МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ

_____

к.т.н, доц. Михелєв І.Л.

"____" _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "магістр"

на тему:

Аналіз потоків даних та інформаційне забезпечення моніторингу відходів руйнації

Виконав: студент групи 6142м

Васильєв О.В. *Прізвище І.Пб.*

(підпис)

Керівник роботи:

_проф.,д.т.н.,доцент_____

(посада, науковий ступінь вчене звання)

Казимиренко Ю.О. *Прізвище І.Пб.*

(підпис)

м. Миколаїв – 2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра інформаційних управляючих систем та технологій

Спеціальність 124 "Системний аналіз"

Освітня програма "Системний аналіз"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

_____

д.ф-м.н, проф. Ушкац М.В.

"____" _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти "магістр"

Студенту Васильєву Олександру Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Аналіз потоків даних та інформаційне забезпечення моніторингу відходів руйнації

Керівник роботи
Казимиренко Юлія Олексіївна

Затверджені наказом ректора № 756-уч. від "25" вересня 2023 року.

2. Термін подання роботи: 15 грудня 2023 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем, матеріали наукового стажування

4. Перелік питань, що належать до розробки (найменування розділів)

1) АНАЛІЗ ПРОБЛЕМНИХ ПИТАНЬ ВПРОВАДЖЕННЯ МОНІТОРИНГУ ВІДХОДІВ РУЙНАЦІЇ

2) МЕТОДИ ВИРІШЕННЯ ЗАДАЧ ВНЕСЕННЯ ТА ОБМІНУ ДАННИМИ ЩОДО ВІДХОДІВ РУЙНАЦІЇ

3) АНАЛІЗ ПОТОКІВ ДАНИХ В ЗАДАЧАХ СИСТЕМАТИЗАЦІЇ ТА ЕКСПЕРИЗИ ВІДХОДІВ РУЙНАЦІЇ

4) ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ РОЗРОБКИ_____

5) ОХОРОНА ПРАЦІ_____

6) ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА_____

5. Перелік презентаційних матеріалів_____
_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Казимиренко Ю.О.		
2			
3			
4			

7. Дата видачі завдання 01 листопада 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	АНАЛІЗ ПРОБЛЕМНИХ ПИТАНЬ ВПРОВАДЖЕННЯ МОНІТОРИНГУ ВІДХОДІВ РУЙНАЦІЇ		
2	Необхідність сортування та проведення експертизи відходів руйнації		
3	Критерії декомпозиції та аналіз елементів системи		
4	Вимоги до інформаційної системи моніторингу		
5	Аналіз вхідних і вихідних даних		

6	Висновки до розділу, аналіз невирішених питань та постановка завдань дослідження		
7	МЕТОДИ ВИРІШЕННЯ ЗАДАЧ ВНЕСЕННЯ ТА ОБМІНУ ДАННИМИ ЩОДО ВІДХОДІВ РУЙНАЦІЇ		
8	Координація цілей задачі та множини їх розв’язання		
9	Вибір і обґрунтування методів вирішення задачі		
10	Вибір і обґрунтування засобів вирішення задачі		
11	Розробка моделей проектування інформаційно пошукової системи «Аналіз Матеріалів»		
12	Технологія і засоби побудови бази даних		
13	Висновки до розділу		
14	АНАЛІЗ ПОТОКІВ ДАНИХ В ЗАДАЧАХ СИСТЕМАТИЗАЦІЇ ТА ЕКСПЕРИЗИ ВІДХОДІВ РУЙНАЦІЇ		
15	Специфіка інформаційно-пошукової системи та бази даних відходів руйнації		
16	Опис механізмів вводу, виводу, пошуку даних		
17	Приклад реалізації		
18	Висновки до розділу		
19	ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ РОЗРОБКИ		
20	Впровадження програми в експлуатацію		
21	Структура та принципи організації виробничого процесу		
22	Технологія розробки і впровадження програмного забезпечення		

23	Розрахунок чисельності розробників програмного забезпечення		
24	Розрахунок вартості матеріально-енергетичних витрат при розробці програмного забезпечення		
25	Розрахунок заробітної плати розробників ПЗ		
26	Розрахунок вартості розробки ПЗ		
27	Розрахунок економічної ефективності та строку окупності ПЗ		
28	ОХОРОНА ПРАЦІ		
29	Вимоги до організації робочого місця оператора ПК		
30	Техніка безпеки при роботі з ПК		
31	ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА		
32	Правові основи охорони середовища		
33	ЗАГАЛЬНІ ВИСНОВКИ		
34	ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ		
35	Додаток А – код розробки		
36	Додаток Б – апробація роботи та сертифікат		

Студент _____ Васильєв О.В.
(підпис) (ПІБ)

Керівник роботи _____ Казимиренко Ю.О.
(підпис) (ПІБ)

АНОТАЦІЯ

Магістерська робота «Аналіз потоків даних та інформаційне забезпечення моніторингу відходів руйнації» за спеціальністю 124 – системний аналіз.

Робота призначена для розв'язання завдань стосовно створення та впровадження інформаційної системи, яка служить для моніторингу відходів внаслідок руйнування.

Мета роботи полягає у систематизації критеріїв та декомпозиції задачі для експертизи та моніторингу при поводженні з відходами руйнації в умовах військових дій. Основними задачами є аналіз проблемниих питань впровадження моніторингу відходів руйнації. Дослідження методів вирішення задач щодо внесення та обміну даними при систематизації відходів руйнації за походженням, складом та структурою. Аналіз потоків та розробка бази даних для систематизації та експертизи відходів руйнації. Розгляд техніко-економічного обгрунтування розробки та розробка заходів щодо охорони праці.[16]

Ця система інформації розробляється для автоматизації процесів збору, обробки та аналізу даних щодо відходів з метою забезпечення точності та своєчасності інформації. Її впровадження сприяє полегшенню ухвалення управлінських рішень у сфері відновлення та управління відходами, підтримуючи екологічно сталі практики та забезпечуючи ефективну діяльність відповідних служб та організацій.[17]

Наукова новизна роботи полягає в наступному: вперше розроблено та впроваджено комплексну систему, яка використовує сучасні технології для автоматизації процесів моніторингу та управління відходами при руйнуванні. Це включає в себе використання передових методів сенсорної техніки, аналізу даних в режимі реального часу та штучного інтелекту для точного прогнозування та оптимізації управлінських рішень у сфері відновлення та обробки відходів. Перевагою цієї новаторської системи є її здатність підвищити рівень ефективності та екологічної сталості управління відходами в умовах руйнування будівель та інших об'єктів.[14]

Практичне значеня роботи полягає у конкретному рішенні та розробці інструментарію для вдосконалення процесів управління відходами під час руйнування, що є важливим в контексті сучасних вимог до екологічної сталості та управління ресурсами. Система, розроблена у рамках цієї роботи, може впроваджуватися на будівельних об'єктах та міських територіях, де активно ведуться роботи з реконструкції, демонтажу або інших видів руйнування.

Методи дослідження включають в себе використання системного аналізу, засади процедурного програмування та традиційні підходи до проєктування баз даних.

Апробація роботи: результати були представлені на міжнародній науково-технічній конференції («Міжнародна конференція для молодих вчених, аспірантів та магістрів – Бекетівські Хімічні Читання. Теорія та практика кризових ситуацій»), на якій зроблено доповідь та отримано сертифікат.

Робота написана на _77_сторінках машинописного тексту, містить _25_ рисунків, _6_ таблиць, _2_ додатки і список використаних джерел з _24_ найменувань.

Ключові слова: моніторинг, декомпозиція, систематизація критеріїв, експертиза**,** відходи руйнації, база даних, SQL-запити.

ABSTRACT

Master's thesis "Analysis of data flows and information support for monitoring of destruction waste" in the specialty 124 - system analysis.

The work is intended to solve problems related to the creation and implementation of an information system that serves to monitor destruction waste.

The purpose of the work is to systematize the criteria and decompose the task for examination and monitoring of the management of destruction waste in the conditions of military operations. The main tasks are to analyze the problematic issues of implementing monitoring of destruction waste. Study of methods for solving problems of data entry and exchange in the systematization of destruction waste by origin, composition and structure. Analysis of flows and development of a database for the systematization and examination of destruction waste. Consideration of the feasibility study for the development and development of labor protection measures.

This information system is being developed to automate the processes of collecting, processing and analyzing waste data to ensure the accuracy and timeliness of information. Its implementation helps to facilitate management decision-making in the field of waste recovery and management, supporting environmentally sustainable practices and ensuring the effective operation of relevant services and organizations.

The scientific novelty of the work is the following: for the first time, a comprehensive system has been developed and implemented that uses modern technologies to automate the processes of monitoring and managing waste during destruction. This includes the use of advanced sensor technology, real-time data analysis, and artificial intelligence to accurately predict and optimize management decisions in the field of waste recovery and treatment. The advantage of this innovative system is its ability to increase the level of efficiency and environmental sustainability of waste management in the context of the destruction of buildings and other facilities.

The practical significance of the work lies in the specific solution and development of tools for improving waste management processes during demolition, which is important in the context of modern requirements for environmental sustainability and resource management. The system developed as part of this work can

be implemented at construction sites and urban areas where reconstruction, dismantling, or other types of destruction are actively underway.

Research methods include the use of system analysis, the principles of procedural programming, and traditional approaches to database design.

Approbation of the work: the results were presented at an international scientific and technical conference ("International Conference for Young Scientists, Postgraduates and Masters – Beckett Chemical Readings. Theory and Practice of Crisis Situations"), where a report was made and a certificate was received.

The work is written on _77__ pages of typewritten text, contains _25_ figures, _6_ tables, _2_ appendices and a list of references of _24_ titles.

Keywords: monitoring, decomposition, systematization of criteria, expertise, destruction waste, database, SQL queries.

Зміст

1. АНАЛІЗ ПРОБЛЕМНИХ ПИТАНЬ ВПРОВАДЖЕННЯ МОНІТОРИНГУ ВІДХОДІВ РУЙНАЦІЇ 10

1.1 Необхідність сортування та проведення експертизи відходів руйнації 10

1.2 Критерії декомпозиції та аналіз елементів системи 11

1.3 Вимоги до інформаційної системи моніторингу.....12

1.4 Аналіз вхідних і вихідних даних.....14

1.5. Висновки до розділу, аналіз невирішених питань та постановка завдань дослідження 15

2. МЕТОДИ ВИРІШЕННЯ ЗАДАЧ ВНЕСЕННЯ ТА ОБМІНУ ДАННИМИ ЩОДО ВІДХОДІВ РУЙНАЦІЇ 17

2.1 Координація цілей задачі та множини їх розв’язання 17

2.2 Вибір і обґрунтування методів вирішення задачі 18

2.3 Вибір і обґрунтування засобів вирішення задачі 20

2.4 Розробка моделей проектування інформаційно пошукової системи «Аналіз Матеріалів» 21

2.5. Технологія і засоби побудови бази даних 23

2.6 Висновки до розділу 25

3. АНАЛІЗ ПОТОКІВ ДАНИХ В ЗАДАЧАХ СИСТЕМАТИЗАЦІЇ ТА ЕКСПЕРИЗИ ВІДХОДІВ РУЙНАЦІЇ 27

3.1. Специфіка інформаційно-пошукової системи та бази даних відходів руйнації 27

3.2. Опис механізмів вводу, виводу, пошуку даних 29

3.3. Приклад реалізації 32

3.4. Висновки до розділу 42

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ РОЗРОБКИ 43

4.1 Впровадження програми в експлуатацію 43

4.2 Структура та принципи організації виробничого процесу.....44

4.3 Технологія розробки і впровадження програмного забезпечення 49

4.4 Розрахунок чисельності розробників програмного забезпечення 50

4.5 Розрахунок вартості матеріально-енергетичних витрат при розробці програмного забезпечення 52
4.6 Розрахунок заробітної плати розробників ПЗ. 55
4.7 Розрахунок вартості розробки ПЗ 58
4.8 Розрахунок економічної ефективності та строку окупності ПЗ 61
5 ОХОРОНА ПРАЦІ 63
5.1 Вимоги до організації робочого місця оператора ПК 63
5.2 Техніка безпеки при роботі з ПК 65
6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА.....68
6.1 Правові основи охорони середовища 68
ЗАГАЛЬНІ ВИСНОВКИ.....71
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 72
Додаток А – код розробки 74
Додаток Б – апробація роботи та сертифікат 74

ВСТУП

У сучасному світі, де технології розвиваються стрімкими темпами, важливо володіти знаннями та інструментами для ефективного аналізу великих обсягів даних. Тема моєї дипломної роботи, "Аналіз потоків даних та інформаційне забезпечення моніторингу відходів руйнації на основі програми", є актуальною та важливою, оскільки вона стосується ефективного використання новітніх технологій для розв'язання проблем управління відходами та їх моніторингу.

Мета роботи полягає у систематизації критеріїв та декомпозиції задачі для експертизи та моніторингу при поводженні з відходами руйнації в умовах військових дій. Практичне застосування включає в себе здатність ефективно відстежувати та керувати відходами в реальному часі, що сприяє оперативному реагуванню на зміни та оптимізації процесів управління. Крім того, система може слугувати інструментом для забезпечення дотримання екологічних норм та стандартів під час будівельних робіт, сприяючи створенню більш сталого та екологічно безпечного середовища.[15]

Основними задачами є аналіз проблемниих питань впровадження моніторингу відходів руйнації. Дослідження методів вирішення задач щодо внесення та обміну даними при систематизації відходів руйнації за походженням, складом та структурою. Аналіз потоків та розробка бази даних для систематизації та експертизи відходів руйнації. Розгляд техніко-економічного обгрунтування розробки та розробка заходів щодо охорони праці.

1. АНАЛІЗ ПРОБЛЕМНИХ ПИТАНЬ ВПРОВАДЖЕННЯ МОНІТОРИНГУ ВІДХОДІВ РУЙНАЦІЇ

1.1 Необхідність сортування та проведення експертизи відходів руйнації

У цьому підрозділі детально розглядається важливість і необхідність сортування та проведення експертизи відходів руйнації, які є ключовими етапами в процесі ефективного моніторингу та управління відходами.

1. **Сортування Відходів:**
    - **Принципи Сортування:** Обговорення загальних принципів та методів сортування відходів, з акцентом на відмінності між різними типами відходів (наприклад, будівельні відходи, промислові відходи, тощо).
    - **Класифікація Відходів:** Детальний опис різних класів та категорій відходів, залежно від їх фізичних, хімічних та токсичних властивостей.
    - **Технології Сортування:** Аналіз сучасних технологій сортування, їх ефективності та екологічної безпечності.
2. **Експертиза Відходів:**
    - **Методи Аналізу Відходів:** Опис різних методик та підходів до аналізу відходів, включаючи хімічний аналіз, токсикологічне тестування, та оцінку фізичних властивостей.
    - **Визначення Ризиків для Здоров'я та Навколишнього Середовища:** Аналіз потенційних ризиків, які несуть різні типи відходів, та їх можливий вплив на здоров'я людей та навколишнє середовище.
    - **Законодавчі та Нормативні Вимоги:** Огляд чинних законодавчих та нормативних вимог до експертизи відходів, включаючи вимоги до безпеки та екологічності.

Цей підрозділ підкреслює важливість ретельного сортування та експертизи відходів руйнації для забезпечення ефективного моніторингу, впровадження заходів щодо їх переробки та зниження негативного впливу на навколишнє середовище. Він також висвітлює різні аспекти, які повинні бути враховані при розробці та впровадженні відповідних стратегій управління відходами.

1.2 Критерії декомпозиції та аналіз елементів системи

У цьому підрозділі дипломної роботи проводиться ретельний аналіз процесу декомпозиції системи моніторингу відходів руйнації, визначаючи ключові критерії для розділення системи на меньші, більш управлінні елементи, та аналізуючи кожен з цих елементів на предмет їх функцій та взаємодії.

1. **Критерії Декомпозиції:**

Функціональність: Розгляд функціональних аспектів системи та їх розділення на окремі модулі або компоненти. Наприклад, сегментація системи на збір даних, обробку, аналітику та звітність.

Масштабованість: Оцінка можливостей системи до масштабування, як важливого критерію для розділення системи на модулі, які можна легко нарощувати або модифікувати.

Незалежність Компонентів: Визначення ступеня незалежності різних елементів системи, що дозволяє зменшити складність управління та забезпечити легшість модифікації та обслуговування.

2. **Аналіз Елементів Системи:**

Збір Даних: Детальний аналіз компонентів системи, відповідальних за збір даних, включаючи типи сенсорів, методи збору даних та їх передачу.

Обробка Даних: Розгляд компонентів, задіяних у обробці даних, включаючи алгоритми фільтрації, аналізу, та виявлення зразків.

Аналітичні Засоби: Опис аналітичних інструментів системи, що дозволяють витягувати змістовні висновки з оброблених даних, включаючи статистичний аналіз, прогнозування та візуалізацію.

Інтерфейс Користувача та Звітність: Аналіз елементів, пов'язаних з користувацьким інтерфейсом та звітністю, обговорення їх важливості для зручності користувачів та ефективності представлення даних.

3. **Взаємодія Між Елементами:**

Інтеграція Компонентів: Аналіз, як різні компоненти системи взаємодіють між собою, та як ця взаємодія впливає на загальну ефективність та надійність системи.

Механізми Зв'язку: Розгляд механізмів зв'язку між різними елементами системи, включаючи протоколи передачі даних та інтеграційні інтерфейси.

Цей підрозділ демонструє глибоке розуміння необхідності структурованого підходу до розробки складних систем моніторингу відходів руйнації. Він детально розглядає кожен компонент системи, його функції, важливість та взаємодію з іншими елементами, надаючи тим самим основу для ефективного проектування та управління такими системами.

1.3 Вимоги до інформаційної системи моніторингу

Цей підрозділ дипломної роботи спрямований на детальний аналіз вимог, які ставляться до інформаційної системи моніторингу відходів руйнації. Важливість цього аналізу полягає у визначенні критичних характеристик та функцій, необхідних для ефективної роботи системи. Аналіз охоплює як технічні, так і функціональні аспекти системи.

1. **Технічні Вимоги:**
    - Надійність та Стабільність: Обговорення важливості надійності системи, здатності працювати безперервно та ефективно обробляти великі обсяги даних.
    - Масштабованість: Аналіз потреби в забезпеченні можливості масштабування системи для адаптації до змінюваних умов та зростаючих обсягів даних.
    - Безпека Даних: Розглядається необхідність захисту даних від несанкціонованого доступу, втрати даних, та інших ризиків.

- Інтегрованість: Оцінка можливостей інтеграції з іншими системами та інструментами, що сприяють обміну даними та координації дій.

2. **Функціональні Вимоги:**
   - Збір та Обробка Даних: Детальний опис функцій, пов'язаних із збором, обробкою та аналізом даних, включаючи автоматизацію цих процесів.
   - Аналітичні Можливості: Розгляд функцій аналітики, необхідних для виявлення тенденцій, визначення проблемних зон та підтримки прийняття рішень.
   - Користувацький Інтерфейс: Аналіз вимог до інтерфейсу користувача, що включає зручність використання, наявність інтуїтивно зрозумілих елементів управління та можливість налаштування.
   - Звітність та Візуалізація: Опис необхідності розробки системи звітності та візуалізації, що дозволяє представляти дані в зручному та зрозумілому форматі.
3. **Загальні Вимоги:**
   - Гнучкість та Адаптивність: Обговорення необхідності створення системи, здатної адаптуватися до змін у вимогах та умовах роботи.
   - Сумісність та Стандартизація: Аналіз важливості сумісності з іншими стандартами та системами, що забезпечує легкість інтеграції та ефективність взаємодії.
   - Підтримка та Сервіс: Розгляд вимог до підтримки системи, включаючи технічну підтримку, оновлення та обслуговування.

Цей підрозділ надає всебічний огляд вимог, які важливі для успішного впровадження та ефективної експлуатації інформаційної системи моніторингу. Він детально описує технічні та функціональні характеристики, які необхідні для забезпечення високої продуктивності, безпеки та користувацької зручності системи.

1.4 Аналіз вхідних і вихідних даних

В рамках аналізу потоків даних у системі моніторингу відходів руйнації, особливу увагу слід приділити процесам вводу, обробки, виводу та пошуку даних, які є ключовими для забезпечення ефективності та точності інформації.

1. **Процес Вводу Даних:**
   - Інтерфейс для вводу даних дозволяє користувачам вносити інформацію про різні аспекти матеріалів, використовуючи форми у графічному інтерфейсі користувача, розробленому через tkinter.
   - Введені дані проходять перевірку на валідність перед зберіганням в базі даних SQLite, що забезпечує їхню точність та надійність.
2. **Обробка Вводу Даних:**
   - Детальний опис різноманітності вхідних даних, таких як сенсорні дані, звіти про відходи, включає аналіз джерел даних і використовуваних форматів.
   - Обробка вхідних даних включає фільтрацію, передобробку та інтеграцію з різних джерел, щоб створити єдину інформаційну базу.
3. **Процес Виводу Даних:**
   - Дані відображаються у вигляді таблиці в графічному інтерфейсі, що сприяє зручності їх перегляду та аналізу.
   - Система забезпечує функціональність перегляду деталей кожного запису, дозволяючи користувачам глибше ознайомитися з відповідними даними.
4. **Механізми Пошуку Даних:**
   - Реалізована функціональність пошуку дозволяє користувачам ефективно фільтрувати та знаходити необхідну інформацію за допомогою різних критеріїв.
5. **Висновки:**
   - Цей підхід до вводу, обробки, виводу та пошуку даних сприяє ефективному управлінню інформацією та полегшує взаємодію користувачів з системою.

- Використання графічного інтерфейсу та зберігання даних у базі даних SQLite робить систему зручною та функціональною для різних сценаріїв використання, особливо у контексті охорони навколишнього середовища.
- Глибокий аналіз вхідних та вихідних даних підкреслює технічну та стратегічну важливість цих процесів, зосереджуючись на точності, надійності та відповідності даних.

1.5. Висновки до розділу, аналіз невирішених питань та постановка завдань дослідження

Цей підрозділ зводить до висновків аналіз, виконаний у попередніх частинах, зосереджуючись на ключових виявлених аспектах, невирішених питаннях та формулюванні завдань для подальшого дослідження в контексті моніторингу відходів руйнації.

1. **Загальні Висновки:**
   - **Підсумок Аналізу:** Зведення основних пунктів аналізу, здійсненого у попередніх частинах, включаючи технічні та функціональні вимоги до системи, аналіз вхідних та вихідних даних.
   - **Значення Виявлених Аспектів:** Визначення важливості різних виявлених аспектів у контексті ефективності моніторингу та управління відходами.
2. **Аналіз Невирішених Питань:**
   - **Ідентифікація Проблемних Зон:** Виявлення та опис ключових невирішених питань, виявлених під час аналізу, таких як труднощі інтеграції, виклики у зборі та обробці даних.
   - **Потенційний Вплив на Систему:** Оцінка, як ці невирішені питання можуть вплинути на загальну ефективність та надійність системи моніторингу.
3. **Постановка Завдань Дослідження:**

- **Формулювання Дослідницьких Завдань:** Визначення конкретних завдань для майбутніх досліджень, які спрямовані на розв'язання ідентифікованих проблемних питань.
- **Рекомендації щодо Подальших Дій:** Надання рекомендацій щодо можливих напрямків розвитку та вдосконалення системи моніторингу, у тому числі технічних удосконалень, дослідження нових методів обробки даних, розробки нових інструментів для аналізу.

4. **Перспективи та Напрямки Майбутнього Розвитку:**
   - **Довгострокові Перспективи:** Огляд потенційних довгострокових перспектив системи моніторингу, зокрема в контексті технологічних інновацій та змін у сфері управління відходами.
   - **Стратегічне Планування:** Важливість стратегічного планування для адаптації системи до майбутніх викликів та можливостей.

Цей підрозділ слугує важливим зв'язуючим елементом між виконаним аналізом та майбутніми діями. Він не лише підсумовує важливі висновки, але й визначає напрямки для подальшого дослідження, спрямовані на вдосконалення та оптимізацію системи моніторингу відходів руйнації.

2. МЕТОДИ ВИРІШЕННЯ ЗАДАЧ ВНЕСЕННЯ ТА ОБМІНУ ДАННИМИ ЩОДО ВІДХОДІВ РУЙНАЦІЇ

2.1 Координація цілей задачі та множини їх розв’язання

У рамках підпункту "Координація цілей задачі та множини їх розв’язання" у розділі "Методи Вирішення Задач Внесення та Обміну Даними Щодо Відходів Руйнації" дипломної роботи, розглядаються способи координації різних цілей і завдань для оптимізації процесів внесення, обробки та обміну даними про відходи руйнації.

1. **Визначення Цілей та Завдань:**
    - **Формулювання Основних Цілей:** Визначення основних цілей задачі, наприклад, підвищення точності даних, покращення ефективності обміну даними, забезпечення прозорості та доступності інформації.
    - **Декомпозиція на Підзадачі:** Розбиття основної задачі на більш дрібні та управлінні підзадачі, які включають збір даних, їх валідацію, аналіз та зберігання. [4]
2. **Стратегії Координації:**
    - **Інтеграція Завдань:** Впровадження системних підходів для інтеграції різних завдань та процесів, забезпечуючи їх взаємодію та синергію.
    - **Планування Ресурсів:** Аналіз та оптимізація використання ресурсів (людських, технічних, фінансових) для ефективного виконання завдань. [8]
3. **Методи Оптимізації Процесів:**
    - **Автоматизація Процесів:** Впровадження автоматизованих інструментів для збору, обробки та обміну даними, знижуючи ризик людських помилок та підвищуючи швидкість обробки.
    - **Стандартизація Процедур:** Розробка та впровадження стандартних процедур для забезпечення однорідності та відповідності даних. [5]

4. **Використання Сучасних Технологій:**
    - **Застосування ІТ-Рішень:** Використання інформаційних технологій, таких як бази даних, облачні технології, для ефективного зберігання та обміну даними.
    - **Залучення Інноваційних Підходів:** Розгляд використання новітніх методів, як-от штучний інтелект та машинне навчання, для аналізу та прогнозування трендів у сфері відходів руйнації. [6]
5. **Моніторинг та Оцінка Результатів:**
    - **Систематичний Моніторинг:** Регулярний перегляд та оцінка ефективності впроваджених процесів та технологій.
    - **Фідбек та Покращення:** Встановлення механізмів зворотного зв'язку для неперервного покращення процесів на основі отриманих даних та досвіду. [7]

Цей підрозділ наголошує на важливості координації між різними аспектами задачі та підкреслює необхідність комплексного підходу до вирішення викликів, пов'язаних з внесенням та обміну даними про відходи руйнації. Особлива увага приділяється використанню інноваційних технологій та методик для забезпечення ефективності та точності управління даними.

2.2 Вибір і обґрунтування методів вирішення задачі

У рамках підпункту "Вибір і обґрунтування методів вирішення задачі" у розділі "Координація Цілей Задачі та Множини Їх Розв’язання" розглядаються специфічні методи та підходи, які оптимізують процеси в рамках координації задач у сфері моніторингу відходів руйнації. Цей підпункт зосереджується на виборі ефективних методів та їх обґрунтуванні для досягнення поставлених цілей.

1. **Аналіз Потреб та Вимог:**

- **Оцінка Специфіки Задачі:** Проведення глибокого аналізу особливостей та вимог задачі, що включає визначення ключових параметрів, з якими потрібно працювати.
- **Ідентифікація Викликів:** Виявлення основних труднощів та викликів, які можуть виникнути при координації цілей і завдань.

2. **Вибір Методів:**
   - **Методи Аналізу Даних:** Вибір методів для обробки та аналізу даних, таких як статистичний аналіз, моделювання, машинне навчання.
   - **Методи Проектування Процесів:** Розгляд методів оптимізації та проектування процесів, наприклад, методи лінійного програмування, методики Agile та Scrum.
3. **Обґрунтування Вибору:**
   - **Ефективність та Адекватність:** Аналіз, чому обрані методи є найбільш ефективними та адекватними для розв'язання поставлених завдань.
   - **Розгляд Альтернативних Варіантів:** Вивчення можливих альтернативних підходів та обґрунтування, чому вони були відхилені або визнані менш підходящими.
4. **Інтеграція Методів у Процес Роботи:**
   - **Планування Впровадження:** Розробка детального плану впровадження обраних методів у вже існуючі процеси.
   - **Координація з Іншими Підрозділами:** Визначення, як методи будуть координуватися з іншими аспектами проекту та підрозділами.
5. **Моніторинг та Оцінка Результатів:**
   - **Систематичний Моніторинг:** Встановлення процедур для регулярного перегляду та оцінки ефективності впроваджених методів.

- **Оцінка Впливу на Цілі та Завдання:** Вимірювання того, наскільки ефективно обрані методи сприяють досягненню загальних цілей проекту.

Цей підрозділ надає детальне обґрунтування та аргументацію вибору конкретних методів для вирішення задач у рамках координації цілей та завдань. Він також визначає, як ці методи інтегруються в загальну структуру проекту, забезпечуючи ефективність та відповідність запланованим цілям.

2.3 Вибір і обґрунтування засобів вирішення задачі

У підпункті "Вибір і обґрунтування засобів вирішення задачі" важливо зосередитися на виборі конкретних інструментів та технік, які будуть застосовані для досягнення цілей задачі, з урахуванням їхньої ефективності, адекватності та здатності інтегруватися в загальну систему.

1. **Оцінка Вимог до Засобів:**
   - **Аналіз Потреб:** Оцінка потреб проекту та визначення функціональних вимог до засобів, необхідних для вирішення задачі.
   - **Технічні Вимоги:** Визначення технічних параметрів, на які повинні відповідати засоби, включаючи сумісність, масштабованість та безпеку.
2. **Вибір Інструментів та Технік:**
   - **Технологічні Рішення:** Вибір відповідних технологій (наприклад, програмне забезпечення для обробки даних, платформи для обміну даними).
   - **Методики Роботи:** Визначення методик та процедур роботи, які будуть використані для виконання завдань (наприклад, методи проектного управління, стратегії розробки).
3. **Обґрунтування Вибору:**

- **Співставлення з Цілями Проекту:** Аргументація, чому обрані інструменти та методики є найбільш підходящими для досягнення цілей задачі.
- **Ефективність та Кост-Ефективність:** Оцінка, як обрані засоби допоможуть досягти максимальної ефективності з оптимальними витратами.

4. **Планування Впровадження:**
   - **Інтеграція в Процеси:** Розробка плану щодо інтеграції обраних інструментів та методик в існуючі процеси.
   - **Адаптація та Налаштування:** Визначення необхідних кроків для адаптації та налаштування інструментів під конкретні умови проекту.
5. **Моніторинг та Оцінка Ефективності:**
   - **Відстеження Виконання:** Встановлення механізмів для моніторингу виконання та прогресу, з використанням обраних засобів.
   - **Аналіз Результатів:** Регулярний аналіз ефективності застосованих засобів та їхнього впливу на загальні результати проекту.

Цей підрозділ вимагає детального та виваженого підходу до вибору засобів вирішення задачі, зосереджуючись на їхній ефективності, відповідності цілям проекту та можливості інтеграції у вже існуючі системи та процеси. Важливим є також забезпечення гнучкості та адаптивності обраних засобів для відповіді на змінні потреби проекту.

2.4 Розробка моделей проектування інформаційно пошукової системи «Аналіз Матеріалів»

При розробці моделей проектування інформаційно-пошукової системи "Аналіз Матеріалів", яка працює з базою даних на Python з використанням бібліотеки SQLite, важливо врахувати ряд ключових аспектів, пов'язаних з функціональністю та користувацьким інтерфейсом. Нижче наведено детальний опис розробки цієї системи. [3]

1. **Структура Бази Даних:**
    - **Схема Бази Даних:** Розробка схеми бази даних, включаючи визначення таблиць, їх атрибутів, типів даних та взаємозв'язків між ними.
    - **Автоматичне Створення Бази Даних:** Програма повинна мати можливість автоматично створювати базу даних, якщо вона не знайдена, забезпечуючи легкість початку роботи з системою.
2. **Інтерфейс Користувача:**
    - **Дизайн Інтерфейсу:** Розробка інтуїтивно зрозумілого та зручного інтерфейсу користувача, включаючи розташування кнопок та візуальні елементи.
    - **Функціональні Кнопки:** Інтеграція кнопок для вибору поточної таблиці, додавання, редагування, оновлення записів, прикріплення та перегляду фото та PDF-протоколів, виходу з програми та пошуку за ID.
3. **Функціональність Системи:**
    - **Обробка Запитів:** Розробка логіки обробки запитів користувача, включаючи пошук, внесення та редагування даних.
    - **Інтеграція з SQLite:** Використання можливостей бібліотеки SQLite для забезпечення ефективного зберігання та обробки даних.
4. **Управління Даними:**
    - **Додавання та Редагування Записів:** Розробка механізмів для легкого та надійного додавання нових записів та редагування існуючих.
    - **Оновлення Відображення Даних:** Забезпечення актуальності відображених даних через функціонал оновлення.
5. **Робота з Файлами:**
    - **Прикріплення Файлів:** Впровадження можливості прикріплення фото та PDF-документів до записів у базі даних.

- **Перегляд Прикріплених Файлів:** Розробка функціоналу для перегляду та доступу до прикріплених файлів.

6. **Безпека та Надійність:**
   - **Захист Даних:** Впровадження заходів безпеки для захисту бази даних від несанкціонованого доступу.
   - **Резервне Копіювання:** Розробка механізмів для резервного копіювання даних, забезпечуючи їх відновлення у випадку втрати. [9]
7. **Тестування та Оптимізація:**
   - **Тестування Функціональності:** Виконання ретельного тестування всіх функцій системи для забезпечення їх коректної роботи.
   - **Оптимізація Продуктивності:** Проведення оптимізації для підвищення швидкості та ефективності роботи системи.

Розробка моделей проектування такої системи вимагає комплексного підходу, що поєднує технічну компетентність з увагою до користувацького досвіду. Важливо забезпечити, щоб система була не тільки функціональною, але й інтуїтивно зрозумілою та легкою у використанні.

2.5. Технологія і засоби побудови бази даних

Розробка бази даних для інформаційно-пошукової системи "Аналіз Матеріалів" на основі використання Python і SQLite включає створення структурної схеми, визначення технологічного стеку та розробку інтерфейсу користувача. Давайте розглянемо детально кожен з цих аспектів. [13]

1. Структура Бази Даних:
   - База даних materials.db містить таблиці Glass, Metal, Other, Plastic, які здаються спеціалізованими під різні типи матеріалів. Кожна таблиця має поля ID, History, ChemicalComposition, Benchmark, ResearchResults, HistoryImage, ChemicalProtocol, EthalonPhoto, PhotoResearch, PDFResearch.
   - Ця структура дозволяє зберігати детальну інформацію про кожен

матеріал, включаючи його історію, хімічний склад, еталони, результати досліджень та пов'язані медіафайли.

2. Технологічний Стек:
    - Використання Python та SQLite забезпечує легкість розробки та гнучкість. Python ідеально підходить для написання скриптів з обробки даних і взаємодії з базою даних, а SQLite пропонує легкий і надійний спосіб для зберігання даних.[22]
    - Використання бібліотеки tkinter для створення графічного інтерфейсу користувача (GUI) дає можливість створити інтуїтивно зрозумілий інтерфейс, який полегшує взаємодію з базою даних.[23]
3. Інтерфейс Користувача:
    - Інтерфейс містить набір кнопок для різноманітних дій: вибір поточної таблиці, додавання, редагування, оновлення записів, прикріплення фотографій та PDF-протоколів, перегляд файлів, вихід та пошук за ID.
    - Такий інтерфейс дозволяє користувачам легко взаємодіяти з даними, вносити зміни, прикріпляти додаткові матеріали та проводити пошукові запити.
4. Безпека та Резервне Копіювання:
    - Важливим аспектом є забезпечення безпеки даних, що може включати шифрування бази даних та контроль доступу.
    - Також важливо передбачити механізми резервного копіювання, щоб забезпечити надійність та безпеку даних.[21]

Розробка такої бази даних та інформаційно-пошукової системи вимагає глибокого розуміння потреб користувачів, технічних можливостей та безпекових вимог. Система має бути легкою у використанні, забезпечувати надійне зберігання даних і бути гнучкою для подальшого розвитку та інтеграції з іншими інструментами аналізу та обробки даних.

2.6 Висновки до розділу

У цьому розділі ми розглядали комплексні методи та підходи, які необхідні для ефективного внесення та обміну даними у контексті моніторингу відходів руйнації. Глибокий аналіз виявив, що успіх вирішення цих задач залежить від ряду ключових факторів.

1. **Інтеграція Різноманітних Завдань:**
   - Спершу була визначена необхідність інтеграції різноманітних завдань та процесів, що включають збір, обробку, аналіз, зберігання та обмін даними. Ця інтеграція вимагає високого рівня координації та взаємодії між різними підрозділами та системами.
2. **Ефективність Використання Ресурсів:**
   - Важливою складовою є оптимізація використання ресурсів, включаючи час, бюджет, технологічні можливості та людський потенціал. Ефективне розподілення та використання цих ресурсів дозволяє досягти максимальної продуктивності та ефективності.
3. **Застосування Сучасних Технологій:**
   - Особливий акцент був зроблений на важливості застосування сучасних технологій, включаючи автоматизовані системи, програмне забезпечення для обробки та аналізу даних, а також інноваційні методи обробки інформації.
4. **Тестування та Оптимізація Систем:**
   - Було підкреслено важливість ретельного тестування та оптимізації всіх систем і процесів. Це забезпечує надійність, точність, та швидкість обробки даних, а також знижує ризик помилок.
5. **Безпека та Конфіденційність Даних:**
   - Захист інформації є критично важливим, особливо в контексті чутливих даних про відходи руйнації. Було визначено, що застосування сучасних методів шифрування та захисту даних є невід'ємною частиною ефективного управління інформацією.

6. **Гнучкість та Адаптивність:**
    - Важливість адаптивності та гнучкості систем підкреслюється з огляду на постійно мінливі умови та потреби в області управління відходами.

У підсумку, для успішного внесення та обміну даними в області відходів руйнації потрібно глибоке розуміння технічних, організаційних та управлінських аспектів. Застосування інтегрованого підходу, що об'єднує ефективну інтеграцію процесів, оптимізацію ресурсів, інноваційні технології та сильну орієнтацію на безпеку та надійність, є ключем до розвитку надійної та ефективної системи управління інформацією про відходи руйнації.

3. АНАЛІЗ ПОТОКІВ ДАНИХ В ЗАДАЧАХ СИСТЕМАТИЗАЦІЇ ТА ЕКСПЕРИЗИ ВІДХОДІВ РУЙНАЦІЇ

3.1. Специфіка інформаційно-пошукової системи та бази даних відходів руйнації

У даному пункті здійснюється глибокий аналіз специфіки інформаційно-пошукової системи та бази даних, яка використовується для управління, аналізу та систематизації відходів руйнації. Давайте розглянемо основні аспекти цієї системи.

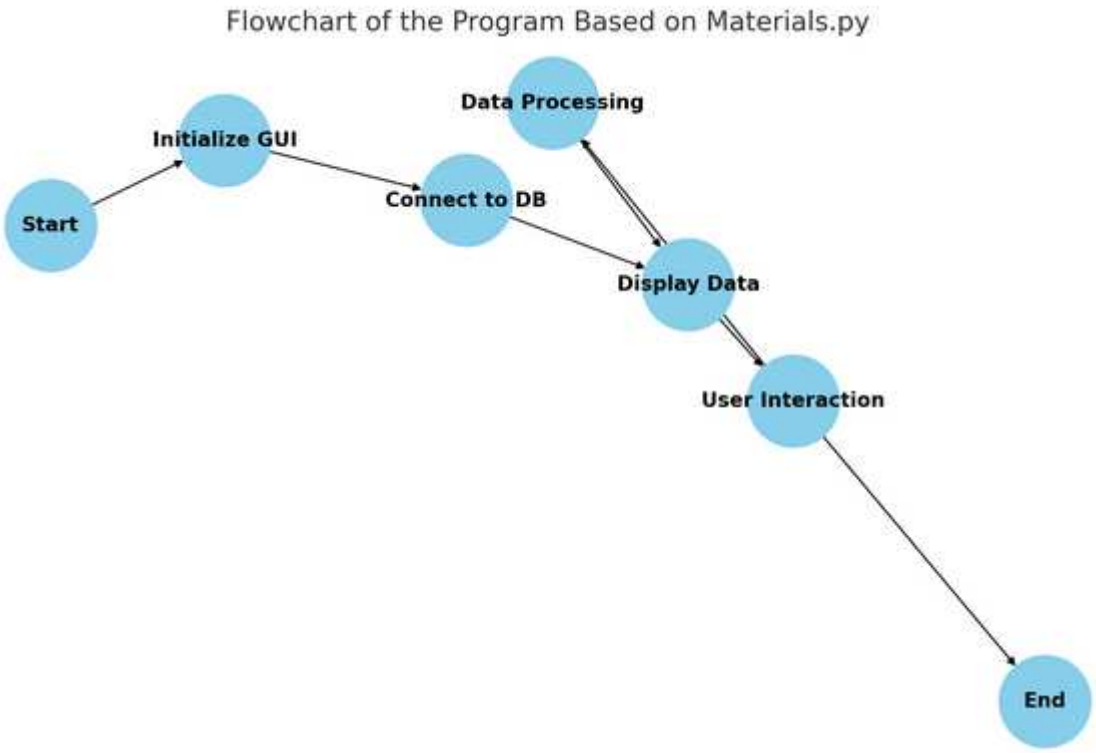


Рисунок 3.1. Блок-схема роботи програми [створено автором]

На зображенні вище представлена блок-схема роботи програми. Схема відображає основні компоненти програми та їх взаємодію:

- **Start**: Початок роботи програми.
- **Initialize GUI**: Ініціалізація графічного інтерфейсу користувача.
- **Connect to DB**: Підключення до бази даних.
- **Display Data**: Відображення даних у графічному інтерфейсі.

- **User Interaction**: Взаємодія з користувачем через графічний інтерфейс.
- **Data Processing**: Обробка даних відповідно до взаємодії з користувачем.
- **End**: Завершення роботи програми.[24]

Ця схема демонструє послідовність дій у програмі, починаючи від запуску і закінчуючи користувацькими діями та обробкою даних. Вона важлива для розуміння загальної структури та логіки програми.

1. **Архітектура Системи:**
    - Система побудована таким чином, щоб оптимізувати процеси збору, обробки, зберігання та аналізу даних відходів руйнації.
    - Важливо, щоб система була модульною, що дозволяє легко вносити зміни або доповнення без необхідності перебудови всієї системи.
2. **База Даних:**
    - База даних повинна бути структурованою таким чином, щоб максимально відображати всі аспекти відходів руйнації - від їх походження до методів переробки та утилізації.
    - Важливо врахувати не тільки текстові дані, але й включити можливість зберігання мультимедійних файлів, таких як фотографії, відео, а також PDF-документи з результатами досліджень. [10]
3. **Функціональність Системи:**
    - Система повинна мати зручний інтерфейс для введення, пошуку, перегляду та аналізу даних.
    - Особлива увага приділяється функціям пошуку та фільтрації даних, що є ключовими для ефективної роботи з великими обсягами інформації.
4. **Інтеграція та Сумісність:**
    - Система повинна бути сумісною з іншими інформаційними системами та базами даних, що дозволяє обмінюватися даними та інтегруватися з широким спектром інструментів.

- Важливо передбачити можливість експорту даних у різні формати для подальшої обробки або аналізу.

5. **Масштабованість та Адаптивність:**
   - Система повинна бути розроблена з урахуванням потреб у масштабуванні, адже обсяги даних про відходи руйнації можуть зростати.
   - Адаптивність системи до змін у законодавстві, технологіях та методологіях управління відходами також є важливим аспектом.

Розробка інформаційно-пошукової системи та бази даних відходів руйнації вимагає забезпечення балансу між технічною складністю, функціональністю та зручністю використання. Це передбачає впровадження новітніх технологій, забезпечення високого рівня безпеки та розробку системи таким чином, щоб вона була здатна адаптуватися до постійно змінних умов управління відходами.

3.2. Опис механізмів вводу, виводу, пошуку даних

У даному пункті розглядаються ключові механізми вводу, виводу та пошуку даних у системі. Ці механізми є фундаментальними для забезпечення ефективності та продуктивності роботи з інформаційною системою.

1. **Механізми Вводу Даних:**

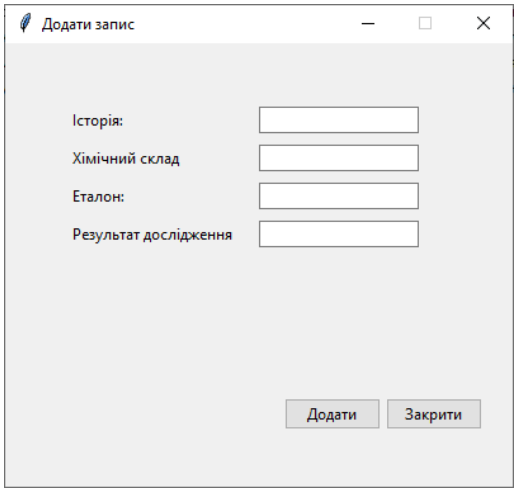


Рисунок 3.2. Вікно Додати запис [створено автором]

Введення даних проходить за допомогою графічного інтерфейсу – заповняються відповідні поля:

- **Історія**
- **Хімічний склад**
- **Еталон**
- **Результат дослідження**

Після заповнення необхідних полів, користувач натискає кнопку “Додати”.

Також в системі є можливість редагування вже занесених даних (представлено на Рисунку)

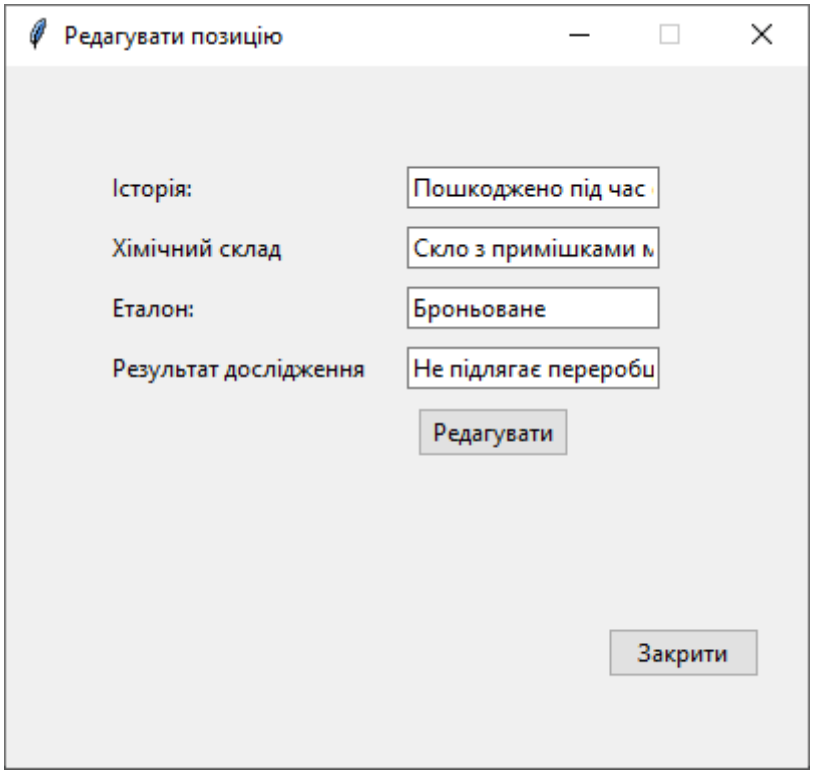
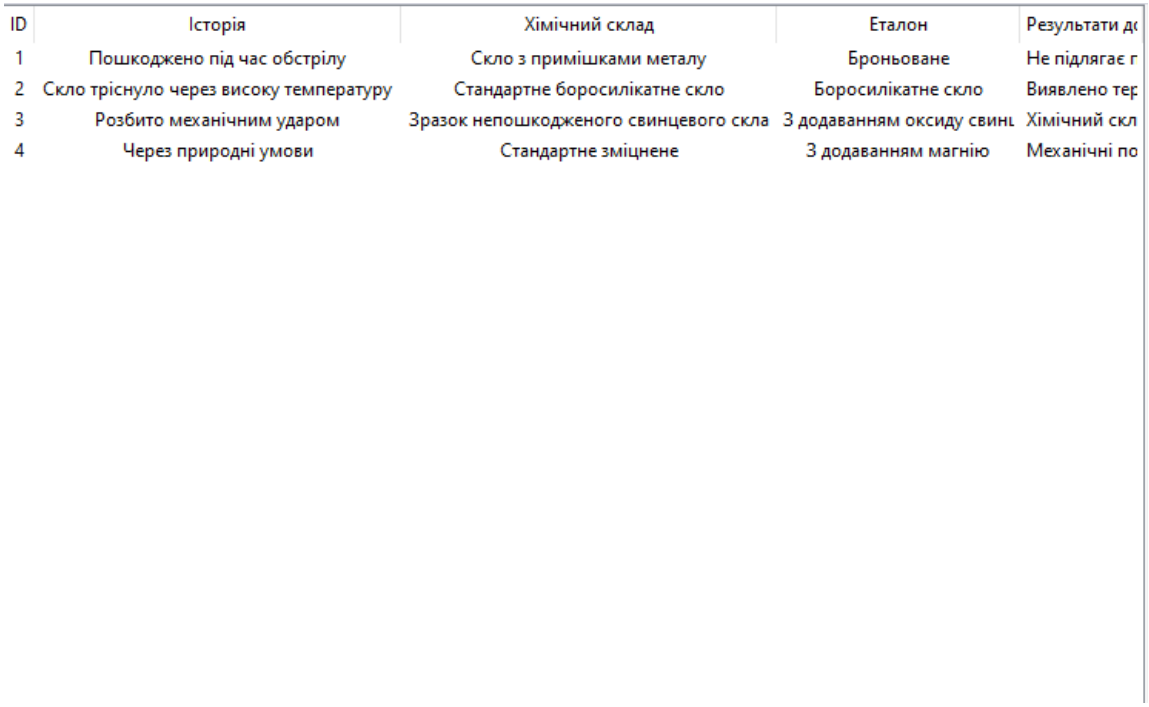


Рисунок 3.3. Вікно Редагувати позицію [створено автором]

Користувач натискає кнопку редагувати та запис зберігається в базі даних вже відредагованим.

2. **Механізми Виводу Даних:**

Виведення даних проходить в спеціальному віджеті для відображення даних (**Treeview**)



ID	Історія	Хімічний склад	Еталон	Результати д
1	Пошкоджено під час обстрілу	Скло з примішками металу	Броньоване	Не підлягає г
2	Скло тріснуло через високу температуру	Стандартне боросилікатне скло	Боросилікатне скло	Виявлено тер
3	Розбито механічним ударом	Зразок непошкодженого свинцевого скла	З додаванням оксиду свинц	Хімічний скл
4	Через природні умови	Стандартне зміцнене	З додаванням магнію	Механічні по

Рисунок 3.4. Віджет Treeview [створено автором]

Кожне поле має адаптивність, наскільки дозволяє розмір віджета, але якщо поле не показує всю занесену в нього інформацію то його можна розтянути щоб відобразити всю інформацію.

3. **Механізми Пошуку Даних:**

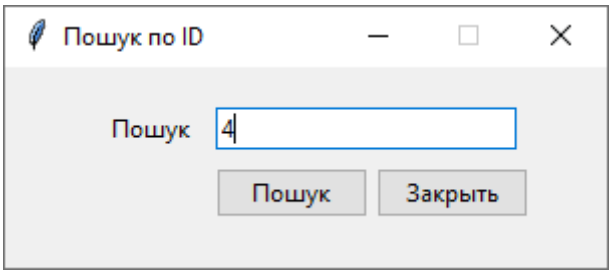


Рисунок 3.5. Віджет Пошук по ID [створено автором]

Пошук даних проходить по ID запису в кожній таблиці окремо, реалізація на пошук саме по полю ID є найкращим варіантом, оскільки дане поле є

індивідуальним для кожного запису, в кожній таблиці окремо (інші поля можуть мати ідентичні дані в своїх записах.

Застосування цих механізмів в інформаційно-пошуковій системі для відходів руйнації дозволяє оптимізувати процеси роботи з даними, забезпечувати їх точність, доступність та корисність для користувачів. Особлива увага приділяється зручності та інтуїтивності інтерфейсу, щоб користувачі могли ефективно використовувати систему незалежно від їхнього досвіду роботи з подібними системами.

3.3. Приклад реалізації

На основі прикріпленого коду програми Materials.py, можемо описати приклад реалізації інформаційно-пошукової системи для відходів руйнації. Програма написана на мові програмування Python і використовує бібліотеку tkinter для створення графічного інтерфейсу користувача та sqlite3 для роботи з базою даних. [11]

1. **Основна Структура Програми:**
    - Клас **Main** розширює **tk.Frame** і є основним класом програми. Він ініціалізує головне вікно програми, створює об'єкти для роботи з різними частинами бази даних (**db_glass, db_metal, db_other, db_plastic**). Головне вікно програми має наступний вигляд:

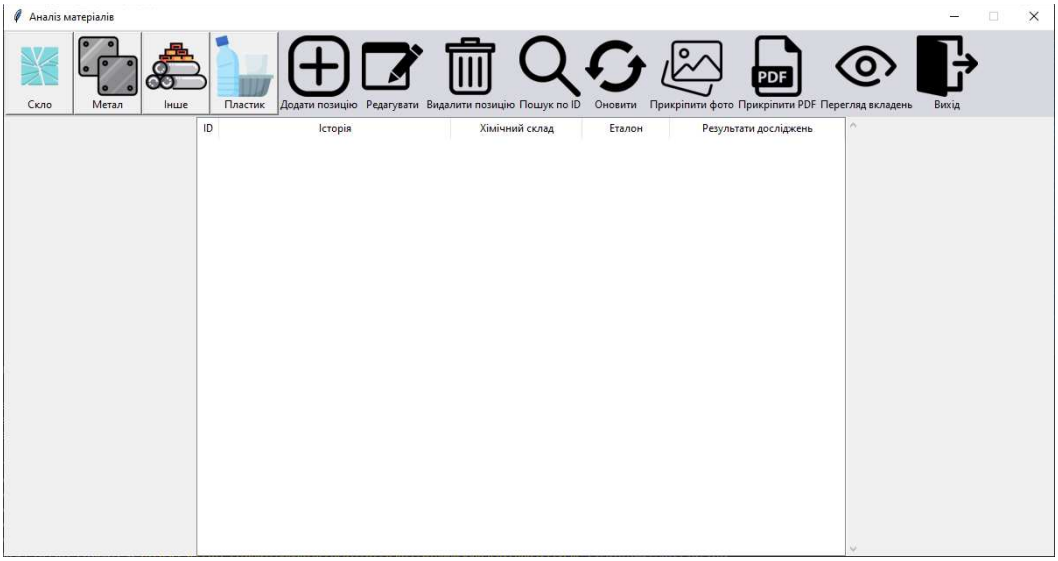


Рисунок 3.6. Головне вікно програмного додатку [створено автором]

- В методі **init_main** створюється тулбар з кнопками для взаємодії з базою даних: вибір таблиці, додавання, редагування, оновлення записів, прикріплення файлів та ін. Тулбар головного вікна виглядає наступним чином:

Рисунок 3.7. Тулбар програмного додатку [створено автором]

Тулбар головного вікна включає в себе головні елементи управління програмним застосунком:

- **Кнопки вибору поточної таблиці:**

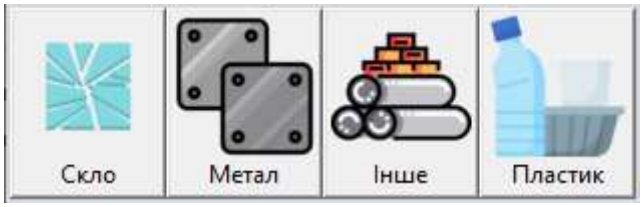


Рисунок 3.8. Кнопки вибору таблиць [створено автором]

Кожна кнопка вибору таблиці відповідає своїй назві та працює з однойменними таблицями, встановлюючи обрану таблицю в якості поточної, що означає виведення на віджет Treeview інформацію лише з поточної таблиці, при наступному виборі таблиці робоча область віджета очищається та виводить відповідні дані для останньої обраної таблиці.

- **Кнопка додати позицію:**

Рисунок 3.9. Кнопка додати позицію [створено автором]

Відповідає за наповнення бази даних новими записами та відкриває наступне дочірнє вікно для заповнення даних

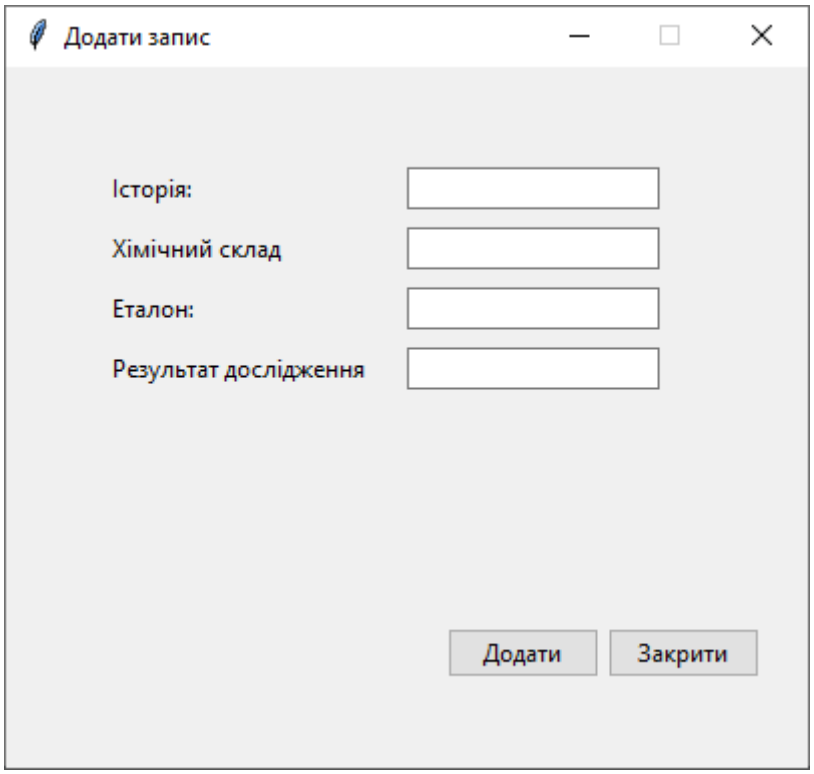


Рисунок 3.10. Вікно додати запис [створено автором]

- **Кнопка Редагувати:**

Рисунок 3.11. Кнопка Редагувати [створено автором]

Виконує функцію редагування вже сформованих записів в базі даних та викликає наступне дочірнє вікно:

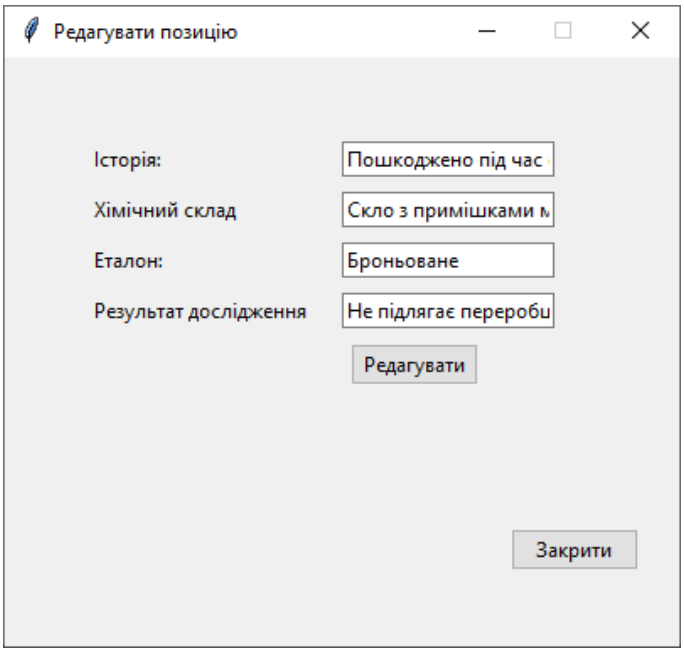


Рисунок 3.12. Вікно редагувати позицію [створено автором]

- **Кнопка Видалити позицію**

Рисунок 3.13. Кнопка Видалити позицію [створено автором]

Виконує функцію видалення вже наявних записів в базі даних.

- **Кнопка Пошук по ID**

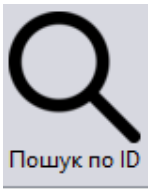


Рисунок 3.14. Кнопка Пошук по ID [створено автором]

Виконує функцію пошуку запису для поточної таблиці по його ID за допомогою наступного дочірнього вікна:

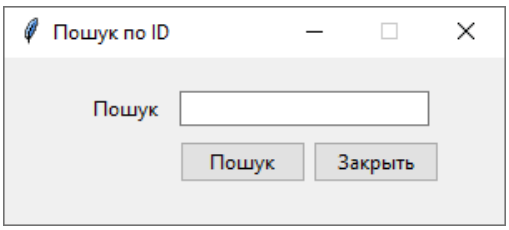


Рисунок 3.15. Вікно Пошук по ID [створено автором]

- **Кнопка Оновити**

Рисунок 3.16. Кнопка Оновити [створено автором]

Виконує функцію оновлення віджета Treeview та даних котрі він включає в собі на поточний момент.

- **Кнопка Прикріпити фото**

Рисунок 3.17. Кнопка Прикріпити фото [створено автором]

Виконує функцію прикріплення фото-звіту до запису в поточній таблиці за допомогою наступного дочірнього вікна:

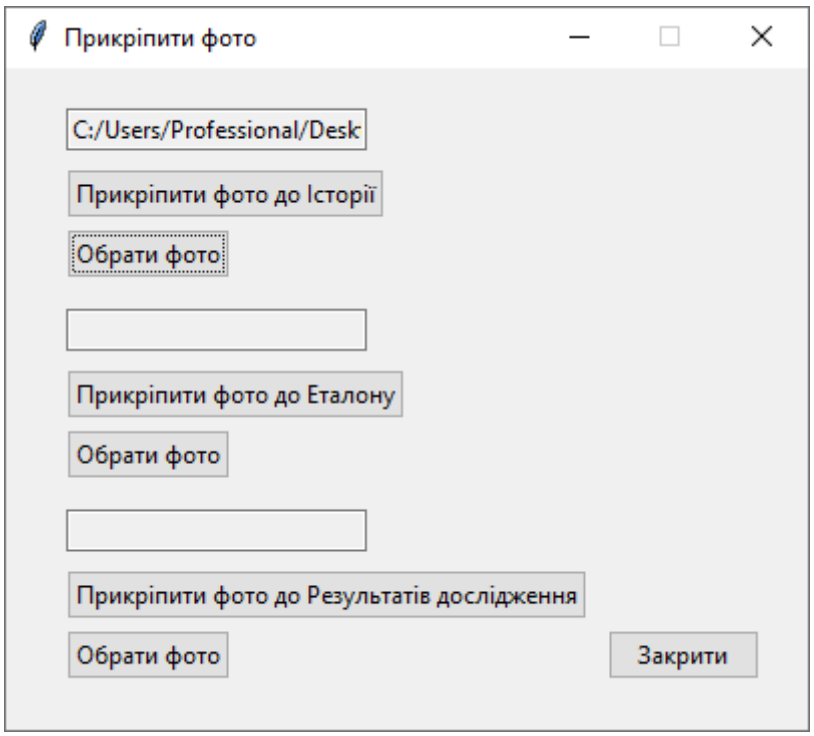


Рисунок 3.18. Вікно Прикріпити фото [створено автором]

Дане дочірнє вікно включає в себе прикріплення фото-звітів відразу до трьох записів на вибір, над кожною парою кнопок, що відповідають за конкретне поле в записі є текстове поле, в котре можна прописати шлях до фото-звіту (з повною назвою файлу фото-звіта) та прикріпити дане фото до обраного запису та обраного поля запису. Вибір запису виконується курсором в області відображення даних на віджеті Treeview.

- **Кнопка Прикріпити PDF**

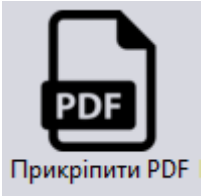


Рисунок 3.19. Кнопка Прикріпити PDF [створено автором]

Виконує функцію прикріплення PDF-протоколів до записів та має дочірнє вікно, функціонал котрого аналогічний функціоналу дочірнього вікна Прикріпити фото

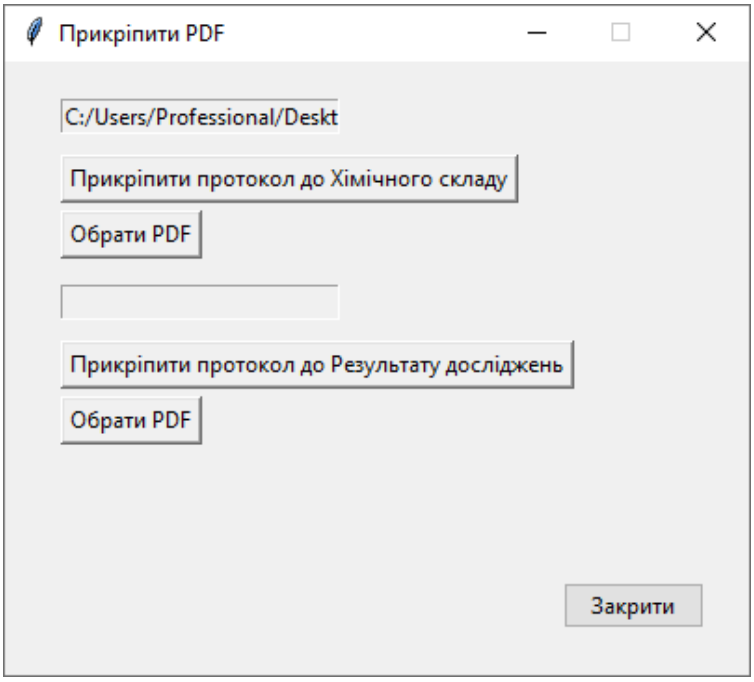


Рисунок 3.20. Вікно Прикріпити PDF [створено автором]

- **Кнопка Перегляд вкладень**

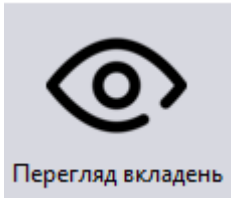


Рисунок 3.21. Кнопка Перегляд вкладень [створено автором]

Виконує функцію перегляду прикріплених фото-звітів та PDF-протоколів, даний функціонал реалізовано за допомогою дочірнього вікна “Переглянути прикріплені файли” та системних інструкцій для відкривання фото стандартним додатком системи, а PDF-протоколи за допомогою веб-браузера.

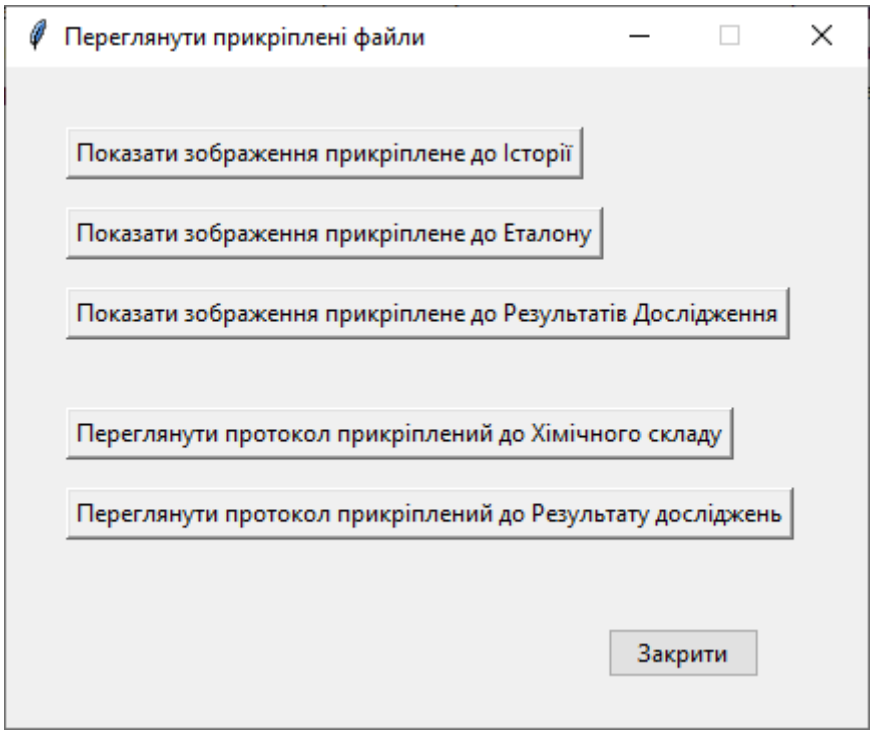


Рисунок 3.22. Вікно Переглянути прикріплені файли [створено автором]

Переглянемо зображення прикріплене до Історії запису обраному в віджеті Treeview

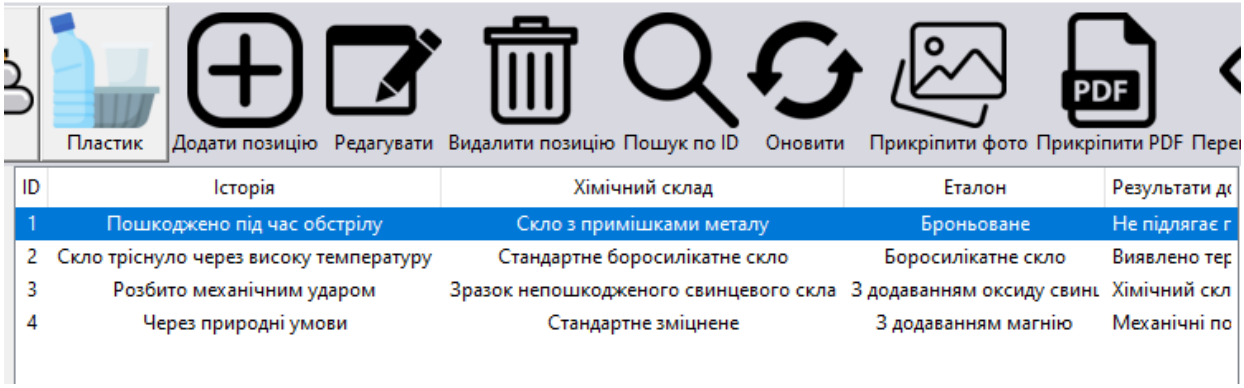


Рисунок 3.23. Віджет Treeview [створено автором]

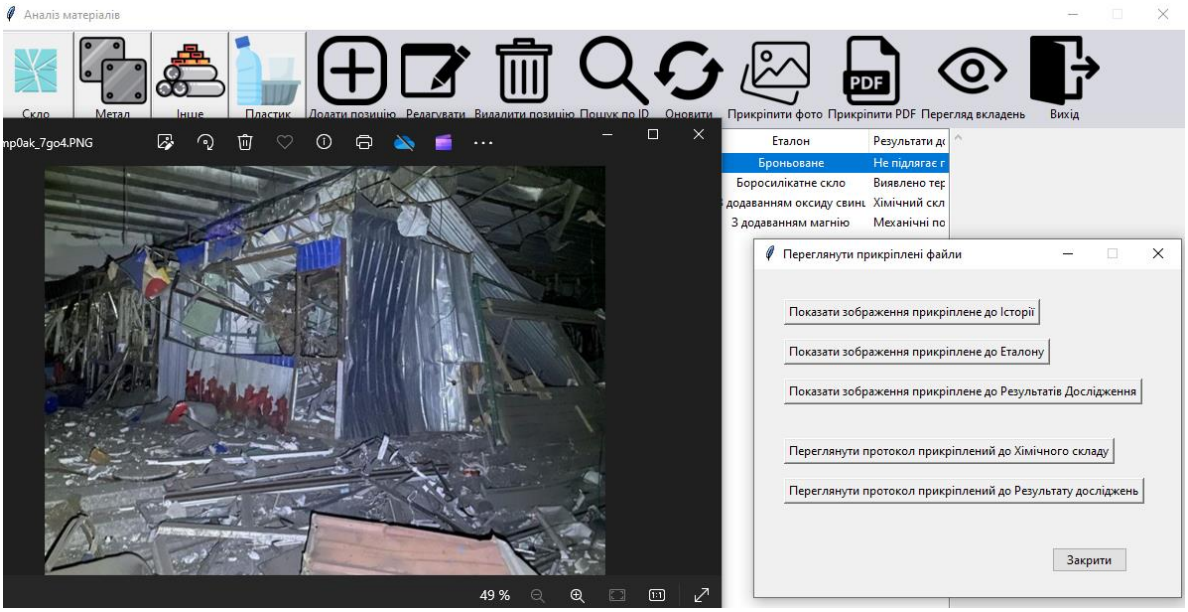


Рисунок 3.24. Перегляд прикріпленого фото [створено автором]

- **Кнопка Вихід**

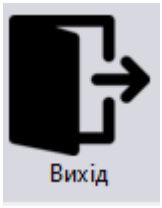


Рисунок 3.25. Кнопка Вихід [створено автором]

Відповідає за вихід з програмного застосунку.

2. **Робота з Базою Даних:**
   - Використання **sqlite3** для підключення до бази даних і виконання SQL-запитів. Це дозволяє виконувати операції читання, запису, оновлення та видалення даних з бази.
   - Програма використовує SQLite для зберігання та обробки даних. Код включає функції для створення та взаємодії з базою даних, включаючи додавання, оновлення та видалення записів.
   - Програма містить окремий клас **DB**, який управляє взаємодією з конкретними таблицями бази даних.

- Автоматичне створення бази даних забезпечується за допомогою логіки перевірки наявності файлу бази даних при запуску програми.

3. **Графічний Інтерфейс Користувача (GUI):**
   - Інтерфейс реалізований за допомогою **tkinter**, що дозволяє створити візуально привабливий та інтуїтивно зрозумілий інтерфейс.
   - Програма має різноманітні елементи управління, такі як кнопки для взаємодії з даними, вікна для введення даних, а також інструменти для відображення та оновлення інформації. [12]
4. **Функціональність Програми:**
   - Програма забезпечує можливість вибору поточної таблиці для роботи, додавання нових записів, редагування та оновлення існуючих записів.
   - Реалізовано функціонал для прикріплення фотографій та PDF-протоколів до записів, а також для перегляду цих файлів.
5. **Робота з Медіафайлами:**
   - Функціонал прикріплення та перегляду фотографій та PDF-документів дозволяє зберігати та відображати додаткові матеріали, пов'язані з конкретними записами в базі даних.
6. **Функціональність Пошуку:**
   - Програма включає механізми для пошуку даних у базі, дозволяючи користувачам ефективно фільтрувати та знаходити необхідну інформацію.

Цей приклад реалізації демонструє, як можна ефективно об'єднати графічний інтерфейс та функціональність бази даних для створення комплексної системи управління даними. Важливою є інтуїтивна навігація, зручність використання та гнучкість системи, що робить її придатною для широкого кола завдань, пов'язаних з відходами руйнації.

3.4. Висновки до розділу

На основі проведеного аналізу можна сформулювати такі висновки:

1. Розробка спеціалізованої інформаційно-пошукової системи та бази даних для відходів руйнації є критично важливою для забезпечення ефективного управління цими відходами. Необхідно враховувати унікальні характеристики та вимоги, пов'язані з такими відходами, забезпечуючи гнучкість та адаптивність системи.
2. Ефективні механізми вводу, виводу та пошуку даних є вирішальними для оптимізації роботи з системою. Важливо забезпечити точність, швидкість та зручність цих процесів, а також розробити надійні методи валідації та перевірки даних.
3. Приклад реалізації, демонстрований на основі конкретного програмного коду, підтверджує практичну застосовність розроблених методів та підходів у реальних умовах. Це демонструє можливість інтеграції різних функціональних елементів та використання сучасних технологій для створення ефективної системи.

Загалом, успіх у сфері управління відходами руйнації залежить від здатності систематично збирати, аналізувати та використовувати дані. Розроблені інструменти та методики повинні враховувати специфіку цієї області, забезпечуючи надійність, доступність та зручність використання.

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ РОЗРОБКИ

4.1 Впровадження програми в експлуатацію

В наш час, в епоху розвитку інформаційних технологій, компанії по всьому світу, включаючи українські, активно використовують різноманітне програмне забезпечення для управління, контролю, організації та проектування різних аспектів своєї діяльності. За останні роки ми спостерігаємо значне збільшення значення цього напряму в бізнесі, оскільки компанії інвестують великі суми в розробку, вдосконалення, тестування та інтеграцію програмного забезпечення у свої процеси.

Очевидно, що створення програмного забезпечення стало ключовою діяльністю для багатьох IT-компаній, які розробляють на замовлення високоякісне програмне забезпечення. При впровадженні програмного продукту на підприємстві, на етапі дослідження важливо оцінити цілі та мету створення цього продукту.

На цій стадії розробляється бізнес-план, визначаються потреби у матеріальних ресурсах, обирається технологія для досягнення потрібних функцій, аналізуються можливі шляхи досягнення цілей, порівнюються пропозиції від постачальників та консультантів, встановлюється юридична основа тощо. Етап досліджень завершується отриманням наукових результатів, які слугують фундаментом для розробки програмного продукту. Це стає наступним кроком у процесі створення програмного продукту. На цьому етапі компанія вже визначається з підходом до розробки майбутнього продукту. Через велику різноманітність функціональних особливостей програмних продуктів, складно узагальнити види робіт, необхідних на цьому етапі.

Капіталізація витрат повинна включати прямі матеріальні витрати, такі як сировина, комплектуючі, запасні частини тощо, а також послуги сторонніх організацій виробничого характеру, використані під час розробки програмного забезпечення. До цього слід додати витрати на оплату праці співробітників, залучених у виробництво такого продукту, внески на соціальні потреби, витрати

на утримання та обслуговування обладнання та інші загальновиробничі витрати.[18]

Подивимося на необхідні витрати, проблеми та цілі, пов'язані з розробкою програмного забезпечення і поліпшенням вже придбаного програмного продукту, а також на процес реінжинірингу. Впровадження програмного продукту на підприємство для управління або організації роботи – це складний процес, який включає аналіз, збір інформації та прийняття рішень. Розглянемо проектування програмного пакету на замовлення, часто описуваного як створення "з нуля".

Переваги такого підходу включають:

- Замовник може розробити вимоги до продукту і замовити тільки необхідний функціонал;
- Можливість точно розрахувати терміни розробки;
- Великий вибір засобів розробки;
- Наймання програмістів потрібної кваліфікації, економія на нескладних задачах;
- Гнучкість у методах і засобах в залежності від фінансування.

Недоліки включають:

- Тривалий час розробки залежно від обсягу функціоналу;
- Якість залежить від кваліфікації виконавців;
- Потреба у доопрацюванні або повній заміні продукту при зміні вимог;
- Необхідність навчання персоналу для роботи з новим програмним забезпеченням;
- Необхідність мати працівника з певною кваліфікацією для технічної підтримки розробленого продукту.

4.2 Структура та принципи організації виробничого процесу

Процес виробництва представляє собою набір взаємопов'язаних дій, які включають людську працю, інструменти та природні ресурси, необхідні для створення товарів. Його ключові складові – це трудова діяльність людей як

свідомий процес, об'єкти та інструменти праці. Ці компоненти є основними ресурсами виробничого процесу і вимагають фінансових витрат. Крім того, в багатьох галузях промисловості задіяні природні процеси, такі як біологічні та хімічні реакції в аграрному та агропромисловому виробництві, сушіння або охолодження деталей після термічної обробки. Ці природні процеси вимагають часу і використовують ресурси лише у випадку, коли їх штучно інтенсифікують.[19]

Ключовим елементом виробничого процесу є технологічний процес, що являє собою комплекс дій, спрямованих на зміну стану і характеристик матеріалу або предмету праці. На підприємствах існують різні типи виробничих процесів, які класифікуються за критеріями, такими як мета, часовий перебіг та рівень автоматизації.

Виробничі процеси поділяють на основні, допоміжні та обслуговуючі. Основні процеси стосуються безпосереднього виробництва головної продукції компанії, яка формує її виробничий профіль і спеціалізацію та постачається на ринок як товар. Такі процеси часто складаються з декількох етапів, включаючи заготівлю, обробку та складання, що утворюють основне виробництво.

Допоміжні процеси пов'язані з виробництвом товарів, які використовуються власне на підприємстві для забезпечення нормальної роботи основних процесів, і включають такі галузі, як ремонт, інструментальне виробництво, енергетика та інші. Обслуговуючі процеси забезпечують необхідні умови для основних і допоміжних процесів і включають логістику, транспортування та складування.

Щодо часового перебігу, виробничі процеси поділяють на дискретні (перервні) та безперервні. Дискретні процеси характеризуються циклічністю і виробництвом продукції, яка має визначену форму та розмір і вимірюється в одиницях (наприклад, машини, прилади, одяг). Безперервні процеси характеризуються виробництвом продукції без фіксованого об'єму та форми, такої як сипкі, рідкі або газоподібні матеріали, і не вимагають технологічної циклічності.

У контексті автоматизації, виробничі процеси поділяються на ручні, механізовані, автоматизовані та повністю автоматичні. Ручні процеси залежать від фізичної сили працівника, який безпосередньо виконує роботу. У механізованих процесах працівник використовує машини для виконання завдань, виконуючи лише допоміжні функції. В автоматизованих процесах машини працюють під наглядом працівника, який відповідає за деякі допоміжні операції. Автоматичні процеси ж повністю керуються машинами згідно з попередньо запрограмованими інструкціями без участі людини.

Основною структурною одиницею виробничого процесу є операція, яка є окремою частиною цього процесу, виконаною на одному робочому місці над одним предметом праці без перенастроювання устаткування. Серед всіх операцій особливо виділяють технологічні операції, які у своїй сукупності утворюють технологічний процес.

Принципи організації виробничого процесу забезпечують його ефективне функціонування у просторі та часі. При проектуванні та організації виробничих процесів застосовуються різні принципи, такі як спеціалізація, пропорційність, паралельність, прямоточність, безперервність, ритмічність, автоматичність, гнучкість та гомеостатичність. Принцип спеціалізації полягає в обмеженні різноманітності елементів виробничого процесу, зокрема, зменшенні номенклатури продукції та різновидів виробничих операцій на робочих місцях. Спеціалізація спрощує організацію виробництва, сприяє його автоматизації, покращує використання ресурсів підприємства, підвищує якість продукції та знижує її собівартість.

Ступінь внутрішньофабричної спеціалізації значно залежить від стандартизації конструкції, технології та організаційних аспектів. Стандартизація полягає у приведенні продуктів, методів виробництва або їх елементів до єдиної форми, розміру, структури чи складу. Це дозволяє скоротити асортимент деталей і вузлів, раціонально обмежити різноманітність виробничих методів, типів і марок обладнання, маршрутів виробництва деталей тощо.

Спеціалізація впливає на впровадження інших принципів ефективної організації виробництва. Принцип пропорційності вимагає гармонізації потужності всіх частин виробничого процесу, усієї системи взаємозв'язаних підрозділів та машин. Пропорційність досягається, коли загальна продуктивність технологічно пов'язаних елементів виробництва відповідає обсягу виконаних робіт.

Недотримання цього принципу може призвести до виникнення "вузьких місць" або недостатнього завантаження окремих підрозділів. На підприємствах зі складною виробничою структурою важко досягнути повної пропорційності потужностей окремих підрозділів. Диспропорції часто виникають через запровадження нових продуктів, різні темпи зниження трудомісткості у різних підрозділах і є природним результатом розвитку і динамічного функціонування виробництва. Однак важливо передбачати та мінімізувати ці диспропорції.

Принцип паралельності передбачає одночасне виконання різних операцій та процесів. Він особливо важливий у виробництві складних продуктів, які складаються з багатьох деталей і вузлів. Паралельність досягається шляхом логічного поділу виробів на компоненти, одночасного виконання різних операцій над ними та одночасного виготовлення різних продуктів. На робочих місцях паралельність забезпечується багатоінструментальною обробкою заготовок та одночасним виконанням основних та допоміжних операцій.

Принцип прямоточності полягає у тому, що предмети праці повинні рухатися найкоротшим шляхом на всіх стадіях та в рамках усіх операцій виробничого процесу, уникаючи зворотних переміщень. Для дотримання цього принципу виробничі цехи, дільниці та робочі місця розміщуються відповідно до ходу технологічного процесу. Допоміжні виробництва та служби розташовуються якомога ближче до основних підрозділів, які вони обслуговують.

Принцип безперервності виробництва вимагає, аби часові проміжки між послідовними технологічними операціями були зведені до мінімуму або повністю усунені. Цей принцип найефективніше реалізується у секторах з безперервним виробництвом, таких як хімічна, металургійна чи енергетична галузі.[1] В

дискретних виробництвах, де технологічний процес відрізняється великою різноманітністю, повне усунення перерв неможливе через технологічні та організаційні обмеження. Тут важливо мінімізувати перерви шляхом синхронізації операцій і впровадження передових методів управління виробництвом. Крім того, неперервність роботи обладнання та персоналу також є важливою.

Принцип ритмічності полягає в регулярному та послідовному виконанні робіт усіма підрозділами підприємства, а також у випуску продукції за певним ритмом. При дотриманні цього принципу, визначена або зростаюча кількість продукції виробляється в однакові часові проміжки, що забезпечує рівномірне навантаження робочих місць і дозволяє максимально використати виробничі потужності.

Принцип автоматичності передбачає максимальне звільнення людини від безпосередньої участі в виробничих процесах, особливо в умовах важкої та шкідливої праці. Автоматизація стосується не лише виробничих процесів, а й інших аспектів діяльності, включаючи управління.

Принцип гнучкості означає, що виробничий процес повинен швидко адаптуватися до змін умов виробництва, наприклад при переході на виготовлення нової продукції або її модифікації. Гнучкість дозволяє швидко освоїти нову продукцію з мінімальними витратами. Важливість гнучкості зростає в умовах швидкого науково-технічного прогресу та частої зміни об'єктів виробництва. Гнучке виробництво ефективно реагує на зміни ринкової кон'юнктури, підвищуючи конкурентоспроможність. Досягається це завдяки універсальності обладнання, автоматизаційних систем, методів обробки, а також використанню верстатів з числовим програмним управлінням та гнучких виробничих систем.

Принцип гомеостатичності виробничої системи полягає у здатності системи стабільно виконувати свої основні функції, навіть при наявності відхилень або зовнішніх негативних впливів. Це досягається завдяки впровадженню технічних і організаційних механізмів, які дозволяють системі самостійно регулювати і стабілізувати свою діяльність. Організаційні системи стабілізації включають,

серед іншого, системи оперативного планування та регулювання виробництва, планового ремонту обладнання, наявності резервних запасів та інші заходи.

Всі розглянуті принципи раціональної організації виробничого процесу є взаємопов'язаними, взаємодоповнюючими і в різній мірі застосовуються на практиці. При проектуванні та організації виробничого процесу необхідно враховувати ці принципи, але вибір оптимальних організаційно-технічних рішень повинен базуватися на критеріях економічної ефективності.

4.3 Технологія розробки і впровадження програмного забезпечення

На сьогоднішній день прогрес у сферах економіки, промисловості, науки, техніки та освіти значною мірою залежить від широкого використання комп'ютерних технологій. Кожен комп'ютер під час своєї роботи використовує програмне забезпечення. Створення програмних продуктів вимагає інтелектуальних зусиль, трудових ресурсів та обладнання, що впливає на вартість розробки програмного продукту.

Функції програмного забезпечення для обчислювальних систем подвійні: вони спрощують підготовку завдань користувачами для комп'ютера та полегшують процес їх виконання та налагодження. На рівні машинного коду електронно-обчислювальна машина (ЕОМ) виконує інструкції, які представлені у вигляді цифр, що робить процес читання даних простішим. Загальний час, необхідний для створення програми, складається з декількох компонентів.

Таблиця 4.1 ТНК на розробку и впровадження ПЗ [створено автором]

Найменування операцій	Трудо місткість (н/год)
1	2
1. Дослідження предметної області	105
2. Дослідження вимог користувача до програмного забезпечення	40
3. Розробка технічного завдання	115

4. Проектування програмного забезпечення	175
4.1 Ескізний проект	45
4.2 Технічний проект	60
4.3 Робочий проект	70
5. Тестування та налагодження програми	90
6. Документування застережень.(Інструкція для користувача)	60
7. Впровадження програмного забезпечення	60
Всього	645

4.4 Розрахунок чисельності розробників програмного забезпечення

Планування трудових ресурсів у компаніях має ключове значення і починається з оцінки наявності і ефективності використання кваліфікованих працівників на робочих місцях. Кількість персоналу організації зумовлена такими факторами, як характер, масштаб, складність і трудомісткість виробничих або управлінських процесів, а також рівень механізації, автоматизації та комп'ютеризації цих процесів. Ці аспекти впливають на нормативну (планову) чисельність персоналу. Фактична чисельність персоналу відображається через кількість працівників, які наразі зайняті в організації.

Облік персоналу регулюється відповідними інструкціями зі статистики персоналу та заробітної плати, які визначають методи обліку, середню чисельність працівників, перелік основних та побічних видів діяльності, категорії персоналу, порядок обліку надходження і вибуття матеріалів та інші аспекти.

Встановлені норми часу включають підготовчо-завершальний час, час на обслуговування робочого місця, час на відпочинок та особисті потреби, у тому числі час на фізкультурні паузи, які становлять 10% від оперативного часу. Норми часу визначаються для виконання робіт одним працівником в людино-годинах на одиницю обсягу робіт і документуються у формі нормативних карт.

Розрахунок загальної річної трудомісткості типових видів робіт, які виконуються, наприклад, працівниками сортування, проводиться за певною формулою:

$$T_З = \sum_{1}^{n} T,$$

де Тз - це загальний обсяг роботи, виконаний протягом року в людино-годинах, для стандартного набору завдань;

Т - річний обсяг роботи, виміряний у людино-годинах, для кожного стандартного виду діяльності, який включає вибір, розподіл, підвищення кваліфікації або перепідготовку персоналу;

п - кількість типів робіт, які здійснюються.

Для обчислення річного обсягу роботи за кожним стандартним видом діяльності використовується відповідна формула.

$$T = H_i \times V_i,$$

де Ні - час, витрачений на виконання кожного типового виду робіт, вимірюваний у людино-годинах;

Vi - обсяг кожного типового виду робіт, виконаних протягом року.

Розрахунок бюджету робочого часу передбачає планування кількості днів та годин, які можуть бути відпрацьовані працівником або службовцем за плановий період. Рахунок часу в людино-годинах, як правило, використовується для певної категорії працівників, тоді як для інших категорій персоналу використовують рахунок в людино-днях. В процесі планування персоналу розрізняють календарний, табельний (номінальний), максимально можливий, плановий (ефективний) і фактичний фонди робочого часу.

Кількість працівників, залучених до добору, розстановки, підвищення кваліфікації або перепідготовки, визначається шляхом ділення загальної річної трудомісткості на річний фонд робочого часу одного працівника та помноження на коефіцієнт, який враховує заплановані пропуски працівників (відпустки, лікарняні тощо).

Для визначення необхідної кількості працівників, задіяних у розробці програмного забезпечення, використовується відповідна формула:

$$N = \frac{T}{F_{еф} \cdot K_{вн}},$$

де N – кількість необхідних розробників програмного забезпечення, вимірювана в особах;

T – загальний обсяг роботи, вимірюваний у нормативних годинах;

$F_{еф}$ – продуктивність робочого часу одного працівника;

$K_{н}$ – коефіцієнт, який відображає ступінь виконання встановлених норм.

$$F_{еф} = Д \cdot \gamma \cdot \eta \cdot S,$$

де Д - це кількість робочих днів у періоді,

S - кількість робочих змін у день (одна зміна),

γ - тривалість однієї робочої зміни у годинах (8 годин),

η - коефіцієнт, який враховує часові втрати протягом робочого дня (зазвичай від 0.93 до 0.96).

Для даного розрахунку припускаємо, що Кн.=1,11, а η=0,95.

$$F_{еф} = 40 \cdot 8 \cdot 0{,}94 = 300{,}8 \text{ год.},$$

$$N = \frac{645}{300{,}8 \cdot 1{,}11} = 1{,}9 \text{ чол.}$$

Для створення програмного забезпечення потрібно залучити двох спеціалістів: інженера-програміста та дизайнера.

4.5 Розрахунок вартості матеріально-енергетичних витрат при розробці програмного забезпечення

Витрати на виробничі ресурси, роботи та послуги, які повністю споживаються в процесі кожного технологічного циклу виробництва (виконання робіт або надання послуг), передають свою вартість на собівартість кінцевого

продукту (роботи, послуги). Витрати, що виникають при використанні виробничих запасів, включають:

1. Витрати на сировину та матеріали, придбані у зовнішніх компаній, які стають основою вироблюваного продукту або необхідним компонентом для його виготовлення;
2. Витрати на куплені матеріали, що використовуються під час виробництва для забезпечення нормального технологічного процесу, упаковки, обслуговування обладнання, ремонту, а також запасні частини та інші потреби;
3. Витрати на придбані комплектуючі вироби та напівфабрикати, які потребують монтажу або додаткової обробки;
4. Витрати на виробничі роботи та послуги, надані сторонніми організаціями;
5. Витрати, пов’язані з використанням природних ресурсів, включаючи плату за воду, деревину, компенсації за вилучення земель тощо;
6. Витрати на паливо, придбане у сторонніх підприємств для виробничих потреб;
7. Витрати на придбану енергію всіх видів для виробничих потреб;
8. Втрати, пов’язані з нестачею або природним убутком матеріальних цінностей.[20]

Ці витрати враховуються в повному обсязі при калькуляції собівартості продукції, робіт чи послуг.

Вартість матеріальних ресурсів визначається на основі цін придбання на дату їх обліку (без урахування податку на додану вартість, окрім випадків, передбачених законодавством). До вартості входять націнки та комісійні винагороди постачальників і зовнішньоекономічних організацій, витрати на послуги товарних бірж, брокерські послуги, мита, митні збори, а також витрати на транспортування, зберігання та доставку, що здійснюються сторонніми організаціями.

При включенні витрат на матеріальні ресурси до собівартості продукції віднімається вартість зворотних відходів. Зворотні відходи - це залишки матеріалів, які утворюються в процесі виробництва і втратили свої первинні споживчі властивості, використовуються з додатковими витратами або не використовуються за прямим призначенням.

Витрати на сировину, матеріали, напівфабрикати та паливо розглядаються як фактичне їх використання безпосередньо у виробництві продукції. Виділення цих ресурсів до цехових складів не вважається витратами на виробництво, а розглядається як переміщення матеріалів.

Фактичні витрати матеріалів на виробництво визначаються їх реальним використанням на виготовленій продукції, що відповідає спеціальним документам. Ці витрати повинні бути підтверджені розрахунками і звітами, у яких вони порівнюються з нормативними витратами та аналізуються відхилення від норм.

Сировина, матеріали, придбані комплектуючі вироби та напівфабрикати, виділені на виробництво, але не використані до кінця місяця, щомісяця вилучаються з витрат на виробництво.

Таблиця 4.2 - Розрахунок вартості матеріальних витрат [створено автором]

Найменування матеріальних затрат	Кількість	Одиниця виміру	Ціна за одиницю	Загальна сума в грн
1	2	3	4	5
1. Підключення до інтернету	1	штук	100,00	100,00
2. Модем	1	штук	700,00	700,00
3. Зберігачі інформації	1	штук	125,00	125,00
4. Блок паперу	2	штук	80,00	160,00
5. Картридж	1	штук	650,00	650,00
Всього:				1735,00

Витрати на енергію охоплюють витрати на паливо та енергію, включно з витратами на електроенергію, пар та гарячу воду, які використовуються для технологічних цілей. Під час розробки програмного продукту головним видом енергетичних витрат є електроенергія, тому точний розрахунок цього показника є ключовим. Основною одиницею вимірювання витраченої енергії є ват-година (Вт-год), але частіше використовується кіловат-година (кВт-год). Вартість однієї кВт-год становить приблизно 1,68 гривень.

$В_{ел}$=N*D*T*Ц, де

N – це потужність комп'ютера або принтера за паспортом,

D – тривалість розробки ПЗ,

T – середня тривалість робочої зміни,

Ц – ціна 1кв/год електроенергії (1,68 грн)

$В_{ел.заг.} = В_{ел1} + \cdots + В_{ел\ n}$.

$В_{ел\ комп1}$= 0,45*40*8*1,68 = 241,92 грн.

$В_{ел\ комп2}$= 0,54*40*8*1,68 = 290,3 грн.

$В_{ел.принт}$= 0,24*40*8*1,68 = 129,02 грн.

$В_{ел.заг.}= В_{ел\ комп1} + В_{ел\ комп2} + В_{ел.принт}$= 241,92 +290,3 +129,02= 661,24 грн.

4.6 Розрахунок заробітної плати розробників ПЗ.

Заробітна плата охоплює всі види оплати праці для постійного та позаштатного виробничого персоналу компанії, а саме тих, хто зайнятий у виробництві продукції, підтримці виробничого процесу та управлінні. Витрати на заробітну плату, які фінансуються з прибутку чи інших спеціалізованих джерел, не включаються до собівартості.

Згідно з Законом України "Про оплату праці", заробітна плата визначається як винагорода, зазвичай у грошовій формі, яку роботодавець виплачує працівнику за виконану роботу. Розмір заробітної плати залежить від складності та умов роботи, кваліфікації працівника, його робочих результатів та ефективності діяльності компанії.

Заробітна плата складається з основної частини (постійної) та додаткової (змінної), а також з включення заохочувальних та компенсаційних виплат. Основна заробітна плата визначається відповідно до встановлених норм праці та виплачується у формі тарифних ставок чи відрядних розцінок. Додаткова зарплата включає винагороду за роботу понад встановлені норми, за успіхи у праці та особливі умови праці.

Інші компенсаційні та заохочувальні виплати охоплюють винагороду за роботу за рік, спеціальні премії, компенсації та інші грошові або матеріальні виплати, які не включені до стандартних законодавчих актів або виплачуються понад встановлені норми.

Мінімальна заробітна плата є обов'язковою державною соціальною гарантією і визначається як законодавчо встановлений розмір оплати за просту, некваліфіковану працю. Вона не включає додаткові виплати, такі як доплати чи надбавки.

Різниця між номінальною та реальною заробітною платою полягає в купівельній спроможності, тобто реальна заробітна плата показує, яку кількість товарів можна придбати на одиницю номінальної зарплати.

Фонд заробітної плати представляє собою загальну суму коштів, спрямованих на оплату праці, включаючи оплату не відпрацьованого часу. Цей фонд диференціюється для планування та аналізу його ефективності, розділяючись на фонд основної та додаткової заробітної плати.

$$ФОП = З_{осн} + З_{дод},$$

де ФОП – Фонд оплати праці.

Основна заробітна плата розраховується по формулі:

$$З_{осн} = Т \times С_{чср},$$

де Т – це загальна трудомісткість виконання робіт, взята з таблиці 1.,

$С_{чср}$ – середньо-годинна тарифна ставка в гривнях.

Відповідно середня годинна тарифна ставка розраховується за формулою:

$$C_{чср} = \frac{C_{ч1р} + C_{ч2р}}{n},$$

де n – нормативна чисельність працівників.

$З_{осн}$ – основна заробітна плата.

$З_{дод}$ – додаткова заробітна плата.

Для розробки програмного забезпечення необхідно 2 працівники, інженер – програміст та дизайнер.

Середня ставка за годину для інженера – програміста становить 36,00 грн.

Середня ставка за годину для дизайнера становить 34,00 грн.

Середня годинна ставка :

$$C_{чср} = \frac{36{,}00 + 34{,}00}{2} = 35{,}00 \text{ грн.}$$

Основна заробітна плата робітників згідно трудомісткості складає:

$$З_{осн} = 645 \times 35{,}00 = 22575{,}00 \text{ грн.}$$

Додаткова заробітна плата:

$$З_{дод} = З_{осн} \times 0{,}25;$$

$$З_{дод} = 22575{,}00 \times 0{,}25 = 5643{,}75 \text{ грн.}$$

Обчислюємо фонд оплати праці:

$$ФОП = 22575{,}00 + 5643{,}75 = 28218{,}75 \text{ грн.}$$

Середня заробітня плата працівника за 1 день розраховується за формулою:

$$З_{сер/д} = \frac{ФОП}{D * K}.$$

Оскільки програмний продукт розробляється 40 днів – D = 40 днів, а кількість робітників дорівнює двом – K = 2, середня заробітня плата працівника за 1 день дорівнює:

$$З_{сер/д} = \frac{28218{,}75}{40 * 2} = 352{,}7 \text{ грн.}$$

Середня заробітня плата працівника за 1 годину розраховується за формулою:

$$З_{сер/год} = \frac{З_{сер/Д}}{j},$$

де J – робоча зміна.

Робоча зміна працівника дорівнює 8 годинам, тому J = 8 годин.

Середня заробітня плата працівника за 1 годину дорівнює:

$$З_{сер/год} = \frac{352,7}{8} = 44,08 \text{ грн.}$$

Згідно законодавству єдиний соціальний внесок у 2017 році складає 22% від загального фонду праці.

$$ЄСВ = 0,22 \times ФОП = 0,22 \times 28218,75 = 6208,12 \text{ грн.}$$

4.7 Розрахунок вартості розробки ПЗ

Собівартість продукції представляє сумарні витрати, виражені у грошовій формі, необхідні для підготовки та виробництва продукції. Більш загально, собівартість можна розглядати як грошову оцінку обсягу ресурсів, використаних з певною метою. Це поняття включає три ключові аспекти:

1. Відображення обсягу та виду ресурсів, які були задіяні у виробництві;
2. Вартість використаних ресурсів виражена у грошовому еквіваленті, що дозволяє оцінити загальну вартість цих ресурсів;
3. Специфікація цілі використання ресурсів вимагає точного визначення об'єкта собівартості, будь то виробництво чи реалізація.

Виробнича собівартість продукції охоплює поточні витрати компанії на виробництво продукції, виконання робіт та надання послуг, виражені у грошовій формі. Ця собівартість є фундаментом для встановлення ціни продукції. Інші витрати, як-от адміністративні чи збутові витрати, не входять до собівартості продукції.

Таким чином, прогнозування виробничої собівартості на одиницю продукції включає аналіз наступних статей витрат:

Таблиця 4.3 Розрахунок виробничої собівартості розробки ПЗ [створено автором]

Статті калькуляції	Методика розрахунку	Сума в грн..
1	2	3
Сировина і матеріали	За розрахунками в п. 2.1	1735,00
Енергетичні витрати	За розрахунками в п. 2.1	661,24
Основна ЗП	За розрахунками в п. 2.2	22575,00
Додаткова ЗП	За розрахунками в п. 2.2	5643,75
Амортизаційні нарахування	За розрахунками в п. 2.3	2686,5
ЄСВ	За розрахунками в п. 2.2	6208,12
Загальновиробничі витрати	170% від $З_{осн}$	38377,5
Виробнича собівартість	Сума всіх попередніх статей	77887,11

Амортизація обладнання проводиться за кумулятивним методом, при якому річна сума амортизації обчислюється шляхом множення вартості обладнання, яка підлягає амортизації, на кумулятивний коефіцієнт

$$А_{рік} = С_{п} * К_{кум.}$$

Кумулятивний коефіцієнт визначається як відношення решти років до кінця строку корисного використання обладнання до загальної кількості років його використання

$$K_{кумІрік} = \frac{2}{1+2} = 0,67$$

$$K_{кумІІрік} = \frac{1}{1+2} = 0,33$$

Так, для обладнання, яким є комп'ютери та принтери, з урахуванням їх швидкого зношування, мінімальний термін використання встановлюється у 2 роки.

Амортизаційні відрахування необхідно розрахувати для комп’ютера,принтера.

Розрахунок амортизаційних відрахувань для комп’ютера:

Розрахунок амортизації для комп'ютера проводиться наступним чином: початкова вартість комп'ютера в 12000,00 грн, мінус ліквідаційна вартість 500,00 грн, дає вартість, що підлягає амортизації, в розмірі 11500,00 грн. Таким чином, річна амортизація складе 7705,00 грн (11500,00 * 0,67). Місячна амортизація враховується за 40 днів розробки програмного продукту і становить 1284,16 грн для двох комп'ютерів:

$$С_А = С_п - С_л, \text{ де}$$

$С_п$ – початкова вартість комп’ютера,

$С_л$ – ліквідаційна вартість комп’ютера.

$С_А = 12000,00 - 500 = 11500,00$ грн.

$А_{рік} = С_А * К_{кум}$

$А_{рік} = 11500,00 * 0,67 = 7705,00$ грн.

$$А_{міс.} = \frac{A_{рік}}{12міс.} = \frac{7705,00}{12} = 642,08 \text{ грн.}$$

За умови, що програмний продукт розроблявся 40 днів:

$А_{комп.} = А_{міс.} * 1 = 642,08 * 2 = 1284,16$ грн.

Так як в мене працює два робітника у статті Амортизаційні нарахування я приведу амортизаційну вартість двох комп’ютерів.

Розрахунок амортизаційних відрахувань для принтера:

Для принтера, вартість якого становить 2149,00 грн і який використовується у другий рік з ліквідаційною вартістю 0, проводиться аналогічний розрахунок амортизації:

$С_А = С_п - С_л = 2149,00 - 0 = 2149,00$ грн.

$А_{рік} = 2149,00 * 0,33 = 709,17$ грн.

$$А_{міс.} = \frac{A_{рік}}{12міс.} = \frac{709,17}{12} = 59,09 \text{ грн..}$$

За умови, що програмний продукт розроблявся 40 днів:

$А_{принт.} = 59,09 * 2 = 118,18$ грн.

Загальна сума амортизаційних відрахувань:

$$А_{заг} = А_{комп.} + А_{принт.}$$

$$А_{заг} = 1284,16 + 1284,16 + 118,18 = 2686,5 \text{ грн.}$$

4.8 Розрахунок економічної ефективності та строку окупності ПЗ

Ефективність представляє собою порівняльний показник, який оцінює продуктивність використаних ресурсів.

Ефект є кінцевим результатом певних дій чи заходів, який зазвичай вимірюється у фінансовому виразі.

Основним показником, що відображає економічну ефективність, є економічний ефект. Він показує різницю між вартістю досягнутих (чи очікуваних) результатів та загальними витратами на ресурси за визначений часовий період.

Пряма економічна ефективність обраховується за допомогою специфічної формули:

$$ЕФ = \Delta C - К_{еф} \cdot ВС \quad (4.1)$$

де $К_{еф}$ – коефіцієнт економічної ефективності 0,5;

$ВС$ – виробнича собівартість;

ΔC – вивільнені кошти після впровадження програмного продукту.

ΔC визначається за формулою 4.2.

$$\Delta C = ФОП - ЕВ, \quad (4.2)$$

де $ЕВ$ – експлуатаційні витрати:

$$ЕВ = А_{відр} + МВ + ВЕ, \quad (4.3)$$

де $А_{відр}$ – амортизаційні відрахування;

$МВ$ – матеріальні витрати;

$ВЕ$ – витрати електроенергії;

$$ЕВ = 2686,5 + 1735,00 + 661,24 = 5082,74 \text{ грн.}$$

Пряму економічну ефективність визначаємо ґрунтуючись на тому, що за експертною оцінкою фахівців підприємства впровадження програмного продукту вивільняє 1 працівника, середня ЗП якого складає ≈ 5000,00 грн. в місяць, відповідно, зекономлений фонд заробітної плати за рік складе:

$$ФЗП = 5000 * 12 = 60000 \text{ грн.}$$

Отже маючи ФОП та ЕВ знайдемо вивільнені кошти після впровадження програмного продукту:

$$\Delta C = 60000 - 5082{,}74 = 54917{,}26 \text{ грн.}$$

Після обчислення можна знайти пряму економічну ефективність:

$$ЕФ = 54917{,}26 - 0{,}6 * 77082{,}00 = 54917{,}26 – 46249{,}2 = 8668{,}06 \text{ грн.}$$

Наступним кроком є обчислення окупності програмного засобу. Строк окупності програмного засобу визначається як відношення вартості розробки до вивільнених коштів (4.4):

$$T_{ок} = BC/\Delta C, \tag{4.4}$$

$$T_{ок} = \frac{77887{,}11}{54917{,}26} = 1{,}41 \text{ роки.}$$

Висновок

У цьому розділі дипломної роботи проведено техніко-економічний аналіз процесу створення інформаційної системи. Було здійснено ретельний розрахунок витрат на матеріали та енергію, а також визначено затрати на оплату праці для команди розробників. Також в розділі розглянуто ключові економічні показники, такі як єдиний соціальний внесок (ЄСВ), форма оплати праці (ФОП), нормо-години та трудомісткість роботи. В результаті було визначено загальну вартість розробки системи та оцінено терміни її окупності, що є важливим для оцінки економічної ефективності проекту.

5 ОХОРОНА ПРАЦІ

5.1 Вимоги до організації робочого місця оператора ПК

Робоче середовище впливає на працездатність та здоров'я людей, які працюють з комп'ютерами. У виробничих приміщеннях, таких як офіси, важливими факторами є мікроклімат, освітлення, наявність шкідливих речовин у повітрі, рівень шуму та випромінювання.

1. Мікроклімат виробничого приміщення характеризується температурою, вологістю, рухом повітря та тепловим випромінюванням. Таке середовище впливає на тепловий баланс тіла людини. Охолодження або нагрівання організму здійснюється через випромінювання та випаровування. Висока вологість повітря ускладнює випаровування, а низька вологість може викликати висихання слизових оболонок. Рух повітря впливає на конвекцію та випаровування, а протяги можуть спричинити простудні захворювання.
2. Оптимальні мікрокліматичні умови забезпечують комфорт та підтримують нормальний тепловий стан організму без додаткового навантаження на систему терморегуляції. Припустимі умови можуть викликати дискомфорт, але вони швидко нормалізуються.

Дослідження показали зміни іонного складу повітря в приміщеннях з відеодисплейними терміналами (ВДТ) протягом робочого дня. Концентрація негативних іонів зменшується, а позитивних - зростає, що негативно впливає на здоров'я. Для покращення умов пропонуються такі заходи:

- Використання генераторів негативних іонів;
- Зволожувачі повітря;
- Кондиціонери;
- Системи вентиляції;

- Заземлені захисні екрани.

У приміщеннях, де основна робота ведеться з ЕОМ, слід забезпечити оптимальні умови мікроклімату. Зволожувачі повітря використовуються для підвищення вологості, їх слід заправляти дистильованою або кип'яченою питною водою.

Таблиця 5.1 - Рівні іонізації повітря приміщень при роботі на ВДТ та ПЕОМ [створено автором]

Рівні	Число іонів в 1 см куб. повітря	
	+ n +	- n -
1	2	3
Мінімально необхідні	400	600
Оптимальні	1500 - 3000	300 - 5000
Максимально допустимі	50000	50000

Таблиця 5.2 - Оптимальні і допустимі параметри температури і відносної вологості повітря в приміщеннях з ВДТ і ПЕОМ. [створено автором]

Оптимальні параметри		Допустимі параметри	
Температура, С °	Відносна вологість,%	Температура, С °	Відносна вологість,%
1	2	3	4
19	62	18	39
20	58	22	31

Таблиця 5.3 - Оптимальні норми мікроклімату для приміщень з ВДТ і ПЕОМ [створено автором]

Період року	Категорія робіт	Температура повітря в ° С не більше	Відносна вологість повітря, у %	Швидкість руху повітря, м / с
1	2	3	4	5
Холодний	легка - 1а	22 -24	40 - 60	0,1
	легка - 1б	21 - 23	40 - 60	0,1
Теплий	легка - 1а	23 - 25	40 - 60	0,1
	легка - 1б	22 - 24	40 - 60	0,2

5.2 Техніка безпеки при роботі з ПК

Правила безпеки та пожежної безпеки при роботі з комп'ютерами включають:

- Слідування інструкціям з експлуатації ПК.
- Під час використання ПК необхідно уникати контакту з елементами, що перебувають під напругою, використовувати лише справні електричні прилади.
- Тільки особи, які пройшли відповідний інструктаж з охорони праці, можуть працювати з ПК. Інструктаж та його фіксація у журналі обов'язкові.
- Під час роботи слід дотримуватися вимог безпеки.
- У разі виявлення несправностей у ПК (наприклад, іскріння або запаху горіння), слід негайно припинити роботу, відключити обладнання та повідомити про це відповідних спеціалістів.
- Важливо вміти надавати допомогу при ураженні електричним струмом та діяти у разі пожежі.

- Знати місцезнаходження засобів пожежогасіння та евакуаційні шляхи.

Перед початком роботи з ПК необхідно:

- Перевірити справність обладнання, включно з корпусами, кабелями та заземленням.
- У разі виявлення несправностей не вмикати ПК.

Під час роботи слід увімкнути обладнання, дотримуючись вимог інструкції з експлуатації. Забороняється:

- Виконувати заміну частин обладнання чи проводити перемонтаж при ввімкненому ПК.
- З'єднувати або роз'єднувати електричні кабелі під напругою.
- Виконувати роботи відкритими струмоведучими частинами.
- Використовувати несертифіковане обладнання та інструменти.

По закінченні робочого дня слід:

- Вимкнути ПК, зняти вилку з розетки та прибрати робоче місце.
- Повідомити про будь-які виявлені проблеми в роботі ПК.

Перед залишенням приміщення:

- Закрити вікна, перевірити відсутність потенційно небезпечних предметів, вимкнути всі електроприлади та освітлення, закрити двері на замок.

За порушення правил безпеки та пожежної безпеки передбачена відповідальність. Контроль за дотриманням цих вимог покладається на керівників відділів та спеціалістів з охорони праці.

Висновок

У цій частині дипломної роботи увага була зосереджена на аспектах охорони праці в процесі розробки інформаційної системи. Проведено детальне вивчення концепції "охорони праці", включно з її основними характеристиками та визначенням. Також було проаналізовано потенційні ризики, пов'язані з роботою

на комп'ютері, включаючи вплив різних шкідливих та небезпечних факторів. Визначено оптимальні параметри робочого середовища, такі як температура та освітлення, що сприяють безпеці та комфорту розробників. Розроблено рекомендації щодо мінімізації ризиків, пов'язаних з тривалою роботою за комп'ютером, та встановлено правила безпечної роботи з ПК.

6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

6.1 Правові основи охорони середовища

Право відіграє ключову роль у регулюванні взаємовідносин між суспільством та природою, встановлюючи науково обґрунтовані норми поведінки людини щодо природного середовища. Ці норми, закріплені в законодавстві, стають обов'язковими для виконання всіма громадянами і підтримуються державним примусом у разі їх порушення.

Екологічні проблеми тісно пов'язані з політичними, соціальними та економічними аспектами життя суспільства, тому стратегія використання природних ресурсів в Україні повинна бути інтегрованою частиною ширшої стратегії розвитку правової, демократичної держави з розвиненою ринковою економікою. Основне право громадян на екологічну безпеку забезпечується через комплекс юридичних, економічних, технологічних і гуманітарних заходів.

Законодавство України надає громадянам право звертатися до суду за відшкодуванням збитків, спричинених впливом на навколишнє середовище, та зобов'язує державні органи сприяти громадянам у здійсненні екологічної діяльності.

Громадяни України мають не тільки права, але й обов'язки щодо збереження природи та раціонального використання її ресурсів. Закон встановлює основні принципи охорони природи, включаючи пріоритетність екологічної безпеки, гарантію безпечного середовища для людей, екологізацію виробництва, узгодження екологічних, економічних та соціальних інтересів, збереження біорізноманіття та інтеграцію наукових підходів у вирішення екологічних проблем.

Закон також включає права громадян на безпечне навколишнє середовище, участь у прийнятті рішень, що стосуються екології, доступ до повної і достовірної інформації про стан довкілля, право на судовий захист та відшкодування збитків, спричинених негативним впливом на навколишнє середовище.[2]

Громадяни України мають обов'язки щодо збереження природи, які включають охорону навколишнього середовища, раціональне використання природних ресурсів, діяльність із дотриманням екологічних стандартів, внесення плати за використання природних ресурсів та компенсацію шкоди, завданої забрудненням або іншим негативним впливом на екосистему.

Закон України визначає повноваження Верховної Ради України, місцевих рад та органів управління щодо охорони природи. Спеціально уповноваженим органом у цій сфері є Міністерство охорони навколишнього природного середовища та ядерної безпеки України.

Закон також надає права громадським організаціям участі в екологічних перевірках, проведенні громадської екологічної експертизи, доступу до екологічної інформації, ініціації проведення референдумів з питань охорони довкілля та подання до суду позовів щодо екологічних порушень.

"Закон про охорону навколишнього природного середовища" України встановлює заходи щодо забезпечення екологічної безпеки, вимоги до розміщення та експлуатації об'єктів, що можуть впливати на довкілля, та передбачає заходи проти забруднення різними видами відходів.

Закон визначає зони екологічної катастрофи та підвищеної екологічної небезпеки та встановлює відповідальність за екологічні порушення, включаючи порушення прав на екологічно безпечне середовище, норм екологічної безпеки, вимоги до екологічної експертизи, невжиття заходів щодо попередження екологічних наслідків, порушення умов зберігання та транспортування небезпечних речовин, неподання або надання неповної інформації про стан навколишнього середовища.

Висновок

У висновках до розділу про правові основи охорони довкілля акцентується на значущості правових механізмів у регулюванні взаємодії між природою і суспільством. Право займає центральне місце у створенні науково підтверджених

принципів поведінки людей у відносинах з природою, закріплюючи ці принципи у законодавчих актах, які є обов'язковими до виконання.

Законодавство України визнає право громадян на екологічну безпеку та встановлює комплекс заходів для його забезпечення, враховуючи юридичні, економічні, технічні та гуманітарні аспекти. Це підтверджує усвідомлення значення екологічних проблем в контексті різних аспектів життя суспільства.

Особливу увагу приділяється праву громадян на судовий захист та компенсацію збитків, завданих через негативний вплив на довкілля, що сприяє відповідальному ставленню підприємств та організацій до їх впливу на довкілля.

Закон також закріплює обов'язки громадян України щодо збереження природи, вказуючи на необхідність збалансованого підходу, де кожен учасник суспільства відповідає за екологічний баланс.

Визначені законом принципи охорони навколишнього середовища підкреслюють інтегрований підхід до екологічних питань і включають пріоритетність екологічної безпеки, екологізацію виробництва, прозорість і демократичність у прийнятті рішень, що впливають на довкілля.

Загалом, цей розділ підкреслює значимість правового регулювання в області екології як фундаментального компонента забезпечення сталого розвитку та охорони навколишнього середовища, враховуючи взаємозв'язок між екологічними, економічними та соціальними аспектами.

ЗАГАЛЬНІ ВИСНОВКИ

Проаналізовано та визначено проблемні питання з впровадження засобів моніторингу відходів руйнації, для чогорозглянутокритерії декомпозиції системи, сформовано вимоги та проаналізовано вхідні/вихідні дані.

Реалізувати задачі моніторингу та експертизи відходів руйнації можливо в результаті їх систематизації з розробкою спеціалізованої бази даних, в основу проектування якої покладено ситуаційний підхід з методологією експертного експеременту. Для розробки програми використано інтерпретовану об'єктно-орієнтовану мову програмування Python та наступні бібліотеки: Tkinter, SQLite3, OS, PIL, Webbrowser.

Проаналізовано потоки даних при проведенні моніторингу ситуації при руйнуванні в умовах військових дій наступних матеріалів: скла, металу, пластику, тощо.

Розраховано техніко-економічне обгрунтування системи моніторингу та розроблення програмного продукту. Розглянуто заходи щодо охорони праці.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

9. Гнатуш В. А., Дорошенко В. С. Сучасне застосування металевих сплавів для автомобілебудування. Met. Lit'e Ukr. 2019. Vol. 27 № 10-12 (317-319), Pp. 74-81.

10. Дрозд О. В. Експертиза відходів руйнації на основі ознак структурної деградації матеріалів та перспективи рециклінгу. Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія: Технічні науки. 2023. Том 34 (73). № 4. С. 176-182.

11. Скороход О. М., Горяшин А. С. Використання можливостей мови Python для роботи з великими об'ємами даних. Прикладні інформаційні технології: збірник наукових праць. Вінниця, ДонНУ імені Василя Стуса. 2023.

12. Антонов А.В. Системний аналіз. – М.: Вища школа, 2004. – 454 с.

13. Гайдес М.А., Загальна теорія систем (системи та системний аналіз). - Вінниця: Глобус-прес, 2005. - 201 с.

14. Сурмін Ю.П. Аналітична діяльність: Посібник для аналітика неприбуткової організації. - К.: Центр інновацій та розвитку, 2002. - 96с.

15. Чорней Н. Б. Теорія систем і системний аналіз: Навч. посіб. для студ. вищ. навч. закл. – К.: МАУП, 2005. – 256с.

16. Кузнєцов В.В., Системний Аналіз 2018.

17. Погромська Г. С., - Бази даних: проектування та реалізація 2019 – 183с.

18. Лосєв М.Ю., - Проектування баз даних 2019 – 233с.

19. Steven F. Lott - Modern Python Cookbook: 133 recipes to develop flawless and expressive programs in Python 3.8 2020

20. Tutorials Point - Python Data Access 2019

21. Eric Matthes - Python Crash Course 2019

22. Звіт про прямі збитки інфраструктури від руйнувань внаслідок військової агресії росії проти України за рік від початку повномасштабного вторгнення.
URL:https://kse.ua/wp-content/uploads/2023/03/UKR_Feb23_FINAL_Damages-Report-1.pdf (дата звернення 22.07.2023)

23. Марченко В. М., Гречко В. В., Корогодова О. О. Циркулярний підхід до управління військовими відходами. Матеріали III Міжнародної науково-практичної конф. «Бізнес, інновації, менеджмент: проблеми та перспективи». К.: КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2022. 292 с.
24. Фісуненко П. А., Герасимова О. Л. Напрями зменшення ризиків воєнного екоциду за допомогою рециклінгу будівельних відходів в девелопменті нерухомості. Економіка і суспільство. 2022. 45. URL:https://economyandsociety.in.ua (дата звернення 22.07.2023).
25. Токарчук Д. М. Особливості утворення і поводження з відходами під час воєнних дій: досвід України. Економіка, фінанси, менеджмент: актуальні питання науки і практики. 2022. 2. С. 109-122.
26. Костенко О.Є., Фера О.В., Костенко Є.Я - НАВЧАЛЬНО-МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ПРАКТИЧНИХ ЗАНЯТЬ З КУРСУ «ОХОРОНА ПРАЦІ В ГАЛУЗІ»
27. Федулова Л.І. Інноваційна економіка: Підручник. - Київ: "Либідь", 2006. - 480 с.
28. Чухно А.А., Юхименко П.І., Леоненко П. М. Сучасні економічні теорії: Підручник / За ред. А.А. Чухна. - К : Знання, 2007. - 878 с. - (Вища освіта XXI століття).
29. https://docs.python.org/uk/3/tutorial/index.html
30. https://uk.wikipedia.org/wiki/SQLite
31. https://docs.python.org/uk/3/library/tkinter.html
32. https://uk.wikipedia.org/wiki/Python_Imaging_Library

Додаток А - код програми

```
import tkinter as tk
import tkinter.ttk as ttk
from tkinter import ttk, messagebox, filedialog
import tkinter.font as tkFont
import sqlite3
import os
from PIL import Image
import webbrowser

class Main(tk.Frame):
    def __init__(self, root, db):
        super().__init__(root)
        self.root = root
        self.db = db
        self.tree = ttk.Treeview(self, columns=('ID', 'History',
'ChemicalComposition', 'Benchmark', 'ResearchResults'), height=50,
show='headings')
        self.view_records()
        self.root.geometry("1200x600")
        self.db_glass = DB('materials.db', 'Glass')
        self.db_metal = DB('materials.db', 'Metal')
        self.db_other = DB('materials.db', 'Other')
        self.db_plastic = DB('materials.db', 'Plastic')
        self.current_db = self.db_glass
        self.init_main()

    def init_main(self):
        toolbar = tk.Frame(bg='#d7d8e0', bd=2)
        toolbar.pack(side=tk.TOP, fill=tk.X)

        current_directory = os.path.dirname(os.path.abspath(__file__))

        self.glass_img =
tk.PhotoImage(file=os.path.join(current_directory, 'glass.png'))
        btn_glass = tk.Button(toolbar, text='Скло', command=lambda:
self.set_table('Glass'), image=self.glass_img, compound=tk.TOP)
        btn_glass.pack(side=tk.LEFT)

        self.metal_img =
tk.PhotoImage(file=os.path.join(current_directory, 'metal.png'))
        btn_metal = tk.Button(toolbar, text='Метал', command=lambda:
self.set_table('Metal'), image=self.metal_img, compound=tk.TOP)
        btn_metal.pack(side=tk.LEFT)

        self.other_img =
tk.PhotoImage(file=os.path.join(current_directory, 'other.png'))
        btn_other = tk.Button(toolbar, text='Інше', command=lambda:
self.set_table('Other'), image=self.other_img, compound=tk.TOP)
        btn_other.pack(side=tk.LEFT)

        self.plastic_img =
tk.PhotoImage(file=os.path.join(current_directory, 'plastic.png'))
        btn_plastic = tk.Button(toolbar, text='Пластик', command=lambda:
self.set_table('Plastic'), image=self.plastic_img, compound=tk.TOP)
        btn_plastic.pack(side=tk.LEFT)

        self.add_img = tk.PhotoImage(file=os.path.join(current_directory,
'add.png'))
```

```
        btn_open_dialog = tk.Button(toolbar, text='Додати позицію',
command=self.open_dialog, bg='#d7d8e0', bd=0,
                                    compound=tk.TOP, image=self.add_img)
        btn_open_dialog.pack(side=tk.LEFT)

        self.update_img =
tk.PhotoImage(file=os.path.join(current_directory, 'update.png'))
        btn_edit_dialog = tk.Button(toolbar, text='Редагувати',
bg='#d7d8e0', bd=0, image=self.update_img,
                                    compound=tk.TOP,
command=self.open_update_dialog)
        btn_edit_dialog.pack(side=tk.LEFT)

        self.delete_img =
tk.PhotoImage(file=os.path.join(current_directory, 'delete.png'))
        btn_delete = tk.Button(toolbar, text='Видалити позицію',
bg='#d7d8e0', bd=0, image=self.delete_img,
                               compound=tk.TOP,
command=self.delete_records)
        btn_delete.pack(side=tk.LEFT)

        self.search_img =
tk.PhotoImage(file=os.path.join(current_directory, 'search.png'))
        btn_search = tk.Button(toolbar, text='Пошук по ID', bg='#d7d8e0',
bd=0, image=self.search_img,
                               compound=tk.TOP,
command=self.open_search_dialog)
        btn_search.pack(side=tk.LEFT)

        self.refresh_img =
tk.PhotoImage(file=os.path.join(current_directory, 'refresh.png'))
        btn_refresh = tk.Button(toolbar, text='Оновити', bg='#d7d8e0',
bd=0, image=self.refresh_img,
                                compound=tk.TOP,
command=self.view_records)
        btn_refresh.pack(side=tk.LEFT)

        self.pin_img = tk.PhotoImage(file=os.path.join(current_directory,
'pin.png'))
        btn_open_pin_dialog = tk.Button(toolbar, text='Прикріпити фото',
command=self.open_pin_dialog, bg='#d7d8e0', bd=0,
                                    compound=tk.TOP, image=self.pin_img)
        btn_open_pin_dialog.pack(side=tk.LEFT)

        self.pdf_img = tk.PhotoImage(file=os.path.join(current_directory,
'pdf.png'))
        btn_open_pdf_dialog = tk.Button(toolbar, text='Прикріпити PDF',
command=self.open_pdf_dialog, bg='#d7d8e0', bd=0,
                                    compound=tk.TOP, image=self.pdf_img)
        btn_open_pdf_dialog.pack(side=tk.LEFT)

        self.view_attachment_img =
tk.PhotoImage(file=os.path.join(current_directory, 'view.png'))
        btn_view_attachment = tk.Button(toolbar, text='Перегляд вкладень',
command=self.view_attachments,
                                        bg='#d7d8e0', bd=0,
compound=tk.TOP, image=self.view_attachment_img)
        btn_view_attachment.pack(side=tk.LEFT)

        self.exit_img = tk.PhotoImage(file=os.path.join(current_directory,
'exit.png'))
```

```
        btn_exit = tk.Button(toolbar, text='Вихід', bg='#d7d8e0', bd=0,
image=self.exit_img,
                             compound=tk.TOP, command=self.exit_app)
        btn_exit.pack(side=tk.LEFT)

        self.tree = ttk.Treeview(self, columns=('ID', 'History',
'ChemicalComposition', 'Benchmark', 'ResearchResults'), height=50,
show='headings')
        self.tree.pack(side=tk.TOP, fill=tk.BOTH, expand=True)

        self.tree.column('ID', width=25, anchor=tk.CENTER)
        self.tree.column('History', width=265, anchor=tk.CENTER)
        self.tree.column('ChemicalComposition', width=150,
anchor=tk.CENTER)
        self.tree.column('Benchmark', width=100, anchor=tk.CENTER)
        self.tree.column('ResearchResults', width=200, anchor=tk.CENTER)

        self.tree.heading('ID', text='ID')
        self.tree.heading('History', text='Історія')
        self.tree.heading('ChemicalComposition', text='Хімічний склад')
        self.tree.heading('Benchmark', text='Еталон')
        self.tree.heading('ResearchResults', text='Результати досліджень')

        self.tree.pack(side=tk.LEFT)

        scroll = tk.Scrollbar(self, command=self.tree.yview)
        scroll.pack(side=tk.LEFT, fill=tk.Y)
        self.tree.configure(yscrollcommand=scroll.set)

    def set_table(self, table_name):
        self.db.set_table_name(table_name)
        self.current_db = self.db
        self.view_records()

    def view_records(self):
        for item in self.tree.get_children():
            self.tree.delete(item)

        data = self.db.get_all_records()

        columns = self.tree["columns"]
        if isinstance(columns, (list, tuple)):
            for col in columns:
                self.tree.column(col, width=0)

            data = self.db.get_all_records()

            for row in data:
                self.tree.insert('', 'end', values=row)

            for col in columns:
                max_width = max(
                    tkFont.Font().measure(col),
                    *[tkFont.Font().measure(str(cell_value)) for
cell_value in [row[columns.index(col)] for row in data]]
                )

                self.tree.column(col, width=max_width + 20)
        else:
            print("Помилка: Невірний тип даних для columns.")
```

```
    def records(self, History, ChemicalComposition, Benchmark,
ResearchResults, HistoryImage=None, ChemicalProtocol=None,
EthalonPhoto=None, PhotoResearch=None, PDFResearch=None):
        self.db.insert_data(History, ChemicalComposition, Benchmark,
ResearchResults, HistoryImage, ChemicalProtocol, EthalonPhoto,
PhotoResearch, PDFResearch)
        self.view_records()

    def update_record(self, History, ChemicalComposition, Benchmark,
ResearchResults):
        self.db.c.execute('''UPDATE {table_name} SET History=?,
ChemicalComposition=?, Benchmark=?, ResearchResults=?, WHERE ID=?''',
                          (History, ChemicalComposition, Benchmark,
ResearchResults, self.tree.set(self.tree.selection()[0], '#1')))
        self.db.conn.commit()
        self.view_records()

        self.tree = ttk.Treeview(self)
        self.tree["columns"] = ("ID", "History", "ChemicalComposition",
"Benchmark", "ResearchResults")

        self.tree.pack(fill=tk.BOTH, expand=1)


    def delete_records(self):
        if self.current_db:
            for selection_item in self.tree.selection():
                query = f"DELETE FROM {self.current_db.table_name} WHERE
id=?"
                self.current_db.c.execute(query,
(self.tree.set(selection_item, '#1'),))
            self.current_db.conn.commit()
            self.view_records()

    def search_records(self, History):
        History = ('%' + History + '%',)
        self.db.c.execute('''SELECT * FROM {table_name} WHERE History LIKE
?''', History)
        [self.tree.delete(i) for i in self.tree.get_children()]
        [self.tree.insert('', 'end', values=row) for row in
self.db.c.fetchall()]

    def exit_app(self):
        self.root.destroy()

    def open_dialog(self):
        Child(self.root, self.current_db)

    def open_update_dialog(self):
        Update(self.root, self.current_db)

    def open_search_dialog(self):
        Search(self.root, self.current_db)

    def open_pin_dialog(self):
        PinPhoto(self.root, self.current_db)

    def open_pdf_dialog(self):
        PinPDF(self.root, self.current_db)

    def view_attachments(self):
```

```
        View(self.root, self.current_db)


class Child(tk.Toplevel):
    def __init__(self, root, current_db, show_fields=True):
        super().__init__(root)
        self.init_child(current_db, show_fields)
        self.view = app
        self.current_db = current_db

    def init_child(self, current_db, show_fields=True):
        self.title('Додати запис')
        self.geometry('400x350+400+300')
        self.resizable(False, False)

        if show_fields:
            label_History = tk.Label(self, text='Історія:')
            label_History.place(x=50, y=50)
            label_Chemical = tk.Label(self, text='Хімічний склад')
            label_Chemical.place(x=50, y=80)
            label_Benchmark = tk.Label(self, text='Еталон:')
            label_Benchmark.place(x=50, y=110)
            label_Research = tk.Label(self, text='Результат дослідження')
            label_Research.place(x=50, y=140)

            self.entry_History = ttk.Entry(self)
            self.entry_History.place(x=200, y=50)

            self.entry_ChemicalComposition = ttk.Entry(self)
            self.entry_ChemicalComposition.place(x=200, y=80)

            self.entry_Benchmark = ttk.Entry(self)
            self.entry_Benchmark.place(x=200, y=110)

            self.entry_ResearchResults = ttk.Entry(self)
            self.entry_ResearchResults.place(x=200, y=140)

        btn_cancel = ttk.Button(self, text='Закрити',
command=self.destroy)
        btn_cancel.place(x=300, y=280)

        self.btn_ok = ttk.Button(self, text='Додати')
        self.btn_ok.place(x=220, y=280)
        self.btn_ok.bind('<Button-1>', lambda event: self.add_record())

        self.grab_set()
        self.focus_set()

    def add_record(self):
        if self.current_db:
            history_value = self.entry_History.get()
            benchmark_value = self.entry_ChemicalComposition.get()
            chemical_value = self.entry_Benchmark.get()
            research_value = self.entry_ResearchResults.get()


            query = f"INSERT INTO {self.current_db.table_name} (History,
ChemicalComposition, Benchmark, ResearchResults) VALUES (?, ?, ?, ?)"
            self.current_db.c.execute(query, (history_value,
chemical_value, benchmark_value, research_value))
            self.current_db.conn.commit()
```

```
            self.view.view_records()
            self.destroy()
        else:
            messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
таблицю перед використанням цієї функції.")

class View(Child):
    def __init__(self, root, current_db):
        super().__init__(root, current_db, show_fields=False)
        self.init_View()
        self.view = app
        self.db = db
        self.current_db = current_db
        self.geometry('430x330')

    def set_table_name(self, table_name):
        if self.current_db:
            self.current_db.set_table_name(table_name)

    def init_View(self):
        self.title('Переглянути прикріплені файли')

        show_image_button = tk.Button(self, text='Показати зображення
прикріплене до Історії', command=self.show_image)
        show_image_button.place(x=30, y=30)

        show_benchmark_image_button = tk.Button(self, text='Показати
зображення прикріплене до Еталону', command=self.show_benchmark_image)
        show_benchmark_image_button.place(x=30, y=70)

        show_research_results_image_button = tk.Button(self,
text='Показати зображення прикріплене до Результатів Дослідження',
command=self.show_research_results_image)
        show_research_results_image_button.place(x=30, y=110)

        show_pdf = tk.Button(self, text='Переглянути протокол прикріплений
до Хімічного складу', command=self.show_attach_pdf)
        show_pdf.place(x=30, y=170)

        show_pdf1 = tk.Button(self, text='Переглянути протокол
прикріплений до Результату досліджень', command=self.show_attach_pdf1)
        show_pdf1.place(x=30, y=210)

        if hasattr(self, 'btn_ok'):
            self.btn_ok.destroy()

    def show_image(self):
        if self.current_db:
            selected_item = self.view.tree.selection()
            if selected_item:
                row_id = self.view.tree.set(selected_item[0], '#1')
                conn = sqlite3.connect('materials.db')
                cursor = conn.cursor()
                cursor.execute(f'SELECT HistoryImage FROM
{self.current_db.table_name} WHERE ID = ?', (row_id,))
                result = cursor.fetchone()

                if result is not None:
                    history_image_path = result[0]
                    conn.close()
```

```
                    if history_image_path and
os.path.exists(history_image_path):
                        image = Image.open(history_image_path)
                        image.show()
                    else:
                        messagebox.showinfo("Повідомлення", "Фото не
прикріплено до даного запису.")
                else:
                    conn.close()
                    messagebox.showinfo("Повідомлення", "Не вдалось знайти
запис с указанным ID.")
            else:
                messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
запис для перегляду фото.")
        else:
            messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
таблицю перед використанням цієї функції.")

    def show_benchmark_image(self):
        if self.current_db:
            selected_item = self.view.tree.selection()
            if selected_item:
                row_id = self.view.tree.set(selected_item[0], '#1')
                conn = sqlite3.connect('materials.db')
                cursor = conn.cursor()

                try:
                    cursor.execute(f'SELECT EthalonPhoto FROM
{self.current_db.table_name} WHERE ID = ?', (row_id,))
                    ethalon_image_path = cursor.fetchone()[0]
                    conn.close()

                    if ethalon_image_path and
os.path.exists(ethalon_image_path):
                        image = Image.open(ethalon_image_path)
                        image.show()
                    else:
                        messagebox.showinfo("Повідомлення", "Фото не
прикріплено до даного запису.")
                except (sqlite3.Error, IndexError) as e:
                    conn.close()
                    messagebox.showinfo("Повідомлення", f"Помилка при
відображенні фото: {str(e)}")
            else:
                messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
запис для перегляду фото.")
        else:
            messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
таблицю перед використанням цієї функції.")

    def show_research_results_image(self):
        if self.current_db:
            selected_item = self.view.tree.selection()
            if selected_item:
                row_id = self.view.tree.set(selected_item[0], '#1')
                conn = sqlite3.connect('materials.db')
                cursor = conn.cursor()

                try:
```

```
                    cursor.execute(f'SELECT PhotoResearch FROM
{self.current_db.table_name} WHERE ID = ?', (row_id,))
                    research_image_path = cursor.fetchone()[0]
                    conn.close()

                    if research_image_path and
os.path.exists(research_image_path):
                        image = Image.open(research_image_path)
                        image.show()
                    else:
                        messagebox.showinfo("Повідомлення", "Фото не
прикріплено до даного запису.")
                except (sqlite3.Error, IndexError) as e:
                    conn.close()
                    messagebox.showinfo("Повідомлення", f"Помилка при
відображенні фото: {str(e)}")
            else:
                messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
запис для перегляду фото.")
        else:
            messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
таблицю перед використанням цієї функції.")


    def show_attach_pdf(self):
        if self.current_db:
            selected_item = self.view.tree.selection()
            if selected_item:
                row_id = self.view.tree.set(selected_item[0], '#1')
                conn = sqlite3.connect('materials.db')
                cursor = conn.cursor()

                try:
                    cursor.execute(f'SELECT ChemicalProtocol FROM
{self.current_db.table_name} WHERE ID = ?', (row_id,))
                    pdf_path_chemical = cursor.fetchone()[0]
                    conn.close()

                    if pdf_path_chemical and
os.path.exists(pdf_path_chemical):
                        webbrowser.open(pdf_path_chemical)
                    else:
                        messagebox.showinfo("Повідомлення", "PDF не
прикріплено до даного запису.")
                except (sqlite3.Error, IndexError) as e:
                    conn.close()
                    messagebox.showinfo("Повідомлення", f"Помилка при
відображенні PDF: {str(e)}")
            else:
                messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
запис для перегляду PDF.")
        else:
            messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
таблицю перед використанням цієї функції.")

    def show_attach_pdf1(self):
        if self.current_db:
            selected_item = self.view.tree.selection()
            if selected_item:
                row_id = self.view.tree.set(selected_item[0], '#1')
                conn = sqlite3.connect('materials.db')
```

```
                cursor = conn.cursor()

                try:
                    cursor.execute(f'SELECT PDFResearch FROM
{self.current_db.table_name} WHERE ID = ?', (row_id,))
                    pdf_path_results = cursor.fetchone()[0]
                    conn.close()

                    if pdf_path_results and
os.path.exists(pdf_path_results):
                        webbrowser.open(pdf_path_results)
                    else:
                        messagebox.showinfo("Повідомлення", "PDF не
прикріплено до даного запису.")
                except (sqlite3.Error, IndexError) as e:
                    conn.close()
                    messagebox.showinfo("Повідомлення", f"Помилка при
відображенні PDF: {str(e)}")
            else:
                messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
запис для перегляду PDF.")
        else:
            messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
таблицю перед використанням цієї функції.")

class PinPhoto(Child):
    def __init__(self, root, current_db):
        super().__init__(root, current_db, show_fields=False)
        self.init_pin()
        self.view = app
        self.db = db
        self.current_db = current_db
        self.geometry('400x330')

    def set_table_name(self, table_name):
        if self.current_db:
            self.current_db.set_table_name(table_name)

    def init_pin(self):
        self.title('Прикріпити фото')

        btn_pin = ttk.Button(self, text='Прикріпити фото до Історії',
command=self.attach_photo)
        btn_pin.place(x=30, y=50)

        self.photo_path_history = tk.StringVar()
        entry_photo = ttk.Entry(self,
textvariable=self.photo_path_history, state='readonly')
        entry_photo.place(x=30, y=20, width=150)

        btn_browse = ttk.Button(self, text='Обрати фото',
command=self.browse_photo)
        btn_browse.place(x=30, y=80)

        btn_pin1 = ttk.Button(self, text='Прикріпити фото до Еталону',
command=self.attach_photo1)
        btn_pin1.place(x=30, y=150)

        self.photo_path_ethalon = tk.StringVar()
        entry_photo1 = ttk.Entry(self,
textvariable=self.photo_path_ethalon, state='readonly')
```

```
        entry_photo1.place(x=30, y=120, width=150)

        btn_browse1 = ttk.Button(self, text='Обрати фото',
command=self.browse_photo1)
        btn_browse1.place(x=30, y=180)

        btn_pin2 = ttk.Button(self, text='Прикріпити фото до Результатів
дослідження', command=self.attach_photo2)
        btn_pin2.place(x=30, y=250)

        self.photo_path_research = tk.StringVar()
        entry_photo2 = ttk.Entry(self,
textvariable=self.photo_path_research, state='readonly')
        entry_photo2.place(x=30, y=220, width=150)

        btn_browse2 = ttk.Button(self, text='Обрати фото',
command=self.browse_photo2)
        btn_browse2.place(x=30, y=280)

        if hasattr(self, 'btn_ok'):
            self.btn_ok.destroy()

    def attach_photo(self):
        if self.current_db:
            history_image_path = self.photo_path_history.get()
            if history_image_path:
                selected_item = self.view.tree.selection()
                if selected_item:
                    row_id = self.view.tree.set(selected_item[0], '#1')
                    try:
                        self.current_db.c.execute(f'UPDATE
{self.current_db.table_name} SET HistoryImage=? WHERE ID=?',
                                                  (history_image_path,
row_id))
                        self.current_db.conn.commit()
                        self.view.view_records()
                        self.destroy()
                    except sqlite3.Error as e:
                        messagebox.showinfo("Повідомлення", f"Помилка при
відображенні фото: {str(e)}")
                else:
                    messagebox.showinfo("Повідомлення", "Будь ласка,
виберіть запис для вставки фото.")
        else:
            messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
таблицю перед використанням цієї функції.")

    def browse_photo(self):
        file_path = filedialog.askopenfilename(filetypes=[("Image files",
"*.png;*.jpg;*.jpeg")])
        self.photo_path_history.set(file_path)

    def attach_photo1(self):
        if self.current_db:
            ethalon_image_path = self.photo_path_ethalon.get()
            if ethalon_image_path:
                selected_item = self.view.tree.selection()
                if selected_item:
                    row_id = self.view.tree.set(selected_item[0], '#1')
                    try:
```

```
                        self.current_db.c.execute(f'UPDATE
{self.current_db.table_name} SET EthalonPhoto=? WHERE ID=?',
                                                  (ethalon_image_path,
row_id))
                        self.current_db.conn.commit()
                        self.view.view_records()
                        self.destroy()
                    except sqlite3.Error as e:
                        messagebox.showinfo("Повідомлення", f"Помилка при
відображенні фото еталону: {str(e)}")
                else:
                    messagebox.showinfo("Повідомлення", "Будь ласка,
виберіть запис для вставки фото еталону.")
        else:
            messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
таблицю перед використанням цієї функції.")

    def browse_photo1(self):
        file_path = filedialog.askopenfilename(filetypes=[("Image files",
"*.png;*.jpg;*.jpeg")])
        self.photo_path_ethalon.set(file_path)

    def attach_photo2(self):
        if self.current_db:
            research_image_path = self.photo_path_research.get()
            if research_image_path:
                selected_item = self.view.tree.selection()
                if selected_item:
                    row_id = self.view.tree.set(selected_item[0], '#1')
                    try:
                        self.current_db.c.execute(f'UPDATE
{self.current_db.table_name} SET PhotoResearch=? WHERE ID=?',
                                                  (research_image_path,
row_id))
                        self.current_db.conn.commit()
                        self.view.view_records()
                        self.destroy()
                    except sqlite3.Error as e:
                        messagebox.showinfo("Повідомлення", f"Помилка при
відображенні фото дослідження: {str(e)}")
                else:
                    messagebox.showinfo("Повідомлення", "Будь ласка,
виберіть запис для вставки фото дослідження.")
        else:
            messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
таблицю перед використанням цієї функції.")

    def browse_photo2(self):
        file_path = filedialog.askopenfilename(filetypes=[("Image files",
"*.png;*.jpg;*.jpeg")])
        self.photo_path_research.set(file_path)


class PinPDF(Child):
    def __init__(self, root, current_db):
        super().__init__(root, current_db, show_fields=False)
        self.init_pdf()
        self.view = app
        self.db = db
        self.current_db = current_db
        self.geometry('400x330')
```

```
    def set_table_name(self, table_name):
        if self.current_db:
            self.current_db.set_table_name(table_name)

    def init_pdf(self):
        self.title('Прикріпити PDF')

        btn_pdf = tk.Button(self, text='Прикріпити протокол до Хімічного
складу', command=self.attach_pdf)
        btn_pdf.place(x=30, y=50)

        self.pdf_path_chemical = tk.StringVar()
        entry_pdf = tk.Entry(self, textvariable=self.pdf_path_chemical,
state='readonly')
        entry_pdf.place(x=30, y=20, width=150)

        btn_browse_pdf = tk.Button(self, text='Обрати PDF',
command=self.browse_pdf)
        btn_browse_pdf.place(x=30, y=80)

        btn_pdf1 = tk.Button(self, text='Прикріпити протокол до Результату
досліджень', command=self.attach_pdf1)
        btn_pdf1.place(x=30, y=150)

        self.pdf_path_results = tk.StringVar()
        entry_pdf1 = tk.Entry(self, textvariable=self.pdf_path_results,
state='readonly')
        entry_pdf1.place(x=30, y=120, width=150)

        btn_browse_pdf1 = tk.Button(self, text='Обрати PDF',
command=self.browse_pdf1)
        btn_browse_pdf1.place(x=30, y=180)

        if hasattr(self, 'btn_ok'):
            self.btn_ok.destroy()

    def attach_pdf(self):
        if self.current_db:
            chemical_pdf_path = self.pdf_path_chemical.get()
            if chemical_pdf_path:
                selected_item = self.view.tree.selection()
                if selected_item:
                    row_id = self.view.tree.set(selected_item[0], '#1')
                    try:
                        self.current_db.c.execute(f'UPDATE
{self.current_db.table_name} SET ChemicalProtocol=? WHERE ID=?',
                                                  (chemical_pdf_path,
row_id))
                        self.current_db.conn.commit()
                        self.view.view_records()
                        self.destroy()
                    except sqlite3.Error as e:
                        messagebox.showinfo("Повідомлення", f"Помилка при
вставці PDF для хімічного протоколу: {str(e)}")
                else:
                    messagebox.showinfo("Повідомлення", "Будь ласка,
виберіть запис для вставки PDF.")
        else:
            messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
таблицю перед використанням цієї функції.")
```

```
    def browse_pdf(self):
        file_path = filedialog.askopenfilename(filetypes=[("PDF files",
"*.pdf")])
        self.pdf_path_chemical.set(file_path)

    def attach_pdf1(self):
        if self.current_db:
            results_pdf_path = self.pdf_path_results.get()
            if results_pdf_path:
                selected_item = self.view.tree.selection()
                if selected_item:
                    row_id = self.view.tree.set(selected_item[0], '#1')
                    try:
                        self.current_db.c.execute(f'UPDATE
{self.current_db.table_name} SET PDFResearch=? WHERE ID=?',
                                                  (results_pdf_path,
row_id))
                        self.current_db.conn.commit()
                        self.view.view_records()
                        self.destroy()
                    except sqlite3.Error as e:
                        messagebox.showinfo("Повідомлення", f"Помилка при
вставці PDF для дослідницьких результатів: {str(e)}")
                else:
                    messagebox.showinfo("Повідомлення", "Будь ласка,
виберіть запис для вставки PDF.")
        else:
            messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
таблицю перед використанням цієї функції.")

    def browse_pdf1(self):
        file_path = filedialog.askopenfilename(filetypes=[("PDF files",
"*.pdf")])
        self.pdf_path_results.set(file_path)

class Update(Child):
    def __init__(self, root, current_db):
        super().__init__(root, current_db)
        self.init_edit()
        self.view = app
        self.db = db
        self.current_db = current_db
        self.default_data()

    def set_table_name(self, table_name):
        if self.current_db:
            self.current_db.set_table_name(table_name)

    def init_edit(self):
        self.title('Редагувати позицію')
        btn_edit = ttk.Button(self, text='Редагувати')
        btn_edit.place(x=205, y=170)
        btn_edit.bind('<Button-1>', lambda event: self.update_record())

        self.btn_ok.destroy()

    def default_data(self):
        if self.current_db:
            row_id = self.view.tree.set(self.view.tree.selection()[0],
'#1')
```

```
            self.current_db.c.execute(f'SELECT * FROM
{self.current_db.table_name} WHERE id=?', (row_id,))
            row = self.current_db.c.fetchone()
            self.entry_History.insert(0, row[1])
            self.entry_ChemicalComposition.insert(0, row[2])
            self.entry_Benchmark.insert(0, row[3])
            self.entry_ResearchResults.insert(0, row[4])

    def update_record(self):
        if self.current_db:
            row_id = self.view.tree.set(self.view.tree.selection()[0],
'#1')
            self.current_db.c.execute(f'''UPDATE
{self.current_db.table_name}
                                          SET History=?, Benchmark=?,
ChemicalComposition=?, ResearchResults=?
                                          WHERE ID=?''',
                                         (self.entry_History.get(),
self.entry_ChemicalComposition.get(),
                                          self.entry_Benchmark.get(),
self.entry_ResearchResults.get(), row_id))
            self.current_db.conn.commit()
            self.view.view_records()
            self.destroy()
        else:
            messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
таблицю перед використанням цієї функції.")

class Search(tk.Toplevel):
    def __init__(self, root, current_db):
        super().__init__(root)
        self.init_search()
        self.view = app
        self.current_db = current_db

    def init_search(self):
        self.title('Пошук по ID')
        self.geometry('300x100+400+300')
        self.resizable(False, False)

        label_search = tk.Label(self, text='Пошук')
        label_search.place(x=50, y=20)

        self.entry_search = ttk.Entry(self)
        self.entry_search.place(x=105, y=20, width=150)

        btn_cancel = ttk.Button(self, text='Закрыть',
command=self.destroy)
        btn_cancel.place(x=185, y=50)

        btn_search = ttk.Button(self, text='Пошук')
        btn_search.place(x=105, y=50)
        btn_search.bind('<Button-1>', lambda event:
self.search_records_by_id())
        btn_search.bind('<Button-1>', lambda event: self.destroy(),
add='+')

    def search_records_by_id(self):
        search_id = self.entry_search.get()

        if self.current_db and search_id:
```

```
            try:
                search_id = int(search_id)
            except ValueError:
                messagebox.showinfo("Повідомлення", "ID повинен бути
числовим значенням.")
                return

            query = f"SELECT * FROM {self.current_db.table_name} WHERE ID
= ?"
            result = self.current_db.c.execute(query,
(search_id,)).fetchone()

            if result:
                for item in self.view.tree.get_children():
                    self.view.tree.delete(item)

                self.view.tree.insert("", "end", values=result)
            else:
                messagebox.showinfo("Повідомлення", f"Запись с ID
{search_id} не найдена в таблице {self.current_db.table_name}.")
        else:
            messagebox.showinfo("Повідомлення", "Будь ласка, виберіть
таблицю перед використанням цієї функції.")

class DB:
    def __init__(self, db_name='materials.db', table_name='Glass'):
        self.conn = sqlite3.connect(db_name)
        self.c = self.conn.cursor()
        self.table_name = table_name
        self.create_table()

    def set_table_name(self, table_name):
        self.table_name = table_name
        self.create_table()

    def create_table(self):
        self.c.execute(f'''
            CREATE TABLE IF NOT EXISTS {self.table_name} (
                ID INTEGER PRIMARY KEY AUTOINCREMENT,
                History TEXT,
                ChemicalComposition TEXT,
                Benchmark TEXT,
                ResearchResults TEXT,
                HistoryImage TEXT,
                ChemicalProtocol TEXT,
                EthalonPhoto TEXT,
                PhotoResearch TEXT,
                PDFResearch TEXT
            )
        ''')
        self.conn.commit()

    def insert_data(self, history, chemical_composition, benchmark,
research_results, history_image, chemical_protocol, ethalon_photo,
photo_research, pdf_research):
        self.c.execute(f'''
            INSERT INTO {self.table_name} (History, ChemicalComposition,
Benchmark, ResearchResults, HistoryImage, ChemicalProtocol, EthalonPhoto,
PhotoResearch, PDFResearch)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
```

```
        ''', (history, chemical_composition, benchmark, research_results,
history_image, chemical_protocol, ethalon_photo, photo_research,
pdf_research))
        self.conn.commit()

    def get_all_records(self):
        self.c.execute(f'SELECT * FROM {self.table_name}')
        return self.c.fetchall()

    def select_data(self, table_name):
        query = f'SELECT * FROM {table_name}'
        self.c.execute(query)
        result = self.c.fetchall()
        return result

    def update_data(self, table_name, column_name, new_value, row_id):
        query = f'UPDATE {table_name} SET {column_name}=? WHERE ID=?'
        self.c.execute(query, (new_value, row_id))
        self.conn.commit()

if __name__ == "__main__":
    root = tk.Tk()
    db = DB()
    app = Main(root, db)
    app.pack()
    root.title("Аналіз матеріалів")
    root.geometry("1200x600+100+50")
    root.resizable(False, False)
    root.mainloop()
```

Додаток Б – тези

УДК 666.75:004.41

ІНФОРМАЦІЙНА ПІДТРИМКА ЕКСПЕРТИЗИ ТА ПОВТОРНОГО ВИКОРИСТАННЯ СТАЛЕВИХ ЕЛЕМЕНТІВ ЗРУЙНОВАНИХ АВТО

Дрозд Оксана Володимирівна
кандидат технічних наук, доцент, доцент
Васильєв Олександр Віталійович
студент магістратури
Національний університет кораблебудування імені адмірала Макарова
oksana.drozd @nuos.edu.ua

З початком масштабного вторгнення на прибережних територіях і уздовж доріг, на лінії зіткнення між Херсонською і Миколаївською областями спостерігається скупчення понівечених авто, на яких місцеві жителі і волонтери намагалися виїхати з зони бойових дій та окупованих територій, власники яких вже не зможуть їх прибрати. Тому виникає найактуальніша на даний час проблема щодо їх утилізації, вирішити яку дуже важко через обмежені можливості їх транспортування та продовження бойових дій. Крім того, процеси металопереробки є достатньо енергоємними, вимагають використання потужного обладнання, а тому не вважаються рентабельними.

Аналіз сучасних матеріалів [1], які використовуються у автомобілебудуванні, показав цінність металу зруйнованих авто як сировинної бази. Це твердосплавні сталі *Bake Hardenable (BH)*, трансформаторні пластичні сталі *Transformation Induced Plasticity (TRIP)*, деформівні високоміцні сплави системи Al-Mg-Si-Mn (зокрема AlSi1MgMn), виготовлені за технологіями холодної і гарячої прокатки, екструзії. З цього можна зробити висновок про доцільність використання фрагментів кузова для виготовлення елементів інших конструкцій, що вимагатиме проведення їх експертизи з розробкою технічних заходів щодо їх обробки. В роботі [2] автором запропоновано методику експертизи відходів руйнації, алгоритм виконання якої ґрунтується на етапах виокремлення зі всього різноманіття матеріалів еталонних зразків і зразків-свідків; експериментальних дослідженнях з аналітичним описом їх макро- і мікроструктури; експериментальних випробуваннях на зразках свідках з визначенням кількісного і якісного характеру пошкодження; складанні калібрувальної шкали для кожного виду зразку матеріалу з вибірки. Для цього було сформульовано головні ідентифікаційні ознаки деградації структури у порівнянні з еталонними зразками; розроблено проєкт мобільної хімічної

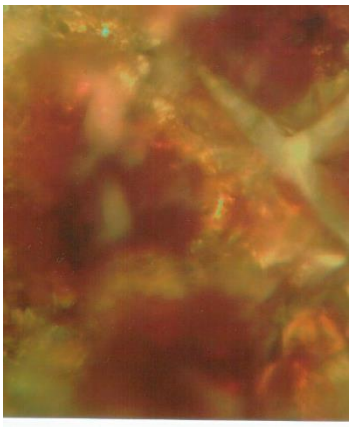
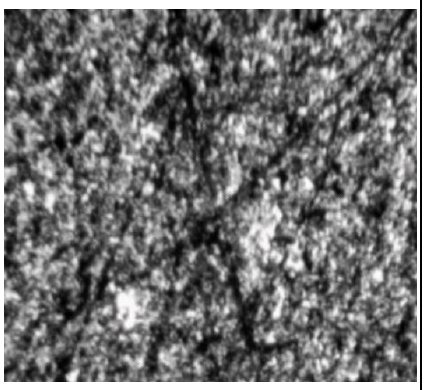
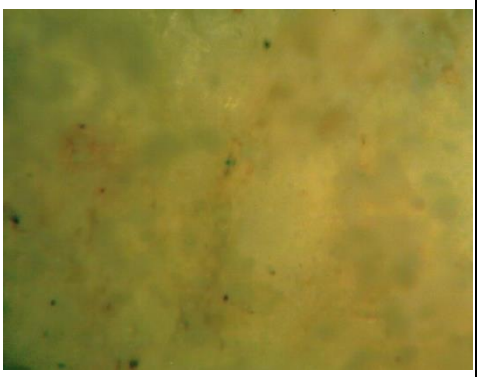
лабораторії. Проте систематизація і обробка інформації вимагатиме розробки професійної інформаційної підтримки з базою даних.

Мета роботи полягає у реалізації задач експертизи сталей пошкоджених через військові дії авто шляхом розробки і впровадження спеціалізованої бази даних.

Науково-практичними передумовами розробки є попередньо проведені результати експериментальних досліджень фрагментів підірваних, пошкоджених, затоплених авто.

Для наочності сприйняття результатів в табл. 1 наведено приклади результатів з оптичними мікрофотографіями.

Таблиця 1. Приклади результатів досліджень зразків-свідків [складено авторами]

Автомобільні сплави * марку сплаву встановлено за літературними джерелами та інформацією щодо марки автомобілю, країни виробника		
TRIP-сталь (після занурення) Зразок (збільшення ×370) вирізано з кузова підірваної машини	BH-сталь (після нагрівання) Зразок (збільшення ×250) вирізано з кузова затопленої машини	AlSi1MgMn (після пожежі та затоплення) Зразок (збільшення ×180) вирізано з кузова затопленої машини
 Піттінгова корозія з розміром виразок до 10 мм	 Розвиток тріщин у місцях пошкодження лакофарбового покриття	 Наявність захисної плівки гальмує вихід продуктів корозії
Висновок: можлива переробка на порошок	*Висновок:* можливо використання для виготовлення елементів конструкцій	*Висновок:* можливо використання у ремонтних технологіях, зокрема у суднoремонті

Структура бази даних документально-фактографічного типу побудована за принципами мінімізації дублювання даних, має віддалений доступ та забезпечує операції введення–виведення при доступі клієнта до інформації. Для написання програмного продукту використано мову програмування Phyton з бібліотекою SQLite [3], функції яких є зручними для обробки і візуалізації даних, що надає можливості проведення інтерактивних досліджень за наступним алгоритмом (рис. 1).

Рисунок 1 – Алгоритм обробки результатів експертизи зруйнованих авто [складено авторами]

Експерт вносить інформацію про «історію» зразку, тобто відомості про марку автомобілю, за яких обставин його вилучено; наводяться результати зважування, вимірювання розмірів; надається світлина яка зберігається у

окремому файлі з розширенням *pdf, *tiff. Фактографічна частина має рядковий запис та заповнюється за результатами вимірювань показників або проведення експериментальних досліджень.

Результати досліджень хімічного складу матимуть вигляд спктрограм, таблиць та оформлюються у протокол, який зберігається у *pdf-форматі.

Паралельно у базу даних заносяться відомості про еталон, який виокремлюється з непошкоджених масивів або належіть незруйнованому авто. Це світлина зразку, відомості про його походження, масу та розміри. У окремих випадках опис може замінювати нормативний документ або наукова стаття у *pdf-форматі.

Результати досліджень містять цифрові документи з оптичними і електронними мікрофотографіями мікроструктур і зламів, звіт у *pdf-форматі.

Всі отримані під час досліджень дані зберігаються у Протоколі, який може бути наданий експертам для подальших висновків. Таким чином, дослідник може отримувати як результати досліджень у режимі реального часу, так і оцінювати ступінь пошкодженості дослідних зразків у порівнянні з еталоном.

Розроблений програмний продукт може працювати під управлінням операційних систем Linux, Windows.

Подальші розробки пов’язані зі створенням та улаштуванням експертної лабораторії, зокрема мобільного типу.

Додатково оброблені елементи авто можуть бути надалі використані для укріплення техніки, формування шаруватих захисних конструкцій, у суднoремонті тощо.

Список використаних джерел

1. Гнатуш В. А., Дорошенко В. С. Сучасне застосування металевих сплавів для автомобілебудування. Met. Lit’e Ukr. 2019. Vol. 27 № 10-12 (317-319), Pp. 74-81.

2. Дрозд О. В. Експертиза відходів руйнації на основі ознак структурної деградації матеріалів та перспективи рециклінгу. Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія: Технічні науки. 2023. Том 34 (73). № 4. С. 176-182.

3. Скороход О. М., Горяшин А. С. Використання можливостей мови Python для роботи з великими об’ємами даних. Прикладні інформаційні технології: збірник наукових праць. Вінниця, ДонНУ імені Василя Стуса. 2023. С. 238-239.