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)

студентки *Левдик Ірини Андріївни*
(ПІБ)

академічної групи *121м-22-2*
(шифр)

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

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

на тему: *Створення та впровадження системи для управління технічними інструкціями з можливістю відокремлення обов'язків адміністрування і користування*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	*доц. Приходченко С.Д.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

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-22-2* *Левдик Ірині Андріївні*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Створення та впровадження системи для управління технічними інструкціями з можливістю відокремлення обов'язків адміністрування і користування*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 09.10.2023 р. № 1227-с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – ефективність роботи співробітників з інструкціями в їхньому робочому процесі.

Предмет досліджень – процес роботи співробітників з інструкціями, винайдені KPI (ключові показники ефективності).

Мета НДР – виявлення слабких місць в роботі з інструкціями та пошук нового, більш оптимального рішення.

Вихідні дані для проведення роботи – теоретичні та експериментальні дослідження, методи описової статистики.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень полягає в поліпшенні ефективності робочого процесу серед працівників, які займаються створенням та переглядом технічних інструкцій, пришвидшенні та оптимізації роботи з документацією, усуненні існуючих дефектів в поточному процесі.

Практична цінність полягає в автоматизації існуючого процесу створення та використання технічних інструкцій, розробці програмного забезпечення для маніпуляцій з цим типом документації.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути подані у вигляді, що дозволяє побачити та оцінити результат аналізу статистичних даних, досягненого запропонованими методами. В результаті роботи повинне бути розроблене програмне забезпечення для управління технічними інструкціями.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі	12.09.2022-30.09.2023
Розробка моделі, метода та програмного забезпечення для управління технічними інструкціями	01.10.2022-31.10.2023
Використання програми та аналіз отриманих результатів	01.11.2022-12.12.2023

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи очікується позитивним завдяки розробці універсального рішення для управління технічною документацією, що дозволило би пришвидшити та полегшити процес роботи з нею.

Соціальний ефект від реалізації результатів роботи очікується позитивним, завдяки полегшенню роботи з технічними інструкціями, автоматизації та усуненню існуючих дефектів поточної системи.

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

Завдання прийняв до виконання _____ *Левдик І.А.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *12.09.2023 р.*

Термін подання кваліфікаційної роботи до ЕК *12.12.2023*

РЕФЕРАТ

Пояснювальна записка: 82 сторінки, 38 рисунків, 2 додатки, 38 джерел.

Об’єкт дослідження: процес автоматизації технічної інструкції.

Мета магістерської роботи: виявлення слабких місць в роботі з інструкціями та пошук нового, більш оптимального рішення..

Методи дослідження: теоретичні та експериментальні дослідження, методи описової статистики.

Наукова новизна даної роботи полягає в поліпшенні ефективності робочого процесу серед працівників, які займаються створенням та переглядом технічних інструкцій, пришвидшенні та оптимізації роботи з документацією, усуненні існуючих дефектів в поточному процесі.

Практична цінність полягає в автоматизації існуючого процесу створення та використання технічних інструкцій, розробці програмного забезпечення для маніпуляцій з цим типом документації.

Область застосування: несе широкий характер, цільовою областю застосування перш за все є великі виробництва з потребою в налагодженні робочого процесу та конвеєру створення інструкцій.

Значення роботи та висновки: економічний ефект від реалізації результатів роботи очікується позитивним завдяки розробці універсального рішення для управління технічною документацією, що дозволило би пришвидшити та полегшити процес роботи з нею. Соціальний ефект від реалізації результатів роботи очікується позитивним, завдяки полегшенню роботи з технічними інструкціями, автоматизації та усуненню існуючих дефектів поточної системи.

Прогнози щодо розвитку досліджень: провести більш детальне дослідження зі збором більш повноцінних та комлпексних статистичних даних для більш глибокого аналізу; створити прогнози щодо поточних рішень та майбутнього рішення, довести вплив певних факторів на результати ефективності при виробництві.

Список ключових слів: МЕТРИКА, ІНСТРУКЦІЯ, ЕФЕКТИВНІСТЬ, АВТОМАТИЗАЦІЯ, ФАКТОРИ ВПЛИВУ.

ABSTRACT

Explanatory note: 82 pages, 38 figures, 2 appendices, 38 sources.

The object of research: the process of automating technical instructions. your goal of the master's thesis: the formation of weak points in working with instructions and the search for a new, most optimal solution. Research methods: theoretical and experimental studies, methods of descriptive statistics.

The scientific novelty of this work contributes to increasing the efficiency of the work process among employees who are engaged in the creation and revision of technical instructions, acceleration and optimization of work with documentation, and the use of existing defects in the current process.

Practical value arises in the automation of the existing process of creation and use of technical instructions, development of software for manipulation of this type of documentation.

Field of application: broad in nature, the target field of application is primarily large-scale production with the need for a well-established workflow and pipeline of creating instructions.

Value of the work and conclusions: the economic effect of the implementation of the results of the work is expected to be positive due to the development of a universal solution for the management of technical documentation, which would allow to come and facilitate the process of working with it. The social effect of the implementation of the results of the work is expected to be positive, thanks to the facilitation of work with technical instructions, automation and elimination of existing defects of the current system.

Predictions for the development of research: to conduct a more detailed study with the collection of more complete and comprehensive statistical data for a deeper analysis; create forecasts regarding current decisions and future decisions, prove the influence of certain factors on the effectiveness of production results.

List of keywords: METRIC, INSTRUCTION, EFFICIENCY, AUTOMATION, INFLUENCING FACTORS.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Інструкція – текстовий конструкторський документ, який містить вказівки і правила щодо виготовлення виробу.

Метрика – вимірювана характеристика, стандарт або система вимірювань, яка використовується для оцінки чого-небудь або визначення його характеристик.

Фіча – окремий функціональний елемент або можливость в програмному продукті.

ЗМІСТ

ВСТУП

Зі зростанням технологічного розвитку та обсягу інформації в сучасному світі, забезпечення доступності та ефективного управління технічними інструкціями стає вельми важливим завданням для підприємств та організацій у різних галузях. Технічні інструкції виконують ключову роль у забезпеченні безпеки, надійності та ефективності виробничих процесів, надаючи працівникам необхідні вказівки та інформацію для виконання різноманітних завдань.

Проте, процес управління технічними інструкціями вимагає покращення і модернізації, особливо в умовах швидкоплинного середовища.

Актуальність даної теми визначається тим, що на сьогоднішній день існує ряд серйозних проблем у сфері управління технічними інструкціями, які потребують невідкладного вирішення.

Однією з найважливіших серед них є неспроможність відокремлювати функції адміністрування системи управління інструкціями від користування ними. Це часто веде до плутанини, невірного використання інструкцій, а також збільшує ризик помилок та непорозумінь серед працівників. Додатково, існують проблеми з оновленням і розповсюдженням інструкцій, що стає перешкодою для підтримання актуальності та безпеки на робочому місці.

Також на сучасному ринку є певний брак схожих систем, які б дозволяли інтегрувати та управляти інструкціями. Зазвичай для цього використовуються засоби загального профілю, але специфічно заточених під цю галузь, на жаль чи на щастя, майже не існує.

Сучасний ступінь дослідження даної теми показує, що існують різні підходи до управління технічними інструкціями, але більшість з них не вирішують проблему відокремлення обов'язків адміністрування та користування в повній мірі. Іншими словами, існує потреба в розробці і впровадженні нового підходу, який дозволить оптимізувати управління технічними інструкціями та покращити доступність та безпеку користування ними.

Метою цієї дипломної роботи є створення та впровадження системи для управління технічними інструкціями з можливістю відокремлення обов'язків адміністрування і користування. Розробка такої системи дозволить підприємствам підвищити ефективність та безпеку робочих процесів, зменшити ризик помилок та сприяти збереженню актуальності інструкцій. В результаті дослідження та розробки системи очікується досягнення значущих практичних результатів та покращення управління технічними інструкціями на підприємствах та організаціях

РОЗДІЛ 1
ЗБІР ТА АНАЛІЗ ДАНИХ, ЗІБРАНИХ В ХОДІ ДОСЛІДЖЕННЯ

1.1 Аналіз предметної області

Об'єктом дослідження є процес створення, редагування та управління технічними інструкціями у великих організаціях, де існує потреба в точній, оновлюваній та керованій технічній документації для забезпечення якості та безпеки виробництва. Цей процес створює проблемну ситуацію, оскільки вимагає збору, оновлення та доступу до великої кількості інформації, а також контролю над доступом різних користувачів до цієї інформації. Існуючі рішення мають ряд недоліків, які будуть розглянуті нижче.

Дослідження спрямоване на розробку системи, яка дозволить керувати цим процесом з відокремленням обов'язків адміністрування і користування для забезпечення ефективності та безпеки управління технічними інструкціями в організаціях. Метою роботи є створення системи управління технічними інструкціями, яка би вирішувала вищевказану проблему найоптимальнішим чином.

Були впроваджені наступні роботи:

– Сбір даних про об’єкт розробки та здійснюваних ним видів діяльності;

– Вивчення та оцінка якості функціонування об’єкта та здійснюваних ним видів діяльності, виявлення проблем, рішення яких можливе завдяки створюваній системі;

– Оцінка технічно-економічної, соціальної, практичної доцільності створення інформаційної системи;

– Виявлення та аналіз вимог користувача до інформаційної системи;

– Аналіз ринку та потенційних конкурентоспроможних вже існуючих рішень;

– Оцінка ефективності прийнятих проектних рішень для створення інформаційної системи.

Загальні відомості з предметної області охоплюють широкий спектр аспектів, пов'язаних з управлінням технічними інструкціями, їх структурою, функціональністю та актуальністю. Ось деякі загальні відомості, які можуть бути корисними для розуміння предметної області:

1. Технічна документація: технічна документація включає в себе різні типи інструкцій, такі як інструкції зі збору, експлуатації, ремонту, безпеки тощо. Ця документація є важливою для правильного функціонування технічних систем та пристроїв.

2. Важливість актуальності і точності: технічні інструкції повинні бути актуальними та точними, оскільки вони впливають на безпеку та ефективність виробництва та обслуговування.

3. Регуляторні вимоги: у багатьох галузях, таких як медицина, авіація, фармація тощо, існують строгі регуляторні вимоги до технічної документації. Дотримання цих вимог є обов'язковим і може вимагати відповідного управління документами.

4. Вплив технологій: сучасні технології, такі як системи управління версіями, хмарні сервіси, інтерфейси користувача тощо, можуть значно полегшити управління технічними інструкціями.

5. Безпека та доступ: забезпечення безпеки та контролю над доступом до технічної документації є важливим завданням. Впровадження систем для відокремлення обов'язків адміністрування і користування допомагає захищати конфіденційну інформацію та забезпечувати безпеку даних.

6. Ефективність та продуктивність: оптимізація процесів створення, редагування та розповсюдження інструкцій може підвищити ефективність та продуктивність організації, а також зменшити ризики помилок.

7. Споживачі та користувачі: розуміння потреб споживачів та користувачів технічних інструкцій є важливим для розробки систем, які відповідають їхнім потребам.

1.2 Використані математичні та статистичні методи

Розрізнють первинні та вторинні підходи до аналізу статистичної вибірки даних.

До первинних належать методи, за допомогою яких можна отримати показники, що безпосередньо відображають результати отриманих в експерименті вимірювань, а саме:

1. середнє арифметичне;
2. дисперсія;
3. мода;
4. медіана.

До вторинних належать методи статистичної обробки, за допомогою яких на базі первинних даних виявляють приховані в них статистичні закономірності, а саме:

1. кореляційний аналіз;
2. регресійний аналіз;
3. факторний аналіз;
4. методи порівняння первинних даних двох або декількох вибірок.

В ході аналізу потреб користувачів та бізнес-процесів було зібрано статистичні дані, які пропонується дослідити за допомогою певних статистичних методів, а саме:

1. Первинні методи описової статистики.

Описова статистика або дескриптивна статистика (англ. descriptive statistics) — розділ статистики, який займається обробкою емпіричних даних, їх систематизацією, наочним представленням у вигляді графіків та таблиць, а також їх кількісним описом через основні статистичні показники. Використовує у свої базі *первинні* методи статистчної обробки (див. вище).

Описова статистика забезпечує короткий підсумок про вибірку та про спостереження, які були зроблені. Такі резюме можуть бути як кількісними, наприклад резюмуюча статистика, так і візуальна, наприклад прості графіки. Ці

резюме можуть бути або основою початкового опису даних як частина більш обширного статистичного аналізу, або вони можуть бути достатніми самі по собі для конкретного дослідження.

За допомогою методів описової статистики пропонується провести аналіз зібраних даних між різними групами співробітників залежно від їхньої методики ведення технічної документації та з'ясувати які методики ведення технічної документації є найбільш вразливими і підлягають аналізу та заміні, а які – навпаки, мають виграшні позиції та рекомендовані до подальшого розгляду та розвитку.

2. Регресійний аналіз

Регресійний аналіз - це статистичний метод, який може перевірити гіпотезу про те, що змінна залежить від однієї або декількох інших змінних. Далі регресійний аналіз може дати оцінку величини впливу зміни однієї змінної на іншу. Ця остання особливість, звичайно, важлива для прогнозування майбутніх значень.

Метод регресійного аналізу вважається найдосконалішим з усіх використовуваних нині нормативно-параметричних методів. Він широко застосовується для аналізу та встановлення рівня і співвідношень вартості продукції, яка характеризується наявністю одного або декількох техніко-економічних параметрів, що характеризують головні споживчі якості. Регресивний аналіз надає можливість знайти емпіричну форму залежності ціни від техніко-економічних параметрів товарів і виробів. При цьому він виступає в ролі цільової функції параметрів.

Метод регресійного аналізу особливо ефективний за умови здійснення розрахунків за допомогою сучасних інформаційних технологій і систем.

За допомогою методу регресійного аналізу пропонується дослідити взаємозалежності між певними метриками, введеними в дослідження за допомогою результатів опитування, та з'ясувати, в яких саме характеристиках мають проблеми використовувані нині методи впровадження та ведення

інструкцій, яка є тенденція ефективності у різних підходах, а також знайти та довести математично найефективніший підхід.

1.3 Інформація про структуру та метрики дослідження

Пропонується ввести в дослідження певні метрики, відштовхуючись від яких можна буде оцінити ефективність попередніх методів ведення та експлуатації технічних інструкцій, а також спрогнозувати, ефективність якого з типів буде найліпшою; дослідити вплив цих факторів на результат роботи співробітників.

Для цього було виділено кілька типів форм управління інструкціями:

1. Усна форма – співробітники на словах пояснюють послідовність дій один одному відповідно до встановленої субординації.
2. Месенджер – співробітники обмінюються повідомленнями в якості інструктажу.
3. Текстовий редактор – співробітники складають інструкцію в текстовому редакторі (в рамках дослідження – Microsoft Word), використовуючи доступний йому функціонал.
4. Специфічне ПЗ – програмний застосунок, через який співробітники мають змогу обмінюватися інформацією та надавати інструктаж.

До кожної з категорій було виділено ряд метрик, які, на думку дослідника, впливають на ефективність робочого процесу:

– Час створення інструкції (в хвилинах) – час, необхідний старшому співробітникові для створення інструкції, наданої одним з вищевказаних способів;
– Час на навчання (в хвилинах) – час для повного засвоєння наданої інформації, потрібний і достатній для того, щоб співробітник міг коректно дотримуватися наданої інструкції;

– Кількість помилок після інструктажу (за добу) – кількість помилок, якої припустився співробітник, дотримуючись інструкції, наданої одним з вищевказаних способів;
– Легкість пошуку потрібної інформації (за шкалою від 1 до 10) – результат опитування серед співробітників щодо легкості та комфортності управління та використання інструкцій, наданими одним з вищевказаних способів.

Нижче (на Рис.1.1.) можна побачити результат зібраної інформації, необхідної для дослідження, згрупованої відповідно до способу створення інструкції.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Усна форма											Середнє значення
2	Час створення	87	150	115	102	108	129	166	82	89	172	**120**
3	Час на навчання	250	276	201	272	249	221	257	210	195	269	**240**
4	К-ть помилок	2	7	3	6	4	6	5	5	6	6	**5**
5	Легкість пошуку І	2	4	3	4	4	2	2	2	3	4	**3**
6												
7												
8	Месенджер											Середнє значення
9	Час створення	155	95	100	105	110	95	90	80	120	50	**100**
10	Час на навчання	300	250	180	220	210	230	190	240	280	300	**240**
11	К-ть помилок	2	1	4	3	2	1	4	5	2	3	**2.7**
12	Легкість пошуку І	8	7	6	6	6	6	7	5	5	7	**6.3**
13												
14												
15	Текстовий редактор											Середнє значення
16	Час створення	500	200	150	180	220	200	190	210	180	200	**223**
17	Час на навчання	200	180	220	200	190	200	210	180	200	200	**198**
18	К-ть помилок	1	3	4	6	5	2	3	5	5	6	**4**
19	Легкість пошуку І	7	6	4	2	3	5	6	7	5	5	**5**
20												
21												
22	Специфічне ПЗ											Середнє значення
23	Час створення	65	55	60	70	50	54	65	66	60	55	**60**
24	Час на навчання	67	55	65	75	55	60	40	50	65	68	**60**
25	К-ть помилок	1	2	2	1	1	4	2	1	3	3	**2**
26	Легкість пошуку І	9	7	8	7	8	7	8	10	9	7	**8**
27												

Рис.1.1. «Статистичний зріз інформації для подальшого дослідження»

Надалі пропонується дослідити зібрану інформацію за допомогою методів описової статистики, довести вплив метрик на результуючий фактор – а саме **кількість виконаних задач за день**, що є прямим показником ефективності робочого процесу.

Також в ході дослідження буде спрогнозовано потенційні результати результуючого фактору **«кількість виконаних задач на день»** за допомогою регресійного аналізу та методу найменших квадратів, що допоможе зробити висновок про найефективніший і найнестабільніший способи ведення інструкцій та довести актуальність та результативність найбільш сучасного, автоматизованого та структурованого на думку автора методу ведення інструкцій, а саме – спеціалізованого програмного забезпечення.

1.4 Хід дослідження. Проведення первинного аналізу даних

До зібраних даних і до самого статистичного спостереження ставиться ряд вимог.

1. Дані повинні бути повними, а не уривчастими або випадковими.

   Під повнотою розуміють:

   - повноту просторового охоплення, тобто повне охоплення одиниць досліджуваної сукупності;
   - повноту охоплення сторін, тобто охоплення всіх істотних сторін явища, що вивчається;
   - повноту охоплення за часом, тобто наявність даних за максимально тривалі, безперервні відрізки часу.

   Особливою формою повноти даних є типові дані, які відображають всі основні сторони, властивості і специфіку досліджуваного явища або процесу.
2. Дані повинні бути достовірними і точними, тобто відповідати об'єктивній дійсності.
3. Дані повинні бути співставними.
4. Дані повинні бути своєчасними (особливо важливо для оперативного управління).

Ретельно розроблений і продуманий план статистичного спостереження - запорука успіху в отриманні достовірних даних про явище, що вивчається. Проте, як би ретельно не були продумані і програма статистичного спостереження і як би точно не слідували вказівкам інструкції особи, які здійснюють збір відомостей, при будь-якому статистичному спостереження можуть виникнути помилки (погрішності).

Вони можуть виникати з різних причин:

- за рахунок описок;
- обмовок;
- округлень;
- неправильного заповнення формуляру;
- забудькуватості тих, хто відповідає, або їх прагнення приховати або перекрутити факти;
- неточності вимірювальних приладів і т.д.

Всі ці помилки можна розділити на *випадкові* і *систематичні*.

Випадкові - результат дії різних випадкових чинників. Вони можуть бути вчинені через неувагу, забудькуватість, неуважність і можуть спотворити результати як в одну, так і в іншу сторону. Вони, як правило, взаємно погашаються і в принципі не в значній мірі впливають на результат.

Систематичні - спотворюють явище тільки в одну сторону, більш небезпечні, оскільки значною мірою впливають на підсумкові показники. Крім того, вони мають здатністю накопичувати помилку.

Але їхній вплив буде максимально зменшений завдяки застосуванню методу регресійного аналізу, який відкидає аномальні значення, тож дослідження буде проведено з якнайбільш допустимою точністю та правдивістю, щоб отримати найбільш реалістичний результат.

Перейдемо до первинного розрахунку та аналізу показників для кожної з вибірок даних.

Середнє арифметичне — це сума n значень спостережень у наборі точок даних, поділена на кількість N значень у наборі точок даних. Середнє

арифметичне записується у вигляді символу X. На Рис.1.2. можна побачити формулу для підрахунку середнього арифметичного.

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{N}$$

Рис.1.2. «Формула середнього арифметичного»

Дисперсія випадкової величини — це один з параметрів розподілу ймовірностей — це середньоквадратичне відхилення від середнього значення. Інакше кажучи, це математичне сподівання квадрату відхилення цієї змінної від її очікуваного значення (її математичного сподівання). Отже дисперсія є вимірюванням величини розпорошеності значень цієї змінної, беручи до уваги всі її значення і їхні ймовірності або ваги. Більші значення дисперсії свідчать про більші відхилення значень випадкової величини від центру розподілу. Іншими словами - якщо дисперсія низька, це означає, що більшість значень близькі до середнього, а якщо дисперсія висока, то значення розкидані ширше. [3]

Дисперсія поділяється на зміщену та незміщену

1. Зміщена дисперсія:

Ця міра розкиду використовується, коли ми маємо дані про всю групу чи популяцію, і ми хочемо оцінити, наскільки значення розподілені навколо середнього.

Формула включає розподіл на кількість значень групи (або популяції) n.

Зміщена дисперсія дає нам оцінку розкиду в даних, припускаючи, що ці дані є повною популяцією.

2. Незміщена дисперсія:

Ця міра розкиду використовується, коли ми маємо лише вибірку даних, і ми хочемо використовувати її для оцінки розкиду в популяції.

Формула включає розподіл на кількість значень у вибірці мінус 1 (n-1).

Незміщена дисперсія коригує оцінку розкиду, щоб зробити її більш точною для випадку, коли ми маємо лише частину даних, а не повну популяцію.

Поділ на (n−1) компенсує втрату інформації, пов'язану з використанням вибірки замість повної популяції.

Зміщена дисперсія використовується для оцінки розкиду у випадку, якщо ми маємо дані про всю групу, а не тільки про вибірку. Незміщена дисперсія використовується, коли ми маємо справу лише з вибіркою даних і хочемо точніше оцінити розкид у популяції.[3]

Розраховується за наступною формулою:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

, де

s^2 – незміщена дисперсія,

n – кількість спостережень в виборці,

x_i – кожне окреме спостереження,

x з чертою – середнє арифметичне

Оскільки у випадку поточного дослідження неможливо дослідити всі можливі компанії, підходи та всіх існуючих співробітників, для аналізу *буде взята формула незміщеної дисперсії*, оскільки використовуються дані про неповну популяцію, за якими планується узагальнити висновки і це потрібно враховувати.

Для характеристики розподілу одиниць сукупності за певною ознакою використовується так звані порядкові або структурні середні — **мода** і **медіана**.[4]

Мода (M0) — це значення ознаки, що найчастіше зустрічається у сукупності. Таким чином, У дискретному ряду вона визначається візуально за найбільшою частотою (часткою), а в інтервальному — таким чином визначається модальний інтервал, а конкретне модальне значення розраховується за формулою:

$$Mo = x_0 + h\frac{f_{mo} - f_{mo-1}}{(f_{mo} - f_{mo-1}) + (f_{mo} - f_{mo+1})};$$

де

- x_0 та h — нижня межа та ширина модального інтервалу,
- f_{mo}, f_{mo-1}, f_{mo+1} — частоти (частки) відповідно модального, передмодального та післямодального інтервалів.

Рис.1.3. «Формула моди»

Медіана — в статистиці це величина ознаки, що розташована посередині ранжованого ряду вибірки, тобто — це величина, що розташована в середині ряду величин, розташованих у зростальному або спадному порядку; в теорії ймовірності — характеристика розподілення випадкової величини.[4]

Пропонується провести розрахунок показників для кожної з вибірок залежно від її типу для подальшого аналізу та порівняння їх між собою.

Розрахуємо всі показники згідно формул вище. Оскільки в послідовностях парна кількість елементів, медіана розраховуватиметься як сума двох величин, що знаходяться в центрі послідовності. Також додатково для дисперсії обчислимо середньоквадратичне відхилення. Отримаємо такі результати:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Усна форма											Середнє значення	Дисперсія	СКВ(відхилення)	Мода	Медіана
2	Час створення	87	150	115	102	108	129	166	82	89	172	**120**	1087	32.83	172	111.5
3	Час на навчання	250	276	201	272	249	221	257	210	195	269	**240**	938	31	269	249.5
4	К-ть помилок	2	7	3	6	4	6	5	5	6	6	**5**	2.4	1.55	6	5.5
5	Легкість пошуку І	2	4	3	4	4	2	2	2	3	4	**3**	0.8	0.9	4	3
6																
7																
8	Месенджер											Середнє значення	Дисперсія	СКВ(відхилення)	Мода	Медіана
9	Час створення	155	95	100	105	110	95	90	80	120	50	**100**	733	27	95	97.5
10	Час на навчання	300	250	180	220	210	230	190	240	280	300	**240**	1822	42.7	300	235
11	К-ть помилок	2	1	4	3	2	1	4	5	2	3	**2.7**	1.78	1.3	2	2.5
12	Легкість пошуку І	8	7	6	6	6	6	7	5	5	7	**6.3**	0.9	0.95	6	6
13																
14																
15	Текстовий редактор											Середнє значення	Дисперсія	СКВ(відхилення)	Мода	Медіана
16	Час створення	500	200	150	180	220	200	190	210	180	200	**223**	9845	99.2	200	200
17	Час на навчання	200	180	220	200	190	200	210	180	200	200	**198**	151	12.3	200	200
18	К-ть помилок	1	3	4	6	5	2	3	5	5	6	**4**	2.8	1.67	5	4.5
19	Легкість пошуку І	7	6	4	2	3	5	6	7	5	5	**5**	2.6	1.6	5	5
20																
21																
22	Специфічне ПЗ											Середнє значення	Дисперсія	СКВ(відхилення)	Мода	Медіана
23	Час створення	65	55	60	70	50	54	65	66	60	55	**60**	41.3	6.4	60	60
24	Час на навчання	67	55	65	75	55	60	40	50	65	68	**60**	104.2	10.2	65	62.5
25	К-ть помилок	1	2	2	1	1	4	2	1	3	3	**2**	1.1	1.04	1	2
26	Легкість пошуку І	9	7	8	7	8	7	8	10	9	7	**8**	1.1	1.04	7	8
27																

Рис.1.4. «Результат обчислень первинних показників»

Тепер знайдемо ті ж самі показники для кожного з параметрів, не відштовхуючись від формату ведення інструкції. Сформуємо вибірку, згрупувавши попередньо зібрані дані за параметрами (Рис.1.4):

	A	B	C	D	E
1	**Час створення**	**Час на навчання**	**К-ть помилок**	**Легкість пошуку І**	
2	87	250	2	2	
3	150	276	7	4	
4	115	201	3	3	
5	102	272	6	4	
6	108	249	4	4	
7	129	221	6	2	
8	166	257	5	2	
9	82	210	5	2	
10	89	195	6	3	
11	172	269	6	4	
12	155	300	2	8	
13	95	250	1	7	
14	100	180	4	6	
15	105	220	3	6	
16	110	210	2	6	
17	95	230	1	6	
18	90	190	4	7	
19	80	240	5	5	
20	120	280	2	5	
21	50	300	3	7	
22	500	200	1	7	
23	200	180	3	6	
24	150	220	4	4	
25	180	200	6	2	
26	220	190	5	3	
27	200	200	2	5	
28	190	210	3	6	
29	210	180	5	7	
30	180	200	5	5	
31	200	200	6	5	
32	65	67	1	9	
33	55	55	2	7	
34	60	65	2	8	
35	70	75	1	7	
36	50	55	1	8	
37	54	60	4	7	
38	65	40	2	8	

Середні показники | Статистичний зріз | **Розрахунки для параметрів**

Рис.1.5. «Згрупована за параметрами вибірка»

Розрахуємо показники для кожного з параметрів (Рис.1.5):

	G	H	I	J	K	L	M
		Середнє значення	Дисперсія	СКВ(відхилення)	Мода	Медіана	
	Час створення	125.75	6413.8	80.08	103.5	200	
	Час на навчання	184.5	6296.6	79.3	200	200	
	К-ть помилок	3	3.27	1.8	3	2	
	Легкість пошуку І	5.575	4.7	2.2	6	7	

Рис.1.6. «Розрахунки показників для кожного з параметрів»

1.5 Висновки ходу дослідження

Тепер в нас є достатньо інформації, щоб можна було зробити загальний первинний аналіз даних. Порівняємо зібрані показники по формах інструктажу зі згрупованими по параметрах:

	A	B	C	D	E	F	G
1	Час створення	Середнє значення	Дисперсія	СКВ	Мода	Медіана	
2	Усна форма	120	1087	32.83	172	111.5	
3	Месенджер	100	733	27	95	97.5	
4	Текстовий редактор	223	9845	99.2	200	200	
5	Спеціальна програма	60	41.3	6.4	60	60	
6							
7	Загальний зріз	125.75	6413.8	80.08	103.5	200	
8							
9							
10	Час на навчання	Середнє значення	Дисперсія	СКВ	Мода	Медіана	
11	Усна форма	240	938	31	269	249.5	
12	Месенджер	240	1822	42.7	300	235	
13	Текстовий редактор	198	151	12.3	200	200	
14	Спеціальна програма	60	104.2	10.2	65	62.5	
15							
16	Загальний зріз	184.5	6296.6	79.3	200	200	
17							
18							
19	К-ть помилок	Середнє значення	Дисперсія	СКВ	Мода	Медіана	
20	Усна форма	5	2.4	1.55	6	5.5	
21	Месенджер	2.7	1.78	1.3	2	2.5	
22	Текстовий редактор	4	2.8	1.67	5	4.5	
23	Спеціальна програма	2	1.1	1.04	1	2	
24							
25	Загальний зріз	3	3.27	1.8	3	2	
26							
27							
28	Легкість пошуку	Середнє значення	Дисперсія	СКВ	Мода	Медіана	
29	Усна форма	3	0.8	0.9	4	3	
30	Месенджер	6.3	0.9	0.95	6	6	
31	Текстовий редактор	5	2.6	1.6	5	5	
32	Спеціальна програма	8	1.1	1.04	7	8	
33							
34	Загальний зріз	5.575	4.7	2.2	6	7	
35							

Рис.1.7. «Порівняльний аналіз даних»

На Рис.1.7. зібрані результати всіх попередніх обчислень. Червоним кольором виділені найбільш впливові в негативному сенсі показники. Зеленим – найбільш впливові в позитивному сенсі показники. Проаналізуємо отримані дані.

1. Час створення

   Бачимо, що дисперсія і СКВ досить великі, що свідчить про вагомий розкид даних. Це вказує на великий можливий діапазон часу, витраченого на створення інструкції. Припускається, що це тісно пов’язано з типом надання інструкції, а також власне індивідуальними випадками, коли інструкція, наприклад, надто легка і можна дуже швидко її описати/показати усно, але в текстовому форматі це займатиме досить багато часу.

   Червоним виділено форму ведення інструкції з максимальним значенням – тобто таку, що займає найбільше часу на створення – **текстовий редактор**. Це може бути пов’язано з тим, що інструктору обов’язково потрібно друкувати текст та інтегрувати в нього багато нюансів, що швидше відбувається в усній формі, месенджері (якій теж в комбінації дозволяє голосове пояснення) або спеціальній програмі, яка надає інструменти для пришвидшення процесу.

   Найоптимальніші значення має спеціальне ПЗ.

2. Час, витрачений на навчання

   Тут дисперсія і СКВ в середньому найбільші серед всіх можливих параметрів вибірки, що свідчить про аномально вагомий розкид даних. Можливо, пов’язано з індивідуальними розумовими здатностями людини, що навчається, а також – безумовно – з інтерфейсом користування та структурованістю даних, їхньою подачею.

   Бачимо, що на **усну форму** та **месенджер** припадають максимальні показники, на останньому місці – специфічне ПЗ. Це свідчить про безумовні переваги ПЗ в сфері **структуризації даних та інтерфейсі їхнього надання.**

3. Кількість помилок

Дисперсія та СКВ знаходяться в нормальних значеннях, що свідчить про те, що в середньому кількість помилок на добу є плюс-мінус рівномірно розподіленою.

Найбільше значення припадає на **усну форму**, що – очевидно – можна пояснити відсутністю занотовування інформації та **людським фактором**, адже єдиним джерелом інформації та збереження даних при цьому способі слугує мозок людини, що є вразливим до купи сторонніх факторів, які можуть впливати на повторне відновлення інформації після інструктажу.

Найменшої кількості помилок також припустилися при використанні **спеціального ПЗ**. Це може бути пов’язано зі способом надання інформації і **чіткістю та зрозумілістю її подачі** для співробітника завдяки **різноманіттю інструментів**, які може надати лише програмне забезпечення.

4. Легкість пошуку інформації

Дисперсія та СКВ знаходяться в нормальних значеннях, що свідчить про не надто невеликий розкид результатів опитування.

Останнє місце займає усна форма – адже в чертогах розуму щось знайти найважче усього, інформацію неможливо фізично структурувати і тому все залежить від індивідуальних характеристик людини, на які не варто повністю покладатися в технічних процесах.

На третьому місці текстовий редактор, що пояснюється відсутністю зручного інтерфейсу та неможливістю ніяк структурувати дані, окрім як засобами операційної системи.

На другому місці месенджер, адже він містить засоби пошуку, проте дані досі є недостатньо структурованими.

Найлегше за результатами опитування шукати інформацію було завдяки спеціалізованому програмному забезпеченню, що пояснюється **технічною комфортабельністю**.

РОЗДІЛ 2
ТЕХНОЛОГІЇ ТА ПРОГРАМИ ЯКІ БУЛИ ЗАДІЯНІ В РОЗРОБЦІ

2.1 Microsoft Visual Studio 2022

Microsoft Visual Studio — серія продуктів фірми Майкрософт, які містять інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Ці продукти дають змогу розробляти як консольні програми, так і програми з графічним інтерфейсом, включно з підтримкою технології Windows Forms, а також вебсайти, вебзастосунки, вебслужби як у рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

- Visual Studio включає один або декілька з наступних компонентів:
- Visual Basic .NET, а до його появи — Visual Basic
- Visual C++
- Visual C#
- Visual F# (входить до складу Visual Studio 2010);
- Visual Studio Debugger

Багато варіантів постачання також містять:

- Microsoft SQL Server або
- MSDE Visual Source Safe — файл-серверна система керування версіями

У минулому, до складу Visual Studio також входили продукти:

- Visual InterDev
- Visual J++
- Visual J#

- Visual FoxPro
- Visual Source Safe — файл-серверна система керування версіями.

Visual Studio .NET 2002 (кодове ім'я Rainier; внутрішня версія 7.0) — випущена в лютому 2002 (містила .NET Framework 1.0). Service Pack 1 для Visual Studio .NET (2002) випущений в березні 2005. Бета — версія була доступною в 2001 році. Найбільшою зміною було впровадження менеджера коду. Застосунки, які були розроблені за допомогою Visual Studio .NET, не компілювались в машинну мову, а перетворювались у формат, який мав назву Microsoft Intermediate Language (MSIL) або Common Intermediate Language (CIL). Коли MSIL-застосунок використовували, він автоматично компілювався в машинну мову для даної платформи, це робило код кросплатформним, що давало змогу виконувати його на різних платформах. Проте такі застосунки могли використовуватись тільки на платформах що підтримували Common Language Infrastructure. Це робило можливим використання застосунків в операційних системах Linux або Mac OS використовуючи спеціальні програми як, Mono та DotGNU. Пакет був випущений одразу в чотирьох версіях: Academic, Professional, Enterprise Developer та Enterprise Architect. Було вперше представлено нову мову програмування C# (сі — шарп), яка була спеціально розроблена для використання в Visual Studio .NET. Також було представлено спадкоємця Visual J++ що мав назву Visual J#. За допомогою Visual Studio .NET можна було створювати звичайні застосунки та вебсайти (використовуючи ASP.NET та Web сервіси). У травні 2005 року було випущено пакет оновлень для Visual Studio .NET.

Visual Studio побудована в архітектурі, що підтримує можливість використання доповнень (Add-Ins), — плагінів від сторонніх розробників, що дає змогу розширювати можливості середовища розробки.

Деякі з найпопулярніших доповнень:

- DevPartner Studio
- Visual Assist
- ReSharper

- IncrediBuild
- Workspace Whiz
- Viva64

Visual Studio 2022 включає велике оновлення для редакторів Blazor і Razor, а також нові можливості для гарячого перезавантаження в ASP.NET Core, включаючи гаряче перезавантаження при збереженні файлу або застосуванні змін до CSS-файлів в реальному часі.

2.2 WPF

Windows Presentation Foundation (WPF, кодова назва — Avalon) — графічна (презентаційна) підсистема (аналог WinForms), яка починаючи з .NET Framework 3.0 в складі цієї платформи. Має пряме відношення до XAML. WPF разом з .NET Framework 3.0 вбудована в Windows Vista, а також доступна для установки в Windows XP Service Pack 2 і Windows Server 2003.

Це перше реальне оновлення технологічного середовища призначеного для користувача інтерфейсу з часу випуску Windows 95. Воно включає нове ядро для заміни GDI і GDI+, використовувані в Windows Forms. WPF є високорівневим об'єктно-орієнтованим функціональним шаром (англ. framework), що дозволяє створювати двовимірні та тривимірні інтерфейси.[9]

XAML (скорочення від Extensible Application Markup Language — розширювана мова розмітки застосунків) є мовою розмітки, яку використовують для створення екземплярів об'єктів .NET. Хоча мова XAML — це технологія, що може бути застосовна до багатьох різних предметних областей, її головне призначення — конструювання інтерфейсів користувачів WPF. Інакше кажучи, документи XAML визначають розташування панелей, кнопок та інших елементів керування, що становлять вікна в застосунку WPF. Малоймовірно, що вам доведеться писати код XAML вручну. Замість цього ви використовуєте інструмент, що генерує необхідний код XAML. [9] Існує кілька підмножин XAML:

- WPF XAML включає елементи, що описують вміст WPF з розряду векторної графіки, елементів керування й документів. У цей час це найважливіше застосування XAML.
- XPS XAML — частина WPF XAML, що визначає XML-подання відформатованих електронних документів. Вона опублікована як окремий стандарт XML Paper Specification (XPS).
- Silverlight XAML — підмножина WPF XAML, призначена для Silverlight-застосунків. Можна відвідати сайт https://web.archive.org/web/20121030190821/http://www.silverlight.net/, щоб ознайомитися з деталями.
- WF XAML включає елементи, що описують уміст Windows Workflow Foundation (WF).

Переваги WPF:

1) Використання традиційних мов .NET-платформи - C#, F# та VB.NET для створення логіки програми

2) Можливість декларативного визначення графічного інтерфейсу за допомогою спеціальної мови розмітки XAML, що базується на xml і представляє альтернативу програмному створенню графіки та елементів керування, а також можливість комбінувати XAML та C#/VB.NET

3) Незалежність від роздільної здатності екрана: оскільки у WPF всі елементи вимірюються в незалежних від пристрою одиницях, програми на WPF легко масштабуються під різні екрани з різною роздільною здатністю.

4) Нові можливості, яких складно було досягти WinForms, наприклад, створення тривимірних моделей, прив'язка даних, використання таких елементів, як стилі, шаблони, теми та ін.

5) Хороша взаємодія з WinForms, завдяки чому, наприклад, у додатках WPF можна використовувати традиційні елементи керування WinForms.

6) Багаті можливості створення різних додатків: це і мультимедіа, і двомірна і тривимірна графіка, і багатий набір вбудованих елементів управління, а також можливість самим створювати нові елементи, створення анімацій, прив'язка даних, стилі, шаблони, теми та багато іншого

7) Апаратне прискорення графіки - незалежно від того, чи працюєте ви з 2D або 3D, графікою або текстом, всі компоненти програми транслюються в об'єкти, зрозумілі Direct3D, а потім візуалізуються за допомогою процесора на відеокарті, що підвищує продуктивність, робить графіку більш плавною.

8) Створення програм під безліч ОС сімейства Windows

У той же час WPF має певні обмеження. Незважаючи на підтримку тривимірної візуалізації, для створення програм з великою кількістю тривимірних зображень, перш за все ігор, краще використовувати інші засоби - DirectX або спеціальні фреймворки, такі як Monogame або Unity. [9]

Також варто враховувати, що в порівнянні з додатками на Windows Forms обсяг програм на WPF та споживання ними пам'яті в процесі роботи в середньому дещо вищий. Але це з лишком компенсується ширшими графічними можливостями та підвищеною продуктивністю при малюванні графіки.

Крім того, незважаючи на те, що WPF працює поверх платформи .NET 5/6/7, але в силу природи WPF і залежності від компонентів Windows, на даний момент створювати програми на WPF можна тільки під ОС Windows.[9]

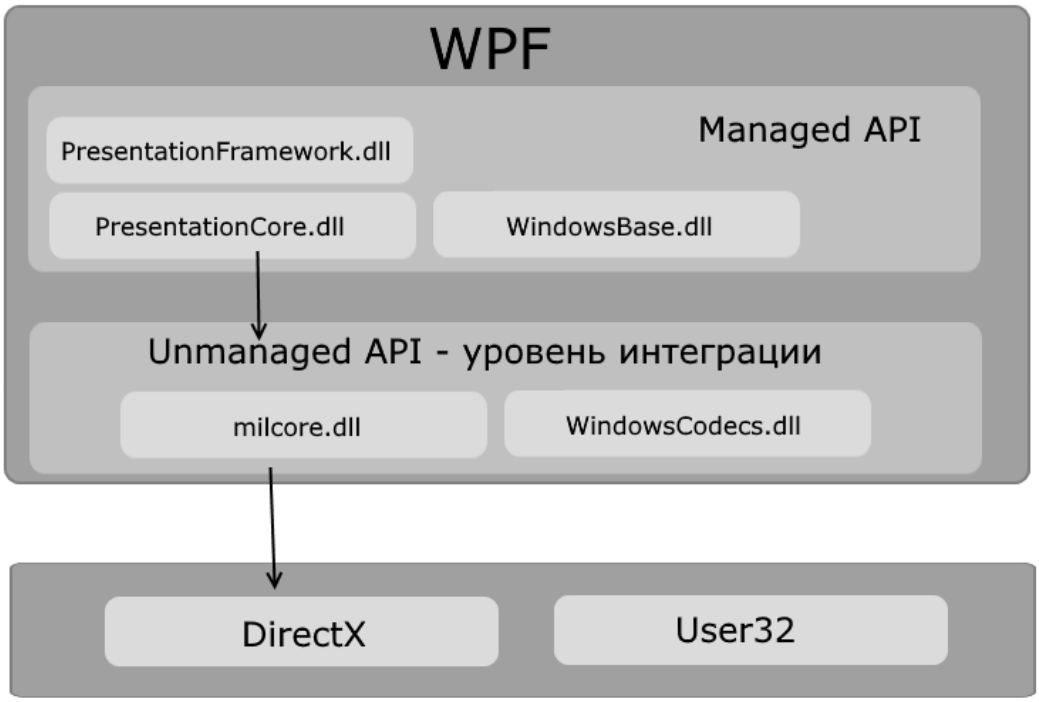

Рис. 2.1. «Архітектура WPF»

Як видно на схемі, WPF розбивається на два рівні: managed API та unmanaged API (рівень інтеграції з DirectX). Managed API (керований API-інтерфейс) містить код, який виконується під управлінням загальномовного середовища виконання .NET - Common Language Runtime. Цей API описує основний функціонал платформи WPF і складається з наступних компонентів:

1) PresentationFramework.dll: містить усі основні реалізації компонентів та елементів керування, які можна використовувати при побудові графічного інтерфейсу

2) PresentationCore.dll: містить усі базові типи для більшості класів з PresentationFramework.dll

3) WindowsBase.dll: містить низку допоміжних класів, які застосовуються у WPF, але можуть також використовуватися і поза цією платформою

4) Unmanaged API використовується для інтеграції вищого рівня з DirectX:

5) milcore.dll: власне забезпечує інтеграцію компонентів WPF із DirectX. Цей компонент написаний на некерованому коді (C/C++) взаємодії з DirectX.

6) WindowsCodecs.dll: бібліотека, яка надає низькорівневу підтримку для зображень у WPF

Ще нижче власне знаходяться компоненти операційної системи та DirectX, які роблять візуалізацію компонентів програми, або виконують іншу низькорівневу обробку. Зокрема, за допомогою низькорівневого інтерфейсу Direct3D, що входить до складу DirectX, відбувається трансляція

Тут також на одному рівні знаходиться бібліотека user32.dll. І хоча вище говорилося, що WPF не використовує цю бібліотеку для рендерингу та візуалізації, проте для низки обчислювальних завдань (що не включають візуалізацію) ця бібліотека продовжує використовуватись.

2.3 Патерн MVVM

Паттерн MVVM (Model-View-ViewModel) дозволяє відокремити логіку програми від візуальної частини (подання). Цей патерн є архітектурним, тобто він задає загальну архітектуру програми.

Цей патерн був представлений Джоном Госсманом (John Gossman) у 2005 році як модифікація шаблону Presentation Model і був спочатку націлений на розробку додатків WPF. І хоча зараз цей патерн вийшов за межі WPF і застосовується в різних технологіях, у тому числі при розробці під Android, iOS, проте WPF є досить показовою технологією, яка розкриває можливості даного патерну.

MVVM складається з трьох компонентів: моделі (Model), моделі уявлення (ViewModel) та уявлення (View).

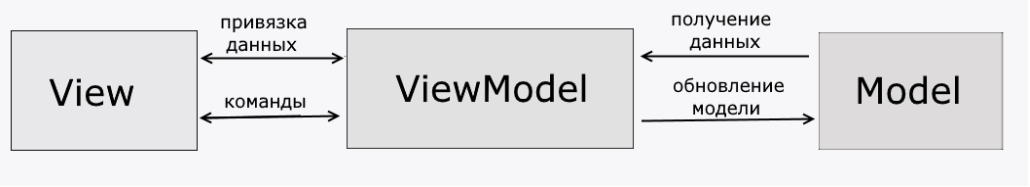

Рис. 2.2. Модель (Model), модель уявлення (ViewModel) та уявлення (View).

1. Model

Модель описує дані, що використовуються в додатку. Моделі можуть містити логіку, безпосередньо пов'язану з цими даними, наприклад, логіку валідації властивостей моделі. У той же час модель не повинна містити жодної логіки, пов'язаної з відображенням даних та взаємодією з візуальними елементами керування.

Нерідко модель реалізує інтерфейси INotifyPropertyChanged або INotifyCollectionChanged, які дозволяють повідомляти систему про зміни властивостей моделі. Завдяки цьому полегшується прив'язка до подання, хоча знову ж таки пряма взаємодія між моделлю і поданням відсутня.

2. View

View або подання визначає візуальний інтерфейс, через який користувач взаємодіє з програмою. Стосовно WPF уявлення - це код xaml, який визначає інтерфейс у вигляді кнопок, текстових полів та інших візуальних елементів.

Хоча вікно (клас Window) у WPF може містити як інтерфейс у xaml, так і прив'язаний до нього код C#, проте в ідеалі код C# не повинен містити якоїсь логіки, крім хіба що конструктора, який викликає метод InitializeComponent і виконує початкову ініціалізацію вікна . Вся основна логіка програми виноситься в компонент ViewModel.

Але коли в файлі пов'язаного коду все може бути певна логіка, яку складно продати в рамках патерну MVVM в ViewModel.

Подання не обробляє події за рідкісним винятком, а виконує дії в основному за допомогою команд.

3. ViewModel

ViewModel або модель подання пов'язує модель і подання через механізм прив'язування даних. Якщо в моделі змінюються значення властивостей, при реалізації моделлю інтерфейсу INotifyPropertyChanged автоматично йде зміна даних, що відображаються в поданні, хоча безпосередньо модель і уявлення не пов'язані.

ViewModel також містить логіку з отримання даних з моделі, які потім передаються у виставу. І також VewModel визначає логіку щодо оновлення даних у моделі.

Оскільки елементи подання, тобто візуальні компоненти типу кнопок, не використовують події, подання взаємодіє з ViewModel за допомогою команд.

Наприклад, користувач бажає зберегти введені в текстове поле дані. Він натискає кнопку і тим самим відправляє команду в ViewModel. А ViewModel вже отримує передані дані і відповідно до них оновлює модель.

Підсумком застосування патерну MVVM є функціональний поділ програми на три компоненти, які простіше розробляти та тестувати, а також надалі модифікувати та підтримувати.

2.4 SQLite

SQLite — полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92. Початковий код SQLite поширюється як суспільне надбання (англ. public domain), тобто може використовуватися без обмежень та безоплатно з будь-якою метою. Фінансову підтримку розробників SQLite здійснює спеціально створений консорціум, до якого входять такі компанії, як Adobe, Oracle, Mozilla, Nokia, Bentley[en] і Bloomberg.[18[

З 2018 року SQLite, як й JSON та CSV, рекомендований Бібліотекою Конгресу США формат зберігання структурованого набору даних[1].

У 2005 році проєкт отримав нагороду Google-O'Reilly Open Source Awards.

Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних

(включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу.

Кілька процесів або потоків можуть одночасно без жодних проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, коли жодних інших запитів у цей час не обслуговується; інакше спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.[20]

У комплекті постачання йде також функціональна клієнтська частина у вигляді виконуваного файлу sqlite3, за допомогою якого демонструється реалізація функцій основної бібліотеки. Клієнтська частина працює з командного рядка, і дозволяє звертатися до файлу БД на основі типових функцій ОС.[22]

Завдяки архітектурі рушія можливо використовувати SQlite як на вбудовуваних (embedded) системах, так і на виділених машинах з гігабайтними масивами даних.

Особливості SQLite[23]

- транзакції атомарні, послідовні, ізольовані, і міцні (ACID) навіть після збоїв системи і збоїв живлення
- Встановлення без конфігурації — не потребує ані установки, ані адміністрування
- Реалізує значну частину стандарту SQL92[2]
- База даних зберігається в одному крос-платформовому файлі на диску
- Підтримка терабайтних розмірів баз даних і гігабайтного розміру рядків і BLOBів
- Малий розмір коду: менше ніж 350KB повністю налаштований, і менш 200KB з опущеними додатковими функціями

- Швидший за популярні рушії клієнт-серверних баз даних для найпоширеніших операцій
- Простий, легкий у використанні API
- Написана в ANSI C, включена прив'язка до TCL; доступні також прив'язки для десятків інших мов
- Добре прокоментований початковий код зі 100 % тестовий покриттям гілок
- Доступний як єдиний файл початкового коду на ANSI C, який можна легко вставити в інший проєкт
- Автономність: немає зовнішніх залежностей
- Крос-платформовість: з коробки підтримується Unix (Linux і Mac OS X), OS/2, Windows (Win32 і WinCE). Легко переноситься на інші системи
- Сирці перебувають в суспільному надбанні
- Поставляється з автономним клієнтом інтерфейсу командного рядка, який може бути використаний для управління базами даних SQLite.

2.5 Мова C#

C # (вимовляється як "сі Шарп") - сучасна об'єктно-орієнтована і типово безпечна мова програмування. C # дозволяє розробникам створювати безліч типів безпечних і надійних програм, які працюють в екосистемі .NET. C # відноситься до широко відомої родини мов C. [5]

C # - це об'єктно-і компонентно-орієнтована мова програмування. C # надає мовні конструкції для безпосередньої підтримки такої концепції роботи. Завдяки цьому C # підходить для створення і застосування програмних компонентів. З моменту створення мова C # збагатилася функціями для підтримки нових робочих навантажень і сучасними рекомендаціями по розробці ПЗ.[10]

Ось лише кілька функцій мови C #, які дозволяють створювати надійні та стійкі додатки.

Прибирання сміття - автоматично звільняє пам'ять, зайняту недоступними невикористовуваними об'єктами.

Типи, що допускають значення null, забезпечують захист від змінних, які не посилаються на виділені об'єкти.

Обробка винятків надає структурований і розширюваний підхід до виявлення помилок і відновлення після них.

Лямбда-вирази підтримують прийоми функціонального програмування. Синтаксис LINQ створює загальний шаблон для роботи з даними з будь-якого джерела.

Підтримка мов для асинхронних операцій надає синтаксис для створення розподілених систем.

У C # діє єдина система типів. Всі типи C #, включаючи типи-примітиви, такі як int і double, успадковують від одного кореневого типу object. Всі типи використовують загальний набір операцій, а значення будь-якого типу можна зберігати, передавати і обробляти схожим чином. Більш того, C # підтримує як визначаються користувачами посилальні типи, так і типи значень. C # дозволяє динамічно виділяти об'єкти та зберігати спрощені структури в стеці. [17]

C # підтримує універсальні методи і типи, що забезпечують підвищену безпеку типів і продуктивність.

C # надає ітератори, які дозволяють розробникам класів колекцій визначати призначені для користувача варіанти поведінки для клієнтського коду.

У C # особлива увага приділяється управлінню версіями для забезпечення сумісності програм і бібліотек при їх зміні. Питання управління версіями істотно вплинули на такі аспекти розробки C #, як роздільні модифікатори virtual і override, правила вирішення перевантаження методів і підтримка явного оголошення членів інтерфейсу.

Програми C # виконуються в .NET, віртуальній системі виконання, що викликає загальномовне середовище виконання (CLR) і набір бібліотек класів. Середа CLR - це реалізація загальномовної інфраструктури мови (CLI), що є міжнародним стандартом, від корпорації Майкрософт. CLI є основою для

створення середовищ виконання і розробки, в яких мови і бібліотеки прозоро працюють один з одним.

Вихідний код, написаний на мові C # компілюється в проміжну мову (IL), яка відповідає специфікаціям CLI. Код на мові IL і ресурси, в тому числі растрові зображення і рядки, зберігаються в збірці, зазвичай з розширенням .dll. Збірка містить маніфест з інформацією про типи, версії, мовою і регіональних параметрах для цієї збірки.

При виконанні програми C # збірка завантажується в середу CLR. Середа CLR виконує JIT-компіляцію з коду на мову IL в інструкції машинної мови. Середа CLR також виконує інші операції, наприклад, автоматичну збірку сміття, обробку винятків і управління ресурсами. Код, що виконується середовищем CLR, іноді називають "керованим кодом", щоб підкреслити відмінності цього підходу від "некерованого коду", який відразу компілюється в машинний мову для певної платформи.[24]

Забезпечення взаємодії між мовами є ключовою особливістю .NET.

Код IL, створений компілятором C #, відповідає специфікації загальних типів (CTS).

Код IL, створений з коду на C #, може взаємодіяти з кодом, створеним з версій .NET для мов F #, Visual Basic, C ++ і будь-яких інших з більш ніж 20 мов, сумісних з CTS.

Одна збірка може містити кілька модулів, написаних на різних мовах .NET, і всі типи можуть посилатися один на одного, як якщо б вони були написані на одній мові.

На додаток до runtime-служб .NET також включає розширені бібліотеки. Ці бібліотеки підтримують безліч різних робочих навантажень. Вони впорядковані по просторах імен, які надають різні корисні можливості: від операцій файлового введення і виведення до управління рядками і синтаксичного аналізу XML, від платформ веб-додатків до елементів управління Windows Forms. Зазвичай додаток C # активно використовують бібліотеку класів .NET для вирішення типових задач.[10]

Розглянемо докладніше сферу застосування мови сі шарп, де вона проявляє себе якнайкраще:

1. Розробка веб-додатків

C# активно використовується у сфері веб-розробки. Одним із ключових інструментів для створення веб-додатків на C# є ASP.NET – платформа, що надає розробникам потужні засоби для побудови сучасних і продуктивних веб-додатків. ASP.NET пропонує широкий набір інструментів для роботи з веб-технологіями, базами даних, безпекою та іншими аспектами веб-розробки.

C# і ASP.NET дають змогу створювати динамічні та масштабовані веб-додатки, обробляти запити від користувачів і взаємодіяти з базами даних. Це означає, що ви можете створювати різноманітні веб-сайти, від корпоративних порталів до електронних магазинів.

2. Створення додатків для Windows

C# має тісну інтеграцію з операційною системою Windows, що робить його чудовим вибором для створення настільних додатків під цю платформу. За допомогою технології Windows Forms або більш сучасної Universal Windows Platform (UWP), ви можете розробляти додатки з графічним інтерфейсом користувача (GUI), які інтегровані з операційною системою Windows.

3. Розробка ігор, зокрема з використанням ігрового рушія Unity

C# також популярний серед розробників ігор, особливо завдяки ігровому рушію Unity. Це потужний засіб для створення 2D і 3D ігор, а також віртуальної та доповненої реальності. Одна з його ключових переваг — підтримка C# для програмування ігрової логіки.

Використовуючи C# і Unity, розробники можуть створювати ігри для різних платформ, включно з ПК, консолями, мобільними пристроями і навіть віртуальними шоломами. Unity надає багату бібліотеку ресурсів, інструменти для моделювання та анімації об’єктів, а також підтримує фізичний рушій для створення реалістичної поведінки об’єктів у грі.

Ці три галузі — лише невеликий огляд того, де використовується мова сі шарп. Можливості цієї мови розширюються щодня, і її гнучкість робить її універсальним інструментом для безлічі різних проєктів.

2.6 Висновки

Використання Visual Studio 2022, C#, SQLite, WPF та патерну MVVM у розробці програмного забезпечення виявляється дуже ефективним та потужним підходом. Інтегроване середовище розробки Visual Studio 2022 забезпечує зручність та продуктивність при написанні коду, тестуванні та налагодженні додатків.

Мова програмування C# забезпечує високий рівень абстракції та об'єднує в собі елементи ефективності та зручності в синтаксисі, дозволяючи розробникам швидко створювати якісний код, який легко розуміти та підтримувати. Об'єктно-орієнтований підхід C# сприяє структуруванню програм та полегшує їхнє розширення.

Застосування мови C# в поєднанні з іншими технологіями визначає високу якість та функціональність програмного забезпечення. Ця комбінація сприяє не лише продуктивності та легкості розробки, але й відкриває нові можливості для розширення та удосконалення продукту в майбутньому.

SQLite, як легка та компактна система керування базами даних, ідеально підходить для проектів з обмеженими ресурсами, таких як мобільні додатки. Використання WPF дозволяє створювати сучасні та естетично привабливі інтерфейси користувача для операційної системи Windows.

Впровадження патерну MVVM в контексті розробки з Visual Studio 2022, SQLite та WPF дозволяє ефективно розділити бізнес-логіку, представлення та логіку взаємодії з користувачем. Це сприяє полегшенню тестування, керуванню кодом та робить програмний код більш підтримуваним.

Узагальнюючи, комбінація Visual Studio 2022, C#, SQLite, WPF та MVVM надає потужний і гнучкий інструментарій для розробки високоякісних, продуктивних та легко підтримуваних додатків для операційної системи Windows.

РОЗДІЛ 3
СТРУКТУРА СИСТЕМИ ТА ЇЇ РЕАЛІЗАЦІЯ

3.1 Загальна модель системи. Опис модулів

Програма розроблена за патерном MVVM і виглядає наступним чином (Рис.3.1.):

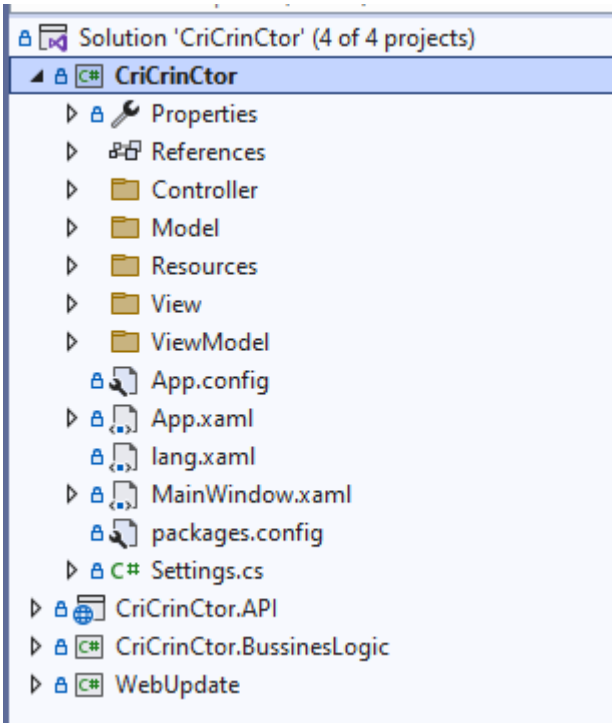

Рис.3.1. «Загальна архітектура програми»

Модуль розбитий на кілька папок, а саме:

1. Model
2. View
3. ViewModel

Згідно до застосованого патерну MVVM.

В папці Model містяться наступні моделі:

1 RichTextBoxExtention.cs – містить логіку для роботи із текстом;

2 Content.cs – модель, відповідна для формування контейнеру зі змістом даних;

3 EnumBooleanConverter.cs – модель для конвертування значення enum-а до списку переліку констант;

4 Enums.cs – модель, що зберігає enum-и;

5 Formulas.cs – модель, відповідальна за зберігання інформації типу Формула, який використовується для відображення формул;

6 ImageExtention.cs – модель, відповідна за роботу з зображеннями;

7 Images.cs – клас для зберігання інформації щодо типу Зображення,який використовується при його відображенні;

8 InfoParameter.cs – модель, що зберігає інформацію про певний код інструкції;

9 Inputs.cs – клас для зберігання інформації типу Змінна, який використовується при її відображенні;

10 Manufacturer.cs – клас для категорії;

11 Operation.cs – клас для пунктів інструкції;

12 OperationNumberVisible.cs – клас для правильного порядку відображення операцій, які входять в інструкцію;

13 PartNumber.cs – модель для ідентифікатора інструкції;

14 RowTable.cs – модель яка містить логіку для доповненої роботи із таблицями;

15 Step.cs – модель для кроків в інструкції;

16 Tables.cs – модель для зберігання інформації щодо типу Таблиця який використовується для відображення компонента;

17 Texts.cs – модель для зберігання інформації щодо типу Текст який використовується для відображення компонента;

18 Types.cs – модель, яка містить дані про тип інструкції;

В папці View містяться відповідні в’юшки для відображення вмісту даних, які репрезентують моделі:

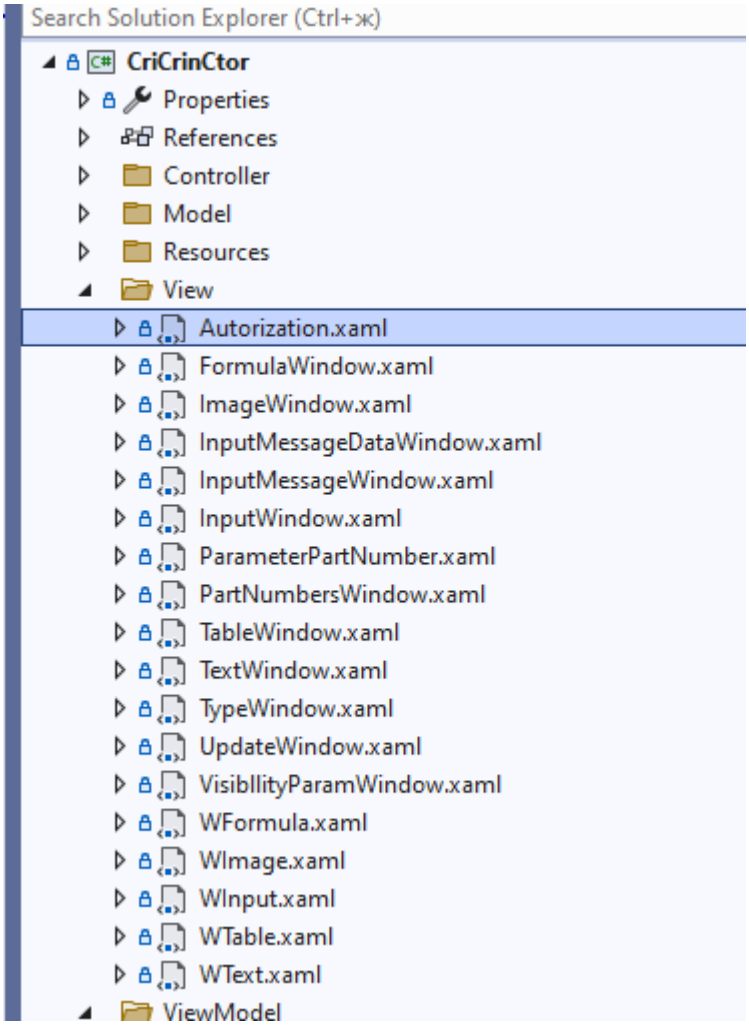

Рис.3.2. «Папка View»

Для зв'язку між абстракцією View та Model додатково є папка з ViewModels, які об’єднують ці два шари.

Також в модулі наявні папки Controller, Resources. В папці контролер наявний модуль CriLiteDBQuery (Рис.3.3).

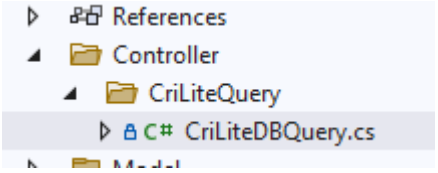

Рис.3.3. «Папка контролер»

Він є медіатором, що дозволяє комунікацію між додатком та базою даних, в ньому виконуються різні запити, CRUD операції та, в цілому, спілкування з базою.

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.

- Image:

Компонент для можливості додавання зображень, дозволяє завантажувати та відображати зображення в інструкції; додатково надана можливість переміщення компонента.

– Text:

Компонент, що надає можливість вводити та редагувати текстові дані, що стосуються інструкцій. Забезпечено можливості форматування, вирівнювання та вибору стилів тексту, можлива інтеграція з іншими компонентами, такими як Рисунок, для вставки їх у текстові блоки.

– Table:

Компонент надає можливість працювати з таблицями (створення, редагування таблиць з заданною кількістю стовпців та строк) ; додатково присутня можливість форматування таблиць, встановлення ширини стовпців, вирівнювання тексту тощо.

– Formula:

Компонент дозволяє вводити, відображати, редагувати математичні формули. Формули можна вставляти в текст.

– Variable:

Цей компонент відповідає за відстеження змін у документах. Забезпечено історію редагувань та можливість відновлення попередніх версій; додатково, компонент може підтримувати колаборативну роботу, де декілька користувачів можуть одночасно вносити зміни в документ.

Всі вищезгадані компоненти інструкції цілком і повністю задовільняють потреби користувача при її створенні, покриваючи більшість акутальних випадків.

3.2 Функціонал оновлення версії системи

Як було написано в першому розділі, система надає користувачу можливість оновлювати її за бажанням користувача, надсилаючи йому відповідні нотифікації про готові оновлення на сервері.

Технічно це реалізовано через ще один компонент системи - Модуль WebUpdate. Він є бібліотекою яка, дозволяє оновлювати систему. Структура модулю виглядає наступним чином (Рис 3.4.).

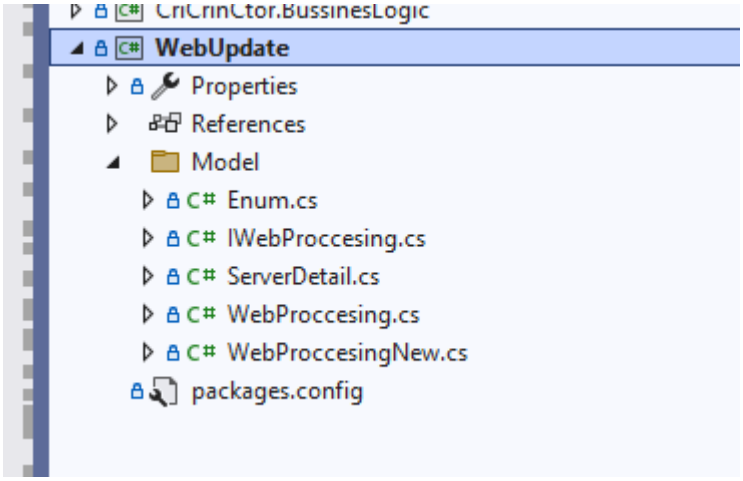

Рис. 3.4. Структура компоненту WebUpdate

Він розбитий на наступні складові:

1 Enum.cs – конфігураційний список для оновлення;

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

3 ServerDetail.cs – модель даних для оновлення системи, містить в собі реалізацію всього необхідного для оновлення версії системи функціоналу (порівняння версій, впровадження оновлення версії і т.д.);

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, є новою функціональністю системи, створеною внаслідок рефакторингу. Деякі версії системи досі підтримують старий функціонал завантаження.

```
public interface IWebProccesing
{
    6 references | HovhannesHovsepyan, 58 days ago | 1 author, 1 change
    bool CheckFileExistsOnServer(string FileCheck);

    5 references | HovhannesHovsepyan, 58 days ago | 1 author, 1 change
    bool DownloadFile(string URL, string NameDownloadFile);

    3 references | HovhannesHovsepyan, 58 days ago | 1 author, 1 change
    bool DownloadFile(string URL, string NameDownloadFile, IProgress<int> progress);

    3 references | HovhannesHovsepyan, 58 days ago | 1 author, 1 change
    void Stoping();
}
}
```

Рис.3.5. «Інтерфейс для оновлення версії системи»

Інтерфейс IWebProcessing (Рис. 3.5.) передбачує ряд наступних дій:

1. Перевірка наявності файлу з оновленням на сервері
2. Можливість завантажити цей файл по заданому URL-маршруту
3. Схоже до пункту 2. Але з можливістю відслідковування прогресу завантаження в реальному часі
4. Зупинка процесу завантаження оновлень

Демонстрація роботи функціональності оновлення системи буде представлена нижче у звіті у відповідному розділі.

3.3 Опис інших розроблених функціональностей

Оскільки робота є комплексною, то розроблені командою фічі були рівномірно розподілені між її учасниками. Ось ряд функціонльностей, створених автором в рамках цієї системи:

1. Проектування та імплементація структури бази даних

2. Впровадження зв'язку між базою даних та кодовою частиною
3. Впровадження мультимовності в системі
4. Впровадження можливості оновлення системи згідно з останньою версією
5. Розподілення та впровадження ролей між користувачами системи
6. Впровадження управління даними, пов'язаними з базовою структурою інструкції, а саме: категорії, підкатегорії, операції, кроки, параметри.

Кожна з вищезгаданих функціональностей буде продемонстрована більш детально далі в звіті, разом із демеонстрацією дизайна та полного функціонала системи.

3.4 Опис структури бази даних

База даних системи сформована в середовщі SQLiteStudio, є локальною для проекта і виглядає наступним чином (Рис.3.6.):

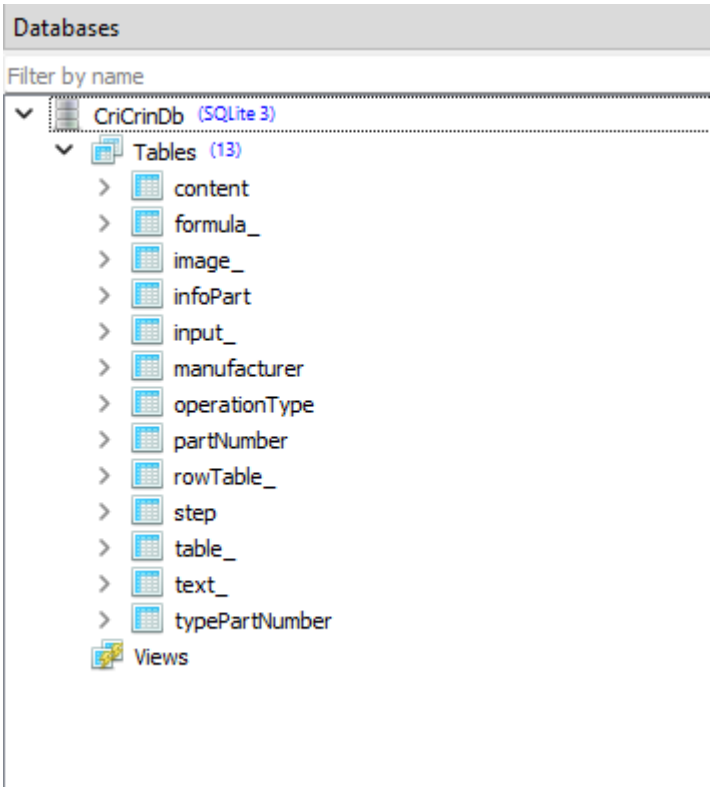

Рис.3.6. «База даних системи»

Вона містить ряд наступних таблиць з відповідними колонками:

1. Content – містить інформацію про вміст конетнту певного компонента. Має наступні колонки (Рис.3.7.)

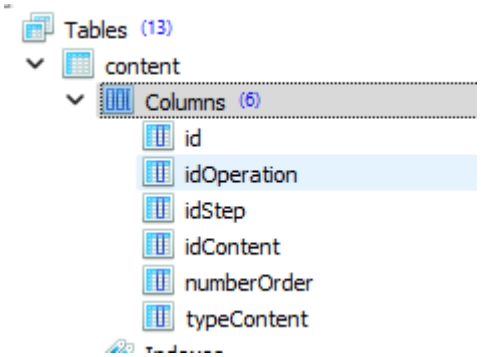

Рис.3.7. «Колонки таблиці content»

2. Formula – містить технічні дані, відповідні для компоненту типу Формула, має ряд колонок, що дозволяють функціональність мултимовності цього компоненту (Рис.3.8.)

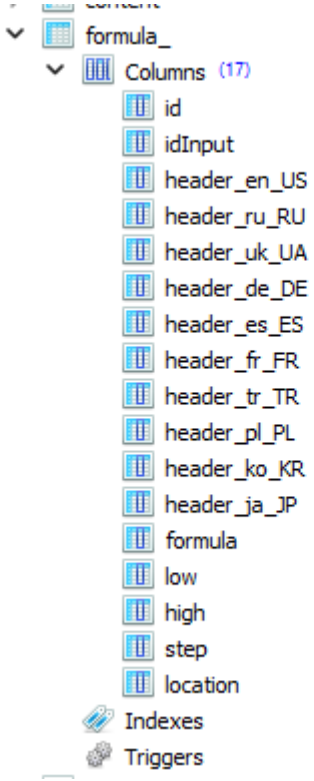

Рис.3.8. «Колонки таблиці formula»

3. Image – як і в випадку з формулою, містить технічні дані, відповідні для компоненту типу Зображення, а також має ряд колонок, що дозволяють підтримувати мультимовні зображення:

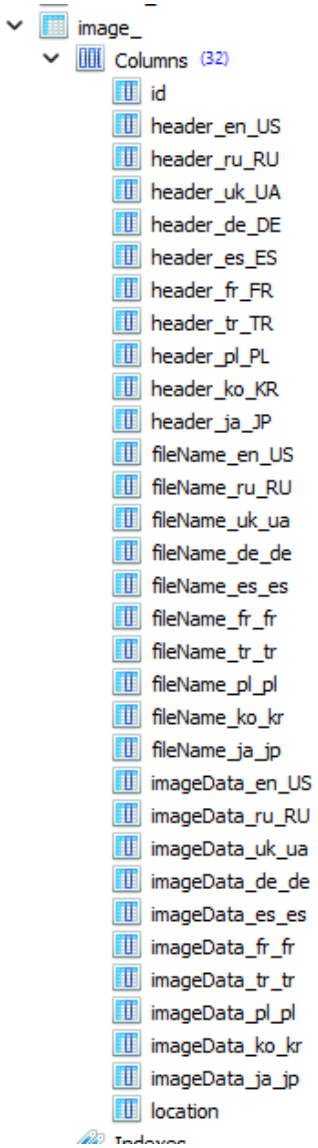

Рис.3.9. «Колонки таблиці image»

4. InfoPart – таблиця, яка містить інформацію про інструкцію та її параметри (інструкції розбиваються за категоріями, в інструкцій одного типу є однакові параметри) – Рис.3.10.

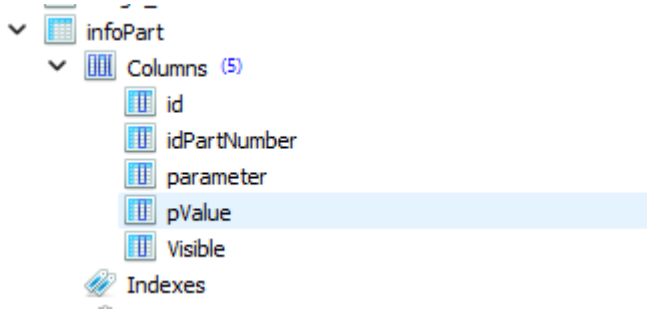

Рис.3.10. «Таблиця infoPart»

5. Manufacturer – таблиця відповідна за категорію, до якої належить інструкція, і містить ряд наступних колонок (див. Рис.3.11.)

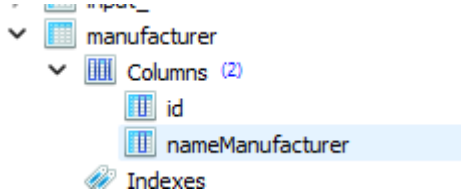

Рис.3.11. «Таблиця Manufacturer»

6. OperationType – відповідна за тип операції, підтримує мультимовність, містить ряд наступних колонок (Рис.3.12.).

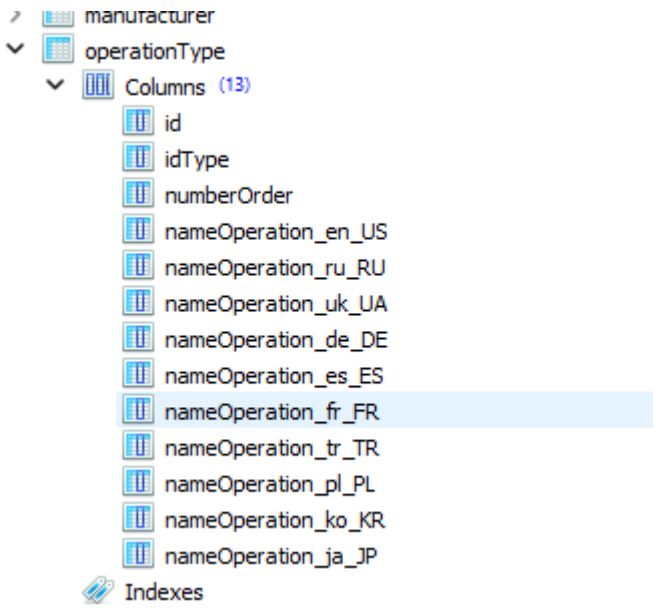

Рис.3.12. «Вміст таблиці operationType»

7. PartNumber – відповідна за ідентифікатори, що призначаються кожній з інструкцій. Має ряд наступних колонок (Рис.3.13.).

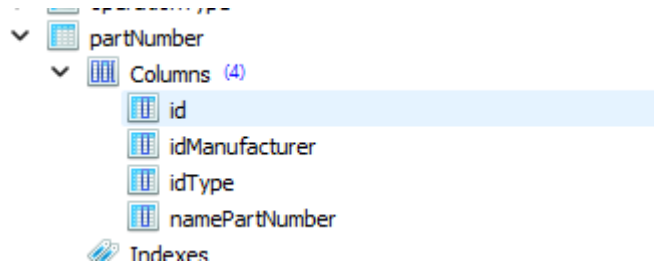

Рис.3.13. «Вміст таблиці PartNumber»

8. RowTable – відповідає за одиницю компоненту Таблиця, а саме – строку таблиці і відповідні їй дані, підтримує мультимовність, має ряд наступних колонок (Рис.3.14.):

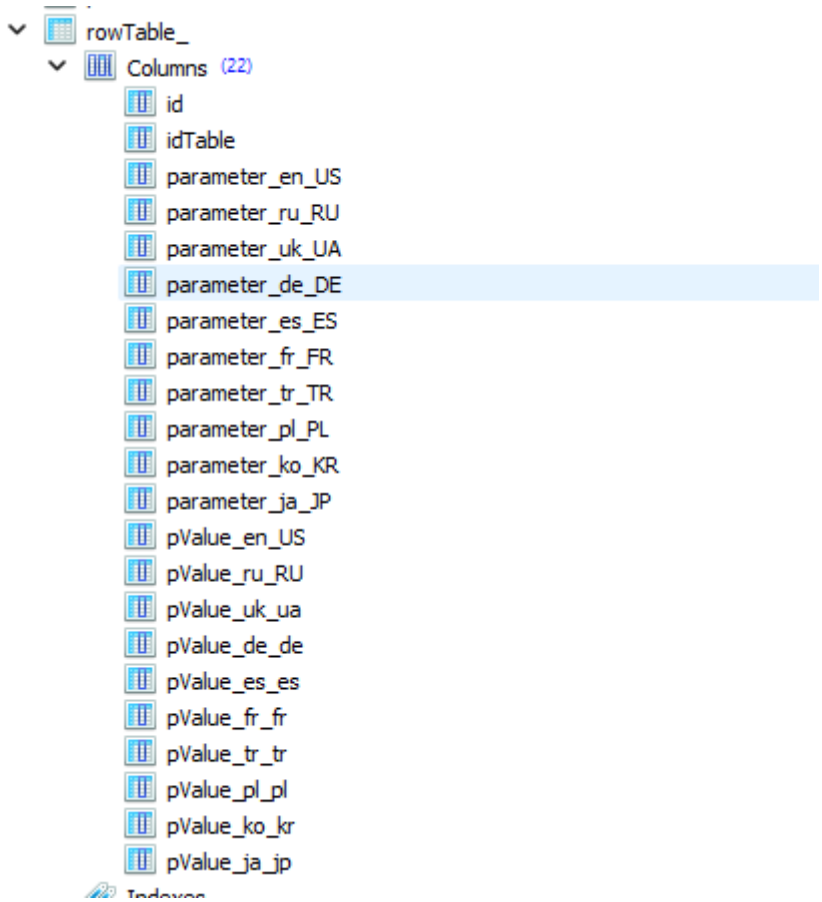

Рис.3.14. «Дизайн таблиці rowTable»

9. Step – відповідає кожному кроку, які описані в операції, що належить інструкції, підтримує мультимовність. Містить наступні колонки:

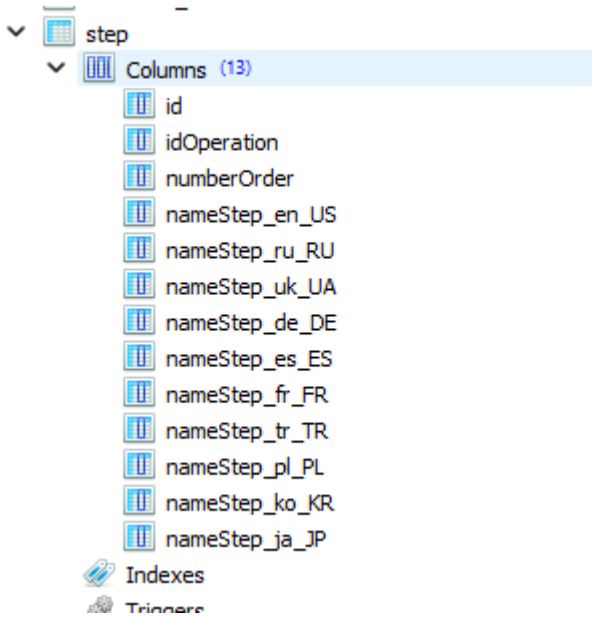

Рис.3.15. «Колонки таблиці step»

10. Table – таблиця (вибачте за тавтологію) компонента Таблиця, містить всі дані, які в ньому зберігаються (Рис.3.16):

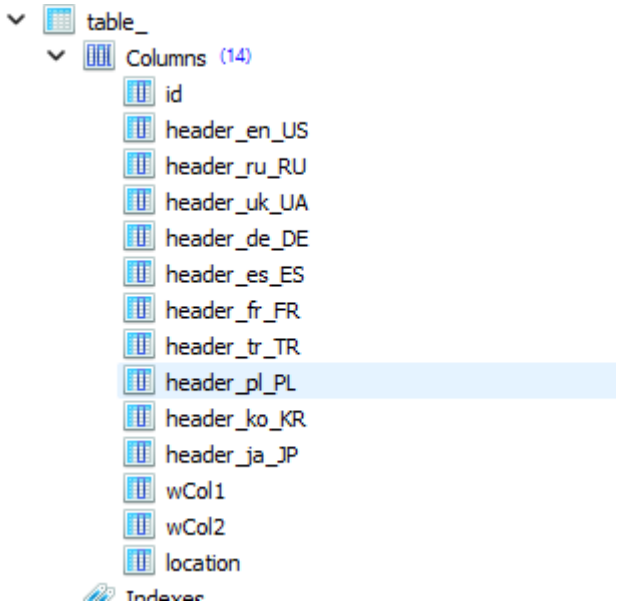

Рис.3.16. «Вміст колонок таблиці Table»

11. Text – зберігає дані, введені в компонент Текст, підтримує мултимовність, містить наступні колонки:

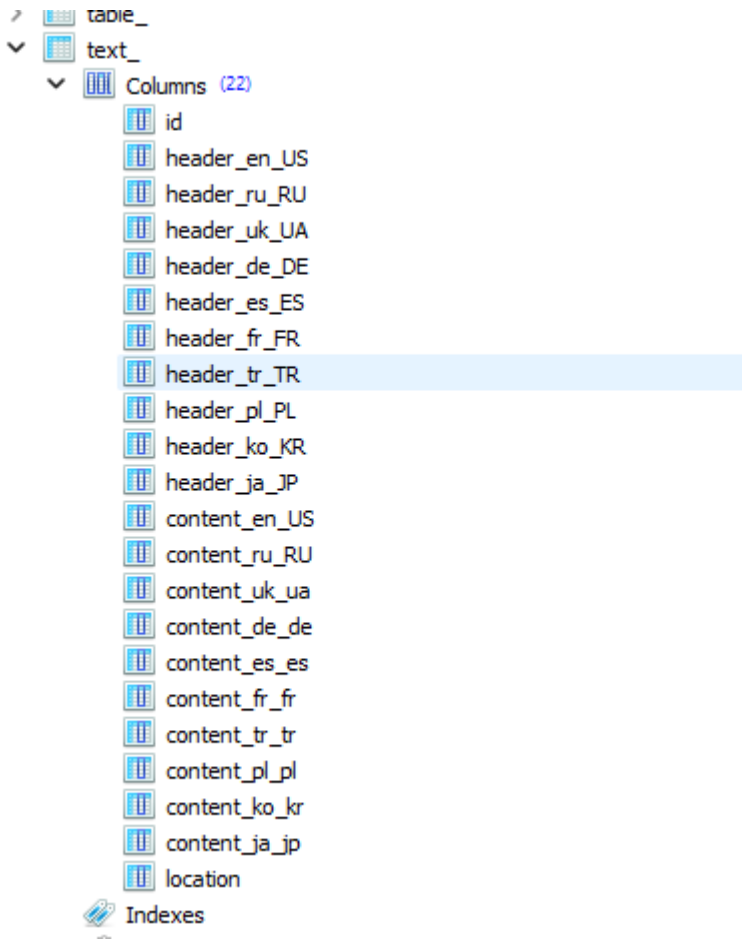

Рис.3.17. «Колонки таблиці Text»

В цілому база була детально спроектована згідно до нормальних форм і є оптимізованою.

3.5 Приклад використання та демонстрація дизайну

Перейдемо до демонстрації дизайну та повного функціоналу програми. На зображеннях нижче можна буде побачити один з окремих випадків використання програми на підприємстві з виготовлення інжекторів, проте це лише поодинокий випадок, система може дозволяти створювати не тільки

конструкторські інструкції, але й будь-які в цілому і є універсальним шаблоном та конструктором для створення будь-якого типу інструкцій.

Нижче (Рис.3.18) можна побачити головне вікно програми. Зліва – згорнуте меню з переліченими кроками та операціями, на головному вікні посередині – вміст поточного кроку операції.

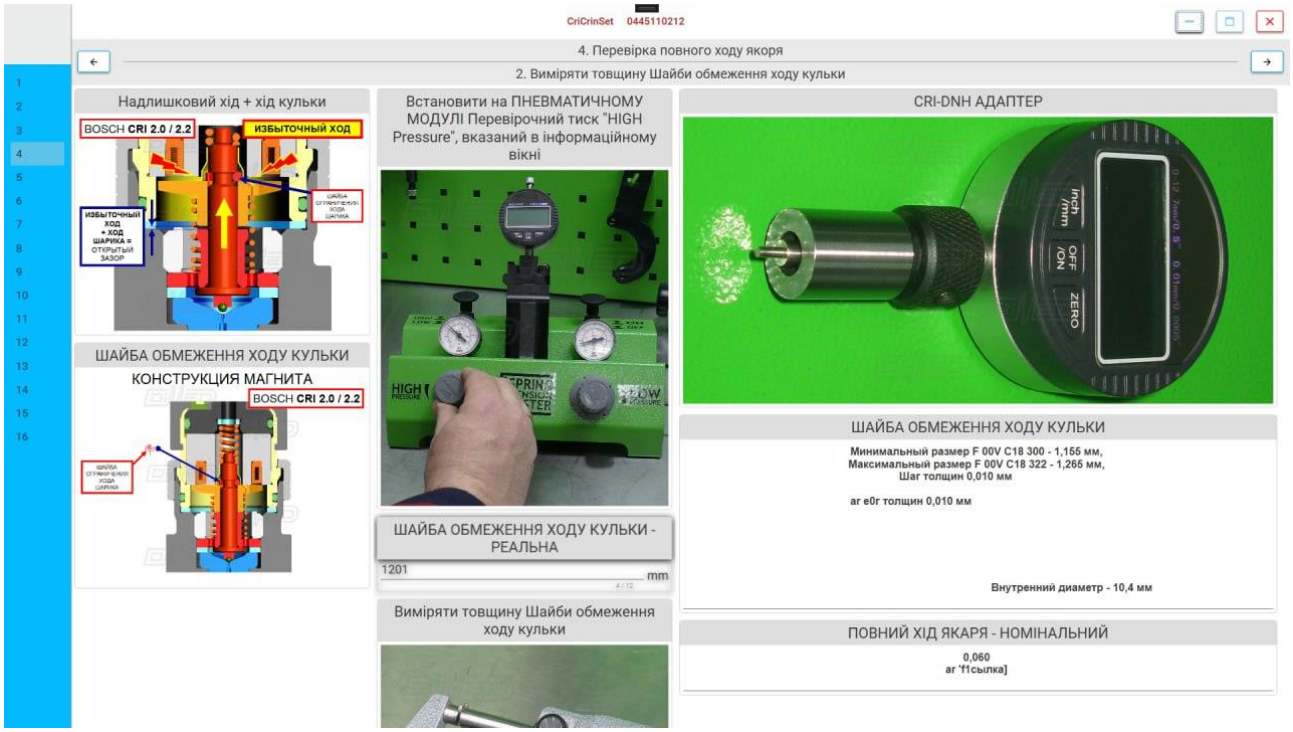

Рис. 3.18. Головне вікно програми

Розгорнемо меню і побачимо структуру власне поточної інструкції зліва (Рис. 3.19.):

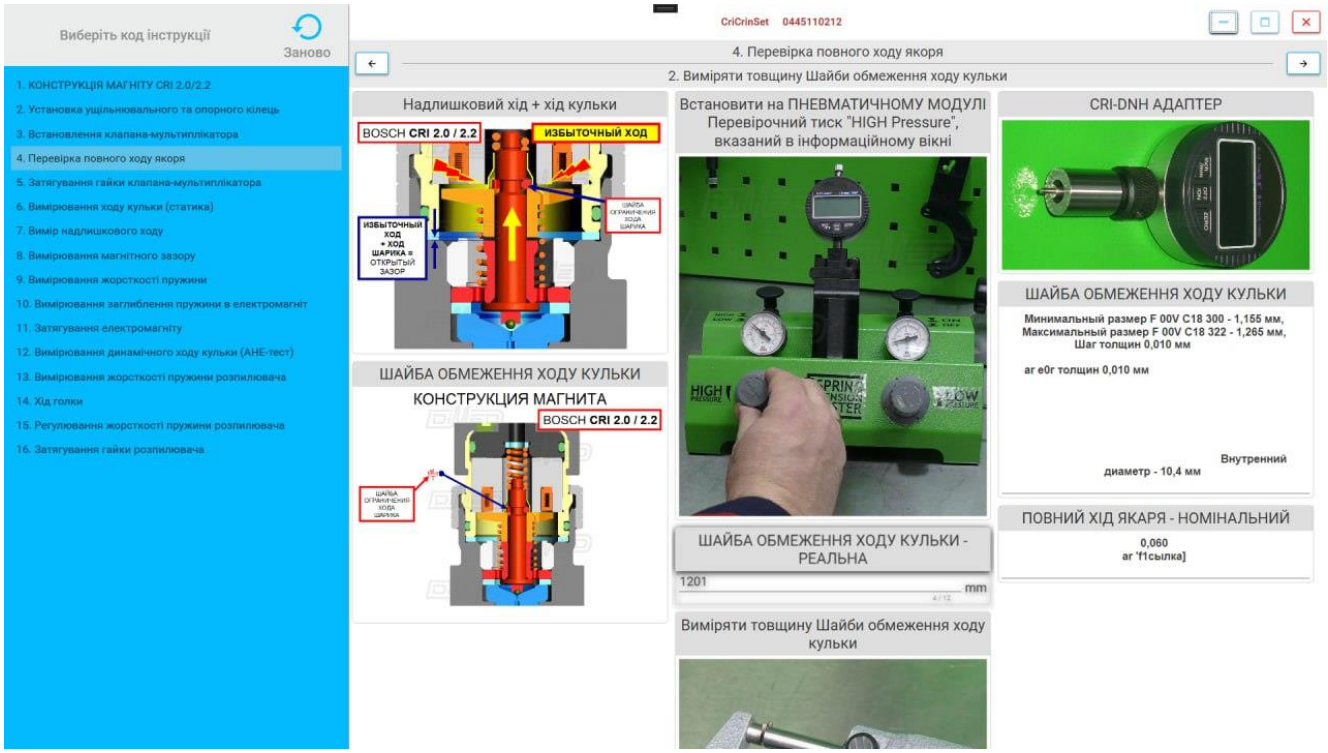

Рис.3.19. «Структура інструкції»

Можемо побачити, що інструкція містить в собі пронумеровані операції, а вони, в свою чергу, розбиті на кроки.

Також наявна можливість навігації по системі. Для цього варто перейти в головне меню (Рис.3.20).

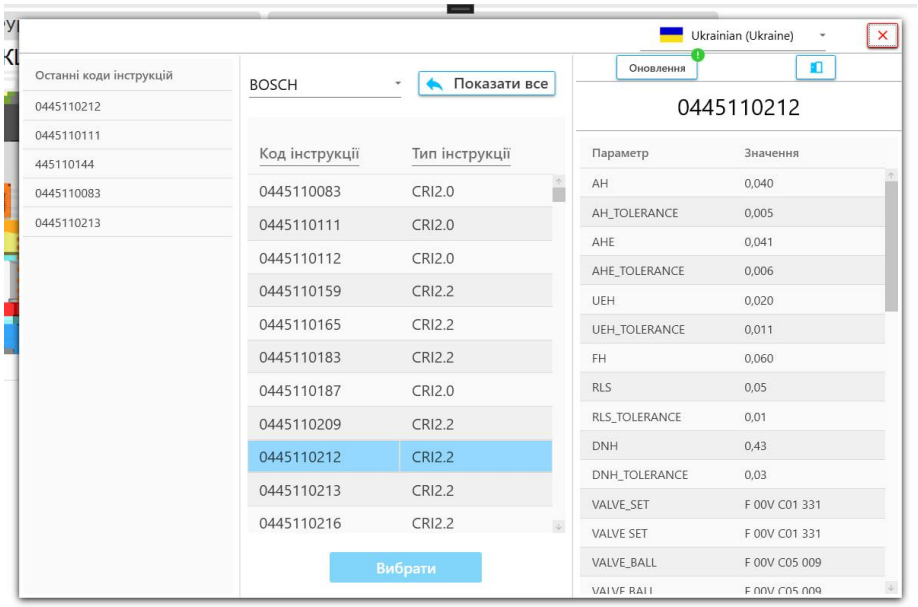

Рис.3.20. «Головне меню системи»

Тут можна побачити зліва історію відкриття інструкцій – останні коди інструкцій, також посередині – список всіх інструкцій, справа – набір параметрів для обраної в списку інструкції. В правому верхньому кутку можна спостерігати нотифікація про оновлення, а також випадаючий список з мовами.
Спробуємо встановити оновлення системи, для цього натиснемо на нотифікацію, отримаємо наступний результат (Рис.3.21).

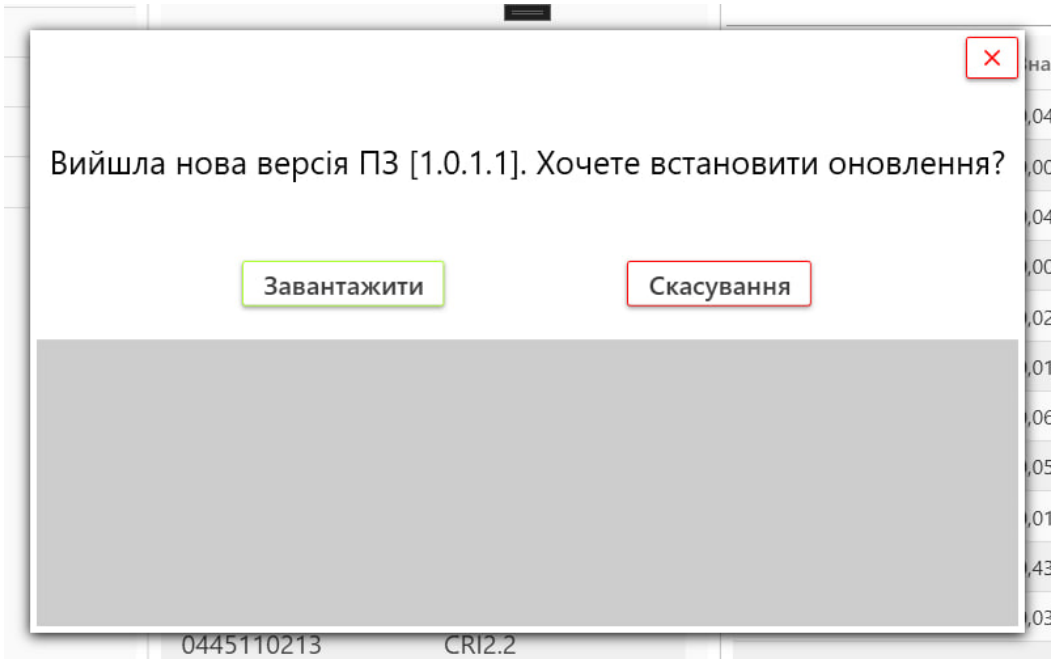

Рис.3.21. «Оновлення версії системи»

Після того, як натиснемо на «Завантажити», з’являється трекбар прогресу завантаження нової версії додатку (Рис.3.22):

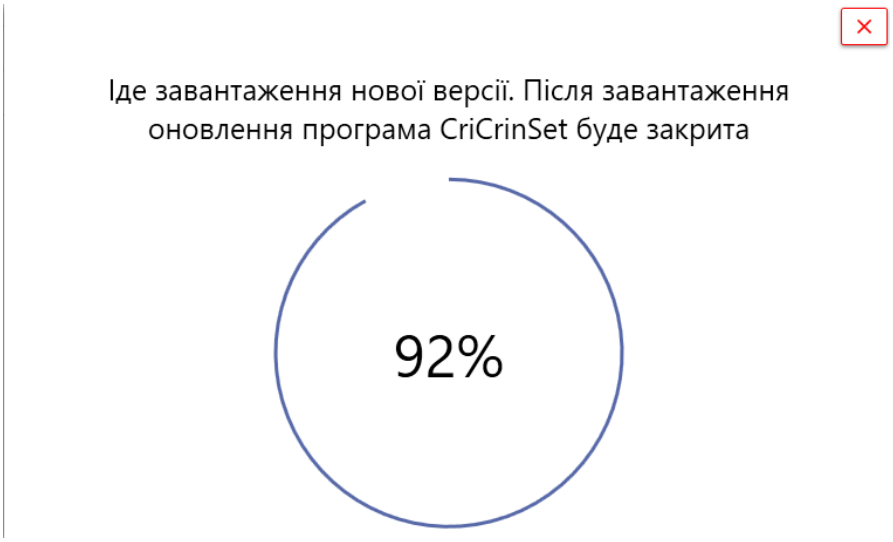

Рис.3.22. «Трекбар прогресу завантаження нової версії»

Після вдалого завантаження поточна програма закривається та запускається інсталятор, що дозволяє встановити оновлену версію додатку (Рис.3.23).

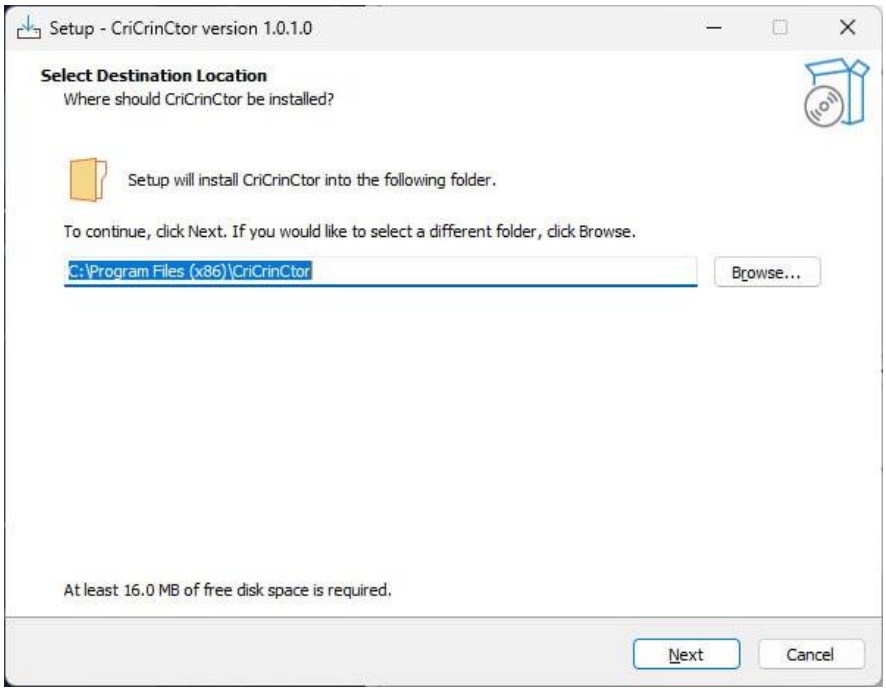

Рис.3.23. «Інсталятор нової версії програми»

Розглянемо основні функції системи. Як можна побачити, надана можливість створення та редагування інструкцій. Перейдемо перегляду можливих дій щодо кроків інструкцій (Рис.3.24.).

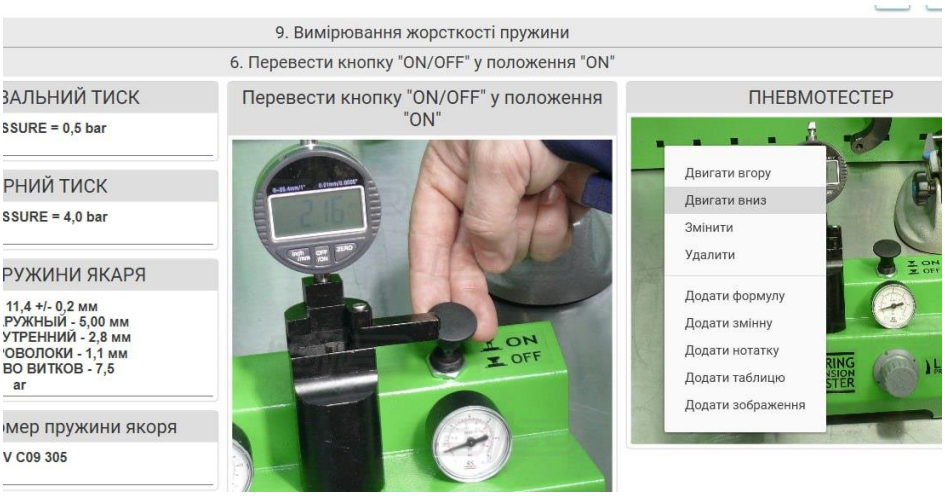

Рис.3.24. «Контекстне меню з можливостями»

Користувачу надається можливість рухати кроки в рамках операції, видаляти та змінювати їх. Також можна додатково додавати компоненти тих типів, які були описані в звіті вище. Розглянемо деякі з них.

1. Створення таблиці (Рис.3.25-3.26).

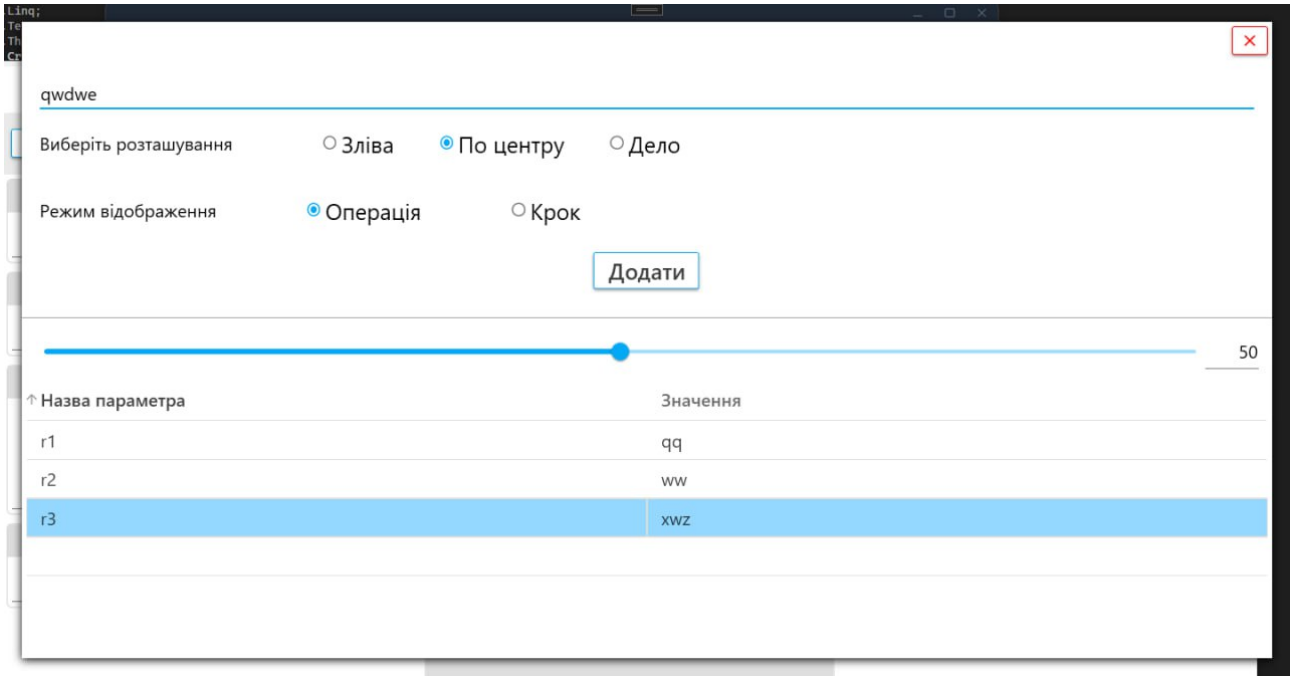

Рис.3.25 «Створення таблиці»

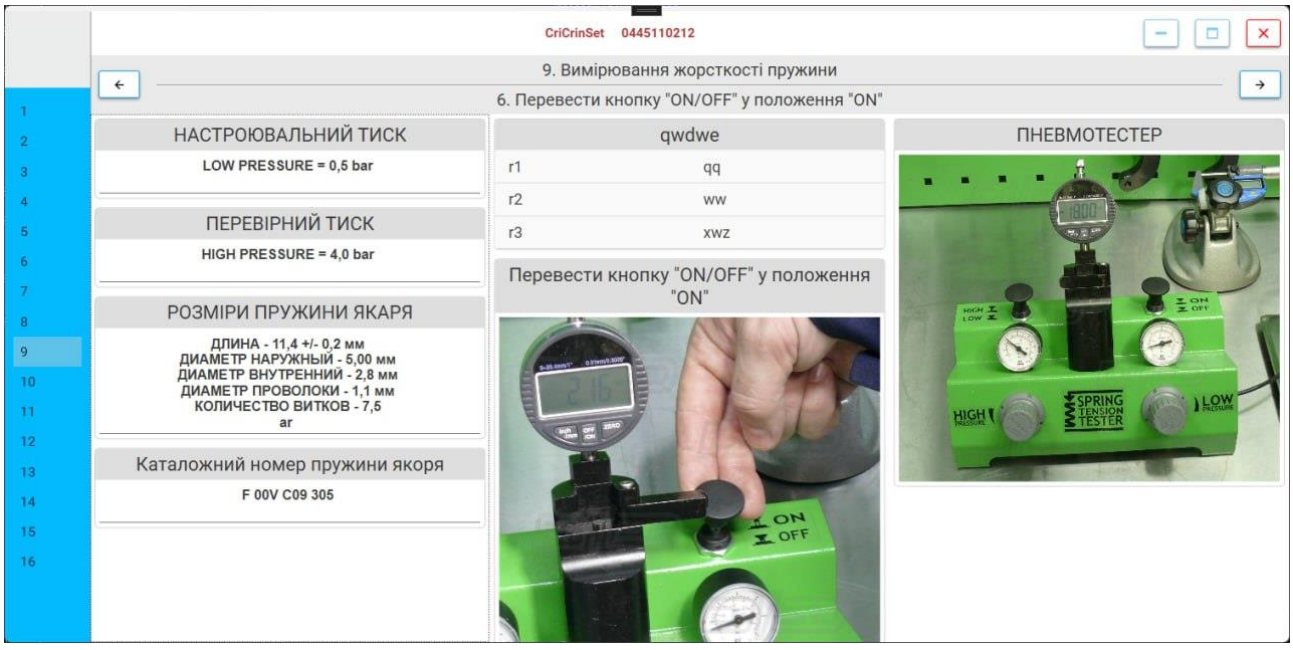

Рис. 3.26 «Відображення створеної таблиці»

2. Створення текстового компоненту - нотатки (Рис.3.27-Рис.3.28):

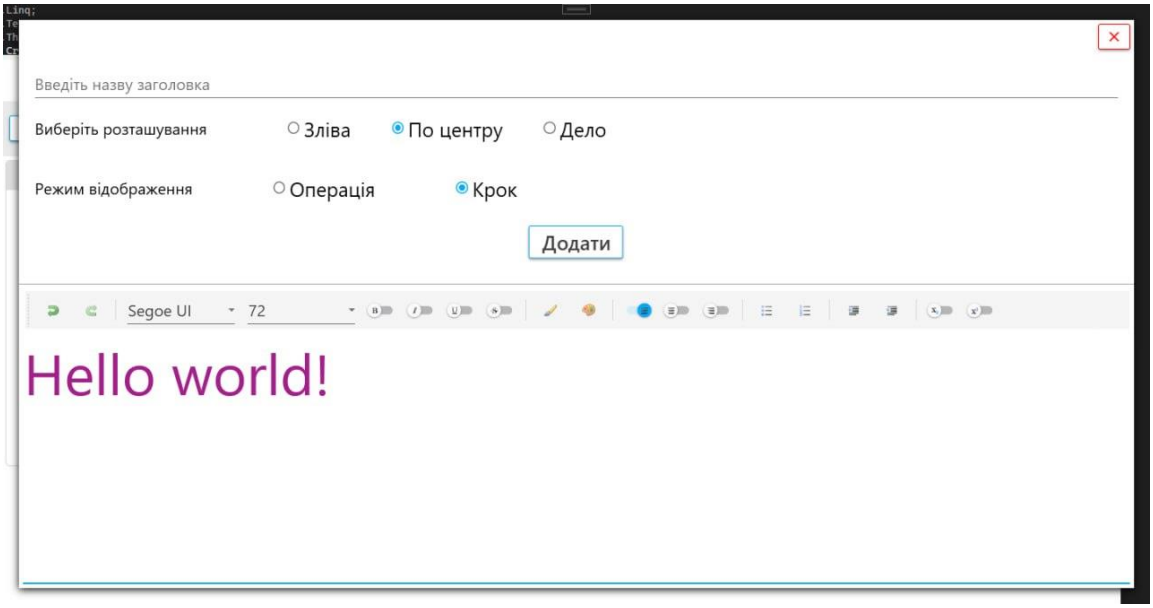

Рис.3.27 «Створення нотатки»

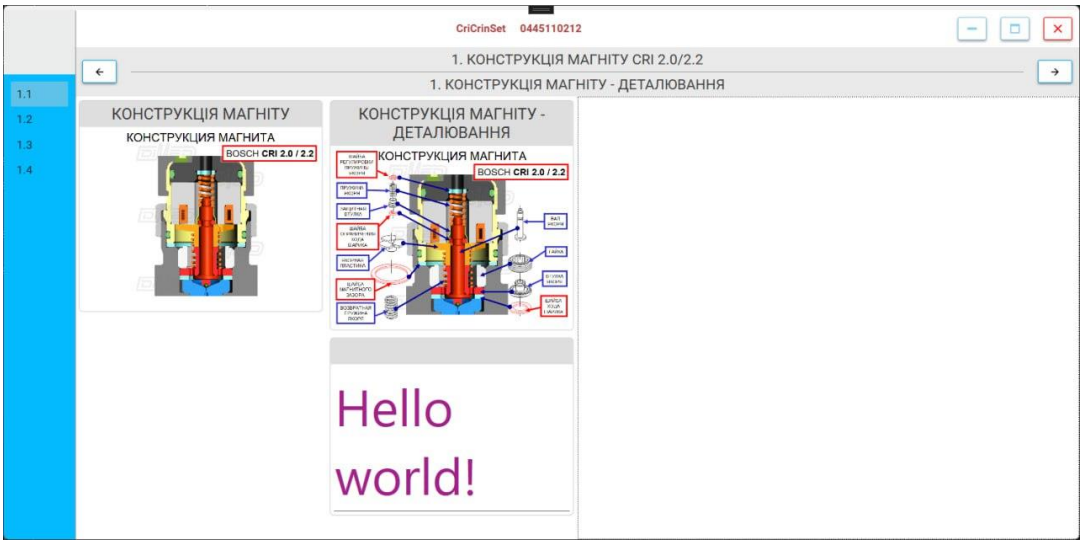

Рис.3.28. «Відображення створеної нотатки»

Далі розглянемо функціональність мультимовності системи. Для цього натиснемо на випадаючий список з мовами в правому верхньому кутку (Рис.3.29.).

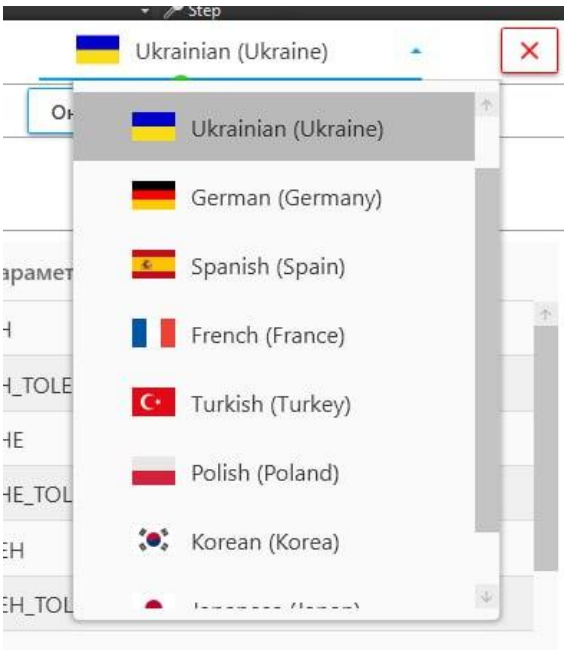

Рис.3.29. «Перелік мов системи»

Та оберемо, наприклад, французьку мову. Після цього перезайдемо в додаток і бачитимемо наступну картину (Рис.3.30):

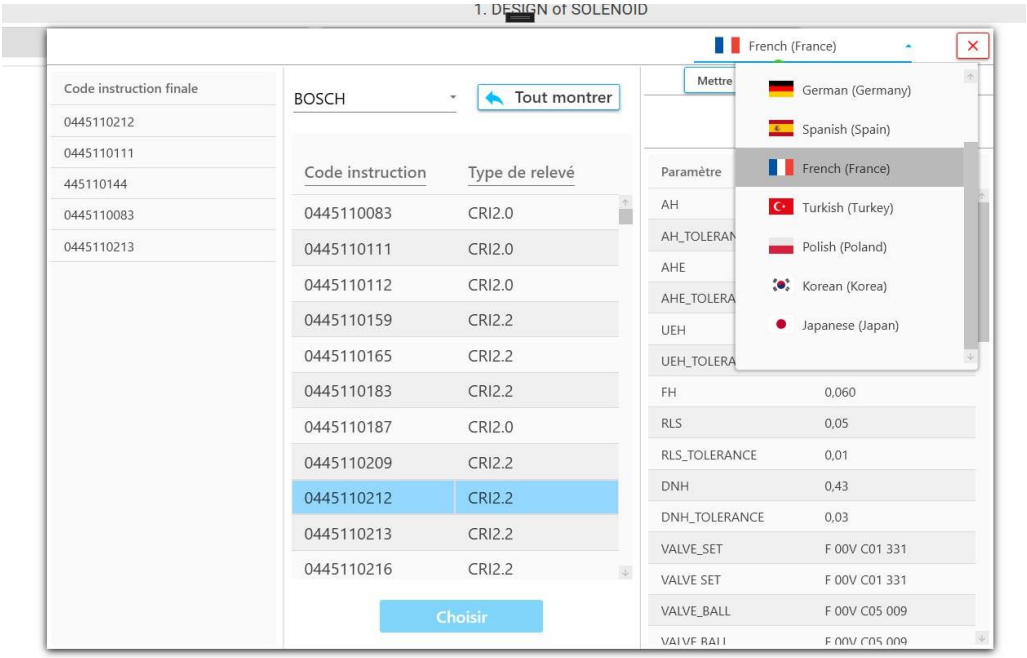

Рис.3.30. «Переключеннння мовного режиму на французьку»

Тепер програма відображає дані французькою мовою.

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

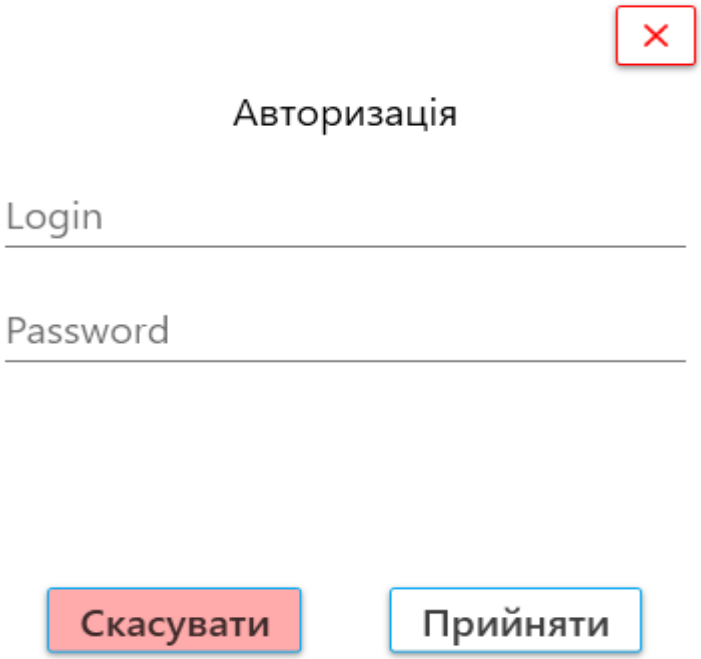

Рис. 3.31. Вікно авторизації

Необхідно зазначити, що в рамках бізнес-логіки створювати та редагувати інструкцію може тільки користувач з роллю адміна, користувач без таких ролі має право тільки на перегляд наданих адміністратором інструкцій.

ВИСНОВКИ

Завдяки дослідженню було проаналізовано та доведено дефекти та пробіли в існуючій системі управління інструкціями та знайдено найбільш оптимальний підхід – маніпулювання ними завдяки специфічному програмному забезпеченню.

Завдяки аналізу існуючих підходів було запропоноване нове, більш досконале рішення.

Об’єктом розробки є система для створення та впровадження системи для управління технічними інструкціями з можливістю відокремлення обов’язків адміністрування і користування.

Актуальність системи зумовлена необхідністю оптимізації процесу управління технічними інструкціями.

В ході виконання роботи була розроблена програма, що організовує систему ведення технічної документації, а саме – інструкцій, дає можливість створення та управління інструкціями залежно від ролі в системі.

Програма надає можливість:

- розбиття користувачів на ролі для визначення подальших прав в системі;
- створення/редагування/видалення категорій та підкатегорій, до яких належатиме інструкція;
- створення та маніпулювання інструкцією згідно з вищеописаними вимогами;
- зміни мови оточення та самих інструкцій;
- оновлення власної версії за можливістю та бажанням користувача;
- оптимізації процесу створення, моніторингу та використання технічних інструкцій.

Програма є повноцінною автоматизованою та оптимізованою одиницею та може використовуватися в різних сферах, таких як: машинобудування, IT-сфера, медична сфера, навчальна, і так далі, вона є унікальним рішенням, що не прив'язується до конкретних умов бізнес-логіки. За потреби рішення може і повинно бути вдосконалено.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. С. А. Л. ВОЗМОЖНОСТИ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ C# (C SHARP) [Електронний ресурс] / А. Л. С., Э. А. Бекирова (Менумерова). – 2018. – Режим доступу до ресурсу: https://www.elibrary.ru/item.asp?id=36605587.

2. Price M. C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET using Visual Studio Code, 4th Edition / Mark Price., 2021. – 816 с. – (4).

3. Ефимова М.Р., Ганченко О.И., Петрова Е.В. Практикум по общей теорий статистики. - М.: Финансы и статистика, 2000. - 280 с.

4. Ткач Є.І. Загальна теорія статистики: Підручник. - Тернопіль.: Лідер, 2004. - 388 с.

5. C#: учебный курс. Герберт Шилдт. – СПб.: Питер; К.: Издательская группа BHV, 2003. – 512с.

6. Мэтью Мак-Дональд WPF: Windows Presentation Foundation в .NET 4.0 с примерами на C# 2010 для профессионалов = Pro WPF in C# 2010: Windows Presentation Foundation with .NET 4.0. — М.: «Вильямс», 2011. — С. 1024.

7. Вигерс К. Разработка требований к программному обеспечению: пер с англ. / Вигерс Карл, Битти Джой — М.: Русская Редакция, 2004. — 314 с.

8. Мейер Д. Теория реляционных баз данных: пер. с англ. / Давид Мейер — М.: Мир, 1987. – 608 с

9. Introduction to Model/View/ViewModel pattern for building WPF apps [Електронний ресурс]/ Microsoft Developer Network. Microsoft — Режим доступу: https://blogs.msdn.microsoft.com/johngossman/2005/10/08/introduction-tomodelviewviewmodel-pattern-for-building-wpf-apps/.

10. Лотка, Рокфорд C# и CSLA .NET Framework. Разработка бизнес-объектов / Рокфорд Лотка. - М.: Вильямс, 2010. - 816 с.

11. Альфред, В. Ахо Компиляторы. Принципы, технологии и инструментарий / Альфред В. Ахо и др. - М.: Вильямс, 2015. - 266 с.

12. Коноваленко І.В. Програмування мовою C# 6.0: навч. посіб. – Тернопіль, ТНТУ- 2016 – 229с.

13. Зубенко В.В., Омельчук Л.Л. Програмування. Поглиблений курс. – К.: Видавничо-поліграфічний центр “Київський університет”, 2011. – 623 с.

14. Троелсен, Эндрю Язык программирования C# 2008 и платформа .NET 3.5 / Эндрю Троелсен. - М.: Вильямс, 2010. - 370 с.

15. Кулямин В. Технологии программирования. Компонентный подход. – М.: Бином, 2007. – 464 с.

16. Troelsen, Andrew. Pro C# 4.0 and the .NET 4.5 Platform, 6th Edition, Apress, 2012

17. Skeet, John. C# in Depth. Manning Publications, 2013

18. Kreibich Jay. Using SQLite. Small. Fast. Reliable. Choose Any Three. – O’Reilly Media, 2010. – 530 p.

19. Харрингтон Дж. Л. Проектирование реляционных баз данных: пер. с англ. / Джен Харрингтон Дж. Л. — М.: Лори, 2006 —232 с

20. Пасічник В. В., Резниченко В. А. Організація баз даних та знань. К. : Видавн. група BHV, 2006. – 384 с

21. Коннолл Т., Карелии Б. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание. : пер. с англ. – М. : Издательский дом «Вильямс», 2003. – 1440 с

22. Дейт К. Дж. Введение в системы баз данных, 7-е издание. : пер. с англ. – М. : Издательский дом «Вильямс», 2001. – 1072 с.

23. What Is SQLite? [Електроний ресурс]. Режим доступу : URL : https://www.sqlite. org/index.html. – Загол. з екрану.

24. C# 2008: ускоренный курс для профессионалов. Трей Неш. – ООО «ИД Вильямс», 2008. – 576с.

25. Ларман К. Применение UML 2.0 и шаблонов проектирования: пер. с англ. / Крэг Ларман. — 3-е изд. М.: Вильямс, 2013 — 736 с.

26. Henry Roediger, Adam L. Putnam, Megan A. Sumeracki. Psychology of Learning and Motivation. Elsevier, 2011. – 32 с.

27. Форми тестових завдань. Форма подання тестового завдання [Електронний ресурс] – Режим доступу: https://pidru4niki.com/12590605/informatika/formi_testovih_zavdan_forma_podannya_testovogo_zavdannya

28. Eye Tracking im digitalen Marketing [Електронний ресурс] – Режим доступу: https://www.webspotting.de/herausforderungenoptimierungsmoeglichkeiten/eye-tracking-im-digitalen-marketing/

29. Chris Richardson. Microservices Patterns: With examples in Java. Manning Publications Co., 2018. – 472 с.

30. Different Types of Web Application Development [Електронний ресурс] https://www.clustox.com/6-different-types-of-web-application-development

31. Automation Testing Life Cycle [Електронний ресурс] https://www.lambdatest.com/blog/all-you-have-to-know-about-automationtesting-life-cycle

31. Tann H. Non-Volatile Memory: A review of past and present concepts and applications / Hokchhay Tann.

32. Davis, F.D., Bagozzi, R.P. andWarshaw, P.R. (1989), “User acceptance of computer-technology – a comparison of 2 theoretical-models”, Management Science 35(8): 98-1003.

33. Chau, P.Y.K. and Hu, P.J. (2002), “Examining a model of information technology acceptance by individual professionals: an exploratory study”, Journal of Information Management Systems, Vol. 18 No. 4, pp. 191-229.

34. Bhardwaj, S., Jain, L. and Jain, S. (2010), “An approach for investigating perspective of cloud software-as-a-service (SaaS)”, International Journal of ComputerApplications, Vol. 10 No. 2, pp. 40-43.

35. Legris, P., Ingham, J. and Collerette, P. (2003), “Why do people use information technology? A critical review of the technology acceptancemodel”, Information andManagement, Vol. 40 No. 3, pp. 191-204.

36. McLeod, J., Childs, S. and Hardiman, R. (2011), “Accelerating positive change in electronic records management – headline indings fromamajor research project”,Archives andManuscripts, Vol. 39 No. 2, pp. 66-94.

37. Rai, R., Sahoo,G. andMehfuz, S. (2013), “Securing software as a servicemodel of cloud computing: issues and solutions”, International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol. 3 No. 4, pp. 1-11.

38. Taylor, S. and Todd, P. (1995), “Understanding information technology: a test of competing models”, Information Systems Research, Vol. 6 No. 2, pp. 144-176.

Додаток А

КОД ПРОГРАМИ

```
      public static bool AddTextToDatabase(int idOperation, int idStep, string rtf,
string header, LocationContent location)
        {
            try
            {
                if (header == null) header = "";
                SQLiteCommand command = new SQLiteCommand()
                {
                    Connection = conn,
                    CommandText = $@"INSERT INTO text_ (header_{App.DefLang},
content_{App.DefLang}, location) VALUES (@header_, @content, @location)"
                };
                command.Parameters.Add("@header_", System.Data.DbType.String);
                command.Parameters.Add("@content", System.Data.DbType.String);
                command.Parameters.Add("@location", System.Data.DbType.String);

                command.Parameters["@header_"].Value = header;
                command.Parameters["@content"].Value = rtf;
                command.Parameters["@location"].Value = location;

                command.ExecuteNonQuery();

                command = new SQLiteCommand("SELECT MAX(id) FROM text_", conn);
                int lastID = Convert.ToInt32(command.ExecuteScalar());

                int numOrd = -1;
                if (idOperation != -1) command = new SQLiteCommand($"SELECT
MAX(numberOrder) FROM content WHERE idOperation = {idOperation}", conn);
                else if (idStep != -1) command = new SQLiteCommand($"SELECT
MAX(numberOrder) FROM content WHERE idStep = {idStep}", conn);
                string numOrd_ = command.ExecuteScalar().ToString();
                if (numOrd_ != null && numOrd_ != "") numOrd =
Convert.ToInt32(numOrd_);
                else numOrd = 0;

                AddContent(idOperation, idStep, lastID, numOrd + 1, TypeContent.Text);
```

```
                return true;
            }
            catch (Exception ex)
            {
                if (App.DEV) MessageBox.Show(ex.Message, string.Empty,
MessageBoxButton.OK, MessageBoxImage.Information);
                return false;
            }
        }

        public static bool AddTableToDatabase(int idOperation, int idStep,
ObservableCollection<RowTable> rowTables, string header,
            double wCol1, double wCol2, LocationContent location)
        {
            try
            {
                if (header == null) header = "";
                SQLiteCommand command = new SQLiteCommand()
                {
                    Connection = conn,
                    CommandText = $@"INSERT INTO table_ (header_{App.DefLang}, wCol1,
wCol2, location) VALUES (@header_, @wCol1, @wCol2, @location)"
                };
                command.Parameters.Add("@header_", System.Data.DbType.String);
                command.Parameters.Add("@wCol1", System.Data.DbType.Double);
                command.Parameters.Add("@wCol2", System.Data.DbType.Double);
                command.Parameters.Add("@location", System.Data.DbType.String);

                command.Parameters["@header_"].Value = header;
                command.Parameters["@wCol1"].Value = wCol1;
                command.Parameters["@wCol2"].Value = wCol2;
                command.Parameters["@location"].Value = location;
                command.ExecuteNonQuery();

                command = new SQLiteCommand("SELECT MAX(id) FROM table_", conn);
                int idTable = Convert.ToInt32(command.ExecuteScalar());

                int numOrd = -1;
```

```
                if (idOperation != -1) command = new SQLiteCommand($"SELECT
MAX(numberOrder) FROM content WHERE idOperation = {idOperation}", conn);

                else if (idStep != -1) command = new SQLiteCommand($"SELECT
MAX(numberOrder) FROM content WHERE idStep = {idStep}", conn);

                string numOrd_ = command.ExecuteScalar().ToString();

                if (numOrd_ != null && numOrd_ != "") numOrd =
Convert.ToInt32(numOrd_);

                else numOrd = 0;


                AddContent(idOperation, idStep, idTable, numOrd + 1,
TypeContent.Table);


                foreach (var item in rowTables)

                {

                    command = new SQLiteCommand()

                    {

                        Connection = conn,

                        CommandText = @$"INSERT INTO rowTable_ (idTable,
parameter_{App.DefLang}, pValue_{App.DefLang}) VALUES (@idTable, @parameter, @pValue)"

                    };

                    command.Parameters.Add("@idTable", System.Data.DbType.Int32);

                    command.Parameters.Add("@parameter", System.Data.DbType.String);

                    command.Parameters.Add("@pValue", System.Data.DbType.String);


                    command.Parameters["@idTable"].Value = idTable;

                    command.Parameters["@parameter"].Value = item.Parametr;

                    command.Parameters["@pValue"].Value = item.PValue;

                    command.ExecuteNonQuery();

                }

                return true;

            }

            catch (Exception ex)

            {

                if (App.DEV) MessageBox.Show(ex.Message, string.Empty,
MessageBoxButton.OK, MessageBoxImage.Information);

                return false;

            }

        }
```

```
        public static bool AddFormulaToDatabase(int idOperation, int idStep, string
header, string formula, string resVariable,
            string low, string high,
            string units, List<string> variable, LocationContent location)
        {
            try
            {
                if (header == null) header = "";
                SQLiteCommand command;
                foreach (var item in variable)
                {
                    command = new SQLiteCommand()
                    {
                        Connection = conn,
                        CommandText = @"INSERT INTO input_ (variable) VALUES
(@variable)"
                    };
                    command.Parameters.Add("@variable", System.Data.DbType.String);
                    command.Parameters["@variable"].Value = item;
                    command.ExecuteNonQuery();
                }

                command = new SQLiteCommand()
                {
                    Connection = conn,
                    CommandText = @"INSERT INTO input_ (variable, units) VALUES
(@variable, @units)"
                };
                command.Parameters.Add("@variable", System.Data.DbType.String);
                command.Parameters.Add("@units", System.Data.DbType.String);
                command.Parameters["@variable"].Value = resVariable;
                command.Parameters["@units"].Value = units;
                command.ExecuteNonQuery();

                command = new SQLiteCommand("SELECT MAX(id) FROM input_", conn);
                int idInput = Convert.ToInt32(command.ExecuteScalar());

                command = new SQLiteCommand()
```

```
                {
                    Connection = conn,
                    CommandText = $@"INSERT INTO formula_ (idInput,
header_{App.DefLang}, formula, low, high, location)
                        VALUES (@idInput, @header_, @formula, @low, @high, @location)"
                };
                command.Parameters.Add("@idInput", System.Data.DbType.Int32);
                command.Parameters.Add("@header_", System.Data.DbType.String);
                command.Parameters.Add("@formula", System.Data.DbType.String);
                command.Parameters.Add("@low", System.Data.DbType.String);
                command.Parameters.Add("@high", System.Data.DbType.String);
                command.Parameters.Add("@location", System.Data.DbType.String);

                command.Parameters["@idInput"].Value = idInput;
                command.Parameters["@header_"].Value = header;
                command.Parameters["@formula"].Value = formula;
                command.Parameters["@low"].Value = low;
                command.Parameters["@high"].Value = high;
                command.Parameters["@location"].Value = location;

                command.ExecuteNonQuery();

                command = new SQLiteCommand("SELECT MAX(id) FROM formula_", conn);
                int lastID = Convert.ToInt32(command.ExecuteScalar());

                int numOrd = -1;
                if (idOperation != -1) command = new SQLiteCommand($"SELECT
MAX(numberOrder) FROM content WHERE idOperation = {idOperation}", conn);
                else if (idStep != -1) command = new SQLiteCommand($"SELECT
MAX(numberOrder) FROM content WHERE idStep = {idStep}", conn);
                string numOrd_ = command.ExecuteScalar().ToString();
                if (numOrd_ != null && numOrd_ != "") numOrd =
Convert.ToInt32(numOrd_);
                else numOrd = 0;

                AddContent(idOperation, idStep, lastID, numOrd + 1,
TypeContent.Formula);
                return true;
```

```
            }
            catch (Exception ex)
            {
                if (App.DEV) MessageBox.Show(ex.Message, string.Empty,
MessageBoxButton.OK, MessageBoxImage.Information);
                return false;
            }
        }

        public static bool UpdateImageToDatabase(int id, int idImage, int idOperation,
int idStep, Images images, string header, LocationContent location)
        {
            try
            {
                if (header == null) header = "";
                SQLiteCommand command = new SQLiteCommand()
                {
                    Connection = conn,
                    CommandText = $@"UPDATE image_ SET fileName_{App.DefLang} =
@fileName, header_{App.DefLang} = @header_,
                        imageData_{App.DefLang} = @imageData, location = @location
WHERE id = @id"
                };
                command.Parameters.Add("@fileName", System.Data.DbType.String);
                command.Parameters.Add("@header_", System.Data.DbType.String);
                command.Parameters.Add("@imageData", System.Data.DbType.Binary);
                command.Parameters.Add("@location", System.Data.DbType.String);
                command.Parameters.Add("@id", System.Data.DbType.Int32);

                command.Parameters["@fileName"].Value = images.FileName;
                command.Parameters["@header_"].Value = header;
                command.Parameters["@imageData"].Value = images.Data;
                command.Parameters["@location"].Value = location;
                command.Parameters["@id"].Value = idImage;

                command.ExecuteNonQuery();

                UpdateContent(idOperation, idStep, id);
```

```
                return true;
            }
            catch (Exception ex)
            {
                if (App.DEV) MessageBox.Show(ex.Message, string.Empty,
MessageBoxButton.OK, MessageBoxImage.Information);
                return false;
            }
        }

        public static bool UpdateTextToDatabase(int id, int idText, int idOperation,
int idStep, string rtf, string header, LocationContent location)
        {
            try
            {
                if (header == null) header = "";
                SQLiteCommand command = new SQLiteCommand()
                {
                    Connection = conn,
                    CommandText = $@"UPDATE text_ SET header_{App.DefLang} = @header_,
content_{App.DefLang} = @content, location = @location WHERE id = @id"
                };
                command.Parameters.Add("@header_", System.Data.DbType.String);
                command.Parameters.Add("@content", System.Data.DbType.String);
                command.Parameters.Add("@location", System.Data.DbType.String);
                command.Parameters.Add("@id", System.Data.DbType.Int32);

                command.Parameters["@header_"].Value = header;
                command.Parameters["@content"].Value = rtf;
                command.Parameters["@location"].Value = location;
                command.Parameters["@id"].Value = idText;

                command.ExecuteNonQuery();

                UpdateContent(idOperation, idStep, id);
                return true;
            }
            catch (Exception ex)
```

```
            {
                if (App.DEV) MessageBox.Show(ex.Message, string.Empty,
MessageBoxButton.OK, MessageBoxImage.Information);
                return false;
            }
        }

        public static bool UpdateTableToDatabase(int id, int idTable, int idOperation,
int idStep, ObservableCollection<RowTable> rowTables,
            string header, double wCol1, double wCol2, LocationContent location)
        {
            try
            {
                if (header == null) header = "";
                SQLiteCommand command = new SQLiteCommand()
                {
                    Connection = conn,
                    CommandText = $@"UPDATE table_ SET header_{App.DefLang} = @header_,
wCol1 = @wCol1, wCol2 = @wCol2, location = @location WHERE id = @id"
                };
                command.Parameters.Add("@header_", DbType.String);
                command.Parameters.Add("@wCol1", DbType.Double);
                command.Parameters.Add("@wCol2", DbType.Double);
                command.Parameters.Add("@location", DbType.String);
                command.Parameters.Add("@id", DbType.Int32);

                command.Parameters["@header_"].Value = header;
                command.Parameters["@wCol1"].Value = wCol1;
                command.Parameters["@wCol2"].Value = wCol2;
                command.Parameters["@location"].Value = location;
                command.Parameters["@id"].Value = idTable;
                command.ExecuteNonQuery();

                UpdateContent(idOperation, idStep, id);

                var listRow = GetRowTable(idTable);
                DifferenceRemove(listRow, rowTables);
                foreach (var item in rowTables)
```

```
                {
                    if (item.ID > 0)
                    {
                        command = new SQLiteCommand()
                        {
                            Connection = conn,
                            CommandText = @$"UPDATE rowTable_
                            SET parameter_{App.DefLang} = @parameter,
pValue_{App.DefLang} = @pValue
                            WHERE id = {item.ID}"
                        };
                        command.Parameters.Add("@parameter",
System.Data.DbType.String);
                        command.Parameters.Add("@pValue", System.Data.DbType.String);

                        command.Parameters["@parameter"].Value = item.Parametr;
                        command.Parameters["@pValue"].Value = item.PValue;
                        command.ExecuteNonQuery();
                    }
                    else
                    {
                        command = new SQLiteCommand()
                        {
                            Connection = conn,
                            CommandText = @$"INSERT INTO rowTable_ (idTable,
parameter_{App.DefLang}, pValue_{App.DefLang}) VALUES (@idTable, @parameter, @pValue)"
                        };
                        command.Parameters.Add("@idTable", System.Data.DbType.Int32);
                        command.Parameters.Add("@parameter",
System.Data.DbType.String);
                        command.Parameters.Add("@pValue", System.Data.DbType.String);

                        command.Parameters["@idTable"].Value = idTable;
                        command.Parameters["@parameter"].Value = item.Parametr;
                        command.Parameters["@pValue"].Value = item.PValue;
                        command.ExecuteNonQuery();
                    }
                }
```

```
                return true;
            }
            catch (Exception ex)
            {
                if (App.DEV) MessageBox.Show(ex.Message, string.Empty,
MessageBoxButton.OK, MessageBoxImage.Information);
                return false;
            }
        }

        private static void DifferenceRemove(ObservableCollection<RowTable> listRow,
ObservableCollection<RowTable> rowTables)
        {
            for (int i = 0; i < listRow.Count; i++)
            {
                bool flagDelete = false;
                for (int j = 0; j < rowTables.Count; j++)
                {
                    if (listRow[i].ID == rowTables[j].ID && listRow[i].IDTable ==
rowTables[j].IDTable)
                    {
                        flagDelete = false;
                        break;
                    }
                    else
                    {
                        flagDelete = true;
                        continue;
                    }
                }
                if (flagDelete)
                {
                    SQLiteCommand command = new SQLiteCommand
                    {
                        Connection = conn,
                        CommandText = @$"DELETE FROM rowTable_ WHERE id =
{listRow[i].ID}"
                    };
```

```
                command.ExecuteNonQuery();
            }
        }
    }

    public static bool UpdateFormulaToDatabase(int id, int idInput, int idFormula,
int idOperation, int idStep, string header, string formula,
        string resVariable, Formulas Formulas_, string units, List<string>
variable, LocationContent location, int flagUpdate)
    {
        try
        {
            if (header == null) header = "";
            SQLiteCommand command;
            if (flagUpdate == 1)
            {
                command = new SQLiteCommand()
                {
                    Connection = conn,
                    CommandText = @"DELETE FROM formula_ WHERE id = @id"
                };

                command.ExecuteNonQuery();

                AddFormulaToDatabase(idOperation, idStep, header, formula,
resVariable, Formulas_.Low, Formulas_.High, units, variable, location);
            }
            else
            {
                command = new SQLiteCommand()
                {
                    Connection = conn,
                    CommandText = $@"UPDATE formula_ SET header_{App.DefLang} =
@header_, formula = @formula, low = @low, high = @high
                    location = @location WHERE id = @id"
                };
                command.Parameters.Add("@header_", System.Data.DbType.String);
                command.Parameters.Add("@formula", System.Data.DbType.String);
```

```
                    command.Parameters.Add("@low", System.Data.DbType.String);
                    command.Parameters.Add("@high", System.Data.DbType.String);
                    command.Parameters.Add("@location", System.Data.DbType.String);
                    command.Parameters.Add("@id", System.Data.DbType.Int32);

                    command.Parameters["@header_"].Value = header;
                    command.Parameters["@formula"].Value = formula;
                    command.Parameters["@low"].Value = Formulas_.Low;
                    command.Parameters["@high"].Value = Formulas_.High;
                    command.Parameters["@location"].Value = location;
                    command.Parameters["@id"].Value = idFormula;

                    command.ExecuteNonQuery();

                    command = new SQLiteCommand()
                    {
                        Connection = conn,
                        CommandText = $@"UPDATE input_ SET units = @units WHERE id =
@id"
                    };
                    command.Parameters.Add("@units", System.Data.DbType.String);
                    command.Parameters.Add("@id", System.Data.DbType.Int32);
                    command.Parameters["@units"].Value = units;
                    command.Parameters["@id"].Value = idInput;
                    command.ExecuteNonQuery();

                    UpdateContent(idOperation, idStep, id);
                }
                return true;
            }
            catch (Exception ex)
            {
                if (App.DEV) MessageBox.Show(ex.Message, string.Empty,
MessageBoxButton.OK, MessageBoxImage.Information);
                return false;
            }
        }
```

```
        public static bool UpdateInputToDatabase(int id, int idInput, int idOperation,
int idStep, string header, string units, LocationContent location)
        {
            try
            {
                if (header == null) header = "";
                SQLiteCommand command = new SQLiteCommand()
                {
                    Connection = conn,
                    CommandText = $"UPDATE input_ SET header_{App.DefLang} = @header_,
units = @units, location = @location WHERE id = {idInput}"
                };
                command.Parameters.Add("@header_", System.Data.DbType.String);
                command.Parameters.Add("@units", System.Data.DbType.String);
                command.Parameters.Add("@location", System.Data.DbType.String);

                command.Parameters["@header_"].Value = header;
                command.Parameters["@units"].Value = units;
                command.Parameters["@location"].Value = location;

                command.ExecuteNonQuery();

                if (id > 0)
                {
                    UpdateContent(idOperation, idStep, id);
                }
                else
                {

                    int numOrd = -1;
                    if (idOperation != -1) command = new SQLiteCommand($"SELECT
MAX(numberOrder) FROM content WHERE idOperation = {idOperation}", conn);
                    else if (idStep != -1) command = new SQLiteCommand($"SELECT
MAX(numberOrder) FROM content WHERE idStep = {idStep}", conn);
                    string numOrd_ = command.ExecuteScalar().ToString();
                    if (numOrd_ != null && numOrd_ != "") numOrd =
Convert.ToInt32(numOrd_);
```

```
                    else numOrd = 0;


                    AddContent(idOperation, idStep, idInput, numOrd + 1,
TypeContent.Input);
                }
                return true;
            }
            catch (Exception ex)
            {
                if (App.DEV) MessageBox.Show(ex.Message, string.Empty,
MessageBoxButton.OK, MessageBoxImage.Information);
                return false;
            }
        }


        public static void DeleteContentToDatabase(int id, int idContent, TypeContent
typeContent)
        {
            try
            {
                SQLiteCommand command = new SQLiteCommand()
                {
                    Connection = conn,
                    CommandText = $"DELETE FROM content WHERE id = {id}"
                };


                command.ExecuteNonQuery();


                string content;
                switch (typeContent)
                {
                    case TypeContent.Table:
                        content = "table_";
                        break;
                    case TypeContent.Input:
                        return;
                    case TypeContent.Formula:
                        content = "formula_";
```

```
                    break;
                case TypeContent.Image:
                    content = "image_";
                    break;
                case TypeContent.Text:
                    content = "text_";
                    break;
                default: return;
            }

            command = new SQLiteCommand()
            {
                Connection = conn,
                CommandText = $"DELETE FROM {content} WHERE id = {idContent}"
            };

            command.ExecuteNonQuery();
        }
        catch (Exception ex)
        {
            if (App.DEV) MessageBox.Show(ex.Message, string.Empty,
MessageBoxButton.OK, MessageBoxImage.Information);
        }
    }
```

Додаток Б

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
121М-22-2, Левдик І.А.docx	Пояснювальна записка роботи. Документ Word.
121М-22-2, Левдик І.А.pdf	Пояснювальна записка роботи. Документ PDF.
Програма	
CriCrinCtor.zip	Архів. Містить код програми.
Презентація	
Презентація Левдик І.А.pptx	Презентація дипломної роботи.
Статистика диплом. xlsx	Файл з даними то розрахунками для дослідження