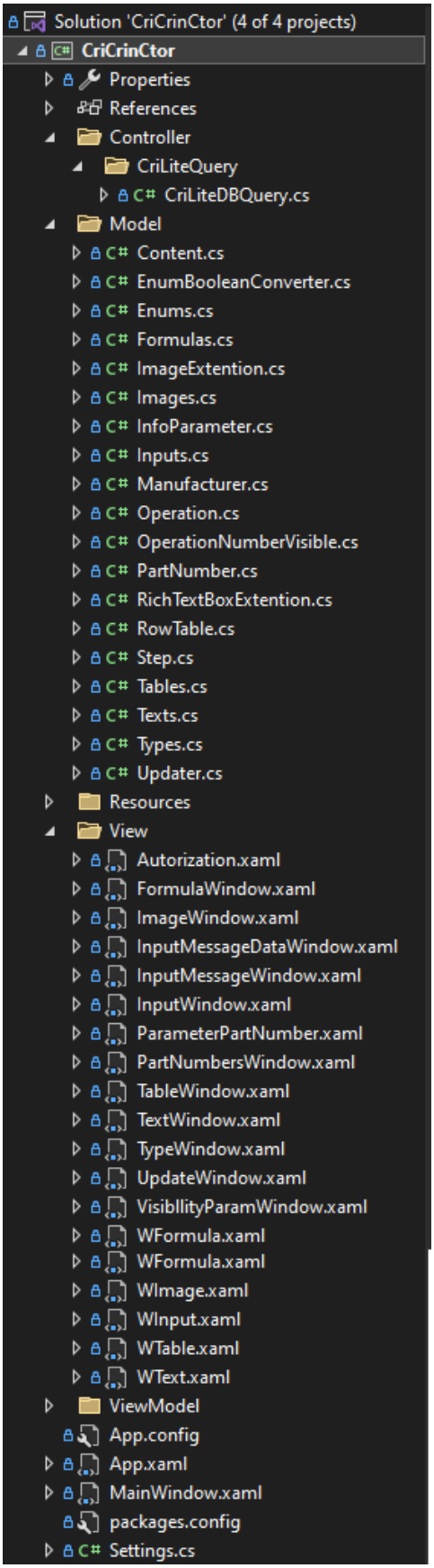


Рис. 3.1. Структура модулю CriCrinCtor

Папка Model містить наступні файли:

1 Content.cs – клас для формування певного контейнеру з певним змістом;

2 EnumBooleanConverter.cs – клас для конвертування радіо-кнопки до списку перелік різних константних значень;

3 Enums.cs – клас для списку перерахувань;

4 Formulas.cs – клас для зберігання інформації щодо типу Формула який використовується при відображені даного компоненту;

5 ImageExtention.cs – клас який містить логіку для доповненої роботи із рисунками;

6 Images.cs – клас для зберігання інформації щодо типу Рисунок який використовується при відображені даного компоненту;

7 InfoParameter.cs – клас який зберігає інформацію про певний код інструкції;

8 Inputs.cs – клас для зберігання інформації щодо типу Змінна який використовується при відображені даного компоненту;

9 Manufacturer.cs – клас для категорії;

10 Operation.cs – клас для основних пунктів інструкції;

11 OperationNumberVisible.cs – клас для правильного порядку відображення;

12 PartNumber.cs – клас для коду інструкції;

13 RichTextBoxExtention.cs – клас який містить логіку для доповненої роботи із текстом;

14 RowTable.cs – клас який містить логіку для доповненої роботи із таблицями;

15 Step.cs – клас для підпунктів інструкції;

16 Tables.cs – клас для зберігання інформації щодо типу Таблиця який використовується при відображені даного компоненту;

17 Texts.cs – клас для зберігання інформації щодо типу Текст який використовується при відображені даного компоненту;

18 Types.cs – клас який містить дані про тип інструкції;

19 Updater.cs – клас який містить дані для роботи із бібліотекою WebUpdate.

Папка View містить наступні файли;

1 Autorization.xaml – це вікно для авторизації адміністратора;

2 FormulaWindow.xaml – це наша візуальна частина, яка відповідає за створення або редагування контейнеру типу Формула в якому також можна задати режим відображення;

3 ImageWindow.xaml – це наша візуальна частина, яка відповідає за створення або редагування контейнеру типу Рисунок в якому також можна задати режим відображення;

4 InputMessageDataWindow.xaml – це вікно для створення параметрів до коду інструкції;

5 InputMessageWindow.xaml – це вікно для введення значення для певного функціоналу;

6 InputWindow.xaml – це наша візуальна частина, яка відповідає за створення або редагування контейнеру типу Зміна в якому також можна задати режим відображення;

7 ParameterPartNumber.xaml – це візуальна частина яка відповідає за параметри коду інструкції;

8 PartNumbersWindow.xaml – це одне з головних вікон де ми обираємо тип інструкції, категорію, код, можемо подивитися параметри, де можна побачити останні вибрані коди інструкції, змінити мову, перевірити оновлення та місце де може авторизуватися адміністратор;

9 TableWindow.xaml – це наша візуальна частина, яка відповідає за створення або редагування контейнеру типу Таблиця в якому також можна задати режим відображення;

10 TextWindow.xaml – це наша візуальна частина, яка відповідає за створення або редагування контейнеру типу Текст в якому також можна задати режим відображення;

11 TypeWindow.xaml – це візуальна частина, де оброблюються дані про тип інструкції;

12 UpdateWindow.xaml – це вікно для зміни видимості параметру для коду інструкції;

13 VisibllityParamWindow.xaml – це наша візуальна частина, яку бачить користувач;

14 WFormula.xaml – частина візуального контейнеру який використовується для відображення типу Формула;

15 WImage.xaml – частина візуального контейнеру який використовується для відображення типу Рисунок;

16 WInput.xaml – частина візуального контейнеру який використовується для відображення типу Зміна;

17 WTable.xaml – частина візуального контейнеру який використовується для відображення типу Таблиця;

18 WText.xaml – частина візуального контейнеру який використовується для відображення типу Текст.

Папка ViewModel містить наступні файли:

1 BaseViewModel.cs – базовий клас для усіх view model, аби у випадку створення нових сторінок не повторювати код, також він містить функціонал який оброблює поведінку при різних діях користувача, а також основна логіка обробки даних;

2 Ще певні класи для обробки даних, але вже для більше спеціальних частинах проекту.

Також мається клас CriLiteDBQuery який спілкується між додатком та базою даних, в ньому виконуються різні запити, CRUD операції та інші специфічні запити завдяки котрим працює додаток.

Для зміни мови використовуються додаткові файли, рисунки та іконки які зберігаються в папці Resources.

Окремо зберігаються конфігураційні файли в яких налаштовуються бібліотеки та файли для мультиновності. Також мається файл MainWindow.xaml який відображає інструкцію користувача, яка збирається з контейнерів.

У рамках даного проекту були розроблені та впроваджені різноманітні компоненти, які забезпечують основні функції та дії. Кожен з них відповідає за конкретний аспект редактора інтерактивних технічних інструкцій.

1 Рисунок:

– завантаження та відображення: компонент для роботи з рисунками дозволяє користувачам завантажувати та відображати графічні елементи у текстових документах;

– інтерактивність: реалізовано можливість масштабування, переміщення та інші інтерактивні дії з рисунками.

2 Текст:

– форматування та редагування: цей компонент дозволяє вводити та редагувати текстові елементи інструкцій. Забезпечено можливості форматування, вирівнювання та вибору стилів тексту;

– вставка елементів: інтеграція з іншими компонентами, такими як Рисунок, для вставки їх у текстові блоки.

3 Таблиця:

– створення та редагування: компонент для роботи з таблицями, який дозволяє користувачам створювати та редагувати таблиці з необхідною кількістю рядків та стовпців;

– форматування: забезпечено можливості форматування таблиць, встановлення ширини стовпців, вирівнювання тексту тощо.

4 Формула:

– математичні вирази: компонент для введення та відображення математичних формул;

– інтеграція з текстом: можливість вставки формул в текстові блоки інструкцій.

5 Зміна:

– відстеження змін: цей компонент відповідає за відстеження змін у документах. Забезпечено історію редагувань та можливість відновлення попередніх версій;

– колаборативність: додатково, компонент може підтримувати колаборативну роботу, де декілька користувачів можуть одночасно вносити зміни в документ.

Ці компоненти в сукупності створюють потужний та зручний редактор інтерактивних технічних інструкцій, забезпечуючи широкий функціонал для користувачів у створенні та редагуванні контенту.

## 3.3 Огляд модуля WebUpdate

Модуль WebUpdate являє собою бібліотеку яка оновлює систему. Структура файлу виглядає наступним чином:

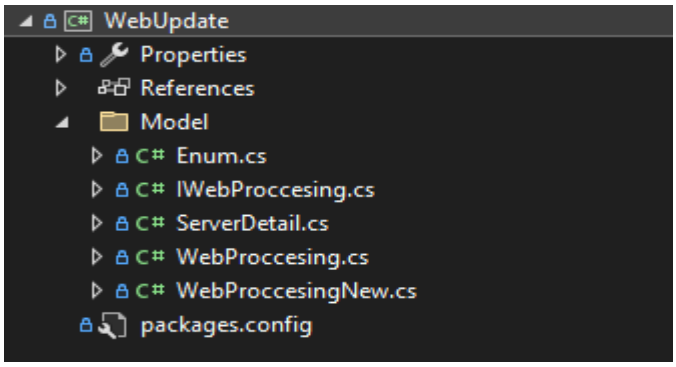


Рис. 3.2. Структура модулю WebUpdate

Ця бібліотека складається з таких модулів:

1 Enum.cs – конфігураційний перелік для оновлення;

2 IWebProccesing.cs – інтерфейс для організації роботи класів які відповідають за оновлення;

3 ServerDetail.cs – модель даних для оновлення системи;

4 WebProccesing.cs – клас який відповідає за оновлення та завантаження даних;

5 WebProccesingNew.cs – клас який відповідає за оновлення та завантаження даних, але з покращеною обробкою.

### 3.4 Способи використання

Для користувача був спеціально створений дизайн. Хочу зауважити, що даний проект вже використовується підприємством для навчання своїх співробітників та продажу створеної інструкції іншим споживачам. Тому даний спосіб користування буде описувати інструкцію яка користується попитом.

Давайте розглянемо головний екран з якого починається наш опис. Це буде сторінка де зразу можна побачити технічну інструкцію, це зроблено для того щоб не потрібно було кожен раз при відкриті програми вибирати потрібний код інструкції, також зберігає останній розділ та підрозділ інструкції.

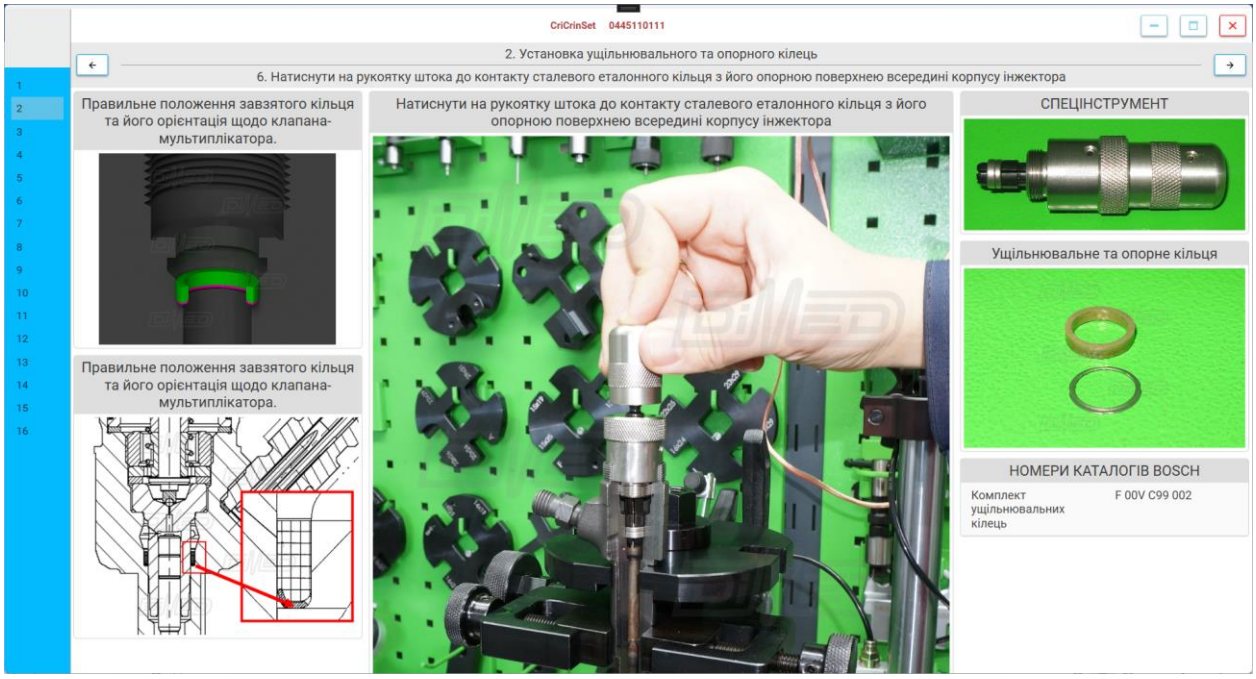


Рис. 3.3. Головне вікно інструкції

Також головний екран був умовно розділено на 3 частини: Ліва, Права, Головна орієнтація. Коли курсор миші пересовується у кордони будь-якої частини орієнтації то вона розширюється, а інші становляться менше.

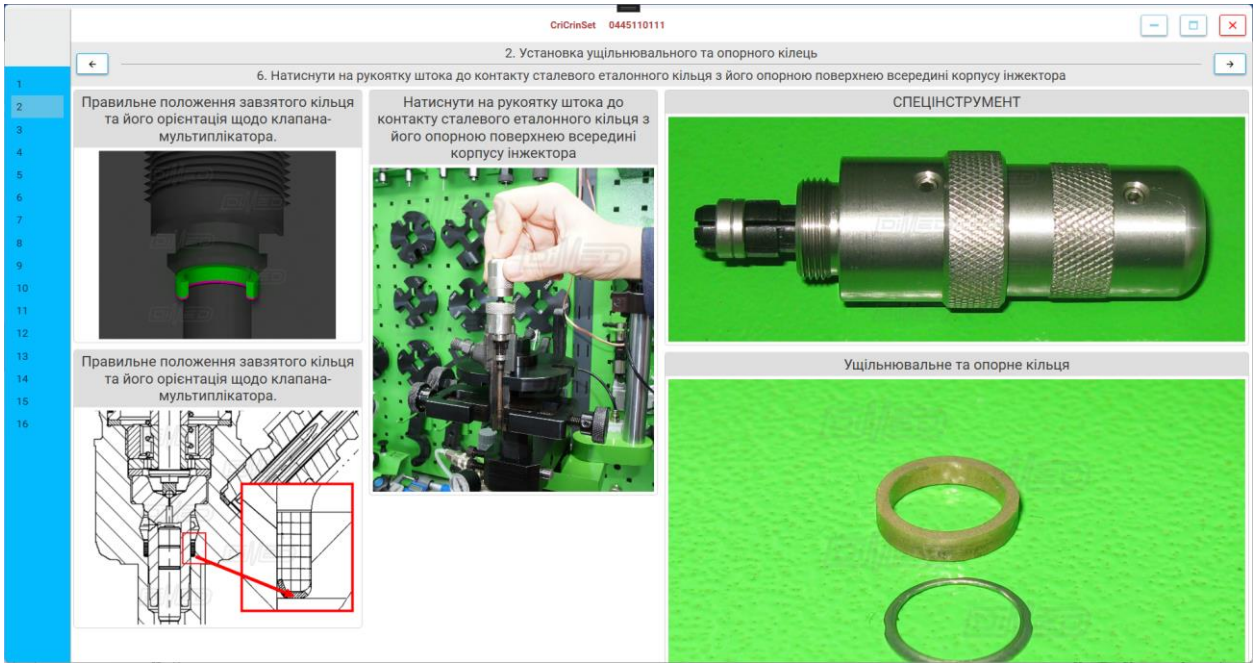


Рис. 3.4. Розширення Правої частини орієнтації

У висуваючому меню на лівій частині екрана можемо побачити усі пункти та підпункти інструкції, на рисунку 3.5. можна побачити підпункти та кнопку виходу до головних пунктів. Також мається 2 кнопки, які відповідають за початок інструкції заново, а також вибору нового коду інструкції та інших налаштувань проекту.

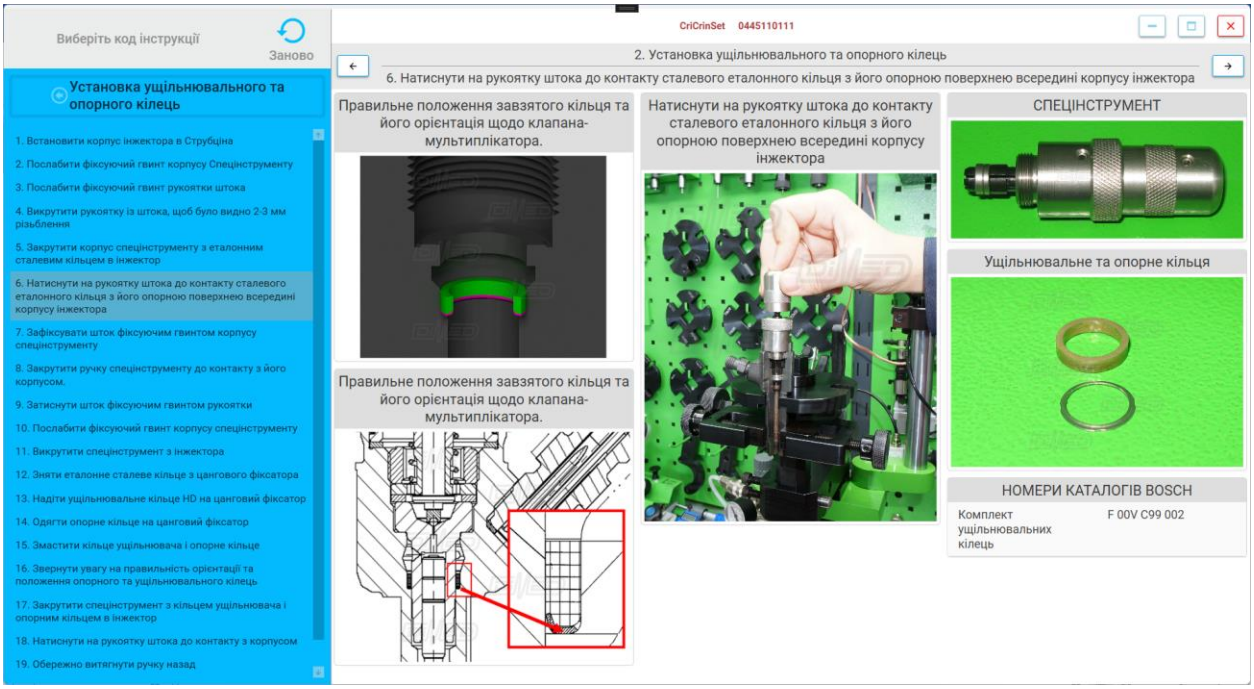


Рис. 3.5. Висуваюче меню на головному вікні

На головному меню для адміністратора є кнопки для переміщення, редагування та створення контейнерів для інструкції. Воно може з’явитися по натиску правого кліку.

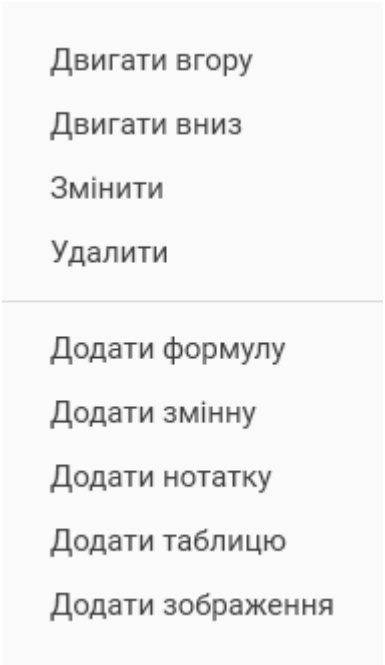


Рис. 3.6. Контекстне меню для редагування інструкції

На рисунку 3.3. можна побачити оригінальну сторінку, а на даному рисунку 3.7. було переміщено в лівій частині зображення вниз, також переміщено таблицю вверх яка знаходиться справа, також було змінено надпис для зображення, більш того було додано новий текстовий контейнер.

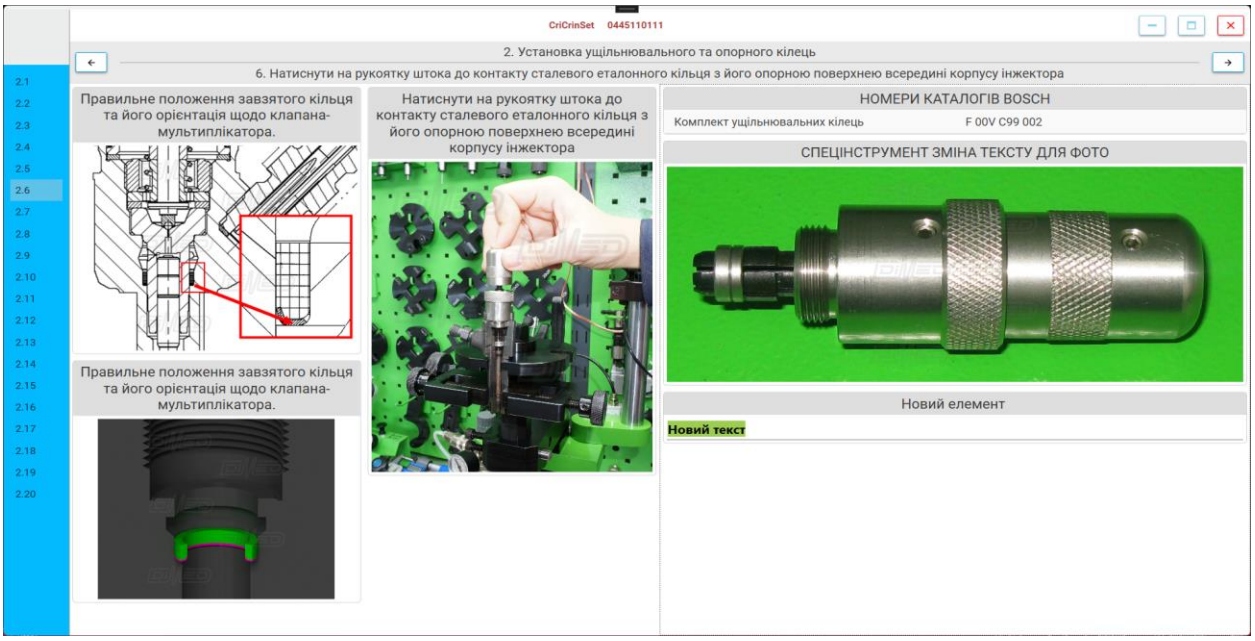


Рис. 3.7. Розширення Правої частини орієнтації

Зазначимо, що дана програма являє собою мультимовний застосунок і за основу була взята інтернаціональна англійська мова. Інструкція пишеться через поточний язик додатку, якщо було обрано англійську мову, то дані з цієї мови будуть відображатись для інших мов якщо там не була прописана інструкція для даного контейнера, тому що вона являє собою основну мову. Інші мови не перевизначать мову інших даних.

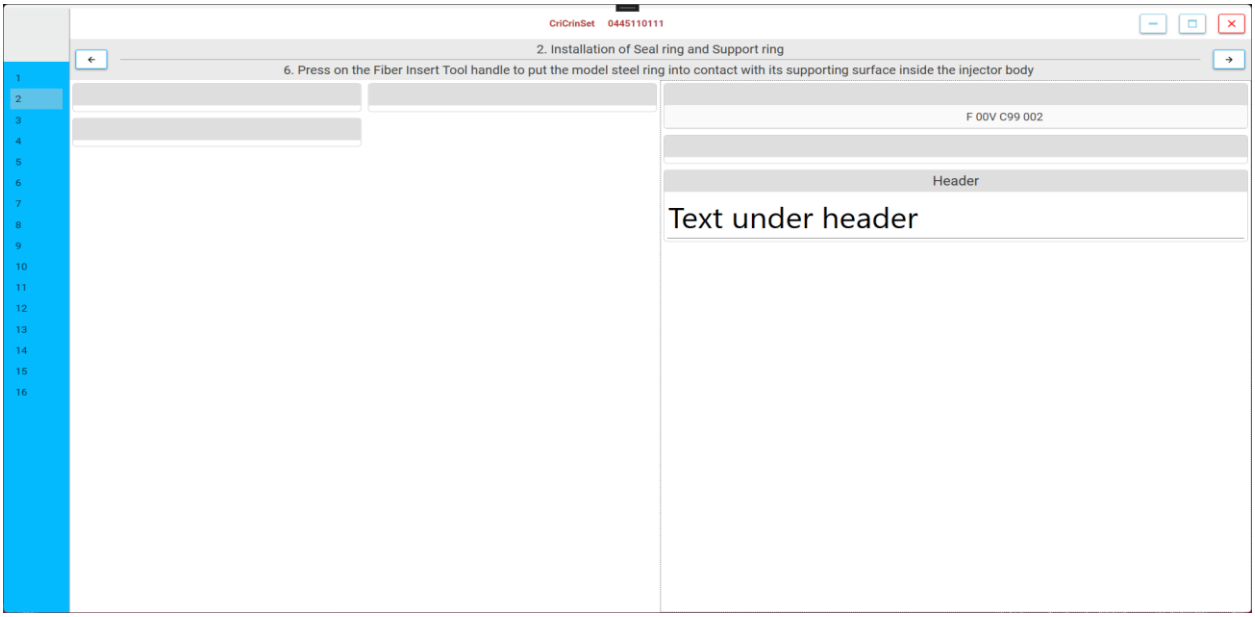


Рис. 3.8. Створення контейнеру в англомовній частині проекту

На рисунку 3.9. буде змінена мова на японську і ми можемо побачити, що при зміні мови перевизначається інформація яка була задана в англомовній версії проекту.

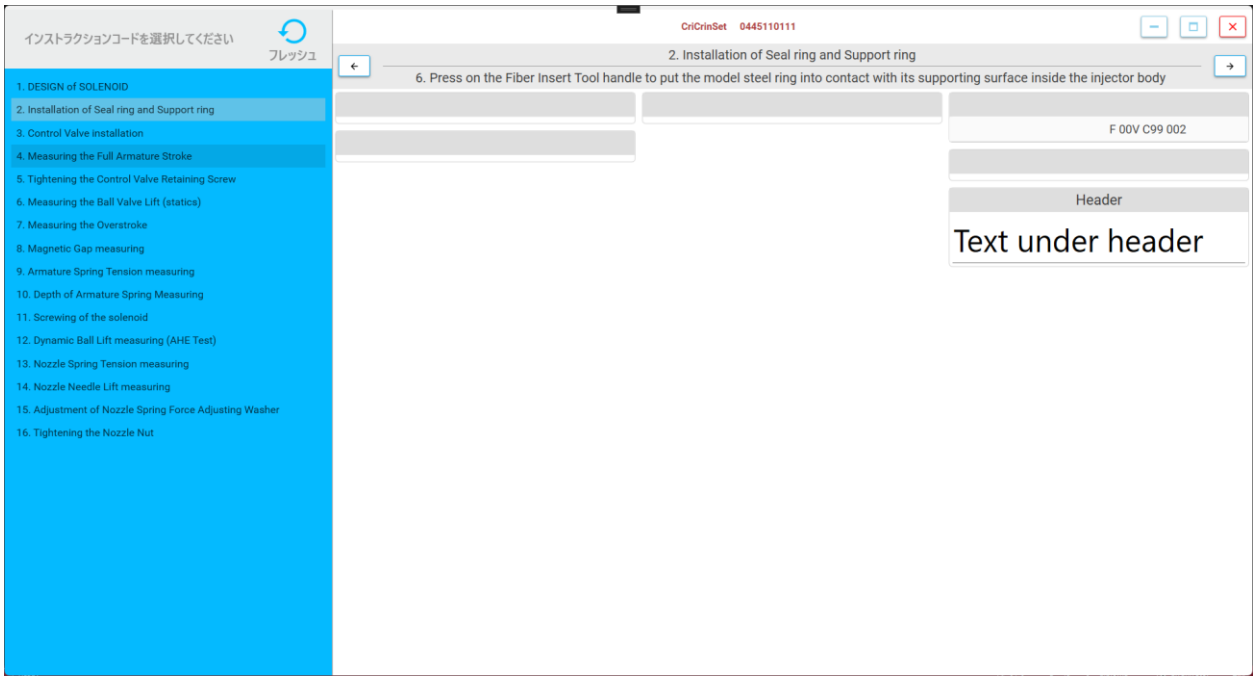


Рис. 3.9. Японська версія додатку

Коли адміністратор хоче додати новий контейнер він обирає пункт із контекстного меню, на рисунках 3.10-14 буде зображені усі вікна які використовуються для створення контейнеру. Для всіх контейнерів можна заповнити надпис, а також є обов’язковим вибір його розташування та режим відображення: Операція чи Крок, тобто контейнер який відображається на протязі розділу чи під підрозділу також для кожного конструктора контейнера використовується індивідуальне налаштування, в залежності від типу.

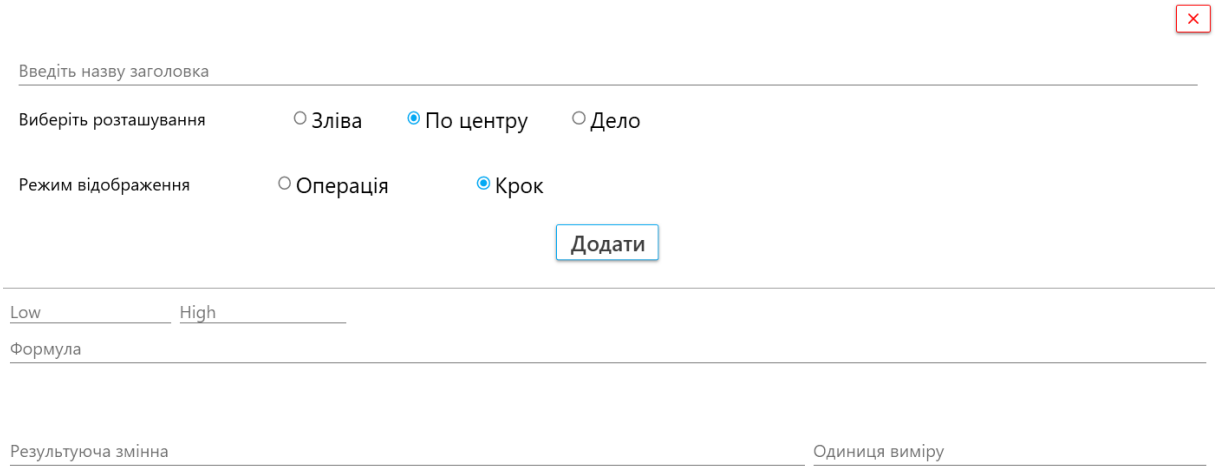


Рис. 3.10. Японська версія додатку

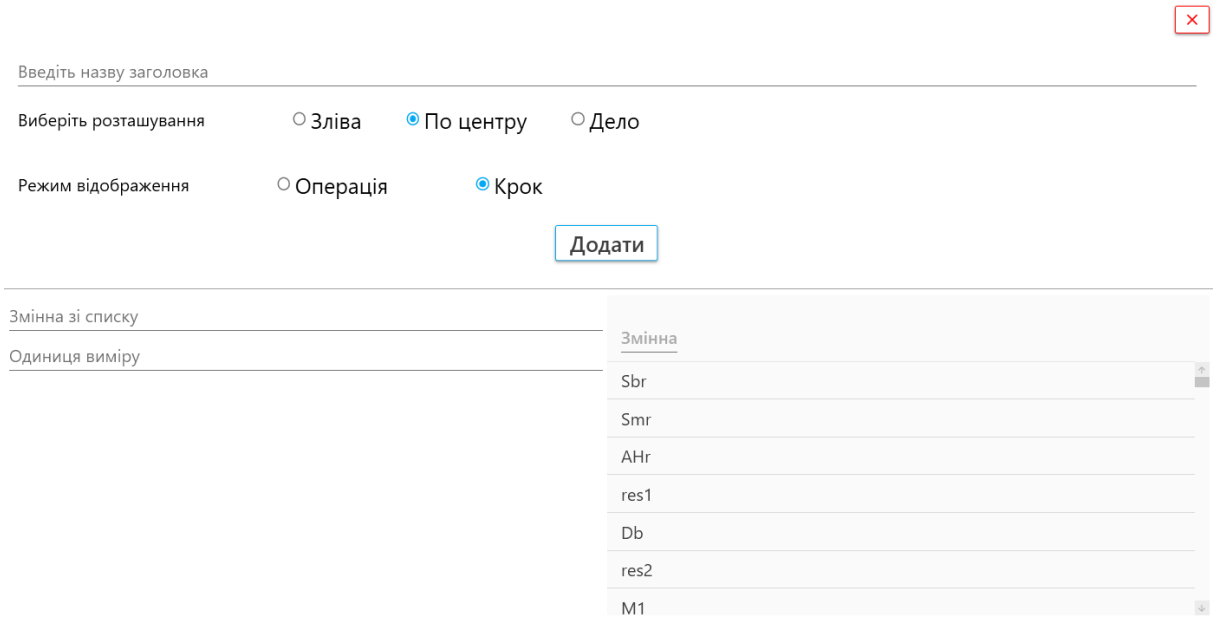


Рис. 3.11. Японська версія додатку

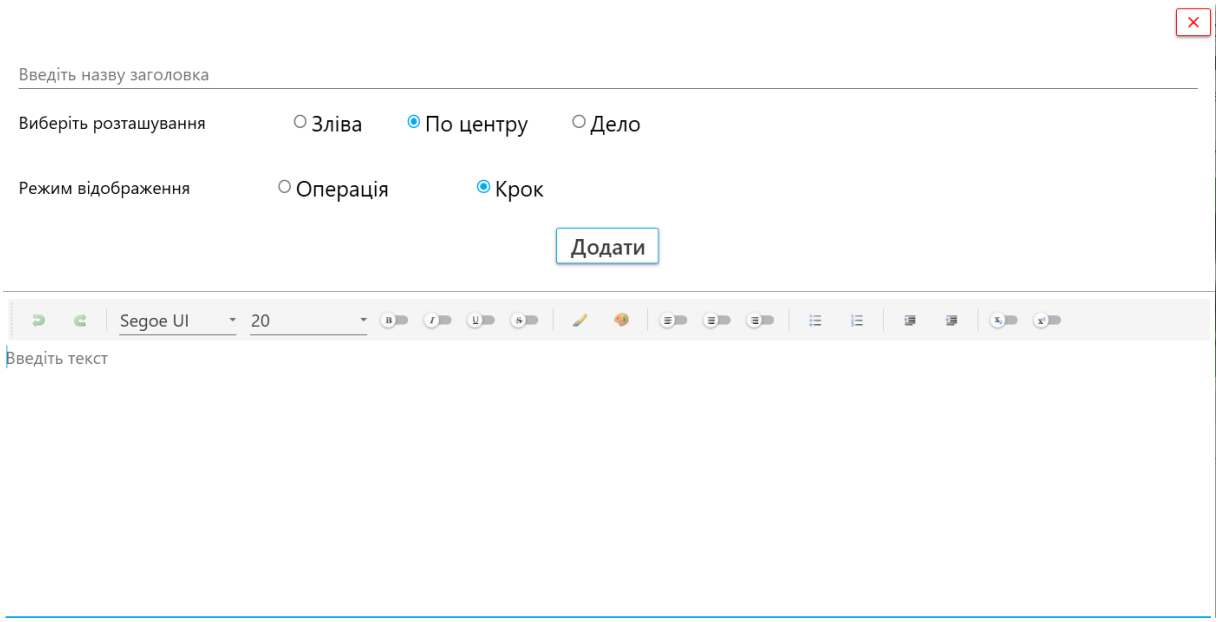


Рис. 3.12. Японська версія додатку

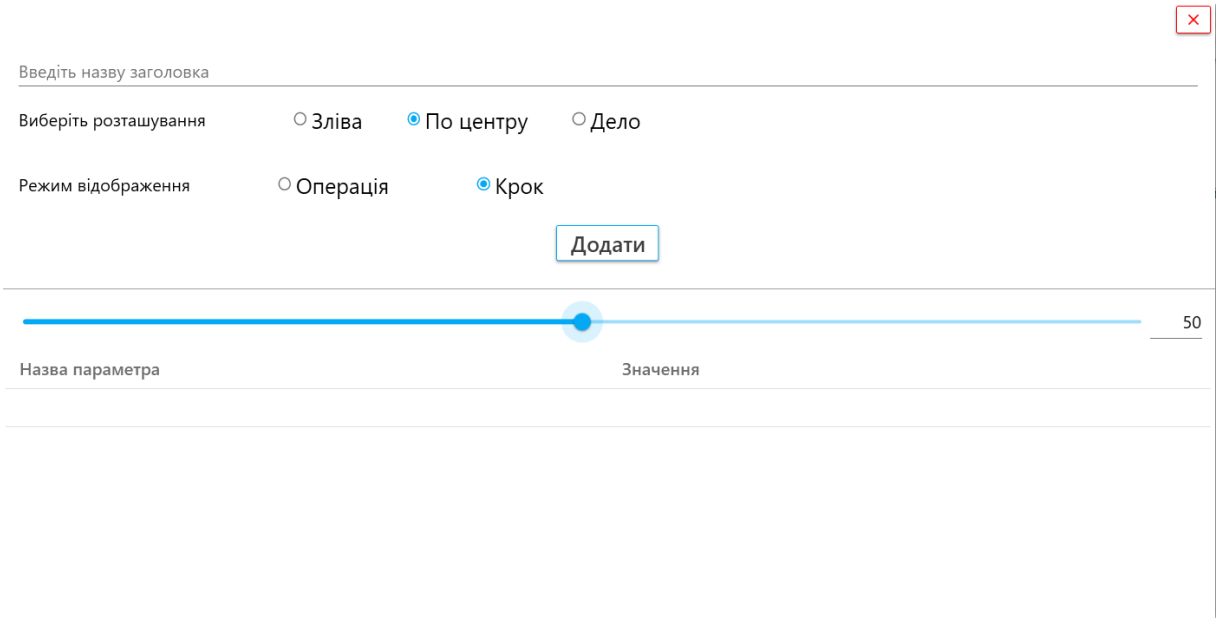


Рис. 3.13. Японська версія додатку

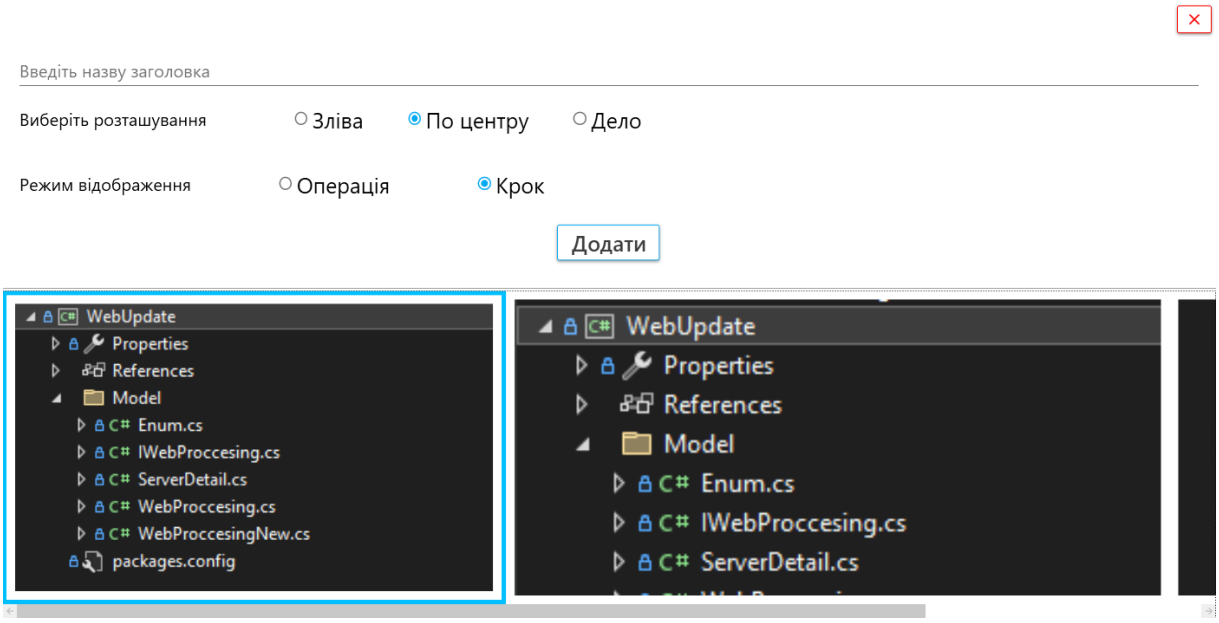


Рис. 3.14. Японська версія додатку

Зараз ми зможемо побачити основне вікно де зможемо побачити місце де зберігається категорія, код інструкції який підпорядковується категорії та тип інструкії який використовується як шаблон для інструкцій. Також можна переглянути останні коди які були використані та переглянути їх. Для кожного з коду інструкції мається набір параметрів які можуть використовуватись пізшіше у формулах, або для відображення. Зверху мається кнопки для зміни мови,

оновлення проекту, але якщо світиться позначка зеленого кольору це буде означати, що маэться оновлення. Для адміністратора мається вікно для авторизації.

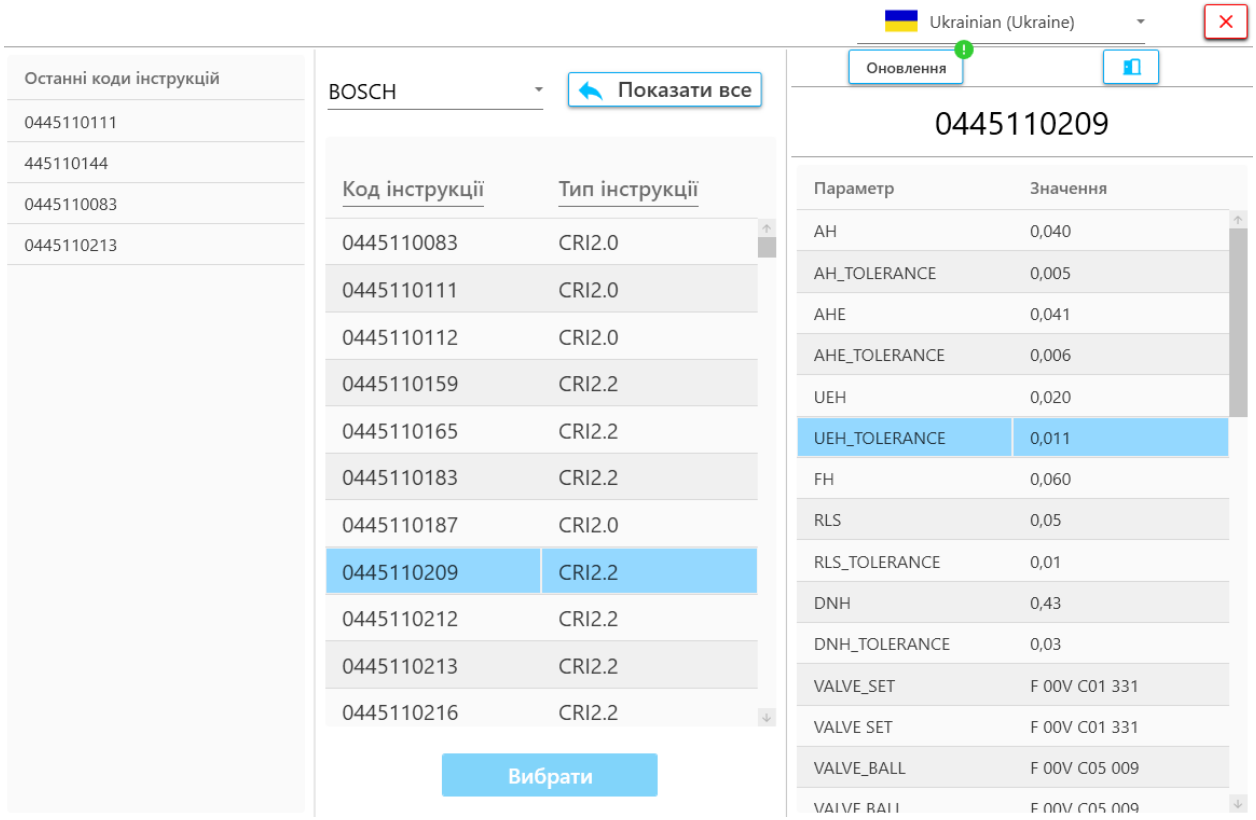


Рис. 3.15. Японська версія додатку

Також у додатку використовується мультимовність і вона підтримує декілька мов, такі як: англійська, українська, німецька, іспанська, французька, турецька, польська, корейська, японська.


Рис. 3.16. Японська версія додатку

Для можливості редагувати або створювати інструкцію, потрібно бути адміністратором, для цього потрібно авторизуватися і ввести логін та пароль.

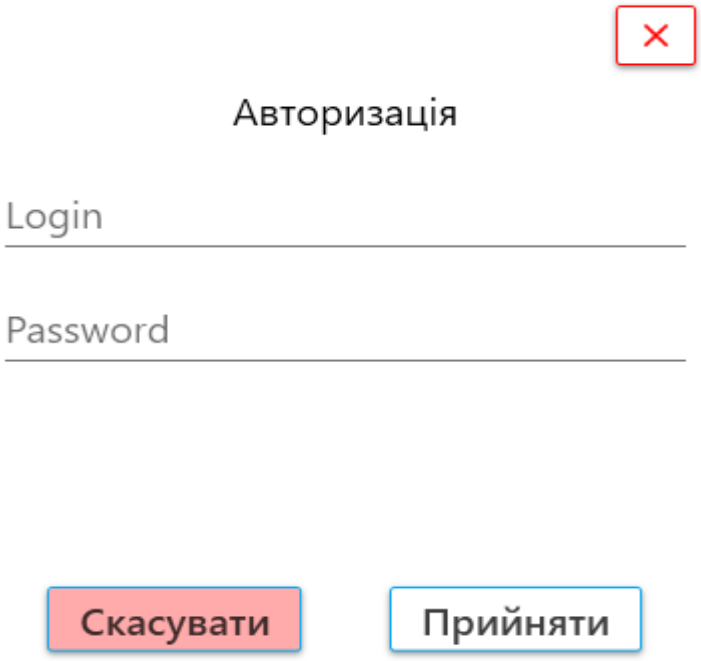


Рис. 3.17. Вікно авторизації

Так як додаток підтримує автоматичне оновлення, ми можемо по кліку перевірити чи є більш нова версія файлу на сервері, якщо мається то відкривається відповідне вікно де можна завантажити нову версію, до того ж якщо на сервері мається відповідна інформація щодо вмісту оновлення.

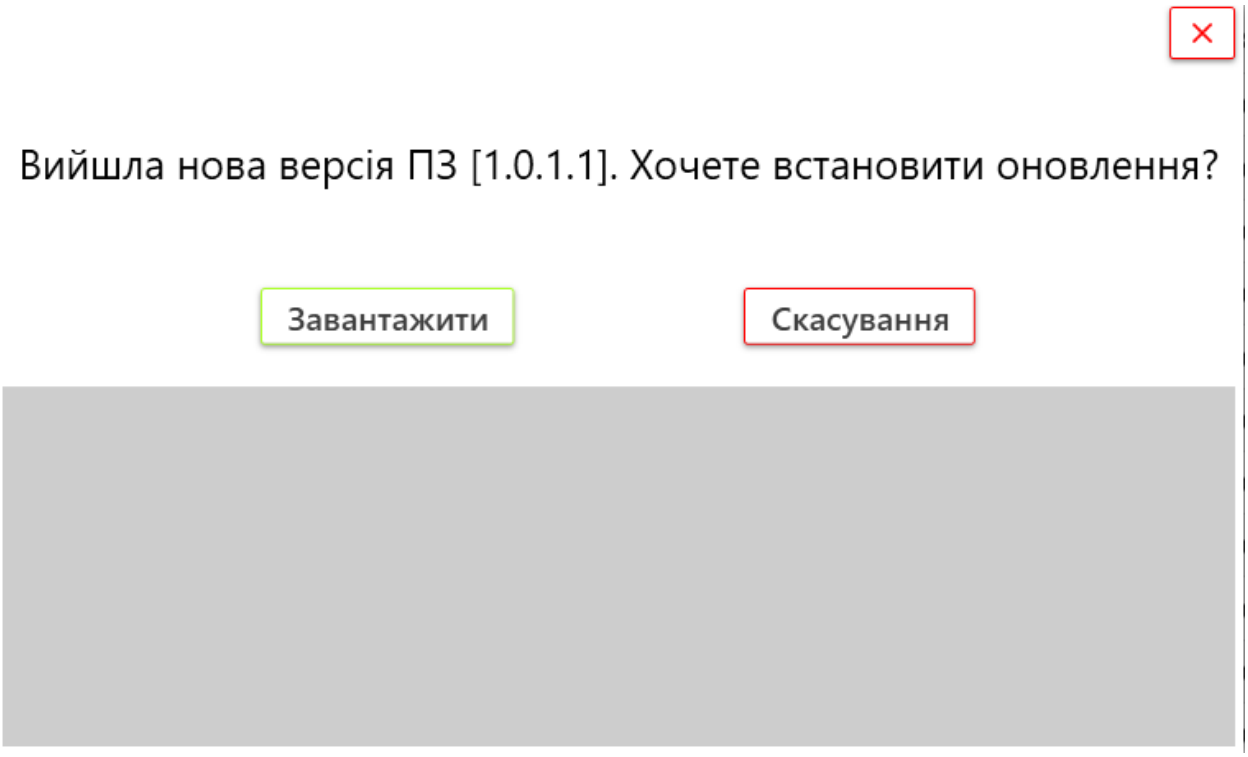


Рис. 3.18. Інформаційне вікно щодо оновлення додатку

Після згоди користувача почнеться завантаження нового файлу інсталятору додатку, буде відображатися прогрес завантаження в відсотковому стані.

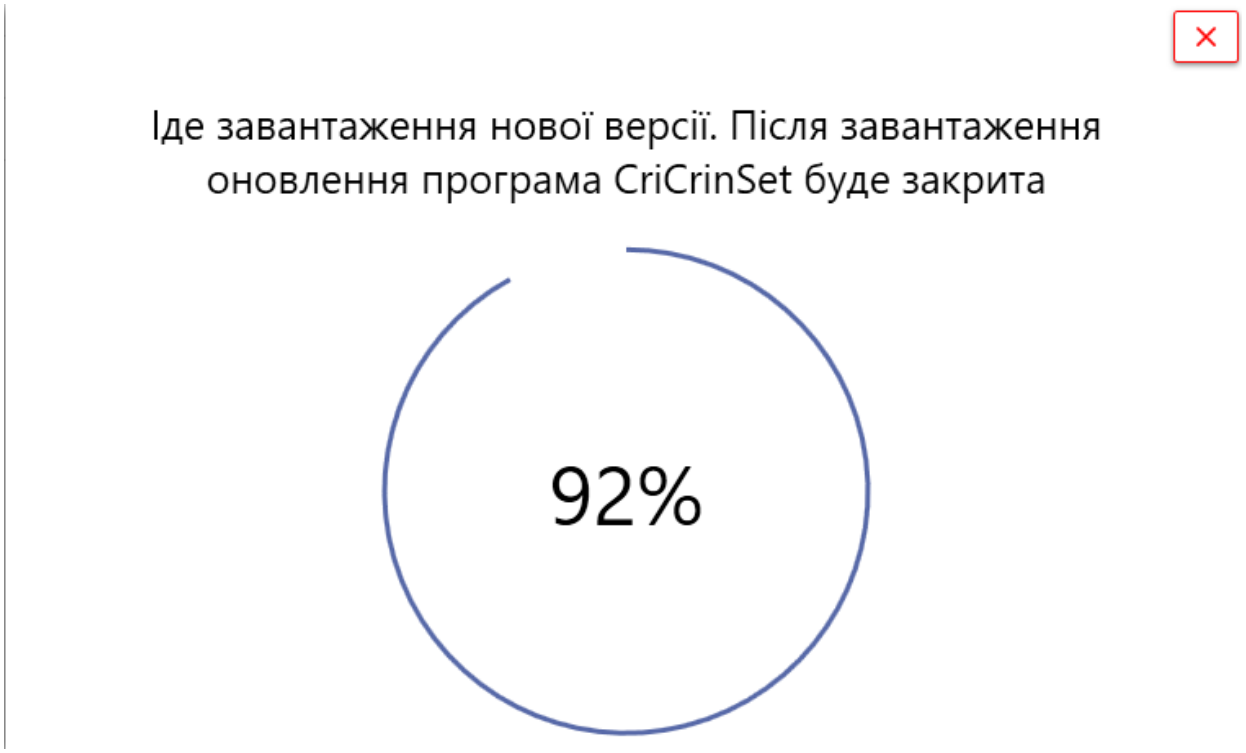


Рис. 3.19. Завантаження додатку

Після завантаження нового додатку, буде закрита поточна программа, та відкрита істалятор в якому можна буде встановити нову версію.

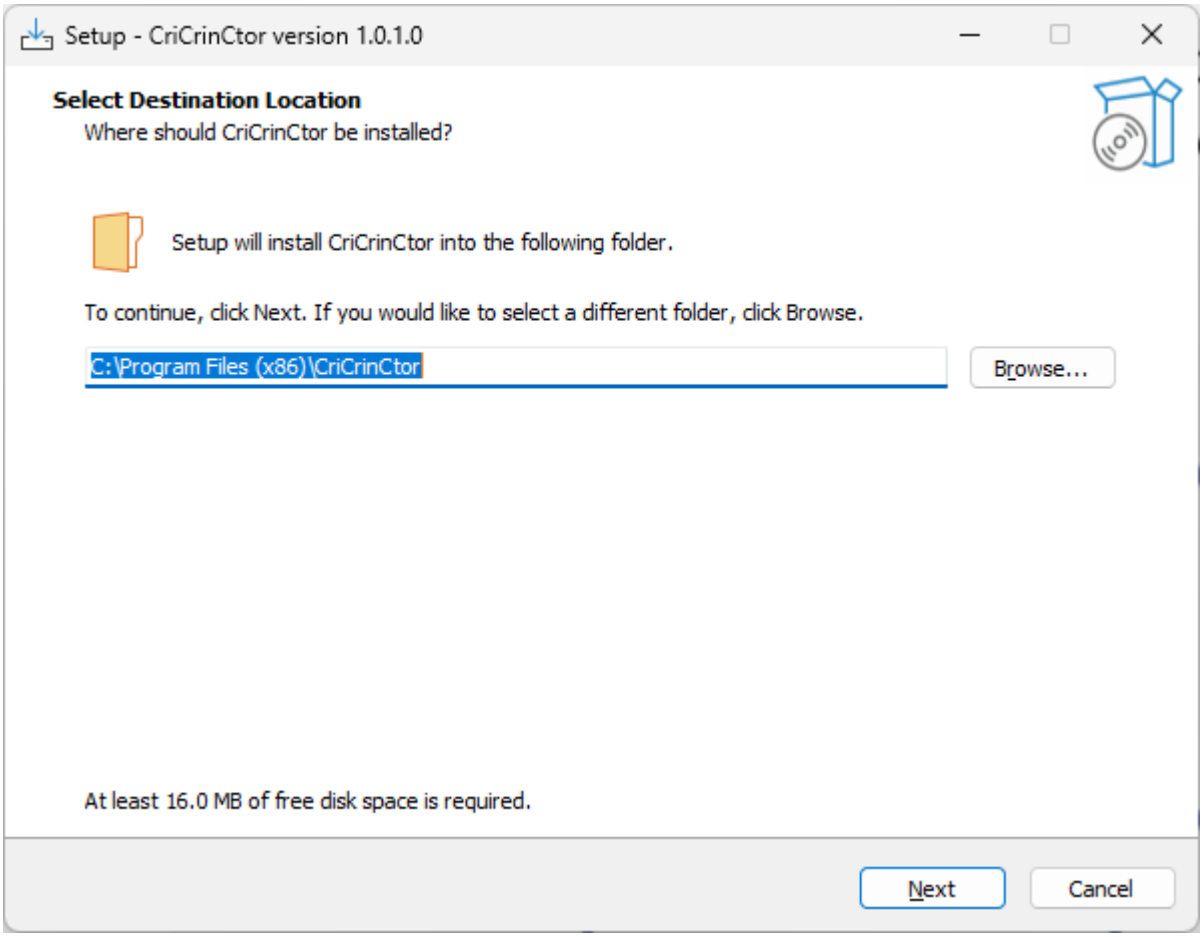


Рис. 3.20. Інсталяція нового додатку

# ВИСНОВКИ

На основі розглянутої роботи щодо «Дослідження ефективності впровадження системи для управління технічними інструкціями з можливістю відокремлення обов’язків адміністрування і користування» можна зробити наступний висновок.

Розроблена програмна система представляє собою потужний інструмент для створення, організації та керування технічними інструкціями різного типу. З врахуванням різновидів параметрів, таких як таблиці, змінні, формули, зображення та текст, користувачі отримують можливість ефективно створювати та редагувати інструкції, дотримуючись контексту та логічної структури.

Забезпечено інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам будь-якого рівня навичок працювати з системою, включаючи адміністраторів та звичайних користувачів. Це сприяє швидкому доступу до основних функцій, таких як створення нових інструкцій, пошук інструкцій та управління користувачами.

Реалізація різних типів параметрів дозволяє користувачам вбудовувати в інструкції різноманітні дані та інформацію, що підвищує багатофункціональність системи. Організація структури інструкцій дозволяє створювати логічну послідовність та зв'язки між розділами.

Загалом, розроблена система відповідає сучасним вимогам управління технічними інструкціями та має великий потенціал для використання у різних галузях. Вона спрощує та оптимізує процес створення та управління інструкціями, що робить її цінним інструментом для підприємств та організацій.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 МЕТОДИЧНІ РЕКОМЕНДАЦІЇ до виконання кваліфікаційних робіт здобувачами другого (магістерського) рівня вищої освіти спеціальностей 121 «Інженерія програмного забезпечення» / / Б.І. Мороз, О.В. Іванченко, О.В. Реута, О.С. Шевцова; М-во освіти і науки України, Нац. техн. ун-т “Дніпровська політехніка”. – Дніпро : НТУ «ДП», 2021. – 56 с.

2 Огляд типів індексів Oracle, MySQL, PostgreSQL, MS SQL [Електронний ресурс] / Огляд типів індексів Oracle, MySQL, PostgreSQL, MS SQL, 2010. – Режим доступа: https://habr.com/ru/post/102785/. – Назва з екрану.

3 Повний посібник з мови програмування C# 12 та платформі .NET 8 [Електронний ресурс] / Повний посібник з мови програмування C# 12 та платформі .NET 8, 2023. – Режим доступа: https://metanit.com/sharp/tutorial/. – Назва з екрану.

4 Посібник з WPF [Електронний ресурс] / Посібник з WPF, 2023. – Режим доступа: https://metanit.com/sharp/wpf/. – Назва з екрану.

5 Документація за C# [Електронний ресурс] / Документація за C#, 2023. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/. – Назва з екрану.

6 What Is SQLite? [Електронний ресурс] / What Is SQLite?, 2023. – Режим доступа: https://www.sqlite.org/index.html. – Назва з екрану.

7 SQLite [Електронний ресурс] / SQLite, 2023. – Режим доступа: https://en.wikipedia.org/wiki/SQLite. – Назва з екрану.

8 SQLite Studio [Електронний ресурс] / SQLite Studio, 2023. – Режим доступа: https://sqlitestudio.pl/. – Назва з екрану.

9 GitHub Copilot та Visual Studio 2022 [Електронний ресурс] / GitHub Copilot та Visual Studio 2022, 2023. – Режим доступа: https://visualstudio.microsoft.com/vs/. – Назва з екрану.

10 Visual Studio [Електронний ресурс] / Visual Studio, 2023. – Режим доступа: https://en.wikipedia.org/wiki/Visual_Studio. – Назва з екрану.

11 Types of instructions [Електронний ресурс] / Types of instructions, 2023. – Режим доступа: https://www.mikroe.com/ebooks/architecture-and-programming-of-8051-mcus/types-of-instructions. – Назва з екрану.

12 Creating a Technical Manual: How, Types & Examples [Електронний ресурс] / Creating a Technical Manual: How, Types & Examples, 2023. – Режим доступа: https://document360.com/blog/technical-manual/. – Назва з екрану.

13 Technical documentation [Електронний ресурс] / Technical documentation, 2023. – Режим доступа: https://whatfix.com/blog/types-of-technical-documentation/. – Назва з екрану.

14 Model-View-ViewModel (MVVM) [Електронний ресурс] / Model-View-ViewModel (MVVM), 2023. – Режим доступа: https://www.techtarget.com/whatis/definition/Model-View-ViewModel#:~:text=Model%2DView%2DViewModel%20(MVVM)%20is%20a%20software%20design,logic%20and%20user%20interface%20controls. – Назва з екрану.

15 Model–view–viewmodel [Електронний ресурс] / Model–view–viewmodel, 2023. – Режим доступа: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel. – Назва з екрану.

16 Model-View-ViewModel (MVVM) [Електронний ресурс] / Model-View-ViewModel (MVVM), 2023. – Режим доступа: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm. – Назва з екрану.

17 WPF – Add a Watermark to a native WPF TextBox [Електронний ресурс] / WPF – Add a Watermark to a native WPF TextBox, 2023. – Режим доступа: https://code.4noobz.net/wpf-add-a-watermark-to-a-native-wpf-textbox/. – Назва з екрану.

18 Основні вимоги до написання науково-дослідницької роботи / URL: http://dvman.dnepredu.com/uploads/editor/4165/ 353853/sitepage_62/files/ vimogi_ do_oformlennya_ndr.docx. дата звернення: 3.12.2017.

19 Складання списку літератури в навчальних виданнях : посіб. для наук.-пед. працівників / В.О. Салов, О.Н. Нефедова, О.Н. Ільченко, В.В. Панченко, Т.О. Недайвода, В.Г. Римар ; М-во освіти і науки України, Нац. гірн. ун-т. – Д. : НГУ, 2013. – 39 с

20 СВО НГУ ІМЗ – 09. Організація видання інформаційно-методичного забезпечення навчального процесу / Розроб.: В.О. Салов, О.І. Додатко, Т.О. Письменкова – Д.: Національний гірничий університет. – 2009. – 60 с.

21 Бюлетень ВАК України, №9-10, 2011. Вимоги до оформлення дисертацій та авторефератів. – 9 с.

22 SQLite Guide [Електронний ресурс] / SQLite Guide, 2023. – Режим доступа: https://metanit.com/sql/sqlite/. – Назва з екрану.

23 A Guide to ADO.NET and Working with Databases in .NET [Електронний ресурс] / A Guide to ADO.NET and Working with Databases in .NET, 2023. – Режим доступа: https://metanit.com/sharp/adonetcore/. – Назва з екрану.

24 Windows technical documentation for developers and IT pros [Електронний ресурс] / Windows technical documentation for developers and IT pros, 2023. – Режим доступа: https://learn.microsoft.com/en-us/windows/. – Назва з екрану.

25 .NET documentation [Електронний ресурс] / .NET documentation, 2023. – Режим доступа: https://learn.microsoft.com/en-us/dotnet/. – Назва з екрану.

**Додаток А**

# КОД ПРОГРАМИ

```
public partial class MainWindow : Window, INotifyPropertyChanged
    {
        public     ObservableCollection<WText>     WTeList     =     new
ObservableCollection<WText>();
        public     ObservableCollection<WInput>     WInList     =     new
ObservableCollection<WInput>();
        public     ObservableCollection<WImage>     WImList     =     new
ObservableCollection<WImage>();
        public     ObservableCollection<WTable>     WTaList     =     new
ObservableCollection<WTable>();
        public    ObservableCollection<WFormula>    WFoList    =    new
ObservableCollection<WFormula>();

        private WImage wImage = null;
        private WText wText = null;
        private WTable wTable = null;
        private WFormula wFormula = null;
        private WInput wInput = null;

        public  PartNumbersWindow  PartWindow  {  get;  set;  }  =  new
PartNumbersWindow();

        public static List<InfoParameter> IParams { get; set; } = new
List<InfoParameter>();

        public PartNumber PartNumber
        {
            get
            {
                return partNumber;
            }
            set
            {
                partNumber = value;
```

```
                RaisePropertyChanged("PartNumber");

                if (value != null)
                {
                    Settings.Default.idPartNumber = value.ID;
                    Settings.Default.Save();

File.AppendAllText(PartNumbersWindow.FILE_LAST_PARTNUMBER,   value.ID   +
Environment.NewLine);
                }
            }
        }
        public PartNumber partNumber = null;

        public Operation Operation
        {
            get => opearation;
            set
            {
                if (opearation != value && value != null)
                {
                    WImList.Clear();
                    WTeList.Clear();
                    WTaList.Clear();
                    WFoList.Clear();
                    WInList.Clear();

                    opearation = value;
                    Instruct(Instruction.Operation, Operation.ID);
                }
                else opearation = value;
                RaisePropertyChanged("Operation");
                for (int i = 0; i < NumberOperationStep.Count && value !=
null; i++)
                {
                    if (OperationList[i] == Operation)
                    {
                        numberOperation.SelectedIndex = i;
```

```
                        break;
                    }
                }

                if (value != null)
                {
                    Settings.Default.idOperation = value.ID;
                    Settings.Default.Save();
                }
            }
        }
        public Operation opearation = null;

        public Step Step
        {
            get => step;
            set
            {
                if (step != value && value != null)
                {
                    DeleteListStep();

                    step = value;
                    Instruct(Instruction.Step, Step.ID);
                }
                else step = value;
                RaisePropertyChanged("Step");
                for (int i = 0; i < NumberOperationStep.Count && value !=
null; i++)
                {
                    if (OperationList[i] == Operation)
                    {
                        numberOperation.SelectedIndex = i;
                        for      (int      j      =      0;      j      <
NumberOperationStep[i].StepNumbers.Count; j++)
                        {
                            if (OperationList[i].Steps[j] == Step)
                            {
```

```
                                numberStep.SelectedIndex = j;
                            }
                        }
                        break;
                    }
                }

                if (value != null)
                {
                    Settings.Default.idStep = value.ID;
                    Settings.Default.Save();
                }
            }
        }
        public Step step = null;

        public ObservableCollection<Operation> OperationList
        {
            get => operationList;
            set
            {
                operationList.Clear();

                NumberOperationStep.Clear();
                int i = 1;

                if (value != null)
                {
                    foreach (var item in value)
                    {
                        operationList.Add(item);

                        List<StepNumberVisible>    stepNumbers    =    new
List<StepNumberVisible>(0);
                        for (int j = 0; j < item.Steps.Count(); j++)
                        {
                            stepNumbers.Add(new StepNumberVisible
                            {
```

```
                                NumberStep = i + "." + (j + 1)
                            });
                        }
                        NumberOperationStep.Add(new
OperationNumberVisible
                        {
                            NumberOperation = i++,
                            StepNumbers = stepNumbers
                        });
                    }
                }
                RaisePropertyChanged("OperationList");
            }
        }
        private ObservableCollection<Operation> operationList = new
ObservableCollection<Operation>();

        public ObservableCollection<OperationNumberVisible>
NumberOperationStep
        {
            get
            {
                return numberOperationStep;
            }
            set
            {
                numberOperationStep.Clear();

                if (value != null)
                {
                    foreach (var item in value)
                    {
                        numberOperationStep.Add(item);
                    }
                }
                RaisePropertyChanged("NumberOperationStep");
            }
        }
```

```
        private ObservableCollection<OperationNumberVisible>
numberOperationStep = new ObservableCollection<OperationNumberVisible>();

        protected void RaisePropertyChanged(string propertyName)
        {
            PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
        }
        public event PropertyChangedEventHandler PropertyChanged;

        private int GridMenuWidth { get; set; } = 114;

        public static RoutedCommand CommandArrowLeft = new
RoutedCommand();

        public static RoutedCommand CommandArrowRigth = new
RoutedCommand();

        private void CommandArrowLeftExecuted(object sender,
ExecutedRoutedEventArgs e)
        {
            BArrowLeft_Click(null, null);
        }

        private void CommandArrowRigthExecuted(object sender,
ExecutedRoutedEventArgs e)
        {
            BArrorRight_Click(null, null);
        }

        public MainWindow()
        {
            InitializeComponent();

            CommandArrowLeft.InputGestures.Add(new KeyGesture(Key.Left));
            CommandArrowRigth.InputGestures.Add(new
KeyGesture(Key.Right));
```

```
            Width = SystemParameters.PrimaryScreenWidth;
            Height = SystemParameters.PrimaryScreenHeight - 50;

            double aContent = (Width - GridMenuWidth) / 3;
            AnimationColumn(scrollLeft, aContent);
            AnimationColumn(scrollMain, aContent);
            AnimationColumn(scrollRigth, aContent);
            scrollLeft.Width = aContent;
            scrollMain.Width = aContent;
            scrollRigth.Width = aContent;

            PartNumber                                                  =
CriLiteDBQuery.GetPartNum(Settings.Default.idPartNumber);
            if (PartNumber != null)
            {
                IParams                                                 =
CriLiteDBQuery.GetInfoParameter(PartNumber.ID).ToList();
                OperationList                                           =
CriLiteDBQuery.GetOperation(PartNumber.IDType);
                if (OperationList.Count > 0)
                {
                    Operation = OperationList.FirstOrDefault(x => x.ID ==
Settings.Default.idOperation);
                    CbOperationSelectedItem(Operation);
                    if (Operation != null && Operation.Steps.Count > 0)
                    {
                        Step = Operation.Steps.FirstOrDefault(x => x.ID ==
Settings.Default.idStep);
                        CbStepSelectedItem(Step);
                    }
                }
            }

            if (!App.DEV)
            {
                conMenu1.Items.Clear();
                conMenu2.Items.Clear();
            }
```

```
            DataContext = this;
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            PartWindow.CheckUpdate();
        }

        private void DeleteListStep()
        {
            List<object> list = new List<object>(0);

            foreach (var item in WImList)
            {
                if ((Instruction)((UserControl)item).Tag == Instruction.Step)
                {
                    list.Add(item);
                }
            }

            foreach (var item in list)
            {
                WImList.Remove((WImage)item);
            }

            list.Clear();

            foreach (var item in WTeList)
            {
                if ((Instruction)((UserControl)item).Tag == Instruction.Step)
                {
                    list.Add(item);
                }
            }
```

```
            foreach (var item in list)
            {
                WTeList.Remove((WText)item);
            }

            list.Clear();

            foreach (var item in WTaList)
            {
                if        ((Instruction)((UserControl)item).Tag        ==
Instruction.Step)
                {
                    list.Add(item);
                }
            }

            foreach (var item in list)
            {
                WTaList.Remove((WTable)item);
            }

            list.Clear();

            foreach (var item in WFoList)
            {
                if        ((Instruction)((UserControl)item).Tag        ==
Instruction.Step)
                {
                    list.Add(item);
                }
            }

            foreach (var item in list)
            {
                WFoList.Remove((WFormula)item);
            }

            list.Clear();
```

```
            foreach (var item in WInList)
            {
                if        ((Instruction)((UserControl)item).Tag        ==
Instruction.Step)
                {
                    list.Add(item);
                }
            }

            foreach (var item in list)
            {
                WInList.Remove((WInput)item);
            }

            list.Clear();
        }

        private void Window_Closing(object sender, CancelEventArgs e)
        {
            Environment.Exit(0);
        }

        private void Grid_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left) this.DragMove();
        }

        private    void    ButtonOpenTakePartNum_Click(object    sender,
RoutedEventArgs e)
        {
            PartWindow.SaveOperationAndStep = true;
            PartWindow.LoadLastNumber();
            PartWindow.ShowDialog();
            if (PartWindow.SaveOperationAndStep)
            {
                SelectAgainContentOperationAndStep();
            }
```

```
            else
            {
                PartNumber = PartWindow.PartNum;
                IParams = PartWindow.ParamList.ToList();

                if (PartNumber != null)
                {
                    OperationList                                          =
CriLiteDBQuery.GetOperation(PartNumber.IDType);

                    spMain.Children.Clear();
                    spLeft.Children.Clear();
                    spRight.Children.Clear();

                    if (OperationList.Count > 0)
                    {
                        Operation = OperationList[0];
                        CbOperationSelectedItem(Operation);
                        if (Operation.Steps.Count > 0)
                        {
                            Step = Operation.Steps[0];
                            CbStepSelectedItem(Step);
                        }
                    }
                }
            }
        }

        private void LeftPanel_ME(object sender, MouseEventArgs e)
        {
            double aContent = (Width - GridMenuWidth) / 4;
            AnimationColumn(scrollLeft, aContent * 2);
            AnimationColumn(scrollMain, aContent);
            AnimationColumn(scrollRigth, aContent);
        }

        private void CenterPanel_ME(object sender, MouseEventArgs e)
        {
```

```
            double aContent = (Width - GridMenuWidth) / 4;
            AnimationColumn(scrollLeft, aContent);
            AnimationColumn(scrollMain, aContent * 2);
            AnimationColumn(scrollRigth, aContent);
        }

        private void RightPanel_ME(object sender, MouseEventArgs e)
        {
            double aContent = (Width - GridMenuWidth) / 4;
            AnimationColumn(scrollLeft, aContent);
            AnimationColumn(scrollMain, aContent);
            AnimationColumn(scrollRigth, aContent * 2);
        }

        private void MainGrid_ML(object sender, MouseEventArgs e)
        {
            double aContent = (Width - GridMenuWidth) / 3;
            AnimationColumn(scrollLeft, aContent);
            AnimationColumn(scrollMain, aContent);
            AnimationColumn(scrollRigth, aContent);
        }

        private  void  AnimationColumn(ScrollViewer  scrollViewer,  double
toChange)
        {
            DoubleAnimation Animation = new DoubleAnimation
            {
                From = scrollViewer.ActualWidth,
                To = toChange,
                Duration = TimeSpan.FromMilliseconds(200)
            };
            scrollViewer.BeginAnimation(WidthProperty, Animation);
        }

        private void Grid_SizeChanged(object sender, SizeChangedEventArgs
e)
        {
            GridMenuWidth = GridMenu.IsMouseOver ? 514 : 114;
```

```
            double aContent = (Width - GridMenuWidth) / 3;
            AnimationColumn(scrollLeft, aContent);
            AnimationColumn(scrollMain, aContent);
            AnimationColumn(scrollRigth, aContent);
        }

        private void GridMenu_MouseEnter(object sender, MouseEventArgs e)
        {
            GridMenuWidth = GridMenu.IsMouseOver ? 514 : 114;
            double aContent = (Width - GridMenuWidth) / 3;
            AnimationColumn(scrollLeft, aContent);
            AnimationColumn(scrollMain, aContent);
            AnimationColumn(scrollRigth, aContent);
        }

        private void GridMenu_MouseLeave(object sender, MouseEventArgs e)
        {
            GridMenuWidth = GridMenu.IsMouseOver ? 514 : 114;
            double aContent = (Width - GridMenuWidth) / 3;
            AnimationColumn(scrollLeft, aContent);
            AnimationColumn(scrollMain, aContent);
            AnimationColumn(scrollRigth, aContent);

            takePartNumber.Visibility = Visibility.Collapsed;
            restartPartNumber.Visibility = Visibility.Collapsed;
            buttOperation.Visibility = Visibility.Collapsed;

            numberOperation.Visibility = cbOperation.Visibility ==
Visibility.Visible || numberOperation.Visibility == Visibility.Visible
                ? Visibility.Visible : Visibility.Collapsed;
            numberStep.Visibility = Visibility.Visible;

            cbOperation.Visibility = Visibility.Collapsed;
            cbStep.Visibility = Visibility.Collapsed;
        }

        private void DoubleAnimationUsingKeyFramesEnter_Completed(object
sender, EventArgs e)
```

```
        {
            if (GridMenu.IsMouseOver)
            {
                if (!(numberOperation.Visibility == Visibility.Collapsed
&& numberStep.Visibility == Visibility.Collapsed))
                {
                    takePartNumber.Visibility = Visibility.Visible;
                    restartPartNumber.Visibility = Visibility.Visible;
                    buttOperation.Visibility = Visibility.Visible;

                    cbOperation.Visibility = numberOperation.Visibility;
                    cbStep.Visibility = Visibility.Visible;

                    numberOperation.Visibility = Visibility.Collapsed;
                    numberStep.Visibility = Visibility.Collapsed;
                }
            }
        }

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

        }

        private void BMinimized(object sender, RoutedEventArgs e)
        {
            WindowState = WindowState.Minimized;
        }

        private void BMaximized(object sender, RoutedEventArgs e)
        {
            WindowState = WindowState == WindowState.Maximized ?
WindowState.Normal : WindowState.Maximized;
        }

        private void CbOperation_PMouseUp(object sender,
MouseButtonEventArgs e)
```

```
        {
            if (cbOperation.Items.Count > 0 && cbOperation.SelectedIndex
> -1)
            {
                cbOperation.Visibility = Visibility.Collapsed;
                spParamStep.Visibility = Visibility.Visible;

                numberOperation.SelectedIndex                          =
cbOperation.SelectedIndex;
            }
        }

        private void HideHeader_Click(object sender, RoutedEventArgs e)
        {
            cbOperation.Visibility = Visibility.Visible;
            spParamStep.Visibility = Visibility.Collapsed;
        }

        private     void     CbStep_SelectionChanged(object     sender,
SelectionChangedEventArgs e)
        {
            if   (cbOperation.Visibility   ==   Visibility.Collapsed   &&
spParamStep.Visibility == Visibility.Visible)
            {
                Operation = (Operation)cbOperation.SelectedItem;
                Step = (Step)cbStep.SelectedItem;
            }
        }

        private  void  CheckNumberOrder(LocationContent  locationContent,
UIElement uIElement, int numOrd)
        {
            switch (locationContent)
            {
                case LocationContent.Main:
                    AddContentInStackPanel(spMain, uIElement, numOrd);
                    break;
                case LocationContent.Left:
```

```
                    AddContentInStackPanel(spLeft, uIElement, numOrd);
                    break;
                case LocationContent.Right:
                    AddContentInStackPanel(spRight, uIElement, numOrd);
                    break;
                default:
                    break;
            }

            void     AddContentInStackPanel(StackPanel     stackPanelAdd,
UIElement uIElement, int numOrd)
            {
                for (int i = 0; i < stackPanelAdd.Children.Count; i++)
                {
                    switch (stackPanelAdd.Children[i])
                    {
                        case WInput wInput:
                            if (wInput.Inputs.NumberOrder > numOrd)
                            {
                                stackPanelAdd.Children.Insert(i,
uIElement);
                                return;
                            }
                            break;
                        case WImage wImage:
                            if (wImage.Images.NumberOrder > numOrd)
                            {
                                stackPanelAdd.Children.Insert(i,
uIElement);
                                return;
                            }
                            break;
                        case WFormula wFormula:
                            if (wFormula.Formulas.NumberOrder > numOrd)
                            {
                                stackPanelAdd.Children.Insert(i,
uIElement);
                                return;
```

```
                            }
                            break;
                        case WText wText:
                            if (wText.Texts.NumberOrder > numOrd)
                            {
                                stackPanelAdd.Children.Insert(i,
uIElement);
                                return;
                            }
                            break;
                        case WTable wTable:
                            if (wTable.Tables.NumberOrder > numOrd)
                            {
                                stackPanelAdd.Children.Insert(i,
uIElement);
                                return;
                            }
                            break;
                        default:
                            break;
                    }
                }
                stackPanelAdd.Children.Add(uIElement);
            }
        }

        private void Instruct(Instruction instruction, int ID)
        {
            if (instruction == Instruction.Step)
            {
                ClearWidgetFromWindow();
            }
            else
            {
                spMain.Children.Clear();
                spLeft.Children.Clear();
                spRight.Children.Clear();
            }
```

```
            ObservableCollection<Images>           ImageList            =
CriLiteDBQuery.GetImage(ID, instruction, TypeContent.Image);
            ObservableCollection<Texts>            TextList             =
CriLiteDBQuery.GetText(ID, instruction, TypeContent.Text);
            ObservableCollection<Tables>           TableList            =
CriLiteDBQuery.GetTable(ID, instruction, TypeContent.Table);
            ObservableCollection<Formulas>         FormulaList          =
CriLiteDBQuery.GetFormula(ID, instruction, TypeContent.Formula);
            ObservableCollection<Inputs>           InputList            =
CriLiteDBQuery.GetInput(ID, instruction, TypeContent.Input);

            for (int i = 0; i < InputList.Count; i++)
            {
                WInput wInput = new WInput()
                {
                    ID = InputList[i].ID_Input,
                    Header = InputList[i].Header,
                    Units = InputList[i].Units,
                    Value = InputList[i].IValue,
                    ID_Content = InputList[i].ID,
                    Inputs = InputList[i],
                    Tag = instruction,
                };

                CheckNumberOrder(InputList[i].Location,           wInput,
wInput.Inputs.NumberOrder);
                WInList.Add(wInput);
            }

            for (int i = 0; i < FormulaList.Count; i++)
            {
                WFormula   wFormula   =   new   WFormula(FormulaList[i],
FormulaList[i].Inputs)
                {
                    ID = FormulaList[i].ID_Formula,
                    Header = FormulaList[i].Header,
                    Formula = FormulaList[i].Formula,
```

```
                    ID_Content = FormulaList[i].ID,
                    Tag = instruction
                };

                CheckNumberOrder(FormulaList[i].Location,       wFormula,
wFormula.Formulas.NumberOrder);
                WFoList.Add(wFormula);
            }

            for (int i = 0; i < TableList.Count; i++)
            {
                WTable wTable = new WTable
                {
                    ID = TableList[i].ID_Table,
                    Header = TableList[i].Header,
                    RowTables = TableList[i].RowTables,
                    ID_Content = TableList[i].ID,
                    WCol1 = TableList[i].WCol1,
                    WCol2 = TableList[i].WCol2,
                    Tables = TableList[i],
                    Tag = instruction
                };

                CheckNumberOrder(TableList[i].Location,           wTable,
wTable.Tables.NumberOrder);
                WTaList.Add(wTable);
            }

            for (int i = 0; i < TextList.Count; i++)
            {
                WText wText = new WText
                {
                    ID = TextList[i].ID_Text,
                    Header = TextList[i].Header,
                    Text = TextList[i].Content,
                    ID_Content = TextList[i].ID,
                    Texts = TextList[i],
                    Tag = instruction
```

```
                };

                CheckNumberOrder(TextList[i].Location,              wText,
wText.Texts.NumberOrder);
                WTeList.Add(wText);
            }

            for (int i = 0; i < ImageList.Count; i++)
            {
                WImage wImage = new WImage
                {
                    ID = ImageList[i].ID_Image,
                    Header = ImageList[i].Header,
                    ImageFile = ImageList[i].Image_,
                    ID_Content = ImageList[i].ID,
                    Images = ImageList[i],
                    Tag = instruction
                };

                CheckNumberOrder(ImageList[i].Location,           wImage,
wImage.Images.NumberOrder);
                WImList.Add(wImage);
            }
        }

        private void ClearWidgetFromWindow()
        {
            List<object> list = new List<object>(0);

            foreach (var item in spMain.Children)
            {
                if        ((Instruction)((UserControl)item).Tag        ==
Instruction.Step)
                {
                    list.Add(item);
                }
            }
```

```
            foreach (var item in list)
            {
                spMain.Children.Remove((UIElement)item);
            }

            list.Clear();

            foreach (var item in spLeft.Children)
            {
                if        ((Instruction)((UserControl)item).Tag        ==
Instruction.Step)
                {
                    list.Add(item);
                }
            }

            foreach (var item in list)
            {
                spLeft.Children.Remove((UIElement)item);
            }

            list.Clear();

            foreach (var item in spRight.Children)
            {
                if        ((Instruction)((UserControl)item).Tag        ==
Instruction.Step)
                {
                    list.Add(item);
                }
            }

            foreach (var item in list)
            {
                spRight.Children.Remove((UIElement)item);
            }

            list.Clear();
```

```
        }

        private void MIEditOperationOrStep(object sender, RoutedEventArgs
e)
        {
            if      (cbOperation.Visibility      ==      Visibility      ||
numberOperation.Visibility == Visibility)
            {
                Operation                    operation                    =
(Operation)cbOperation.SelectedItem;
                if (operation != null)
                {
                    InputMessageWindow       inputWindow       =       new
InputMessageWindow(FindResource("txt42").ToString(),
operation.NameOperation);
                    inputWindow.ShowDialog();

                    if (inputWindow.FlagAccept)
                    {
                        operation.NameOperation                          =
inputWindow.ReturnString;
                        CriLiteDBQuery.EditOperation(operation);
                        SelectAgainOperationAndStep();
                    }
                }
            }
            else
            {
                Step step = (Step)cbStep.SelectedItem;
                if (step != null)
                {
                    InputMessageWindow       inputWindow       =       new
InputMessageWindow(FindResource("txt43").ToString(), step.NameStep);
                    inputWindow.ShowDialog();

                    if (inputWindow.FlagAccept)
                    {
                        step.NameStep = inputWindow.ReturnString;
```

```
                    CriLiteDBQuery.EditStep(step);
                    SelectAgainOperationAndStep();
                }
            }
        }
    }

    private void MIDeleteOperationOrStep(object sender,
RoutedEventArgs e)
    {
        if (cbOperation.Visibility == Visibility ||
numberOperation.Visibility == Visibility)
        {
            Operation operation =
(Operation)cbOperation.SelectedItem;
            if (operation != null)
            {
                var res = MessageBox.Show($"{FindResource("txt44")}
{operation.NameOperation} {FindResource("txt14")}?", string.Empty,
MessageBoxButton.YesNo, MessageBoxImage.Question);
                if (res == MessageBoxResult.Yes)
                {
                    CriLiteDBQuery.DeleteOperation(operation.ID);
                    SelectAgainOperationAndStep();
                }
            }
        }
        else
        {
            Step step = (Step)cbStep.SelectedItem;
            if (step != null)
            {
                CriLiteDBQuery.DeleteStep(step.ID);
                SelectAgainOperationAndStep();
            }
        }
    }
```

```
        private void MIAddOperationOrStep(object sender, RoutedEventArgs
e)
        {
            if (cbOperation.Visibility == Visibility ||
numberOperation.Visibility == Visibility)
            {
                if (PartNumber != null)
                {
                    InputMessageWindow inputWindow = new
InputMessageWindow(FindResource("txt45").ToString(), "");
                    inputWindow.ShowDialog();

                    if (inputWindow.FlagAccept)
                    {
                        CriLiteDBQuery.AddOperation(PartNumber.IDType,
inputWindow.ReturnString);
                        SelectAgainOperationAndStep();
                    }
                }
            }
            else
            {
                Operation operation =
(Operation)cbOperation.SelectedItem;
                if (operation != null)
                {
                    InputMessageWindow inputWindow = new
InputMessageWindow(FindResource("txt46").ToString(), "");
                    inputWindow.ShowDialog();

                    if (inputWindow.FlagAccept)
                    {
                        CriLiteDBQuery.AddStep(operation.ID,
inputWindow.ReturnString);
                        SelectAgainOperationAndStep();
                    }
                }
            }
```

```
        }

        private void MIUpOperationOrStep(object sender, RoutedEventArgs e)
        {
            if (cbOperation.Visibility == Visibility ||
numberOperation.Visibility == Visibility)
            {
                Operation operation =
(Operation)cbOperation.SelectedItem;
                if (operation != null)
                {
                    CriLiteDBQuery.SwapOperation(operation, SWAP.Up);
                    SelectAgainOperationAndStep();
                }
            }
            else
            {
                Step step = (Step)cbStep.SelectedItem;
                if (step != null)
                {
                    CriLiteDBQuery.SwapStep(step, SWAP.Up);
                    SelectAgainOperationAndStep();
                }
            }
        }

        private void MIDownOperationOrStep(object sender, RoutedEventArgs
e)
        {
            if (cbOperation.Visibility == Visibility ||
numberOperation.Visibility == Visibility)
            {
                Operation operation =
(Operation)cbOperation.SelectedItem;
                if (operation != null)
                {
                    CriLiteDBQuery.SwapOperation(operation, SWAP.Down);
                    SelectAgainOperationAndStep();
```

```
                }
            }
            else
            {
                Step step = (Step)cbStep.SelectedItem;
                if (step != null)
                {
                    CriLiteDBQuery.SwapStep(step, SWAP.Down);
                    SelectAgainOperationAndStep();
                }
            }
        }

        private void MIUpContent(object sender, RoutedEventArgs e)
        {
            if (wImage != null)
            {
                if         (CriLiteDBQuery.SwapContent(wImage.ID_Content,
wImage.Images.IDOperation,                              wImage.Images.IDStep,
wImage.Images.NumberOrder, SWAP.Up, wImage.Images.Location))
                    SelectAgainContentOperationAndStep();
            }
            if (wText != null)
            {
                if          (CriLiteDBQuery.SwapContent(wText.ID_Content,
wText.Texts.IDOperation,   wText.Texts.IDStep,   wText.Texts.NumberOrder,
SWAP.Up, wText.Texts.Location))
                    SelectAgainContentOperationAndStep();
            }
            if (wTable != null)
            {
                if         (CriLiteDBQuery.SwapContent(wTable.ID_Content,
wTable.Tables.IDOperation,                              wTable.Tables.IDStep,
wTable.Tables.NumberOrder, SWAP.Up, wTable.Tables.Location))
                    SelectAgainContentOperationAndStep();
            }
            if (wFormula != null)
            {
```

```
                if      (CriLiteDBQuery.SwapContent(wFormula.ID_Content,
wFormula.Formulas.IDOperation,                  wFormula.Formulas.IDStep,
wFormula.Formulas.NumberOrder, SWAP.Up, wFormula.Formulas.Location))
                    SelectAgainContentOperationAndStep();
            }
            if (wInput != null)
            {
                if         (CriLiteDBQuery.SwapContent(wInput.ID_Content,
wInput.Inputs.IDOperation,                          wInput.Inputs.IDStep,
wInput.Inputs.NumberOrder, SWAP.Up, wInput.Inputs.Location))
                    SelectAgainContentOperationAndStep();
            }
        }

        private void MIDownContent(object sender, RoutedEventArgs e)
        {
            if (wImage != null)
            {
                if         (CriLiteDBQuery.SwapContent(wImage.ID_Content,
wImage.Images.IDOperation,                          wImage.Images.IDStep,
wImage.Images.NumberOrder, SWAP.Down, wImage.Images.Location))
                    SelectAgainContentOperationAndStep();
            }
            if (wText != null)
            {
                if          (CriLiteDBQuery.SwapContent(wText.ID_Content,
wText.Texts.IDOperation,   wText.Texts.IDStep,   wText.Texts.NumberOrder,
SWAP.Down, wText.Texts.Location))
                    SelectAgainContentOperationAndStep();
            }
            if (wTable != null)
            {
                if         (CriLiteDBQuery.SwapContent(wTable.ID_Content,
wTable.Tables.IDOperation,                          wTable.Tables.IDStep,
wTable.Tables.NumberOrder, SWAP.Down, wTable.Tables.Location))
                    SelectAgainContentOperationAndStep();
            }
            if (wFormula != null)
```

```
            {
                if       (CriLiteDBQuery.SwapContent(wFormula.ID_Content,
wFormula.Formulas.IDOperation,                  wFormula.Formulas.IDStep,
wFormula.Formulas.NumberOrder, SWAP.Down, wFormula.Formulas.Location))
                    SelectAgainContentOperationAndStep();
            }
            if (wInput != null)
            {
                if         (CriLiteDBQuery.SwapContent(wInput.ID_Content,
wInput.Inputs.IDOperation,                              wInput.Inputs.IDStep,
wInput.Inputs.NumberOrder, SWAP.Down, wInput.Inputs.Location))
                    SelectAgainContentOperationAndStep();
            }
        }

        private void BArrorRight_Click(object sender, RoutedEventArgs e)
        {
            bool flagNextUp = false;

            for (int i = 0; i < OperationList.Count(); i++)
            {
                for (int j = 0; j < OperationList[i].Steps.Count; j++)
                {
                    if (flagNextUp)
                    {
                        Operation = OperationList[i];
                        CbOperationSelectedItem(Operation);
                        Step = OperationList[i].Steps[j];
                        CbStepSelectedItem(Step);
                        return;
                    }
                    else if (OperationList[i].Steps[j].ID == Step.ID)
                    {
                        flagNextUp = true;
                    }
                }
            }
        }
```

```
        private void BArrowLeft_Click(object sender, RoutedEventArgs e)
        {
            bool flagNextDown = false;

            for (int i = OperationList.Count() - 1; i > -1; i--)
            {
                for (int j = OperationList[i].Steps.Count - 1; j > -1; j-
-)
                {
                    if (flagNextDown)
                    {
                        Operation = OperationList[i];
                        CbOperationSelectedItem(Operation);
                        Step = OperationList[i].Steps[j];
                        CbStepSelectedItem(Step);
                        return;
                    }
                    else if (OperationList[i].Steps[j].ID == Step.ID)
                    {
                        flagNextDown = true;
                    }
                }
            }
        }

        private void MIEditElement(object sender, RoutedEventArgs e)
        {
            if (wImage != null)
            {
                Images images = new Images
                {
                    ID_Image = wImage.ID,
                    Header = wImage.Header,
                    Image_ = wImage.ImageFile,
                };
                ImageWindow imageWindow = new ImageWindow(Operation, Step,
TypeContent.Text, images);
```

```
                imageWindow.ShowDialog();
                SelectAgainContentOperationAndStep();
            }
            if (wText != null)
            {
                Texts texts = new Texts
                {
                    Content = wText.rich.SaveData(),
                    ID_Text = wText.ID,
                    Header = wText.Header
                };
                TextWindow textWindow = new TextWindow(Operation, Step,
TypeContent.Text, PartNumber, texts);
                textWindow.ShowDialog();
                SelectAgainContentOperationAndStep();
            }
            if (wTable != null)
            {
                Tables tables = new Tables
                {
                    ID_Table = wTable.ID,
                    Header = wTable.Header,
                    RowTables = wTable.RowTables,
                    WCol1 = wTable.WCol1,
                    WCol2 = wTable.WCol2
                };
                TableWindow tableWindow = new TableWindow(Operation, Step,
TypeContent.Text, PartNumber, tables);
                tableWindow.ShowDialog();
                SelectAgainContentOperationAndStep();
            }
            if (wFormula != null)
            {
                Formulas formulas = new Formulas
                {
                    ID_Formula = wFormula.ID,
                    Header = wFormula.Header,
                    Formula = wFormula.Formula,
```

```
                    Inputs = wFormula.Inputs
                };
                FormulaWindow formulaWindow = new
FormulaWindow(Operation, Step, TypeContent.Text, formulas);
                formulaWindow.ShowDialog();
                SelectAgainContentOperationAndStep();
            }
            if (wInput != null)
            {
                Inputs inputs = new Inputs
                {
                    ID_Input = wInput.ID,
                    Header = wInput.Header,
                    Units = wInput.Units,
                };
                InputWindow inputWindow = new InputWindow(Operation, Step,
TypeContent.Text, inputs);
                inputWindow.ShowDialog();
                SelectAgainContentOperationAndStep();
            }
        }

        private void MIDeleteElement(object sender, RoutedEventArgs e)
        {
            if (wImage != null)
            {
                var ob = MessageBox.Show($"{FindResource("txt47")}
\"{wImage.Header}\"", string.Empty, MessageBoxButton.YesNo,
MessageBoxImage.Question);
                if (ob == MessageBoxResult.Yes)
                {

CriLiteDBQuery.DeleteContentToDatabase(wImage.ID_Content, wImage.ID,
TypeContent.Image);
                    SelectAgainContentOperationAndStep();
                }
            }
            if (wText != null)
```

```
            {
                var    ob    =    MessageBox.Show($"{FindResource("txt47")}
\"{wText.Header}\"",         string.Empty,         MessageBoxButton.YesNo,
MessageBoxImage.Question);
                if (ob == MessageBoxResult.Yes)
                {

CriLiteDBQuery.DeleteContentToDatabase(wText.ID_Content,         wText.ID,
TypeContent.Text);
                    SelectAgainContentOperationAndStep();
                }
            }
            if (wTable != null)
            {
                var    ob    =    MessageBox.Show($"{FindResource("txt47")}
\"{wTable.Header}\"",         string.Empty,         MessageBoxButton.YesNo,
MessageBoxImage.Question);
                if (ob == MessageBoxResult.Yes)
                {

CriLiteDBQuery.DeleteContentToDatabase(wTable.ID_Content,      wTable.ID,
TypeContent.Table);
                    SelectAgainContentOperationAndStep();
                }
            }
            if (wFormula != null)
            {
                var    ob    =    MessageBox.Show($"{FindResource("txt47")}
\"{wFormula.Header}\"",       string.Empty,       MessageBoxButton.YesNo,
MessageBoxImage.Question);
                if (ob == MessageBoxResult.Yes)
                {

CriLiteDBQuery.DeleteContentToDatabase(wFormula.ID_Content,  wFormula.ID,
TypeContent.Formula);
                    SelectAgainContentOperationAndStep();
                }
            }
```

```
            if (wInput != null)
            {
                var ob = MessageBox.Show($"{FindResource("txt47")} \"{wInput.Header}\"", string.Empty, MessageBoxButton.YesNo, MessageBoxImage.Question);
                if (ob == MessageBoxResult.Yes)
                {

CriLiteDBQuery.DeleteContentToDatabase(wInput.ID_Content, wInput.ID, TypeContent.Input);
                    SelectAgainContentOperationAndStep();
                }
            }
        }

        private void MIAddFormula(object sender, RoutedEventArgs e)
        {
            Dispatcher.BeginInvoke(new Action(() =>
            {
                if (Operation != null && Step != null)
                {
                    FormulaWindow frmWnd = new FormulaWindow(Operation, Step, TypeContent.Image);
                    frmWnd.ShowDialog();
                    SelectAgainContentOperationAndStep();
                }
            }));
        }

        private void MIAddInput(object sender, RoutedEventArgs e)
        {
            Dispatcher.BeginInvoke(new Action(() =>
            {
                if (Operation != null && Step != null)
                {
                    InputWindow frmWnd = new InputWindow(Operation, Step, TypeContent.Image);
                    frmWnd.ShowDialog();
```

```
                SelectAgainContentOperationAndStep();
            }
        }));
    }

    private void MIAddText(object sender, RoutedEventArgs e)
    {
        Dispatcher.BeginInvoke(new Action(() =>
        {
            if (Operation != null && Step != null)
            {
                TextWindow inpWnd = new TextWindow(Operation, Step,
TypeContent.Image, PartNumber);
                inpWnd.ShowDialog();
                SelectAgainContentOperationAndStep();
            }
        }));
    }

    private void MIAddTable(object sender, RoutedEventArgs e)
    {
        Dispatcher.BeginInvoke(new Action(() =>
        {
            if (Operation != null && Step != null)
            {
                TableWindow tblWnd = new TableWindow(Operation, Step,
TypeContent.Image, PartNumber);
                tblWnd.ShowDialog();
                SelectAgainContentOperationAndStep();
            }
        }));
    }

    private void MIAddImage(object sender, RoutedEventArgs e)
    {
        Dispatcher.BeginInvoke(new Action(() =>
        {
            if (Operation != null && Step != null)
```

```
                {
                    ImageWindow imageWindow = new ImageWindow(new
List<string>(0), Operation, Step, TypeContent.Image);
                    imageWindow.ShowDialog();
                    SelectAgainContentOperationAndStep();
                }
            }));
        }

        private void MainContent_Drop(object sender, DragEventArgs e)
        {
            Dispatcher.BeginInvoke(new Action(() =>
            {
                if (e.Data.GetDataPresent(DataFormats.FileDrop) &&
Operation != null && Step != null)
                {
                    var arr =
((string[])e.Data.GetData(DataFormats.FileDrop)).ToList();

                    ImageWindow imageWindow = new ImageWindow(arr,
Operation, Step, TypeContent.Image);
                    imageWindow.ShowDialog();
                    SelectAgainContentOperationAndStep();
                }
            }));
        }

        private void SelectAgainOperationAndStep()
        {
            int indexOperation = cbOperation.SelectedIndex, indexStep =
cbStep.SelectedIndex;

            OperationList =
CriLiteDBQuery.GetOperation(PartNumber.IDType);

            cbOperation.SelectedIndex = -1;
            cbStep.SelectedIndex = -1;
            CbOperationSelectedIndex(indexOperation);
```

```
            CbStepSelectedIndex(indexStep);
        }

        public void SelectAgainContentOperationAndStep()
        {
            int indexOperation = cbOperation.SelectedIndex, indexStep =
cbStep.SelectedIndex;

            if (PartNumber != null && indexOperation != -1 && indexStep !=
-1)
            {
                OperationList                                          =
CriLiteDBQuery.GetOperation(PartNumber.IDType);

                cbOperation.SelectedIndex = -1;
                cbStep.SelectedIndex = -1;
                cbOperation.SelectedIndex = indexOperation;
                cbStep.SelectedIndex = indexStep;
                Operation = OperationList[indexOperation];
                Step = OperationList[indexOperation].Steps[indexStep];
            }
        }

        private       void       MainGrid_MouseDown(object       sender,
MouseButtonEventArgs e)
        {
            wImage   =   WImList.Count(x   =>   x.IsMouseOver)   ==   1   ?
WImList.First(x => x.IsMouseOver) : null;
            wText   =   WTeList.Count(x   =>   x.IsMouseOver)   ==   1   ?
WTeList.First(x => x.IsMouseOver) : null;
            wTable   =   WTaList.Count(x   =>   x.IsMouseOver)   ==   1   ?
WTaList.First(x => x.IsMouseOver) : null;
            wFormula   =   WFoList.Count(x   =>   x.IsMouseOver)   ==   1   ?
WFoList.First(x => x.IsMouseOver) : null;
            wInput   =   WInList.Count(x   =>   x.IsMouseOver)   ==   1   ?
WInList.First(x => x.IsMouseOver) : null;
        }
```

```
        private void RestartPartNumber_Click(object sender,
RoutedEventArgs e)
        {
            CriLiteDBQuery.UpdateValueVariableNull();

            Operation = null;
            Step = null;

            if (OperationList.Count > 0)
            {
                Operation = OperationList[0];
                CbOperationSelectedItem(Operation);
                if (Operation.Steps.Count > 0)
                {
                    Step = Operation.Steps[0];
                    CbStepSelectedItem(Step);
                }
            }
        }

        private void CbOperationSelectedItem(object obj)
        {
            cbOperation.SelectedItem = obj;
            for (int i = 0; i < OperationList.Count; i++)
            {
                if (OperationList[i] == obj)
                {
                    numberOperation.SelectedIndex = i;
                    break;
                }
            }
        }

        private void CbOperationSelectedIndex(int ind)
        {
            cbOperation.SelectedIndex = ind;
            numberOperation.SelectedIndex = ind;
        }
```

```
    private void CbStepSelectedItem(object obj)
    {
        cbStep.SelectedItem = obj;
    }

    private void CbStepSelectedIndex(int ind)
    {
        cbStep.SelectedIndex = ind;
        numberStep.SelectedIndex = ind;
    }
}
```

**Додаток Б**

# ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

| **Ім’я файлу** | **Опис** |
|---|---|
| Пояснювальні документи | |
| ПЗ Олійник І.К.docx | Пояснювальна записка роботи. Документ Word. |
| ПЗ Олійник І.К.pdf | Пояснювальна записка роботи. Документ PDF. |
| Програма | |
| CriCrinCtor.zip | Архів. Містить код програми. |
| Презентація | |
| Презентація Олійник.pptx | Презентація дипломної роботи. |