Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи управління електронним документообігом”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-22М-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Теплухін В.С.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Смірнова Т.В.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Теплухіну Віктору Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи управління електронним документообігом*

2. Керівник роботи *Смірнова Тетяна Віталіївна, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 35-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи управління електронним документообігом*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Теплухін В.С. Дослідження та програмна реалізація системи управління електронним документообігом. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи управління електронним документообігом.

Метою розробки є дослідження та програмна реалізація системи управління електронним документообігом.

Об'єктом дослідження є процес управління електронним документообігом.

Предметом дослідження є методи управління електронним документообігом.

Методи дослідження базуються на методах цифровізації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи управління електронним документообігом.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Visual C++.

Ключові слова: комп'ютерна інженерія, електронний документообіг

ABSTRACT

Teplukhin V.S. Research and software implementation of the electronic document management system. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software was developed, which is intended for the electronic document management system.

The purpose of the development is the research and software implementation of the electronic document management system.

The object of the research is the process of electronic document flow management.

The subject of the study is the methods of electronic document flow management.

Research methods are based on digitization methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the electronic document management system.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Visual C++ environment.

Keywords: computer engineering, electronic document management

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП..... 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 7
1.1 Призначення системи..... 7
1.2 Область застосування..... 8
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 9
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 9
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 26
2.3 Розгорнута постановка завдання 28
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 30
3.1 Опис функціонування системи 30
3.2 Розробка структурної схеми..... 33
3.3 Розробка функціональної схеми 41
3.4 Розробка діаграми процесів..... 46
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 48
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 48
4.2 Захист розробленого програмного забезпечення..... 63
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 66
6 НАУКОВА НОВИЗНА 77

7 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ 78
7.1 Техніко економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти 78
7.2 Розрахунок трудомісткості розробки програмної продукції 80
7.3 Визначення чисельності виконавців і планового фонду зарплати 82
7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника 86
7.5 Визначення собівартості розробки та ціни програмної продукції. 91
7.6 Визначення об'єму капітальних вкладень та експлуатаційних витрат у споживача програмної продукції. 94
7.7 Визначення експлуатаційних витрат 94
7.8 Визначення економічної ефективності програмної продукції. 96
7.9 Висновок 98
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 99
8.1 Вступ 99
8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером 100
8.3 Пожежна безпека 101
8.4 Розробка заходів з умов поліпшення охорони праці 103
8.5 Розрахункова частина 104
9 ОСНОВНІ ВИСНОВКИ 109
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 111

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЕА	–	електронний архів
ЕДО	–	електронний документообіг
САДД	–	система автоматизації електронного документообігу й діловодства
СЕА	–	системи електронних архівів
СДЕ	–	системи діловодства електронні
СУЕД	–	система управління електронним документообігом
ТЕ	–	типові елементи
ФФО	–	формат файлових об'єктів
EDMS	–	Electronic Document Management Systems

ВСТУП

Актуальність теми. Незалежно від того, чи йдеться про обробку замовлень на купівлю, рахунків-фактур або даних про співробітників, надзвичайно важливо мати надійну систему для ефективного керування документами. Пам’ятаючи про це, багато організацій вирішують запровадити систему електронного документообігу (EDMS), щоб гарантувати, що всі дані будуть доступні, надійно зберігаються та легко доступні за потреби.

EDMS також надає підприємствам додаткові переваги, включаючи відповідність вимогам і автоматизацію щоденних процесів. Це дозволяє співробітникам зосередитися на більш стратегічних завданнях, включаючи інновації, співпрацю та спілкування.

EDMS – це скорочення від електронної системи управління документами. Простіше кажучи, це програмне забезпечення, яке використовується для управління внутрішніми документами як частина ширшої стратегії управління бізнесом і людським капіталом. У порівнянні з іншими HR-рішеннями, спрямованими на оптимізацію конкретних внутрішніх процесів (HR SaaS-додатки), EDMS слугує для централізації документів організації на всіх рівнях компанії. Сюди входять файли співробітників, внутрішні правила, рахунки-фактури, посібники, контракти, замовлення на купівлю та дані з платформ залучення талантів, серед інших типів внутрішніх документів.

EDMS зазвичай розміщується на внутрішньому сервері компанії або в хмарі. Користувачі можуть зберігати документи на сервері та легко шукати їх пізніше на різних пристроях. Зберігаючи дані в єдиному цифровому сховищі, ви отримуєте легкий доступ до них, спрощену співпрацю, підвищену відповідність і спрощений контроль версій документів. І все це, зрештою, призводить до підвищення ефективності та рентабельності інвестицій.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи управління електронним документообігом.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем управління електронним документообігом.

– Дослідження системи управління електронним документообігом.

– Програмна реалізація системи управління електронним документообігом.

Об'єктом дослідження є процес управління електронним документообігом.

Предметом дослідження є методи управління електронним документообігом.

Методи дослідження базуються на методах цифровізації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод управління електронним документообігом.

– Розроблено вітчизняний продукт управління електронним документообігом, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі управління електронним документообігом.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи управління електронним документообігом, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Тепер, коли ми розглянули значення EDMS, давайте дослідимо деякі з ключових функцій, які можна знайти в системі електронного документообігу:

– Створення та оцифрування документів.

– Масштабоване та централізоване зберігання документів.

– Організація файлів за допомогою систем тегів, пошуку метаданих і папок.

– Автоматичні та/або ручні процеси резервного копіювання та архівування.

– Системи контролю документів.

– Контроль доступу.

– Інструменти для спільної роботи для легкого обміну та редагування в реальному часі.

– Розширені протоколи безпеки.

– Функції для керування життєвими циклами документів і записів.

– Можливості внутрішнього та зовнішнього обміну файлами.

– Пошук на багатьох ресурсах і платформах.

– Інструменти для відстеження історії документів і активності системи.

– Інтеграція інструментів EDMS з іншими платформами, такими як програмне забезпечення для управління витратами, системи відстеження заявників, програмне забезпечення для баз даних співробітників, системи реєстрації співробітників, програмне забезпечення для адаптації персоналу, портали самообслуговування для співробітників, програмне забезпечення для прогнозної аналітики для відділу кадрів і програмні рішення для управління людьми, серед іншого.

1.2 Область застосування

Система електронного документообігу (СЕД) – це програмна система для організації та зберігання різних видів документів. Цей тип системи є більш конкретним типом системи управління документами, більш загальним типом системи зберігання, яка допомагає користувачам упорядковувати та зберігати паперові або цифрові документи. EDMS більш конкретно відноситься до системи програмного забезпечення, яка обробляє цифрові документи, а не паперові документи, хоча в деяких випадках ці системи можуть також обробляти цифрові скановані версії оригінальних паперових документів. Електронний документообіг забезпечує спосіб централізованого зберігання великого обсягу цифрових документів. Багато з цих систем також містять функції для ефективного пошуку документів. Деякі експерти відзначають, що система електронного документообігу має багато спільного з системою управління контентом (CMS). Однією з основних відмінностей є те, що більшість систем CMS передбачають обробку різноманітного веб-вмісту з центрального сайту, тоді як система керування документами часто використовується переважно для архівування. Щоб забезпечити якісну класифікацію цифрових документів, багато систем керування електронними документами покладаються на детальний процес зберігання документів, включаючи певні елементи, які називаються метаданими. Метадані навколо документа забезпечать легкий доступ до ключових деталей, які допоможуть тим, хто шукає в архівах, знайти те, що їм потрібно, за хронологією, темою, ключовими словами чи іншими асоціативними стратегіями. У багатьох випадках конкретна документація для оригінальних протоколів зберігання є основною частиною того, що робить систему керування електронними документами настільки цінною для бізнесу чи організації.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи управління електронним документообігом, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Ведення бізнесу пов’язане з великою кількістю паперів: контракти, інформація про співробітників, рахунки-фактури, фінансові звіти, технологічна документація, протоколи зустрічей. Керування цими документами вручну витрачає дорогоцінні робочі години, знижує продуктивність і ставить під загрозу конфіденційну інформацію. Ось чому програмне забезпечення для керування документами (DMS) є таким вирішальним: воно оптимізує все, від зберігання документів до контролю записів, пошуку інформації та спільного використання файлів.

Додайте автоматизацію документообігу в бізнес-процеси

Дізнайтесь як

Послуги керування документами представлені в різних варіантах, тож є хороші шанси, що ви знайдете той, який відповідатиме вашому типу бізнесу, робочому процесу та бюджету. Щоб скласти список найкращих служб керування документами, я поспілкувався з різними безпаперовими компаніями про їхні улюблені програми, отримавши відгуки від експертів, які насправді використовують ці інструменти щодня. Потім я самостійно дослідив програми, щоб переконатися, що вони вирішують важливу проблему для бізнесу.

При цьому ось системи керування документами, які виділялися.

– Найкраще програмне забезпечення для керування документами для автоматизації та робочих процесів.

– Найкраще програмне забезпечення для керування документами для простоти використання.

– Найкраще програмне забезпечення для керування документами для спільної роботи.

– Найкраще програмне забезпечення для керування документами для інтеграції.

– Найкраще програмне забезпечення для керування документами для безпеки та відповідності.

– Найкраще програмне забезпечення для керування документами для підприємств.

Що таке програмне забезпечення для керування документами?

Програмне забезпечення для керування документами забезпечує центральне сховище для зберігання, керування та доступу до документів в електронному вигляді. З ним ви можете впорядковувати, отримувати та ділитися інформацією, не переглядаючи гору паперів у дюжині картотечних шаф площею дев'ять квадратних футів. Окрім зберігання та спільного використання, служби керування документами також надають інструменти для автоматизації всього життєвого циклу документів, щоб оптимізувати внутрішні процеси та підвищити продуктивність.

Коли я спілкувався з експертами та перевіряв ці служби керування документами, я звузив рекомендації на основі деяких ключових критеріїв:

– **Цільові можливості керування документами.** Головною відмінністю між програмним забезпеченням для хмарного зберігання та службою керування документами є робочий процес. Хоча рішення для хмарних сховищ переважно збирають, зберігають і обмінюються файлами, менеджери документів організовують, керують і направляють документи. Програми, які я тут включив, зосереджені безпосередньо на управлінні документами.

– **Автоматизація робочого процесу:** робочі процеси включають створення документів, маршрутизацію, оновлення, архівування, затвердження та будь-які інші автоматизовані процеси, які покращують продуктивність. Програмне забезпечення Solid DMS повинно мати вбудовані

робочі процеси та автоматичні правила для зберігання, упорядкування та позначення документів.

– **Покращена безпека:** служби керування документами мають вбудовану систему безпеки, яка забезпечує безпеку ваших файлів. Завдяки таким функціям, як контроль доступу та контрольний журнал, ви можете відстежувати дії з кожним файлом і вирішувати, хто має доступ до чого. А оскільки програмне забезпечення DMS зазвичай базується на хмарі, ваші файли захищені від фізичних пошкоджень і втрати.

– **Функції співпраці.** Використовуючи програмне забезпечення для керування документами, ви зможете спільно працювати над документами в реальному часі, відстежувати зміни, контролювати робочі процеси та контролювати версії документів.

– **Можливості збору даних:** програмне забезпечення DMS використовує програмне забезпечення оптичного розпізнавання символів (OCR) для оцифрування паперових документів, тому ви можете пропустити аспект введення даних під час керування документами.

– **Зручність використання:** чудова система керування документами має бути простою у використанні та адаптації навіть для людей, які не знаються на техніці. Адміністратори повинні мати повний контроль і авторизацію над своїми документами, а файли мають бути легкими для моніторингу, відстеження та обміну.

З огляду на все це, ось деякі з найкращих систем керування електронними документами для оцифрування вашого бізнесу.

Найкраще програмне забезпечення для керування документами для автоматизації та робочих процесів

Автоматизація звільняє час співробітників, щоб вони могли зосередитися на найважливішому: обслуговуванні клієнтів і розвитку бізнесу. Ці програми допомагають вам автоматизувати рутинні завдання для керування документами. Перш ніж занурюватися, варто перевірити, чи

може Zapier підключити додатки, якими ви вже користуєтеся, щоб допомогти вам створити власні власні робочі процеси з наявним стеком технологій.

Zapier є лідером у сфері автоматизації без використання коду, інтегрується з понад 6000 програмами таких партнерів, як Google, Salesforce і Microsoft. Створюйте безпечні автоматизовані системи для важливих для бізнесу робочих процесів у технологічному стеку вашої організації. Дізнайтеся більше.

DocuPhase

DocuPhase допомагає компаніям трансформувати свою діяльність за допомогою автоматизованого керування документами та робочими процесами. Ріва Джин Мей Кабурог з юридичної фірми Nadrich and Cohen покладається на можливості збору даних і пошуку документів DocuPhase на базі ШІ. Здатність DocuPhase швидко отримувати важливу інформацію з різноманітних документів і отримувати дані зі старих файлів дозволяє їхній команді оперативно реагувати на потреби клієнтів.

Ціни DocuPhase: Індивідуальні.

M-файли

Якщо ви шукаєте просте, але інтуїтивно зрозуміле програмне забезпечення для керування документами, щоб упорядкувати свою цифрову картотеку, M-Files може підійти саме вам. Джошу Степлінгу з агентства нерухомості Treasure Coast MLS Search подобається, як M-Files автоматично класифікує та категоризує файли, контракти та інформацію про клієнтів. Ця автоматизація зменшує введення вручну та ризик неправильного розміщення – і це дає їхнім агентам доступ до важливих документів без попередньої роботи.

Ціни M-Files: Індивідуальні.

Fluix

Fluix – це програмне забезпечення DMS, яке вибирають компанії з мобільною робочою силою. Його офлайн-режим оптимізує документообіг, оскільки команди на місцях можуть швидко отримувати інформацію у віддалених місцях і синхронізувати її з системою, коли вони мають доступ до

Інтернету. Вони також можуть створювати та отримувати доступ до звітів або даних про клієнтів, забезпечуючи максимальну ефективність у польових умовах. А його доступність як програми для iOS робить його ще більш доступним.

Ціни Fluix: починаючи з 30 доларів США за користувача на місяць (мінімум 10 користувачів) плюс додаткові додаткові компоненти для покращеної автоматизації.

Найкраще програмне забезпечення для керування документами для простоти використання

Зручний менеджер документів економить час і підвищує продуктивність. Ці системи DMS є найкращими за простотою використання.

LogicalDOC

LogicalDOC став одним із небагатьох рішень для керування документами, сумісним із кількома операційними системами (ОС) і пристроями. Якщо ви шукаєте багатофункціональну та інтуїтивно зрозумілу платформу, просту у використанні та доступну з будь-якого місця, ця система керування документами може вам підійти.

Ціни на LogicalDOC: Індивідуальні.

Folderit

Folderit – це електронна система керування файлами, широко відома своєю зручністю для користувача. Платформа, створена для Windows, проста в налаштуванні та містить стандартні навороти, очікувані від будь-якого програмного забезпечення DMS. Він також інтегрується з DocuSign для електронних підписів, має кілька параметрів обміну файлами та пропонує офлайн-доступ.

Ціни Folderit: від 27 доларів США на місяць із щомісячним виставленням рахунків; включає до 5 користувачів.

Найкраще програмне забезпечення для керування документами для спільної роботи

Ефективна співпраця зменшує вузькі місця та покращує робочий процес. Ці програми надають вам необхідні ресурси, щоб зробити більше за менший час.

Microsoft SharePoint

Корпорація Майкрософт давно займається продуктивністю, тому не дивно, що кілька компаній, з якими я спілкувався, згадали SharePoint. Гаррі Джонс Вайт з NBA Blast сказав мені, що найбільше цінує функції співпраці SharePoint, особливо можливість установлювати дозволи на конфіденційні документи під час роботи з товаришами по команді в різних місцях. Тим часом Абдулу Сабуру Хану з PCB Insider подобається функція керування версіями SharePoint через ітераційний характер їхніх проектів і процесів.

SharePoint легко інтегрується з іншими продуктами Microsoft, і ви можете змусити його працювати з тисячами інших програм, підключивши його до Zapier.

Ціни SharePoint: від 6,80 дол. США за користувача на місяць (річне виставлення рахунків).

Revver

Менеджер документів Revver спрощує організацію файлів і керування робочим процесом, як і всі програми в цьому списку, але однією з його видатних функцій є розширена співпраця. Роббі Бенардаут з Nature Roamer покладається на надійну систему контролю версій Revver, щоб віддалено співпрацювати зі своєю командою та керувати документацією, маршрутом і створенням вмісту для планування пригод на природі.

Ціна оборотів: на замовлення

Hightail

Якщо ви творчий бізнес, який готовий перенести процес спільної роботи на безпечне програмне забезпечення для зберігання документів, Hightail може вам підійти. Платформа сяє тим, що фокусується на великих зображеннях і відеофайлах. Крім того, він пропонує безкоштовний обліковий запис і доступні

плани, що робить його ідеальним для малого бізнесу.

Якщо підключити Hightail до Zapier, ви зможете автоматизувати співпрацю та стати ще продуктивнішими. Ось кілька готових робочих процесів, які допоможуть вам почати.

Високі ціни: доступний безкоштовний план; платні плани починаються від 12 доларів США на місяць.

Найкраще програмне забезпечення для керування документами для інтеграції

Уміння працювати з інструментами, з якими ви знайомі, підтримує послідовність і організованість. Ці системи керування документами підключаються до інших програм, які ви використовуєте для повсякденних завдань.

DocuWare

Якщо ви шукаєте програмне забезпечення для керування файлами, яке може працювати у вашій IT-екосистемі, DocuWare пропонує безпечну інтеграцію з понад 500 програмами в CRM, системах управління персоналом, командних порталах тощо. Derrick Hathaway з VEM Medical вважає це найкращим інструментом керування документами, оскільки він оптимізує всю їхню систему продажів завдяки синхронізації з CRM та бухгалтерським програмним забезпеченням. Повна інтеграція зменшує загрозу помилок у їхніх файлах і економить багато часу.

Ціни на DocuWare: Індивідуальні.

Коли ви переходите до цифрового офісу, робочі процеси, співпраця та безпека стають критичними, і функції керування документами Box пропонують усе це. Але особливо яскравим є інтеграція з 1500 програмами. Завдяки безпечній інтеграції Box підтримує співпрацю в масштабах організації, що дає змогу командам працювати якнайкраще. І ви можете підключити Box до тисяч інших програм через Zapier. Ось кілька прикладів робочого процесу, які ви можете використовувати.

PandaDoc, Box

Детальніше

Ціна коробки: починається від 20 доларів США за користувача на місяць (річне виставлення рахунків).

Найкраще програмне забезпечення для керування документами для безпеки та відповідності

Безпека є важливою частиною управління документами. Правильна платформа DMS має служити програмним забезпеченням для контролю документів, забезпечуючи шифрування та керуючи доступністю всіх файлів у системі. Ці інструменти забезпечують надійну безпеку та функції відповідності.

MasterControl

Якщо ви працюєте в регульованій галузі, як-от медична, фармацевтична, харчова та напоїв, MasterControl допоможе вам дотримуватися вимог. Ця система керування документами відповідає 21 CFR Part 11, тому всі документи, над якими ви співпрацюєте, мають контрольний журнал із мітками часу, звітування та функцію електронного підпису відповідно до федеральних норм. Менеджер документів також містить функції безпеки, такі як контроль доступу та керування версіями, щоб сприяти співпраці та гарантувати доступ потрібних людей до правильних документів і їх затвердження.

Ціноутворення MasterControl: індивідуальне

IronVault

Завдяки особливому фокусу на безпеці, EisenVault має надійний алгоритм шифрування для збереження документів. Йогеш Чоудхарі з Finoit покладається на контрольний журнал EisenVault для забезпечення відповідності та безпеки. У той же час його контроль доступу дозволяє його команді обмежувати доступ до конфіденційного коду та контролювати, хто переглядає, редагує та затверджує документи.

Ціни EisenVault: Індивідуальні.

Найкраще програмне забезпечення для керування документами для

підприємств

Більшість систем електронного документообігу в цьому списку розроблено для масштабування вашого бізнесу незалежно від розміру, але ось кілька рішень, розроблених переважно для корпоративних організацій.

Alfresco

Система керування корпоративним контентом (ECM) від Alfresco об'єднує керування документами, співпрацю, штучний інтелект і збір даних для забезпечення продуктивності корпоративних організацій. За допомогою розумних папок інструмент може автоматично групувати схожі файли на основі вмісту, щоб оптимізувати організацію та пошук інформації. Він також містить такі функції, як вбудовані робочі процеси та багаті метадані, щоб автоматично переміщувати документи через попередньо визначені процеси.

Ціни на свіжому повітрі: на замовлення.

FileHold

FileHold пропонує вилучення даних, керування версіями, інструменти адміністрування та керування записами для оптимізації автоматизації документообігу на великих підприємствах. У Deep Cognition John Pennypacker покладається на свої потужні пошукові можливості, щоб швидко отримувати, переглядати та ділитися файлами з клієнтами та колегами, особливо в дорозі. Крім того, платформа містить такі функції, як портал анонімного доступу та посилання на гарячі клавіші, які менеджери проектів можуть використовувати для налаштування своїх облікових записів.

Ціни FileHold: індивідуальні.

OpenKM

Якщо ви віддаєте перевагу спеціальній системі керування файлами, програмне забезпечення OpenKM для керування документами з відкритим вихідним кодом дозволяє створювати власні функції, адаптовані до робочого процесу вашої організації та вимог до IT. DMS забезпечує сумісну платформу, яка підтримує інтеграцію репозиторіїв і додатків у масштабах підприємства.

Ціни OpenKM: доступний безкоштовний план; для платної/професійної версії потрібна індивідуальна ціна.

Хмарне сховище та офісні пакети для управління документами

У моїх дослідженнях і розмовах я виявив, що деякі хмарні рішення для зберігання й офісні пакети також слугують менеджерами документів, тому я пропоную їх тут як альтернативу, якщо вам потрібно щось трохи легше.

Dropbox Business

Незважаючи на те, що вона не така надійна, як інші системи керування файлами, які я перерахував вище, функція керування версіями Dropbox полегшує співпрацю. У Offices.net Тереша Ейрд і її гібридна команда покладаються на історію версій і спільні папки Dropbox, щоб співпрацювати над проектами та легко впорядковувати вміст.

Коли ви платите за Dropbox як службу хмарного зберігання, ви також отримуєте рішення для керування документами, як-от електронний підпис і журнали аудиту. А коли ви підключите Dropbox до Zapier, ви зможете використовувати автоматизацію, щоб оптимізувати робочі процеси з документами. Дізнайтеся більше про те, як автоматизувати Dropbox, або перегляньте деякі з цих прикладів.

Завантажте щойно підписані конверти DocuSign у Dropbox

Gmail, Dropbox

Детальніше

Ціни Dropbox: від 15 доларів США за користувача на місяць (річне виставлення рахунків)

Диск Google

Диск Google намагається запропонувати все, що потрібно вашій команді для ефективного керування документами, без вбудованих автоматизованих робочих процесів. Абхішеку Шаху з Testlify особливо подобається це для співпраці в режимі реального часу: кілька членів команди можуть одночасно працювати над документом, відстежувати зміни та залишати коментарі,

сприяючи динамічному та згуртованому робочому середовищу.

Google Drive має вбудовану інтеграцію з іншими продуктами та службами в Google Workspace, але ви можете налаштувати його на роботу з тисячами інших програм сторонніх розробників за допомогою інтеграції Zapier. Дізнайтеся більше про те, як автоматизувати Google Диск, і перегляньте ці готові робочі процеси, щоб почати.

Надсилайте електронні листи через Gmail, коли нові файли додаються на Диск Google

Gmail, Filter by Zapier, Google Drive

Детальніше

Ціни Google Drive : безкоштовно; платні плани починаються від 12 доларів США за користувача на місяць (річне виставлення рахунків).

OnlyOffice

OnlyOffice пропонує систему керування документами, яка підтримує співпрацю. Кіт Донован із Startup Stumbles цінує те, як він обробляє версії: ви можете побачити, хто вніс зміни в документ, переглянути версію, завантажити її або залишити коментар у вбудованому вікні чату.

Ціни OnlyOffice: хмарна версія доступна від 1 долара США за користувача на місяць (3-річний план) принаймні для 6 користувачів. Корпоративні плани починаються від 2200 доларів США за версію, що розміщується самостійно.

Залишайтеся без паперу завдяки системі електронного документообігу

Якщо ваш бізнес вимагає багато паперової роботи та внутрішньої співпраці, вам потрібна система управління документами. Звукове програмне забезпечення для керування файлами допоможе організувати ваш процес, підвищити продуктивність ваших співробітників і зменшити операційні витрати.

Вибір правильної системи керування файлами залежить від типу вашого бізнесу, робочого процесу та бюджету. Більшість програмного забезпечення може обслуговувати підприємства будь-якої галузі, але деякі спеціально

розроблені для регульованих підприємств, польових робіт або підприємств. І майте на увазі, що послуги DMS можуть дуже швидко подорожчати, тому вибирайте систему, яка відповідає розміру вашої організації та її діяльності також з точки зору бюджету.

Інженерний документообіг Меридіан

Зрозумійте основи та переваги програмного забезпечення EDMS, а також те, як власна платформа EDMS від Accruent – Meridian – може принести значну користь вашому бізнесу.

Програмне забезпечення EDMS, або програмне забезпечення для системи управління технічними документами, є надійною платформою, яка може централізувати вашу інженерну документацію та оптимізувати ваші операції.

Що таке програмне забезпечення EDMS?

Програмне забезпечення EDMS або програмне забезпечення системи керування технічними документами допомагає організаціям централізувати, зберігати, шукати та отримувати доступ до інженерної документації, креслень і даних із різноманітних систем і місць, таким чином підвищуючи продуктивність, зменшуючи кількість помилок через неправильні версії документів і, зрештою, допомагаючи компаніям зберігати гроші.

Багато компаній вважають свою EDMS одним із найпотужніших інструментів у своїй колекції програмного забезпечення, оскільки вона допомагає їм керувати складною інженерною документацією, оптимізувати роботу заводу, покращувати оперативне уявлення, досягати та підтримувати державні нормативні акти та стандарти, а також оптимізувати робочі процеси у відділах для збільшення ефективність. Це також може допомогти організаціям подолати найгостріші повсякденні проблеми, зокрема:

– Труднощі із забезпеченням дотримання процедур і вимог контролю.

– Проблеми з нумерацією, сортуванням і зберіганням важливих електронних і паперових документів.

– Непослідовна співпраця та закрита інформація між відділами.

– Низький доступ до даних, які можуть максимально підвищити продуктивність персоналу та забезпечити безпеку персоналу.

– Складне управління робочими замовленнями та помилки введення даних.

– Питання безпеки даних і аварійного відновлення.

Представляємо Meridian, платформу EDMS компанії Accruent

Основні функції програмного забезпечення EDMS

Надійне , комплексне програмне забезпечення EDMS має кілька основних функцій. Він може зберігати вашу важливу інженерну документацію централізованою, точною та легкодоступною. Це може оптимізувати ручні процеси та замінити ручку та папір, зрештою підвищуючи продуктивність співробітників, час безвідмовної роботи активів і рентабельність інвестицій у підприємствах і відділах.

– Аналізуйте дані для отримання корисної інформації.

– Автоматизуйте процеси робочих нарядів.

– Допоможіть користувачам автоматично імпортувати та перевіряти дані.

– Ефективно переміщуйте документи на різних етапах робочого процесу.

– Сприяти перегляду та погодженню документів.

– Інтегруйте з іншими критично важливими для бізнесу системами для повного аналізу.

Чого ви можете досягти за допомогою програмного забезпечення EDMS?

Ви можете досягти багато чого за допомогою правильного програмного забезпечення EDMS, включаючи ефективне зберігання та керування точною інформацією, інтеграцію з програмним забезпеченням EAM та іншими критично важливими системами, автоматизацію даних, надійні налаштування та підтримку відповідності.

Налаштуйте керування документами

Програмне забезпечення Meridian EDMS можна налаштувати так, щоб

воно було настільки надійним чи простим, як це потрібно вашому бізнесу. Одразу після впровадження ви можете використовувати програмне забезпечення EDMS для ефективного зберігання та керування вашими електронними документами, і його можна за потреби інтегрувати з іншими системами, включаючи програмне забезпечення EAM. З Meridian ви можете:

– Імпортуйте та впорядковуйте документи з пером і папером.

– Легко архівуйте електронні записи, які більше не є активними.

– Створюйте налаштовувані робочі процеси, щоб інформувати користувачів про їхні обов’язки та автоматизувати ключові завдання.

– Автоматично передає коментарі та фотографії з технічного обслуговування до інженерних відділів.

– Отримайте доступ до інформації про керування документами через мобільні пристрої, усуваючи таким чином потребу в роздруківках.

– Підтримуйте зв’язки між моделями САПР, активами та документами.

– Отримайте доступ до важливої технічної документації та даних EAM лише одним клацанням миші.

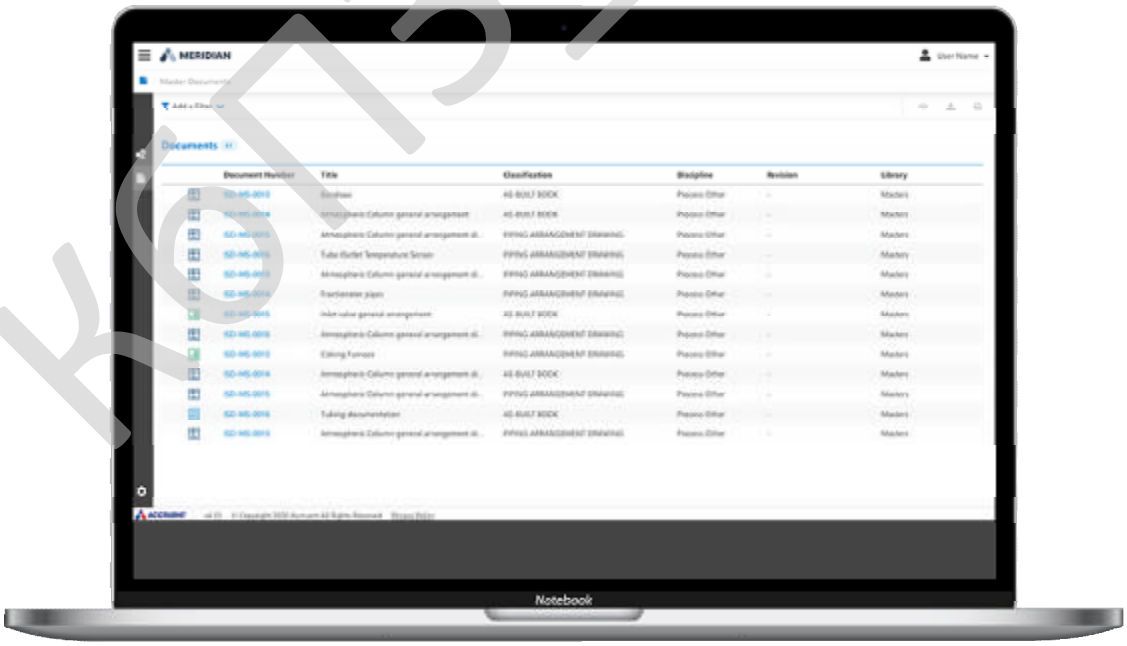

Рисунок 2.1 – Інформаційна панель Meridian Documents

Спрощення відповідності

За допомогою системи управління технічною документацією Meridian ви можете оптимізувати перевірки та переконатися, що ваш бізнес завжди готовий до процесу аудиту. Легко підтримувати відповідність:

– Автоматизована маршрутизація документів, ескалація та затвердження документів.

– Елементи керування змінами документів, які дозволяють легко помічати та контролювати будь-які зміни у ваших документах.

– Надійні можливості звітування.

– Контроль версій документів, щоб допомогти вам підтримувати відповідність регуляторним органам.

– Ефективні стандартні операційні процедури (SOP), які спрощують отримання сертифіката ISO.

Збільште ваше підключення

Підключіть своє програмне забезпечення для керування проектною документацією до інших критично важливих систем і процесів, щоб підвищити ефективність і допомогти своїм користувачам отримувати вичерпну інформацію. Зокрема, підключіть програмне забезпечення EDMS до:

– Ваша комп'ютеризована система керування технічним обслуговуванням (CMMS) для усунення розбіжностей, розробки комплексної інформації, підвищення безпеки документів і забезпечення точності інформації про ваші активи протягом усього життєвого циклу активів.

– Ваші системи програмного забезпечення САПР, щоб ви могли зберігати, керувати, візуалізувати та візуалізувати свої 2D і 3D креслення.

– Мобільні пристрої, щоб ви могли переглядати та редагувати документи з планшета чи смартфона.

– Ваша платформа управління взаємовідносинами з клієнтами (CRM).

– Бухгалтерське програмне забезпечення для комплексного розрахункового та фінансового аналізу.

Переконайтеся, що ваша документація повна

Неповна або застаріла документація може призвести до збільшення часу простою, переробки та зниження продуктивності співробітників. Meridian EDMS може допомогти забезпечити повну документацію за допомогою:

– Правильна реалізація.

– Поглиблене навчання продукту.

– Технічна підтримка.

– Автоматизації.

– Доступність на різних пристроях.

Автоматизуйте свої дані

Ефективний технічний документообіг вимагає значного ступеня автоматизації. Програмне забезпечення Meridian EDMS забезпечує такий вид автоматизації, зрештою підвищуючи продуктивність співробітників, забезпечуючи точність документів і зменшуючи кількість помилок, що виникають вручну. Можливості автоматизації включають:

– Імпорт документів.

– Групування документів із пов’язаними замовленнями на придбання, квитанціями, кресленнями тощо.

– Робочі процеси, керовані документами.

– Затвердження рахунків.

– Створення замовлень на закупівлю.

Робота з експертами EDMS компанії Ackruent

Ви не самі, коли справа доходить до впровадження програмного забезпечення Meridian EDMS. Коли прийде час почати, експерти Accruent розроблять реалізацію, протестують її та нададуть шаблони, плани, відгуки та навчання, необхідні для досягнення успіху. Вони також допоможуть вам вирішити будь-які проблеми, які можуть перешкодити реалізації, як-от чистота даних, IT та бізнес-процеси. Заплануйте телефонну розмову з професіоналом Accruent сьогодні

Після впровадження рішення Meridian ви й надалі матимете доступ до багатьох ресурсів Ackruent, включаючи підтримку після запуску та Академію Ackruent. Ці послуги створені, щоб допомогти вам прискорити впровадження, побудувати прибутковий бізнес і просуватися вперед у розвитку культури постійного вдосконалення.

Розуміння переваг і рентабельності інвестицій програмного забезпечення EDMS

Завдяки оцифровці та оптимізації керування документами за допомогою програмного забезпечення EDMS ваша компанія може побачити миттєвий приріст доходу та значне підвищення рентабельності інвестицій.

У короткостроковій перспективі це можна розглядати як скорочення витрат на документообіг, підвищення продуктивності співробітників і перерозподіл бюджетів канцелярських товарів. У загальному плані система EDMS:

– Надайте єдине джерело правдивої інформації для критично важливої інформації.

– Оптимізуйте співпрацю між відділами.

– Допоможіть забезпечити належне відстеження та керування кожною зміною документа.

– Сприяти дотриманню нормативних документів усіма відповідними органами.

– Оптимізуйте виконання завдань.

– З’єднайте всі відділи вашої організації.

Ці вдосконалення, у свою чергу, допоможуть вам автоматизувати й оптимізувати процеси, розробити бізнес-модель на основі даних і змінити ефективність вашого бізнесу.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації програми мною була використана мова програмування Visual C++. У зв'язку з тим, що сьогодні рівень складності програмного забезпечення дуже високий, розробка застосунків Windows з використанням тільки якої-небудь мови програмування значно утрудняється. Програміст повинен затратити масу часу на рішення стандартних завдань по створенню багатовіконного інтерфейсу. Реалізація технології зв'язування й вбудовування об'єктів – OLE – зажадає від програміста ще більш складної роботи. Щоб полегшити роботу програміста практично всі сучасні компілятори з мови C++ містять спеціальні бібліотеки класів. Такі бібліотеки містять у собі практично весь програмний інтерфейс Windows і дозволяють користуватися при програмуванні засобами більш високого рівня, чим звичайні виклики функцій. За рахунок цього значно спрощується розробка застосунків, що мають складний інтерфейс користувача, полегшується підтримка технології OLE і взаємодія з базами даних. Сучасні інтегровані засоби розробки застосунків Windows дозволяють автоматизувати процес створення застосунка. Для цього використовуються генератори застосунків. Програміст відповідає на питання генератора застосунків і визначає властивості застосунка – чи підтримує воно багатовіконний режим, технологію OLE, тривимірні органи керування, довідкову систему. Генератор застосунків, створить додаток, що відповідає вимогам, і надасть вихідні тексти. Користуючись їм як шаблоном, програміст зможе швидко розробляти свої застосунки. Подібні засоби автоматизованого створення застосунків включені в компілятор Microsoft Visual C++ і називаються MFC AppWizard. Заповнивши кілька діалогових панелей, можна вказати характеристики застосунка й одержати його тексти, постачені великими коментарями. MFC AppWizard дозволяє створювати одновіконні й багатовіконні застосунки, а також застосунки, що не мають головного вікна, – замість нього

використовується діалогова панель. Можна також включити підтримку технології OLE, баз даних, довідкової системи. Звичайно, MFC AppWizard не всесильний. Прикладну частину застосунка програмістові прийдеться розробляти самостійно. Вихідний текст застосунка, створений MFC AppWizard, стане тільки основою, до якої потрібно підключити інше. Але працюючий шаблон застосунка – це вже половина всієї роботи. Вихідні тексти застосунків, автоматично отриманих від MFC AppWizard, можуть становити сотні рядків тексту. Набір його вручну був би дуже стомлюючий. Потрібно відзначити, що MFC AppWizard створює тексти застосунків тільки з використанням бібліотеки класів MFC (Microsoft Foundation Class library). Тому тільки вивчивши мову C++ і бібліотеку MFC, можна користуватися засобами автоматизованої розробки й створювати свої застосунки в найкоротший термін. Як уже згадувався, MFC – це базовий набір (бібліотека) класів, написаних мовою C++ і призначених для спрощення й прискорення процесу програмування для Windows. Бібліотека містить багаторівневу ієрархію класів, що нараховує близько 200 членів. Вони дають можливість створювати Windows-застосунки на базі об'єктно-орієнтованого підходу. З погляду програміста, MFC являє собою каркас, на основі якого можна писати програми для Windows. Бібліотека MFC розроблялася для спрощення завдань, що стоять перед програмістом. Як відомо, традиційний метод програмування під Windows вимагає написання досить довгих і складних програм, що мають ряд специфічних особливостей. Зокрема, для створення тільки каркаса програми таким методом знадобиться близько 75 рядків коду. У міру ж збільшення складності програми її код може досягати воістину неймовірних розмірів. Однак та ж сама програма, написана з використанням MFC, буде приблизно в три рази менше, оскільки більшість приватних деталей приховано від програміста.

Одною з основних переваг роботи з MFC є можливість багаторазового використання того самого коду. В зв'язку з тим, що бібліотека містить багато елементів, загальних для всіх Windows-застосунків, немає необхідності щораз писати їх заново. Замість цього їх можна просто успадковувати (говорячи мовою

об’єктно-орієнтованного програмування). Крім того, інтерфейс, забезпечуваний бібліотекою, практично незалежний від конкретних деталей, його що реалізують. Тому програми, написані на основі MFC, можуть бути легко адаптовані до нових версій Windows (на відміну від більшості програм, написаних звичайними методами). Ще однією істотною перевагою MFC є спрощення взаємодії із прикладним програмним інтерфейсом (API) Windows. Будь-який додаток взаємодіє з Windows через API, що містить кілька сотень функцій. Значний розмір API утрудняє спроби зрозуміти й вивчити його цілком. Найчастіше навіть складно простежити, як окремі частини API зв’язані один з одним! Але оскільки бібліотека MFC поєднує (шляхом інкапсуляції) функції API у логічно організовану безліч класів, інтерфейсом стає значно легше управляти.

Оскільки MFC являє собою набір класів, написаних мовою C++, тому програми, написані з використанням MFC, повинна бути в той же час програмами на C++. Для цього необхідно володіти відповідними знаннями. Для початку необхідно вміти створювати власні класи, розуміти принципи спадкування й уміти перевизначати віртуальні функції. Хоча програми, що використовують бібліотеку MFC, звичайно не містять занадто специфічних елементів з арсеналу C++, для їхнього написання проте потрібні солідні знання в даній області.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи управління електронним документообігом.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Зберігання безпаперових офісних документів лише на комп'ютері чи локальному сервері створює ризики виходу з ладу жорсткого диска, пожежі, повені чи крадіжки зі зломом. А що, якщо ви хочете отримати доступ до одного з цих важливих файлів поза офісом?

Система повинна забезпечувати наступні функції:

1. Контроль доступу та інтеграція Office 365 / DocuSign. Діліться документами, папками, розділами або навіть цілим обліковим записом з різними дозволами (засіб попереднього перегляду, перегляд, редактор, лише завантаження) будь-кому, кому завгодно. Використовуйте Okta або Microsoft Entra ID (Active Directory) для централізованого керування користувачами. Створюйте групи співавторів, щоб ділитися багатьма людьми одночасно! Працюйте разом у програмах Office 365 і надсилайте документи в робочі процеси електронного підпису.

2. Потужний пошук із багатомовним OCR. Потужна функція прямого пошуку вбудована в наше програмне забезпечення для керування хмарними документами, тож ви можете шукати документи не лише за назвою файлу та метаданими, наприклад ключовими словами, але й за вмістом файлів завдяки технології оптичного розпізнавання символів (OCR). PDF-файли, файли Docx або навіть знімки екрана та скановані документи тепер доступні для повного пошуку.

3. Робочий процес і автоматизоване збереження. Рахунок-фактуру, заяву на відпустку чи інший документ потрібно затверджувати однією чи кількома особами перед наступним кроком? Не проблема! Запросіть людей схвалити та додати свої коментарі. У встановленому порядку або все відразу. Ще ніколи не було так просто отримати та відстежувати схвалення! Дізнайтеся

більше про процес затвердження! А може просто потрібно надіслати документ для підтвердження? Для цього також є робочий процес! Ви також можете встановити додатковий період зберігання для будь-якої з папок і файлів – у днях, тижнях, місяцях або роках!

4. Нумерація документів, сповіщення та журнали аудиту. Використовуйте автоматичну нумерацію документів на основі визначених схем. Налаштуйте та отримуйте автоматичні сповіщення про зміни в документах або папках так часто, як забажаєте. Тоді як контрольний журнал реєструє кожну дію кожного користувача для підзвітності.

5. Налаштовувані метадані та зв'язування файлів. Додайте теги, примітки, дату та термін виконання, щоб допомогти впорядкувати документи. Ви можете легко додавати власні поля метаданих різних типів, наприклад списки, прапорці та багато іншого! Ви також можете зв'язувати файли, щоб створювати зв'язки між документами в різних структурах папок. Шаблони метаданих також можна визначити на рівні папки.

6. Версії файлів. Ви можете завантажити нову версію документа та зберегти існуючі метадані та всі попередні версії файлу, які завжди можна легко відновити лише одним клацанням миші! Функція реєстрації/виписки документів дозволяє вам заблокувати документ для інших, поки ви редагуєте його на своєму комп'ютері. І кожен варіант документа може бути затверджений самостійно.

7. Нагадування. Щоразу, коли вам потрібно сповіщення в певну дату та час, просто встановіть нагадування! Ви можете додати скільки завгодно нагадувань і вибрати іншу адресу електронної пошти для сповіщень про кожне з них.

8. Імпорт документів електронною поштою. Кожна папка має окрему адресу електронної пошти. Просто надішліть/перешліть електронний лист на цю адресу, і вкладений файл опиниться у вибраній папці з незмінною інформацією про відправника, тему та вміст. Це не може бути простіше, і воно ідеально підходить для зберігання документів у дорозі або для надсилання їх у ваш DMS зі

сканера.

9. Підтримуються дочірні компанії. Хоча ви можете створити необмежену ієрархію під своєю основною роллю облікового запису, ви можете додати окремі місця для своїх компаній-філій! Таким чином, ви можете мати по одному для кожної вашої компанії або її підрозділів. Команди та дані зберігаються окремо, якщо ви не хочете надати комусь доступ до кількох ваших компаній.

10. Мобільний DMS. Завдяки онлайн-системі управління документами Folderit ви можете легко отримати доступ до своїх документів з будь-якого пристрою, підключеного до Інтернету, будь то ПК, Mac, планшет або смартфон. Це означає, що незалежно від того, де ви знаходитесь у світі, ви можете безпечно отримувати доступ до своїх документів і керувати ними без жодних проблем.

11. Також локальне резервне копіювання. Ваші документи зберігаються в захищеній хмарі та є цілковито безпечними, але якщо ви хочете, ви можете легко завантажувати весь обліковий запис або окремі файли/папки на жорсткий диск скільки завгодно для локальної резервної копії. Або скористайтеся нашою програмою синхронізації для Windows

12. Безпечно та надійно. У хмарній системі управління документами Folderit ваші дані не тільки надійно зберігаються, але й створюються потрійні резервні копії та шифруються на рівні банку. Наша платформа використовує захищену технологію рівня SSL, щоб гарантувати, що всі передачі даних здійснюються безпечно та безпечно. Ви можете налаштувати політику паролів, застосувати 2FA для всіх користувачів і використовувати Azure Active Directory (AD).

3.2 Розробка структурної схеми

Впровадження системи електронного документообігу дає низку переваг, як ми побачимо більш детально нижче. Простіше кажучи, він пропонує підвищену безпеку, ефективність, прозорість і продуктивність**.** Це допомагає вашій компанії залишатися організованою, а вашим співробітникам повертає дорогоцінний час, збільшуючи прибутки. Це також зменшує ймовірність втрати даних, допомагаючи вашому бізнесу працювати безперебійно та залишатися сумісним.

Отже, хто робить вибір щодо впровадження системи EDMS? Коли найкращий час інвестувати в програмне забезпечення HRMS, наприклад EDMS?

Це варіюється.

Розгортання EDMS часто ініціюється IT-відділами з метою стандартизації практик управління даними між відділами. Наприклад, один відділ може зберігати інформацію на локальних пристроях, а інший може використовувати незахищений спільний диск. Результатом цього є те, що відділи не можуть отримати доступ до даних, якими керують інші сфери діяльності, коли це необхідно. Замість простого доступу до спільного сховища, коли їм щось потрібно, вони мають спеціально запитувати інформацію, що часто призводить до затримок.

Іншою причиною впровадження системи EDMS може бути **те, що ви вирішили перейти від фізичного до цифрового зберігання.** Це може бути через зростаючі витрати на архівування або просто тому, що у вас не вистачає офісного приміщення. Крім того, оцифровуючи всі свої файли, ви також відкриваєте можливість автоматизувати свої процеси та отримати цінну інформацію про ваші бізнес-операції.

Ось кілька інших поширених проблемних моментів, які можуть спонукати компанії до впровадження програмного забезпечення EDMS для оцифровування та організації внутрішніх документів:

– Пошук і пошук документів, що займає багато часу.

– Створення інформаційних силосів між відділами, що ускладнює прийняття рішень на основі даних.

– Постійні затримки, пов’язані з даними, і неефективність основних бізнес-процесів.

– Порушення даних або помилки в результаті ручної обробки документів.

– Проблеми безпеки та вразливі місця.

Як користуватися системою електронного документообігу

Важливо, щоб ваша система електронного документообігу була максимально організованою та актуальною. EDMS має на меті забезпечити, щоб ваші співробітники могли легко знаходити все, що їм потрібно, одним натисканням кнопки. Навіть якщо дані, які вони шукають, були збережені багато років тому.

Подумайте про це так: якщо хтось із ваших співробітників відсутній, вам потрібно мати можливість легко знайти будь-які документи, створені або керовані ними, не надсилаючи їм повідомлення та не чекаючи відповіді. Іншими словами, наша EDMS має бути інтуїтивно зрозумілою, інстинктивною та зручною для користувача.

Існує низка найкращих практик, які ви можете встановити, щоб отримати максимальну віддачу від інструменту. Ці поради допоможуть вам створити логічну та інтуїтивно зрозумілу систему, де всі дані можуть бути легко знайдені та доступні будь-якому співробітнику. Переконайтеся, що ви пропонуєте всім своїм співробітникам навчання EDMS, щоб вони ознайомилися з цими правилами:

– **Створіть одну кореневу папку** (наприклад, «Спільні документи») і зберігайте всі документи у вкладених папках у кореневій папці. Це полегшує пошук речей і запуск резервних копій і архівів.

– **Створюйте папки в логічну ієрархію** та вкладайте папки в папки.

– **Встановіть правила іменування файлів і дотримуйтеся їх.** Використовуйте логічні, конкретні імена та включайте дати в імена файлів. Ви повинні мати можливість миттєво дізнатися, що таке файл, не відкриваючи його.

– **Створіть стратегію управління документами.** Цей план має містити детальні вказівки щодо всіх аспектів роботи з документами, включаючи зберігання, пошук, резервне копіювання та безпеку.

– **Переконайтеся, що всі пройшли навчання**, щоб вони знали ці вказівки.

Переваги системи EDMS

Найбільш очевидною перевагою використання EDMS, яку ви спочатку побачите, є те, що ви різко зменшите обсяг паперу, який потрібно придбати для керування внутрішніми процесами**.** Оцифровуючи свої записи, ви також усуваєте потребу у фізичному архівуванні, допомагаючи відновити простір і зменшивши витрати для вашого бізнесу.

Однак це не зупиняється на цьому.

Давайте закінчимо сьогоднішню публікацію вивченням деяких найбільших переваг впровадження електронної системи управління документами, такої як та, що включена до комплексного рішення Factorial HR. Ви можете дізнатися більше про те, як набір функцій програмного рішення Factorial може допомогти вашому бізнесу тут.

Покращена ефективність

Замість того, щоб витрачати час на копання в картотеках або запити інформації в інших відділах, ваші співробітники можуть знайти саме те, що їм потрібно, у будь-який час у кількох сховищах, пристроях і форматах документів. Вони також можуть використовувати автоматизовані процеси та

функції індексування документів, щоб швидше видобувати дані. Це спрощує ваші робочі процеси та зменшує вузькі місця, сприяючи плавній та ефективній роботі вашого щоденного бізнесу та, зрештою, заощаджуючи ваш час і гроші.

Багатоплатформні можливості

Зі збільшенням кількості моделей дистанційної та гібридної роботи та діапазону пристроїв, якими користуються співробітники в наші дні, може бути важко відслідковувати всі ваші дані. Віддалений працівник може зберігати документи, наприклад, на своєму особистому ноутбуці. Або продавець може зберегти цей важливий рахунок-фактуру на свій телефон. Єдиний спосіб отримати ці дані, коли вони вам знадобляться, це зв'язатися з ними та сподіватися, що вони швидко відреагують.

Навпаки, за допомогою EDMS дані можна зберігати в централізованому сховищі з будь-якого пристрою та з будь-якого місця. Це допомагає забезпечити цілісність даних, керування вмістом і контроль документів на багатьох пристроях і місцях.

Документом, який ви шукаєте, керував хтось, хто працював на іншому кінці світу? Без проблем. Просто перейдіть до цифрового сховища, і ви зможете завантажити те, що вам потрібно, одним натисканням кнопки.

Безпека та відповідність

Одна з найбільших переваг впровадження цифрової трансформації та впровадження електронної системи керування документами полягає в тому, що ви отримуєте підвищену безпеку та відповідність вимогам. Найкращі рішення включають інструменти для встановлення дозволів і контролю доступу до документів, щоб сприяти безперервній відповідності, зменшуючи ризик витоку даних, втрати документів і порушень відповідності. Це особливо важливо, якщо ваш бізнес працює в контрольованому середовищі або в строго регульованій галузі. Наприклад, фінанси чи охорона здоров'я.

Розширення співпраці

Система керування електронними документами може спростити спільну співпрацю ваших співробітників. Наприклад, багато рішень включають такі функції для опцій розмітки, як анотації та штампи, керування версіями та одночасне редагування документа. Це означає, що співробітники можуть разом працювати над змінами документів, незалежно від фізичного розташування. Це допоможе вам приймати більш обґрунтовані бізнес-рішення, а вашим співробітникам набагато легше працювати разом як команда.

Цілісність даних

Нарешті, EDMS може допомогти вам забезпечити цілісність ваших даних, створюючи єдине джерело правди. Завдяки функціям спільної роботи, таким як керування версіями та доступом, ви можете бути впевнені, що інформація, яку ви зберігаєте, є актуальною, правдивою та актуальною. Це підвищує точність ваших бізнес-даних, щоб ви могли приймати більш обґрунтовані рішення. Це також суттєво зменшує ймовірність людської помилки внаслідок застарілої інформації, неповних даних або неправильних активних версій. І це, зрештою, допомагає вашому бізнесу працювати більш гладко, збільшуючи прибутки.

Розглянемо більш докладно елементи структурної схеми СУЕД.

Система СУЕД, побудована за допомогою предметно-орієнтованого інструмента, має багаторівневу архітектуру. Архітектура виступає гарантом доступності, надійності й безпеці системи, що дозволяє системі СУЕД охопити всіх комп'ютеризованих співробітників і підвищити ефективність роботи організації в цілому.

Основними структурними елементами архітектури є:

– Предметно-орієнтований інструмент розробки – середовище виконання коду, що реалізує інтерфейс служб і користувальницьких додатків (у тому числі сторонньої розробки) для доступу до системи. Зокрема, сервер веб-доступу СУЕД, реалізований на платформі ASP.NET, використовує предметно-орієнтований інструмент розробки для реалізації всіх функцій системи, які стають доступні користувачам через веб-браузер.

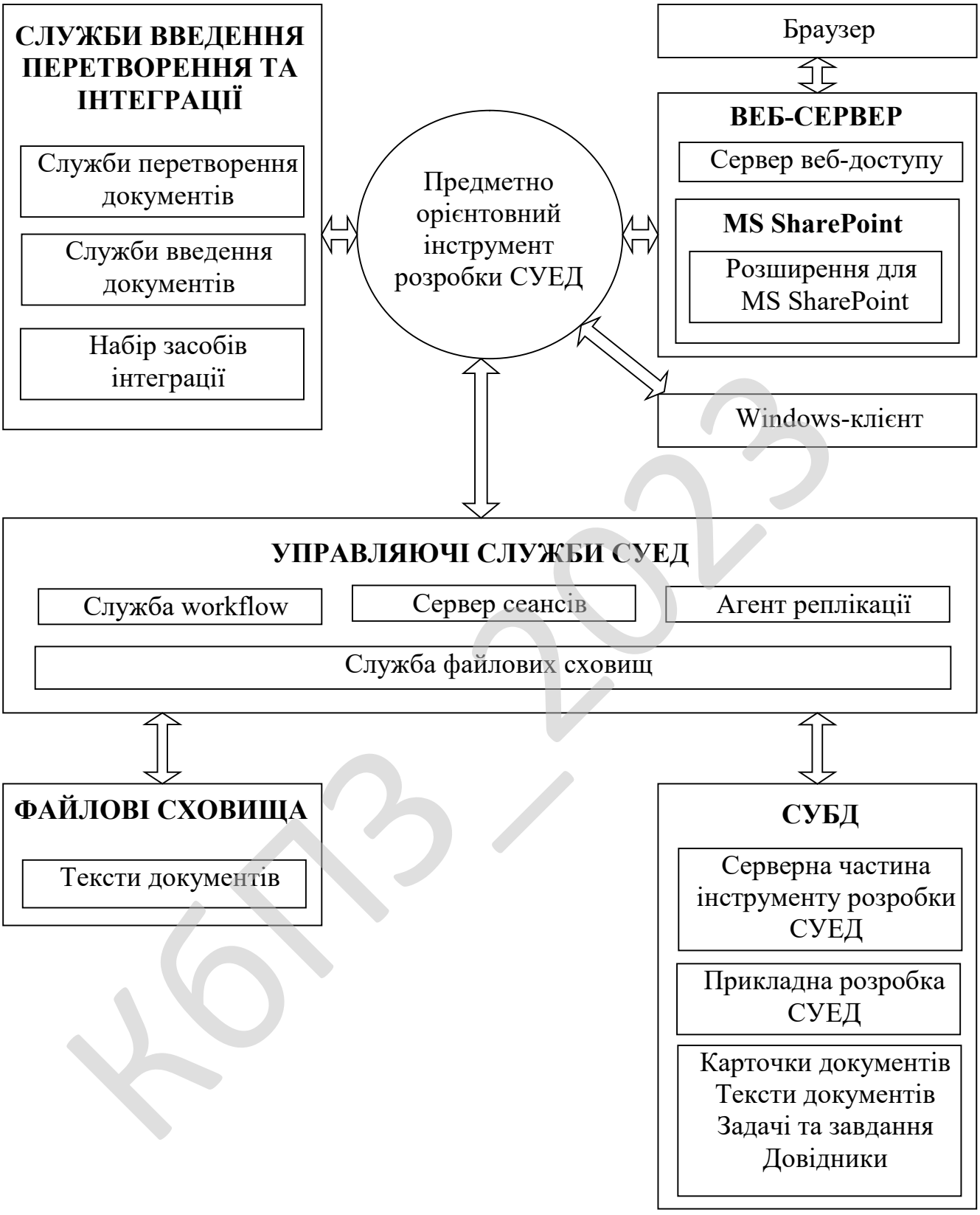

Рисунок 3.1 – Структурна схема системи управління електронної документації

– Служби перетворення документів. Перетворення документів в інші формати, добування з документів корисної інформації

– Служби введення документів. Масове введення документів у СУЕД з різних джерел (сканери, БФП, файлова система, факси, електронна пошта й т.д.).

– Набір засобів інтеграції. Легка інтеграція з ERP-системами: двостороння синхронізація довідників, включення об'єктів системи в workflow, робота з документами з ERP-системи. Workflow – це повна або часткова автоматизація бізнес-процесу, при якій документи, інформація або завдання передаються від одного учасника (бізнес-процесу) до іншого для виконання дій відповідно до набору керівних правил.

– Сервер веб-доступу до СУЕД. Робота з електронними документами, завданнями й завданнями через веб-браузер.

– Розширення для SharePoint. Набір готових веб-частин і інтеграційних механізмів, що забезпечують доступ до даних СУЕД з порталу на базі Microsoft SharePoint.

– Клієнти системи СУЕД – додатки для кінцевих користувачів, інструментарій розробки, утиліти адміністрування системи. Клієнтом може бути як Windows-додаток, що використовує для доступу до системи предметно-орієнтований інструмент розробки, так і веб-браузер.

– Керуючі служби СУЕД – служби, що забезпечують керування системою. Наприклад, служба workflow управляє роботою завдань СУЕД, а сервіси зберігання СУЕД відповідають за файлові сховища документів. Всі керуючі служби можуть бути встановлені як на один комп'ютер, так і на різні – з метою розподілу навантаження.

– Сервер реплікації. Створення ієрархічної системи розподілених систем, що обмінюються даними в режимі off-line; налаштовується состав даних, що, реплікуються.

– Служба файлових сховищ. Керування довгостроковим і архівним зберіганням великого обсягу документів, у тому числі медіаданих; налаштовування політик переміщення даних по сховищах.

– СУБД – сховище даних і метаданих системи. Одним з важливих компонентів системи, що зберігаються в СУБД, є прикладна розробка СУЕД, що визначає функціональність предметних модулів системи, замовлених, а також розроблених партнерами СУЕД рішень.

– Файлові сховища – архіви більших або рідко використовуваних документів, які ефективніше тримати за межами СУБД; управляються власними службами.

Архітектура системи СУЕД, будучи частиною інформаційної інфраструктури організації, демонструє характеристики, важливі для будь-якої корпоративної системи:

– Відкритість. Основа системи СУЕД – платформа предметно-орієнтованого інструмента розробки – підтримує технології Microsoft COM і.NET. Вона містить готові інструменти інтеграції з корпоративними додатками, у тому числі набір функцій для обробки XML-документів. Корпоративні стандарти й відкрита структура даних дозволяють легко інтегрувати СУЕД в інформаційну інфраструктуру організації.

– Розширюваність. Як правило, у кожній організації висувають унікальні вимоги до побудови електронного документообігу й рішенню завдань взаємодії. Об'єктна модель і предметно-орієнтований інструмент розробки дозволяють створювати власні й змінювати існуючі об'єкти для рішення специфічних завдань. Оскільки ядром системи є COM-сервер, що управляють функції системи можна використовувати в будь-яких сторонніх додатках.

– Масштабованість. Виділення декількох рівнів архітектури дозволяє підвищувати продуктивність системи не тільки за допомогою нарощування потужності апаратних засобів, але й завдяки розподілу служб по різних серверах. Механізм реплікації предметно-орієнтованого інструмента розробки

дозволяє побудувати територіально розподілену систему, мінімізуючи як вимоги до пропускної здатності каналів зв'язку за рахунок обсягу переданих даних між серверами, так і технічні вимоги до вторинних серверів. Виділення як SQL-серверних, так і файлових сховищ документів дозволяє гнучко управляти розподілом навантаження на сервера організації при доступі до документів.

– Надійність. Архітектура СУЕД підтримує транзакційну модель, що гарантує цілісність дані системи протягом всіх стадій їхнього життєвого циклу. Керовані SQL– і файлові сховища документів дозволяють організувати надійне зберігання документів.

– Безпека. Для кожного об'єкта системи може бути задано, які користувачі або групи мають право виконувати з ним певні дії. Конфіденційні електронні документи й завдання можуть бути зашифровані безпосередньо в системі будь-яким CryptoAPI-сумісним криптопровайдером (у тому числі сертифікованим ДСТЗІ СБУ), що гарантує захист навіть від осіб, що мають необмежений доступ до даних. Протоколювання всіх дій користувача дозволить відновити історію роботи з об'єктами системи у випадку порушення режиму безпеки. Забезпечується високий захист від несанкціонованого доступу до сховищ документів всіх типів.

Таким чином, архітектура системи СУЕД розроблена з урахуванням максимального використання всіх переваг сучасних технологій, платформ і предметно-орієнтованого підходу до побудови інформаційних систем керування.

3.3 Розробка функціональної схеми

На основі структурної схеми моделі середовища системи обміну даними була розроблена функціональна схема моделі середовища відкритої СУЕД, при цьому визначені наступні об'єкти стандартизації:

– служби засобів проектування;

– служби засобів управління змістом;

– служби засобів моделювання процесів;

– служби засобів моделювання даних;

– формати файлових об'єктів;

– служби засобів забезпечення безпеки;

– інтерфейс користувача;

– служби засобів управління даними;

– служби засобів управління системою;

– служби засобів управління сховищем;

– служба комунікацій.

Отримані дані дозволили синтезувати функціональну схему еталоної моделі середовища відкритої СУЕД (рисунок 3.2).

Дана модель використана при розробці функціонального стандарту середовища відкритої СУЕД. Призначенням даного стандарту є:

– Забезпечення мобільності (переміщуваності) СУЕД.

– Забезпечення інтероперабельності СУЕД.

– Забезпечення масштабованості СУЕД.

– Забезпечення адаптуємості системи.

Визначено, що область застосування функціонального стандарту СУЕД установлює загальні положення по створенню й експлуатації відкритої системи управління документообігом на основі окремих модулів, що входять у єдину інформаційну систему організації.

Положення функціонального стандарту підлягають застосуванню при рішенні завдань:

– створення, модернізації СУЕД;

– організації доступу користувачів до ресурсів СУЕД;

– інтеграції обчислювальних і інформаційних ресурсів із СУЕД.

Для заповнення профілю в роботі була розроблена методика вибору стандартів на основі експерименту.

Результатом розробки функціонального стандарту є набір вимог до елементів відкритої СУЕД, розроблений відповідності з ДЕРЖСТАНДАРТ Р ИСО/МЭК 10000-1-99 «Інформаційна технологія. Основи й таксономія міжнародних функціональних стандартів. Частина 1. Загальні положення й основи документування» у вигляді функціонального стандарту – Профілю середовища відкритої СУЕД.

Проведемо дослідження особливості функціонування елементів системи управління електронним документообігом і експериментальним дослідженням з визначення ефективного формату файлових об'єктів СУЕД, як засобу підвищення техніко-економічних характеристик СУЕД.

Виходячи з вимог до функціонала систем управління й аналізу існуючих СУЕД, у системах управління виділені типові елементи (ТЕ):

– ТЕ «обробки інформації».

– ТЕ «передачі інформації».

– ТЕ «сполучна лінія».

– ТЕ «масив зберігання інформації».

– ТЕ «точка діалогу».

Дані типові елементи служать для опису інформаційної системи й інформаційних потоків у будь-якій системі, що неможливо зробити через описані вище служби еталонної моделі середовища відкритої СУЕД.

Таким чином показано, що через подібний базис типових елементів може бути представлена кожна, як завгодно складна інформаційна система.

Для проведення функціональної стандартизації ТЕ здійснемо перехід від ТЕ до служб СУЕД.

Для перевірки експериментальним шляхом залежності ефективності роботи типових елементів СУЕД від форматів використовуваних файлових об'єктів і вибору стандартів для відповідного розділу профілю розроблені:

– вимірювальна установка для виміру часу мережної затримки;

– програма статистичного аналізу файлових об'єктів СУЕД;

– методики проведення вимірів часу мережної затримки й розмірів файлів у сховище СУЕД.

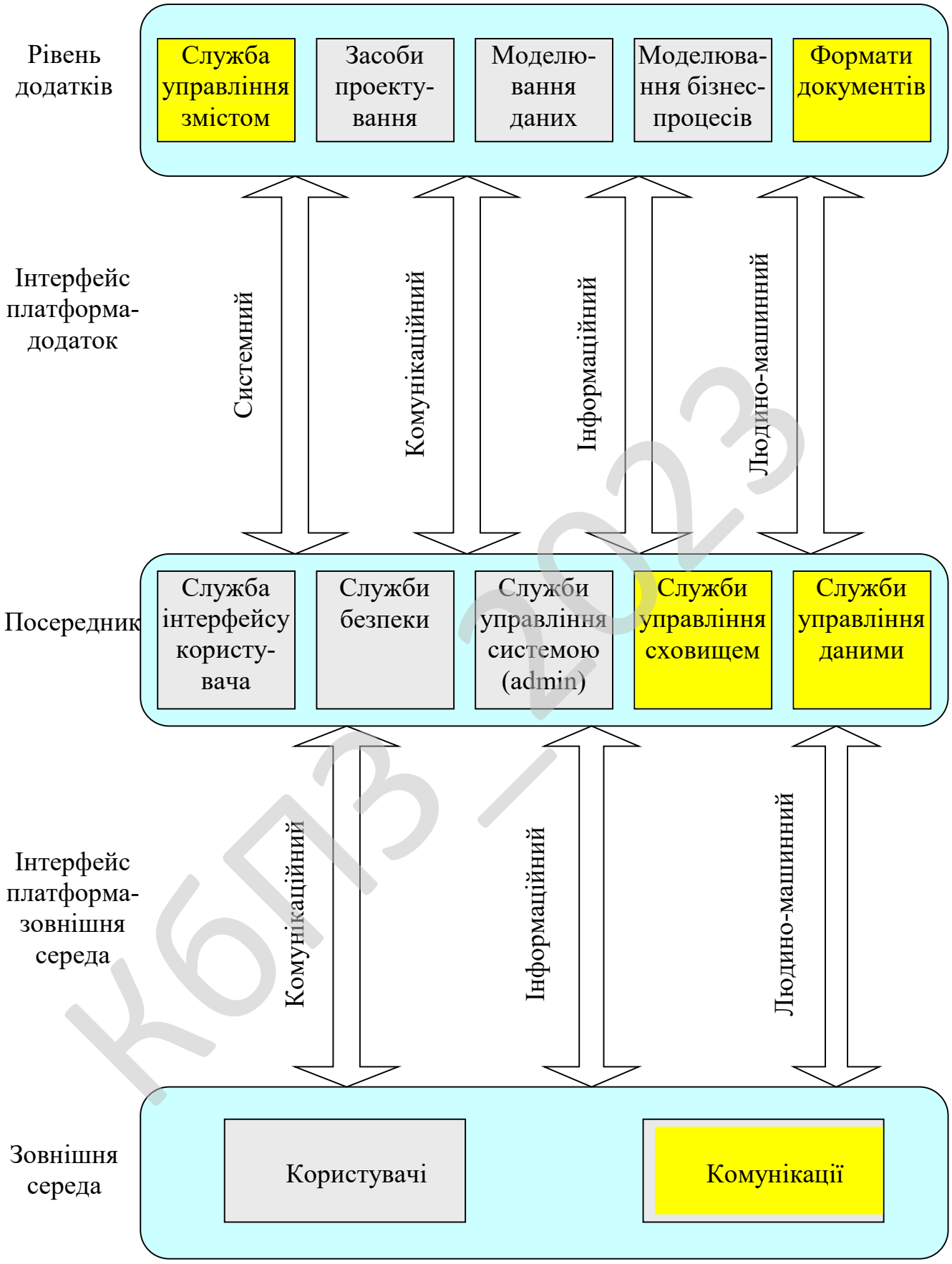

Рисунок 3.2 – Функціональна схема системи управління електронним документообігом

Визначено співвідношення розмірів файлів залежно від типу інформації у файлах і їхніх розмірах. Визначено формати, у яких розмір файлу виходить мінімальним (таблиця 3.1). Визначено, як саме впливає формат файлових об'єктів (ФФО) на роботу служб СУЕД. На рисунку 3.3, на моделі СУЕД показані служби, на роботу яких впливає формат файлових об'єктів. Ці служби виділені жовтим кольором.

Результати досліджень дозволили визначити залежність розмірів файлових об'єктів від типу їхнього вмісту й залежність часу мережної затримки, що підтвердило, що швидкодія системи залежить від розміру файлових об'єктів СУЕД. Також дослідження дозволило підтвердити залежність ефективності роботи СУЕД від ФФО й визначити ефективний формат файлових об'єктів СУЕД, що дозволяє мінімізувати необхідні для роботи СУЕД комунікаційні й обчислювальні ресурси, що дозволило гармонізувати стандарти файлових об'єктів у функціональному стандарті (профілі) середовища відкритої системи управління електронним документообігом і підвищити ефективність роботи ТЕ.

Таблиця 3.1 – Співвідношення типу інформації у файлах і їхніх розмірах

	Тест	Таблиці	Графіка	Формати, у яких розмір файлу виходить мінімальним
1.	100%	0	0	PDF, DOC
2.	80%	0	20%	PDF, DOC
3.	80%	20%	0	PDF, XML
4.	60%	0%	40%	PDF, XML
5.	60%	40%	0%	PDF, XML
6.	40%	0%	60%	XML, PDF
7.	40%	60%	0%	XML, PDF
8.	20%	0%	80%	XML, PDF
9.	20%	80%	0%	XML, PDF
10.	0%	0%	100%	XML, PDF
11.	0%	100%	0%	PDF, XML

Таким чином показано, що залежно від структури документів є ефективні можливості по поліпшенню роботи СУЕД залежно від сфери її застосування.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3.

З рисунку видно, що на початку роботи програми запускається процес початку/кінця роботи програми.

Він взаємодіє з наступними двома процесами:

– Процес введення документів.

– Процес управління базою даних.

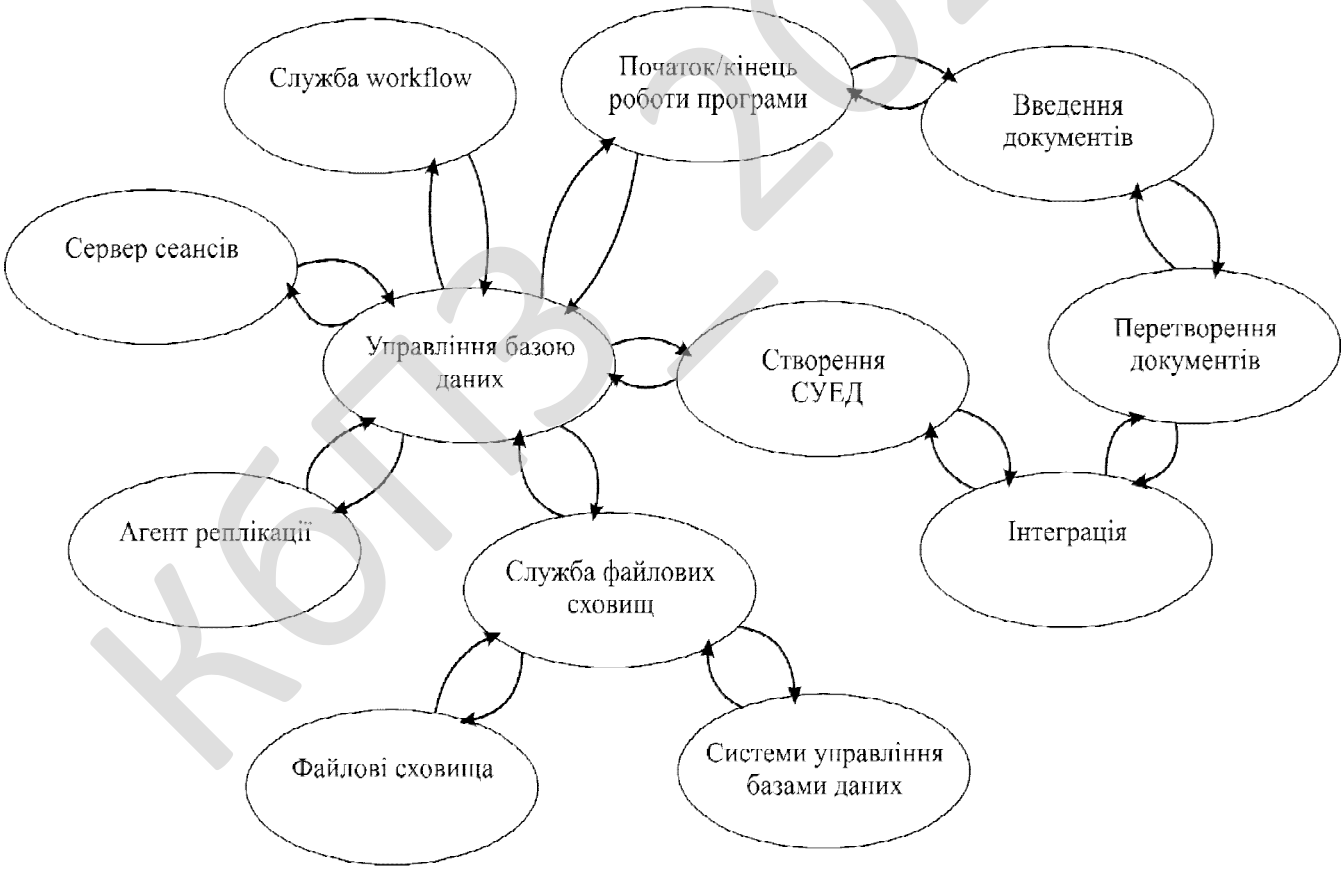

Рисунок 3.4 – Діаграма процесів системи

Процес введення документів взаємодіє з процесом перетворення документів.

Процес перетворення документів у свою чергу взаємодіє з процесом інтеграції.

Процес інтеграції взаємодіє з процесом створення СУЕД.

Останній процес, у свою чергу, взаємодіє з процесом управління базою даних.

Процес управління базою даних взаємодіє з наступними процесами:

– Процесом реалізації служби файлових сховищ.

– Процесом реалізації агенту реплікації.

– Процесом реалізації серверу сеансів.

– Процесом реалізації служби workflow. Workflow – це повна або часткова автоматизація бізнес-процесу, при якій документи, інформація або завдання передаються від одного учасника (бізнес-процесу) до іншого для виконання дій відповідно до набору керівних правил.

– Процесом початку/кінця роботи програми.

Процес реалізації служби файлових сховищ взаємодіє з наступними процесами:

– Процесом реалізації файлових сховищ.

– Процесом реалізації системи управління базами даних.

Процес управління базою даних є кінцевим процесом, тому саме він взаємодіє з процесом початку/кінця роботи програми.

Розглянувши у цьому розділі структурну схему розробленої відкритої системи обміну даними, структурну схема системи управління електронної документації, функціональну схем, діаграму взаємодії процесів, перейдемо до опису блок-схеми алгоритмів розробленого, у результаті виконання магістерської роботи, програмного забезпечення системи управління електронним документообігом.

4 Реалізація роботи. Розрахунки і експериментальні дані, що підтверджують правильність проектних рішень

4.1 Блок–схеми та опис алгоритмів функціонування системи

Блок-схема алгоритму роботи основної програми зображена на рисунку 4.1. Після запуску програми на екран виводиться вікно авторизації користувача програми.

Користувач вводить логін та пароль у виведеному для цього вікні авторизації.

Відбувається перевірка введеного логіну та паролю.

Якщо обліковий запис не існує, то виводиться повідомлення про помилку, й система переходить у режим чекання введення логіну та паролю, який дозволить увійти у систему.

Якщо ж логін та пароль є легітимними, то відбувається перевірка, під якими правами зайшов користувач. При цьому існують наступні права користувача:

– Права адміністратора.

– Права користувача.

Якщо користувач отримав права адміністратора, то відбувається наступна послідовність дій:

– Виводиться вікно адміністратора.

– Створюються, редагуються та видаляються документи.

– Формуються звіти.

– Відбувається перегляд історії надходження документів.

– Відбувається перегляд історії виконань документів.

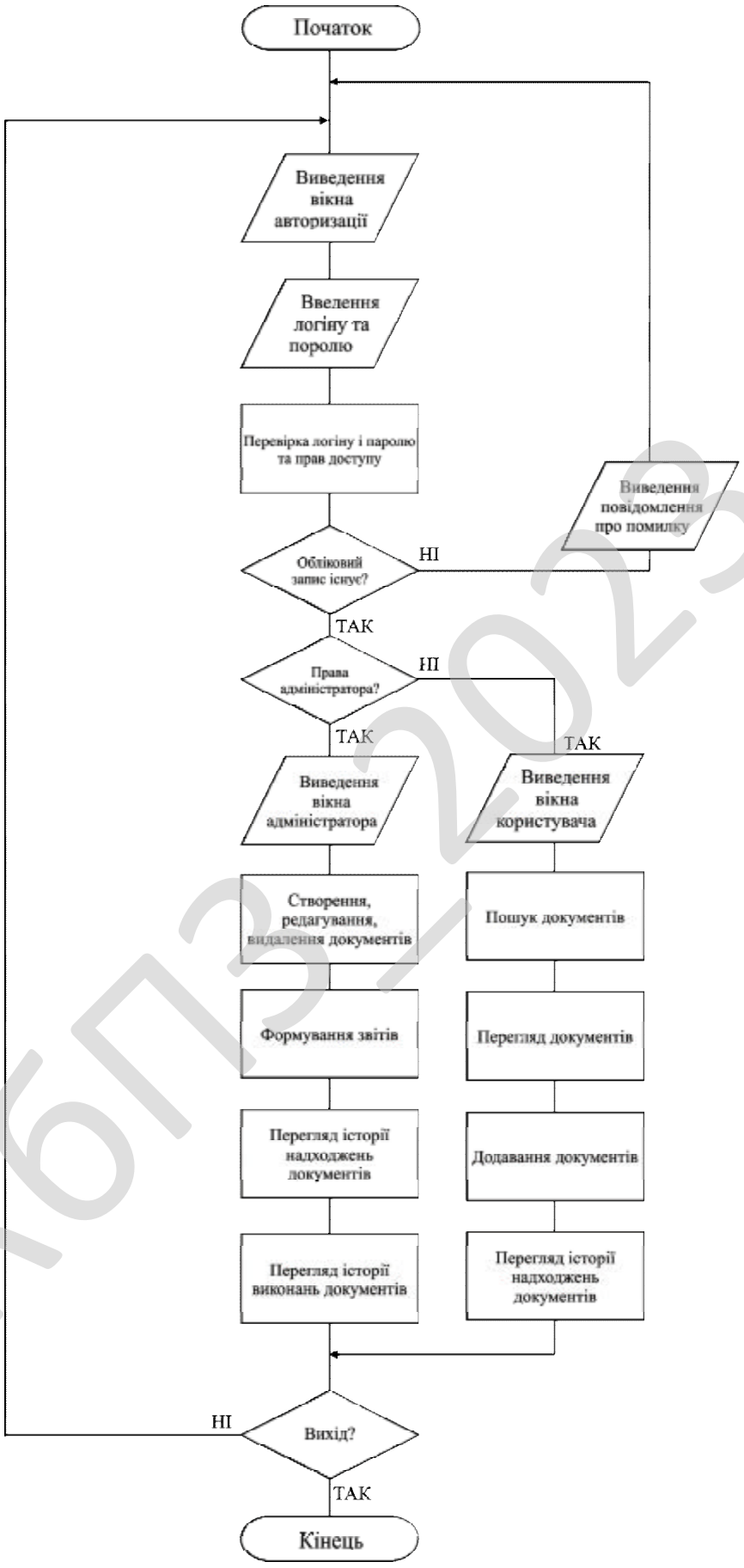

Рисунок 4.1 – Блок-схема алгоритму роботи основної програми

Якщо користувач отримав не права адміністратора, то відбувається наступна послідовність дій:

– Виводиться вікно користувача.

– Відбувається пошук документів.

– Відбувається перегляд документів.

– Відбувається додавання документів.

– Відбувається перегляд історії надходження документів.

Після цього користувач обирає працювати йому далі з системою, або ні.

Якщо він обирає, що працювати, тоді відбувається перехід у початок програми, до завантаження вікна авторизації.

У іншому випадку програма закінчує свою роботу.

Блок-схема алгоритму роботи підпрограми пошуку документів зображена на рисунку 4.2.

З неї ми бачимо, що спершу відбувається виведення вкладки пошуку документів.

Після цього відбувається підключення бази даних.

Наступним кроком є введення користувачем ключової фрази для пошуку документів.

При цьому вибирається категорія, у якій потрібно здійснити пошук.

Коли користувачем введена ключова фраза, та вибрана категорія у якій потрібно вести пошук, запускається процес пошуку.

Для цього спершу відбувається перехід до вказаного розділу.

Після цього, у цьому розділі, відбувається пошук документу за вказаною фразою.

Якщо пошук не знайшов документів, які відповідають заданим параметрам пошуку, то виводиться вікно про те, що такі документи з заданими параметрами не знайдені.

Після цього пропонується повторити пошук.

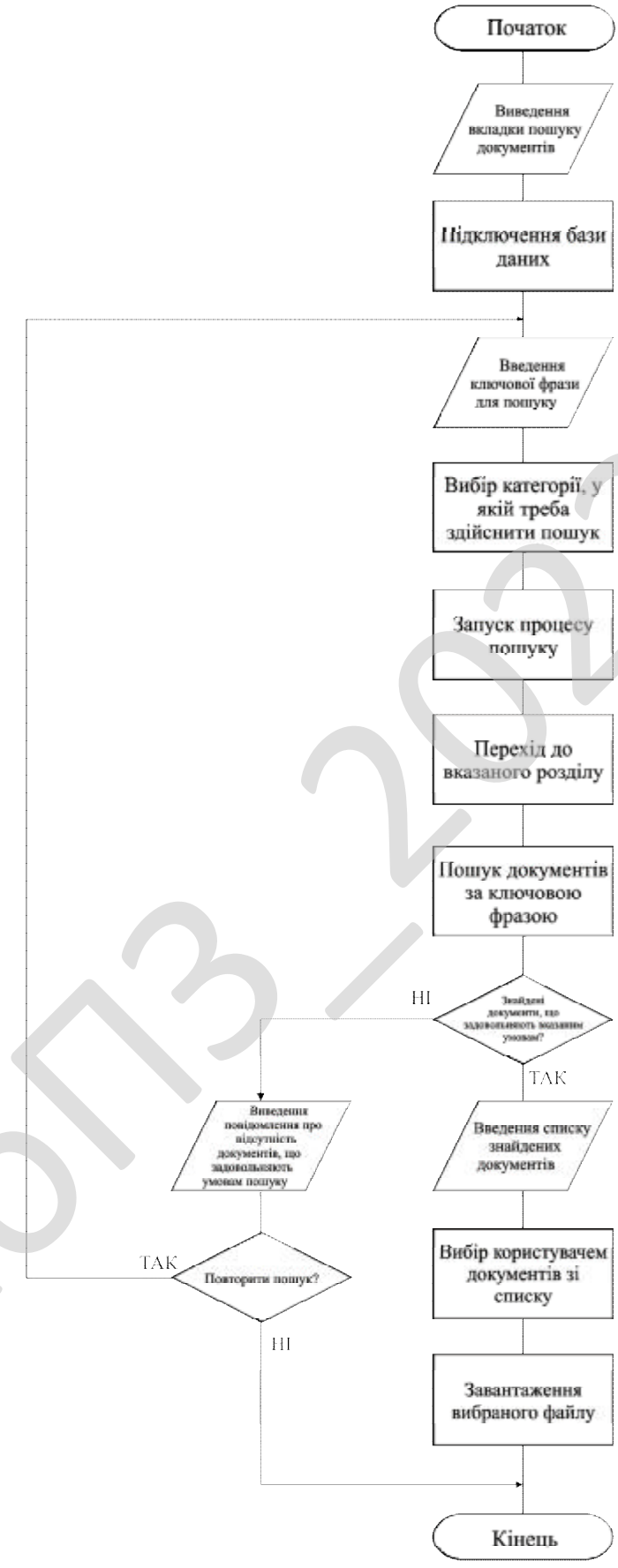

Рисунок 4.2 – Блок-схема алгоритму роботи підпрограми пошуку документів

Якщо користувач обирає вибір повтору пошуку, то переходить до виведення вікна пошуку, та його параметрів.

Якщо пошук знайшов документи, які відповідають заданим параметрам пошуку, то виводиться вікно, у якому відображено список знайдених документів.

Після цього користувач обирає документи з цього списку.

Кінцевим етапом підпрограми пошуку є завантаження вибраного файлу.

Приведемо частину програмного коду, яка реалізує функції головного вікна.

```
/*****************************************************/
// NAME:       CWorker
// DESCRIPTION:    Конструктор класу
// INPUT:      N/D
/*****************************************************/
CWorker::CWorker(void)
{
  listDoc = gcnew ArrayList();
  logInput = gcnew CLogger();
  logOutput = gcnew CLogger();
}
/*****************************************************/
// NAME:                GetIndexByISBNHash
// DESCRIPTION:         Функція одержання індексу документа в загальному списку по
                        вхідному номеру документа
// INPUT:               ddHashValue – значення хеша вхідного номера документа
/*****************************************************/
Int32 CWorker::GetIndexByISBNHash(Int32 HashValue)
{
  // Цикл пошуку документа із заданим вхідним номером документа
  for(Int32 i = 0; i < listDoc->Count; i++)
  {
     // Хеш-значення вхідного номера документа заданого документа збігається з
     // хеш-  значенням вхідного номера документа знайденого документа?
     if(HashValue == ((CDoc^)listDoc[i])->GetISBNHash())
     {
          // Документ знайдений, повернути індекс
          return i;
     }
  }
  // Документ не знайдений, повернути -1
```

```
  return -1;
}
/*****************************************************/
// NAME:                AddDoc
// DESCRIPTION:         Функція додавання нового документа
// INPUT:               SourceDoc – об'єкт "документ" для включення в список
// OUTPUT:              TRUE – документ успішно доданий
//                      FALSE – такий документ уже існує в системі
/*****************************************************/
Boolean CWorker::AddDoc(CDoc^ SourceDoc)
{
  // Документи з таким вхідним номером документа не існує?
  if(this->GetIndexByISBNHash(SourceDoc->GetISBNHash()) == -1)
  {
     // Додати заданній документ у загальний список
     listDoc->Add(SourceDoc);
     // Записати подія в лог
     logInput->WriteEvent(
          "Новий документ '" + ((CDoc^)SourceDoc)->GetName() +
          "', Автор '" + ((CDoc^)SourceDoc)->GetAuthor() +
          "', Вхідний номер документа " + ((CDoc^)SourceDoc)->GetISBN());
     // Функція відробила успішно
     return true;
  }
  else
  {
     // Документ із таким же вхідним номером документа існує, повернути помилку
     return false;
  }
}
/*****************************************************/
// NAME:                RemoveDoc
// DESCRIPTION:         Функція видалення документів із системи
// INPUT:               ddISBNHash – значення хеша вхідного номера документа
// OUTPUT:              0 – видалення зроблене
//                      1 – видалення неможливо, не всі екземпляри перебувають у
                        системі
//                      2 – документ із заданим вхідним номером документа не
                        знайдений
/*****************************************************/
Int32 CWorker::RemoveDoc(Int32 ddISBNHash)
{
```

```
  // Знайти документ по вхідному номеру документа
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);
  // Документ із таким вхідним номером документа існує?
  if(ddIndex != -1)
  {
     // Перевірити, що жодного документа немає в користувача
     if(((CDoc^)listDoc[ddIndex])->GetTotalNumber() ==
          ((CDoc^)listDoc[ddIndex])->GetFreeNumber())
     {
          // Записати подія в лог
          logOutput->WriteEvent(
               "Виконаний документ '" + ((CDoc^)listDoc[ddIndex])->GetName() +
               "', Автор '" + ((CDoc^)listDoc[ddIndex])->GetAuthor() +
               "', Вхідний номер документа " + ((CDoc^)listDoc[ddIndex])-
>GetISBN());
          // Видалити знайдений документ зі списку
          listDoc->RemoveAt(ddIndex);
          // Функція відробила успішно
          return 0;
     }
     else
     {
          // Повернути помилку, не всі документи перебувають у системі
          return 1;
     }
  }
  else
  {
     // Повернути помилку, документ із таким вхідним номером документа не був
знайдений
     return 2;
  }
}
/*************************************************/
// NAME:                GiveDoc
// DESCRIPTION:         Функція видачі документів користувачеві
// INPUT:               listDoc – список документів користувача (для виконання
                        додавання)
//                      ddISBNHash – хеш вхідного номера документа, що потрібно
                        одержати
// OUTPUT:              TRUE – операція пройшла успішно
```

```
//                      FALSE – у наявності немає жодного екземпляра заданого
                        документа
/*****************************************************/
Boolean CWorker::GiveDoc(ArrayList^ listReader, Int32 ddISBNHash)
{
  // Визначити індекс по вхідному номеру документа
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);
  // Зменшити кількість вільних документів
  if((ddIndex != -1) && ((CDoc^)listDoc[ddIndex])->DecFreeNumber())
  {
     // Додати документ у список користувача
     listReader->Add(listDoc[ddIndex]);
     // Функція відробила успішно
     return true;
  }
  else
  {
     // Такого документа не є в наявності
     return false;
  }
}
/*****************************************************/
// NAME:                TakeDoc
// DESCRIPTION:         Функція прийняття документів від користувача назад у
                        систему
// INPUT:               // listDoc – список документів користувача (для виконання
                        видалення)
//                      ddISBNHash – хеш вхідного номера документа, що потрібно
                        повернути
/*****************************************************/
void CWorker::TakeDoc(ArrayList^ listReader, Int32 ddISBNHash)
{
  // Визначити індекс по вхідному номеру документа (документ існує, тому що його
вхідний номер документа був повідомлений користувачеві)
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);
  // Видалити документ зі списку користувача
  // listReader->RemoveAt(ddIndex);
  // Збільшити кількість вільних документів
  ((CDoc^)listDoc[ddIndex])->IncFreeNumber();
}
/*****************************************************/
// NAME:                LoadDocList
```

```
// DESCRIPTION:         Функція завантаження документів системи з файлу
// INPUT:               N/D
/*****************************************************/
void CWorker::LoadDocList()
{
  // Перевірити існування файлу
  if(!File::Exists(DOCS_FILE))
  {
     // Файл не знайдений, закінчити виконання функції
     return;
  }
  // Відкрити файл
  StreamReader^ srDoc = gcnew StreamReader(DOCS_FILE);
  // Продовжувати, поки не буде знайдений кінець файлу
  while(srDoc->EndOfStream == false)
  {
     // Створити новий об'єкт "документ"
     CDoc^ NewDoc = gcnew CDoc();
     // Завантажити з файлу й установити параметри документа
     NewDoc->SetName(srDoc->ReadLine());
     NewDoc->SetAuthor(srDoc->ReadLine());
     NewDoc->SetISBN(srDoc->ReadLine());
     NewDoc->SetTheme(srDoc->ReadLine());
     NewDoc->SetPages(Convert::ToInt32(srDoc->ReadLine()));
     NewDoc->SetTotalNumber(Convert::ToInt32(srDoc->ReadLine()));
     NewDoc->SetFreeNumber(Convert::ToInt32(srDoc->ReadLine()));
     // Включити документ у загальний список документів
     listDoc->Add(NewDoc);
  }
  // Закрити файл
  srDoc->Close();
}
/*****************************************************/
// NAME:                SaveDocList
// DESCRIPTION:         Функція збереження документів системи у файл
// INPUT:               N/D
/*****************************************************/
void CWorker::SaveDocList()
{
  // Перевірити існування файлу
  if(File::Exists(DOCS_FILE))
  {
```

```
      // Видалити файл
      File::Delete(DOCS_FILE);
   }
   // Створити й відкрити файл
   StreamWriter^ swDoc = gcnew StreamWriter(DOCS_FILE);
   // Цикл запису по одному документу
   for(Int32 i = 0; i < listDoc->Count; i++)
   {
      // Одержати параметри документа й записати їх у файл
      swDoc->WriteLine(((CDoc^)listDoc[i])->GetName());
      swDoc->WriteLine(((CDoc^)listDoc[i])->GetAuthor());
      swDoc->WriteLine(((CDoc^)listDoc[i])->GetISBN());
      swDoc->WriteLine(((CDoc^)listDoc[i])->GetTheme());
      swDoc->WriteLine(Convert::ToString(((CDoc^)listDoc[i])->GetPages()));
      swDoc->WriteLine(Convert::ToString(((CDoc^)listDoc[i])->GetTotalNumber()));
      swDoc->WriteLine(Convert::ToString(((CDoc^)listDoc[i])->GetFreeNumber()));
   }
   // Закрити файл
   swDoc->Close();
}
/*****************************************************/
// NAME:                FindDoc
// DESCRIPTION:         Функція пошуку документа по заданих параметрах
// INPUT:               strFindDoc – рядок для пошуку
/*****************************************************/
ArrayList^ CWorker::FindDoc(String^ strFindValue)
{
   // Список для результату
   ArrayList^ listResult = gcnew ArrayList();
   // Шукане значення без обліку регістра
   String^ strValue = strFindValue->ToLower();
   // Задана порожній рядок?
   if(String::IsNullOrEmpty(strValue->Trim()))
   {
      //Вивести весь список
      for(Int32 i = 0; i < listDoc->Count; i++)
      {
           // Додати документ у результуючий список
           listResult->Add(listDoc[i]);
      }
   }
   else
```

```
  {
     // Вибрати з умовою
     for(Int32 i = 0; i < listDoc->Count; i++)
     {
          // Пошук рядка в кожному полі без обліку регістра
          if(((((CDoc^)listDoc[i])->GetName())->ToLower())->IndexOf(strValue) != -
1 ||
             ((((CDoc^)listDoc[i])->GetAuthor())->ToLower())->IndexOf(strValue) !=
-1 ||
             ((((CDoc^)listDoc[i])->GetISBN())->ToLower())->IndexOf(strValue) != -
1 ||
             ((((CDoc^)listDoc[i])->GetTheme())->ToLower())->IndexOf(strValue) !=
-1 ||
            Convert::ToString((((CDoc^)listDoc[i])->GetPages()))-
>IndexOf(strValue) != -1)
          {
               // Додати документ у результуючий список
               listResult->Add(listDoc[i]);
          }
     }
  }
  // Повернути список збігів
  return listResult;
}
/*****************************************************/
// NAME:                ViewDoc
// DESCRIPTION:         Функція перегляду інформації про заданий документ
// INPUT:               ddISBNHash – хеш вхідного номера документа, інформацію
                        про яку треба переглянути
/*****************************************************/
CDoc^ CWorker::ViewDoc(Int32 ddISBNHash)
{
  // Визначити індекс по вхідному номеру документа (документ існує, тому що його
вхідний номер документа був повідомлений користувачеві)
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);
  // Повернути елемент "документ" із загальне списку
  return (CDoc^)listDoc[ddIndex];
}
/*****************************************************/
// NAME:                ReadLogs
// DESCRIPTION:         Функція зчитування історії з файлу
// INPUT:               N/D
```

```
/*****************************************************/
void CWorker::ReadLogs()
{
  // Файл із історією надходжень існує?
  if(File::Exists(LOG_INPUT))
  {
     // Зчитування історії надходжень
     logInput->strLog = File::ReadAllText(LOG_INPUT);
  }
  else
  {
     logInput->strLog = "";
  }
  // Файл із історією списань існує?
  if(File::Exists(LOG_INPUT))
  {
     // Зчитування історії списань
     logOutput->strLog = File::ReadAllText(LOG_OUTPUT);
  }
  else
  {
     logOutput->strLog = "";
  }
}
/*****************************************************/
// NAME:                WriteLogs
// DESCRIPTION:         Функція збереження історії у файл
// INPUT:               N/D
/*****************************************************/
void CWorker::WriteLogs()
{
  // Запис історії
  File::WriteAllText(LOG_INPUT, logInput->strLog);
  File::WriteAllText(LOG_OUTPUT, logOutput->strLog);
}
/*****************************************************/
// NAME:                ClearLogs
// DESCRIPTION:         Функція очищення історії
// INPUT:               N/D
/*****************************************************/
void CWorker::ClearLogs()
{
```

```
  logInput->strLog = "";
  logOutput->strLog = "";
}
/*****************************************************/
// NAME:                ViewInputLog
// DESCRIPTION:         Функція перегляду історії надходжень
// INPUT:               N/D
/*****************************************************/
String^ CWorker::ViewInputLog()
{
  return logInput->strLog;
}
/*****************************************************/
// NAME:                ViewOutputLog
// DESCRIPTION:         Функція перегляду історії списань
// INPUT:               N/D
/*****************************************************/
String^ CWorker::ViewOutputLog()
{
  return logOutput->strLog;
}
/*****************************************************/
```

Взаємозв’язок класів які використані при реалізації програми, наведено на рисунку 4.3.

Проведемо оцінку якості роботи системи управління електронним документообігом.

В [2] показано основні переваги застосування теорії нечітких множин для рішення завдання оцінки ефективності роботи СУЕД у порівнянні із традиційними підходами теорії автоматичного управління.

Розроблено критерії оцінки якості роботи системи управління електронним документообігом. Показано архітектуру нечіткої моделі управління якістю роботи СУЕД. Основним модулем системи оцінки якості роботи СУЕД є блок прийняття рішень, хоча інші блоки не менш важливі для нормального функціонування моделі.

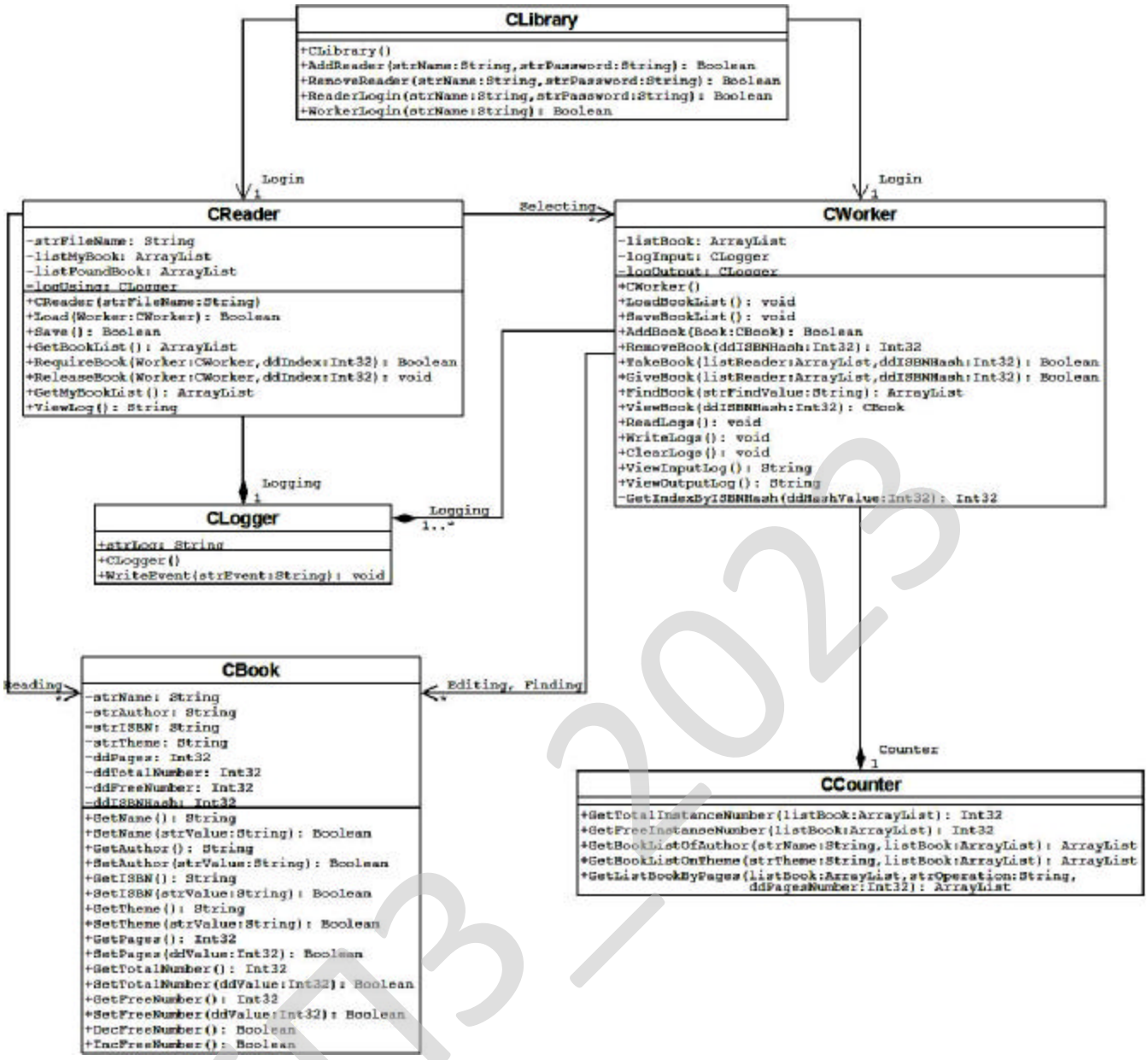

Рисунок 4.3 – Class_Digramm

Блок оцінки станів, на основі вступник на його вхід інформації, будує формалізований опис ситуації, що виникла при роботі СУЕД.

У блоці видачі керуючих впливів здійснюється перехід від внутрішньої форми завдання керуючих рішень до зовнішньої форми.

Для оцінки якості роботи СУЕД скористаємося критерієм, вираженим безліччю M, елементи якого є інтервалами якісної (нечіткої) шкали:

М={«дуже добре», «добре», «задовільно» «незадовільно»}, для цього методом експертних оцінок були визначені конкретні значення процентного вмісту тексту, графіки й таблиць у файлах певного формату.

Виходячи з аналізу експериментальних даних, був отриманий масив $\{W_N(X)\}$, що містить інтервали значень фізичної величини, що характеризує ефективність роботи елемента, при яких забезпечується ефективна робота елементів пристрою (системи) і масив $\{WW_N(X)\}$, що містить інтервали значень фізичної величини, що характеризує функціонування елемента, і значення функції приналежності стану елемента системи до одного з можливих станів. Розроблено вирішальні правила виходячи зі значень функції приналежності стану елемента одному з можливих станів для кожного з інтервалів базових значень.

Для перевірки запропонованих правил була зроблена побудова моделі оцінки якості роботи системи управління електронним документообігом у середовищі MatLab 7.0 для СУЕД фінансової організації. Модель заснована на методі Мамдани, що має у своїй основі базу знань у вигляді сукупності нечітких предикатних правил виду:

П1: якщо x є A1, тоді z є B1,

П2: якщо x є A2, тоді z є B2,

.....

Пn: якщо x є An, тоді z є Bn,

З метою перевірки даних про те, що використовувані критерії можуть застосовуватися для діючих систем проведений ряд експериментів– інструментальних досліджень на діючій системі, для чого якого були обмірювані й визначені величини:

– кількість переданих у СУЕД файлів;

– формати переданих у СУЕД файлів.

Отримано підтвердження того, що використовувані для оцінки ефективності роботи СУЕД критерії можуть застосовуватися для діючих систем.

Розроблено методику оцінки ефективності роботи елементів СУЕД.

Зроблено оцінку ефективності й повернення інвестицій від впровадження відкритої системи управління електронним документообігом.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму Camellia – блоковий шифр на основі мережі Фейстеля. У криптографії, Camellia – це симетричний ключ блоковий шифр із розміром блоку 128 біт і розмірами ключа 128, 192 і 256 біт. Він був розроблений спільно Mitsubishi Electric і NTT з Японії. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу NESSIE і Японським CRYPTREC проект. шифр має рівні безпеки й можливості обробки, порівнянні з Advanced Encryption Standard.

Шифр був розроблений, щоб підходити як для програмних, так і для апаратних реалізацій, від недорогих смарт-карти для високошвидкісних мережних систем. Він є частиною криптографічного протоколу Transport Layer Security (TLS), призначеного для забезпечення безпеки зв'язки в комп'ютерній мережі, такий як Інтернет

Camellia – це шифр Фейстеля з 18 раундами (при використанні 128-бітних ключів) або 24 раундами (при використанні 192- або 256-бітних ключів). Кожні шість раундів застосовується шар логічного перетворення: так звана «FL-функція» або її зворотна. Camellia використовує чотири 8 × 8-бітних S-блоку із вхідними й вихідними афінними перетвореннями й логічними операціями. Шифр також використовує введення й вивід відбілювання клавіш. Шар дифузія використовує лінійне перетворення на основі матриці з номером галузей 5.

Аналіз безпеки

Камелія вважається сучасним надійним шифром. Навіть при використанні параметра меншого розміру ключа (128 біт) уважається неможливим зламати його за допомогою атаки грубої сили на ключі за допомогою сучасних технологій. Немає відомих успішних атак, що значно послабляють шифр. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу NESSIE і Японським CRYPTREC проект. Японський шифр має рівні безпеки й можливості обробки, порівнянні із шифром AES/Rijndael.

Camellia – це блоковий шифр, який може бути повністю визначені мінімальними системами багатомірних багаточленів:

– Камелія (а також AES) S-блоки можуть бути описані системою 23 квадратних рівнянь в 80 членах.

– Розклад ключів можна описати 1120 рівняннями в 768 змінні з використанням 3328 лінійних і квадратичних членів.

– Увесь блоковий шифр можна описати 5104 рівняннями в 2816 змінні з використанням 14 592 лінійних і квадратичних членів.

– Усього потрібно 6224 рівняння з 3584 змінними з використанням 17 920 лінійних і квадратичних членів.

– Кількість вільних членів становить 11 696, що приблизно таке ж число, що й для AES.

Теоретично, такі властивості можуть дозволити зламати Camellia (і AES) за допомогою алгебраїчної атаки, такий як розширена розріджена лінеаризація, у т Майбутнє за умови, що атака стане можливою.

Хоча Camellia запатентована, вона доступна за безоплатною ліцензією. Це дозволило шифру Camellia стати частиною проекту OpenSSL під ліцензією з відкритим вихідним кодом з листопада 2006 року. Це також дозволило йому стати частиною Mozilla Модуль NSS (Служби мережної безпеки).

Підтримка Camellia була додана в остаточний випуск Mozilla Firefox 3 в 2008 році (за замовчуванням відключене починаючи з Firefox 33 в 2014 році в

дусі «Пропозиції по зміні стандартних наборів шифрів TLS, пропонованих браузерами», який був виключено з версії 37 в 2015 році). Pale Moon, відгалуження Mozilla / Firefox, продовжує пропонувати Camellia і розширив свою підтримку, включивши в нього набори Galois / Counter mode (GCM) із шифром, але вилучив GCM знову у випуску 27.2.0, пославшись на очевидну відсутність інтересу до них.

Пізніше, в 2008 році, група розробки релізу FreeBSD оголосила, що цей шифр також був включений в FreeBSD 6.4. Крім того, Йошисато Янагисава додав підтримку шифру Camellia у дисковий клас зберігання geli FreeBSD.

У вересні 2009 року GNU Privacy Guard додала підтримку Camellia у версії 1.4.10.

Veracrypt (відгалуження Truecrypt) включав Camellia як один з підтримуваних алгоритмів шифрування.

Крім того, різні популярні бібліотеки безпеки, такі як Crypto ++, Gnutls, mbed TLS і Openssl також включають підтримку Camellia.

26 березня 2013 р. було оголошено, що Camellia була знову обрана для включення в новий список рекомендованих шифрів для електронного уряду Японії як єдиний 128-бітний алгоритм блокового шифрування, розроблений у Японії. Це збігається з тим, що список CRYPTREC обновляється вперше за 10 років. Вибір був заснований на високій репутації Camellia у плані простоти придбання, а також характеристик безпеки й продуктивності, порівнянних з такими з Advanced Encryption Standard (AES). Камелія залишається незмінною у своєму повному втіленні. Неможлива диференціальна атака на Camellia з 12 раундами без шарів FL / FL дійсно існує.

Продуктивність

S-блоки, використовувані Camellia, мають структуру, аналогічну S-блоку AES. У результаті можна прискорити реалізацію програмного забезпечення Camellia за допомогою наборів команд ЦП, розроблених для AES, таких як x86 AES-NI.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблене програмне забезпечення реалізує систему управління електронним документообігом.

Програмно-апаратні вимоги:

– Загальний обсяг ОЗП: 512 Мбайт.
– Жорсткий диск не менше 10 Гбайт.
– Операційна система Microsoft Windows 10/11.

Початок роботи

При запуску програми система відображає вікно для введення даних (рисунок 5.1) облікового запису користувача – логін і пароль.

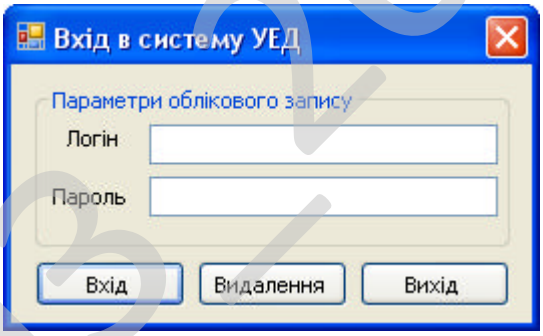

Рисунок 5.1 – Стартове вікно програми

Створення облікового запису в режимі «Адміністратор»

Створення облікового запису для адміністратора відбувається автоматично при першому запуску програми. При цьому запам'ятовується введений пароль, що буде використовуватися при подальшій роботі в режимі «Адміністратор».

Вхід у систему в режимі «Адміністратор»

Для входу в систему в режимі адміністратора поле для введення логіна повинне містити значення «Workers» (регістр не має значення). Поле «Пароль»

повинне містити пароль для входу під обліковим записом адміністратора. При невірному значенні пароля видається відповідне повідомлення.

Створення облікового запису режиму «Користувач»

Для того, щоб створити новий обліковий запис користувача в поле «Логін» (вікно на рисунку 5.1) необхідно ввести бажане ім'я облікового запису й пароль у поле «Пароль», що після цього буде використовуватися для входу. Ім'я може складатися із символів російського, українського або англійського алфавітів, а так само може містити символ нижнього підкреслення «_». Ім'я користувача не може бути «Worker», так це значення зарезервоване для облікового запису адміністратора. Якщо введене ім'я не відповідає перерахованим вимогам, то буде видане повідомлення про помилку. Після цього необхідно нажати кнопку «Вхід». Системи видасть повідомлення із запитом на підтвердження створення нового облікового запису, необхідно відповісти «Так».

Вхід у систему в режимі «Користувач»

Для входу в систему в режимі користувача необхідно в стартовому вікні програми (рисунок 5.1) увести логін і пароль у відповідні поля, а потім нажати кнопку «Вхід». Якщо введений пароль не збігається з паролем, введеним під час реєстрації, то буде видано відповідне повідомлення про помилку.

Видалення облікового запису режиму «Користувач»

Для видалення облікового запису режиму користувача необхідно в стартовому вікні програми (рисунок 5.1) увести у відповідні текстові поля логін і пароль користувача, чий обліковий запис повинна бути вилучена. Після цього необхідно нажати кнопку «Видалення». Якщо обліковий запис буде знайдений і пароль буде вірний, то видалення буде зроблено. У противному випадку буде видане повідомлення про відповідну помилку: обліковий запис із таким ім'ям користувача не знайдений або пароль не збігається.

Видалення облікового запису режиму «Адміністратора»

Виконання даної операції неможливо. При спробі видалити обліковий запис адміністратора буде видане повідомлення про помилку.

Робота із програмою в режимі «Адміністратор»

При вході в систему в режимі адміністратора відображається вікно, наведене на рисунку 5.2.

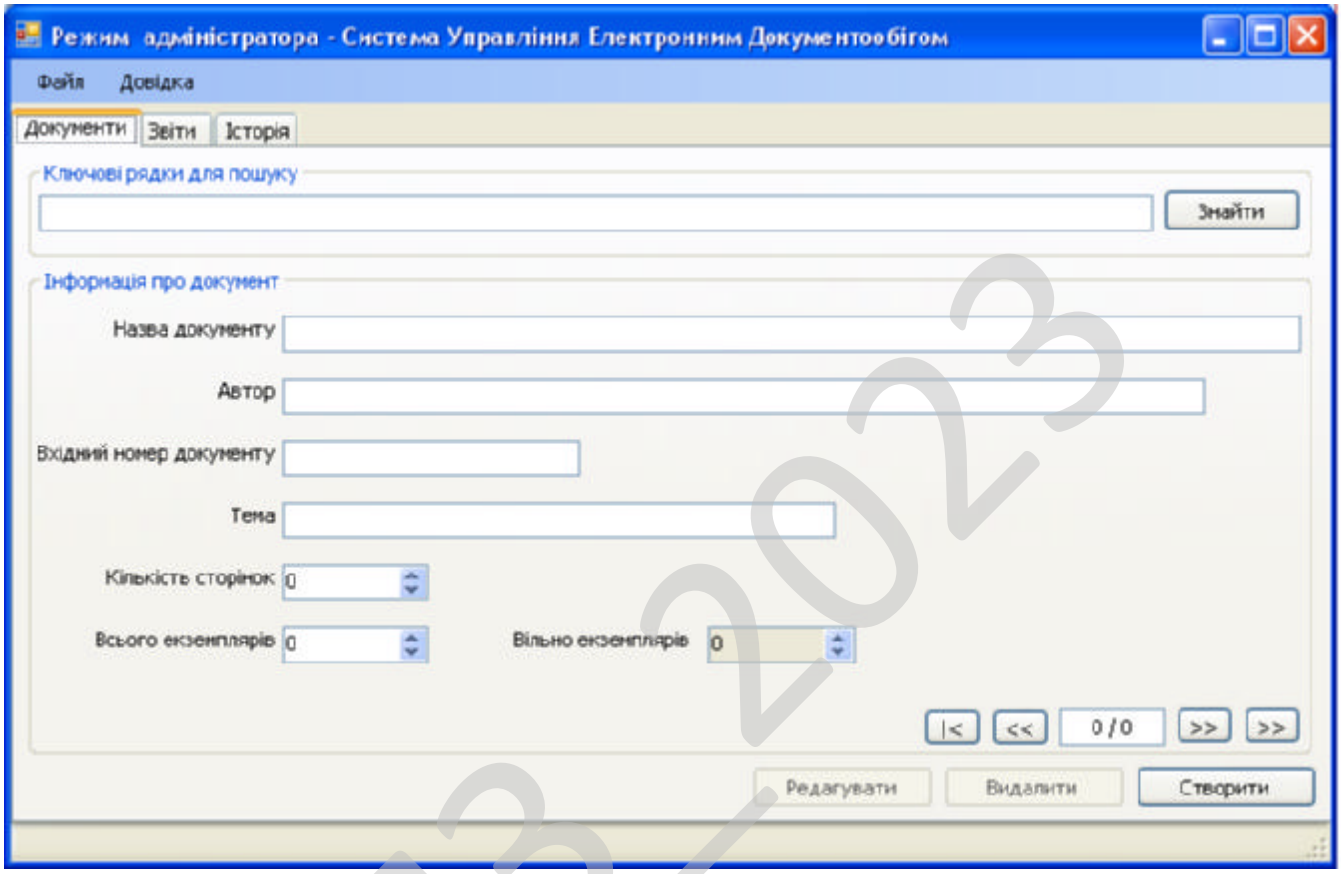

Рисунок 5.2 – Програма в режимі «Адміністратор»

Редагування списку документів і пошук

Додавання нового документа

Для додавання нового документа необхідно нажати кнопку перейти на вкладку «Документи» (рисунок 5.2) і нажати кнопку «Створити». Всі текстові поля будуть очищені для введення інформації про нову книгу. Поля «Назва документа» і «Вхідний номер документа» є обов'язковими для заповнення. Інші поля можуть бути заповнені пізніше. Для завершення операції необхідно повторно нажати кнопку «Зберегти» (кнопка «Створити» міняє напис). При додаванні нового документа система перевіряє значення вхідного номера

документа, якщо документ із таким же значенням цього параметра знайдений, то видається повідомлення про те, що такий вже є в системі.

Видалення документа

Для видалення існуючого документа необхідно перейти на вкладку «Документи», вибрати відповідний документ і нажати кнопку «Видалити». Обов'язковою умовою для видалення документу є наявність всіх його екземплярів у системі (поля «Усього екземплярів» і «Вільно екземплярів» мають однакові значення), якщо це не так, то буде видано відповідне повідомлення про помилку.

Редагування опису документа

Для редагування опису документа необхідно перейти на вкладку «Документи», вибрати необхідний документ і нажати кнопку «Редагувати». Відредагувати необхідні поля, а потім нажати кнопку «Зберегти» (кнопка «Редагувати» міняє напис). Система перевіряє наявність значення поля «Назва документа», і, якщо воно заповнено, то застосовує зроблені зміни. У противному випадком видається повідомлення про помилку.

Навігація

Для перегляду списку документів необхідно вибрати вкладку «Документи». Навігація за списком здійснюється в такий спосіб:

– Натискання кнопки «|<<» приводить до зсуву поточної позиції на перший елемент.

– Натискання кнопки «<<» приводить до зсуву поточної позиції на попередній елемент.

– Натискання кнопки «>>» приводить до зсуву поточної позиції на наступний елемент.

– Натискання кнопка «>>|» приводить до зміни поточної позиції на останній елемент.

Поточна позиція відображається у вікні між кнопками «<<» і «>>». За допомогою цього вікна теж може виробляється позиціонування, для цього досить

ввести номер необхідного елемента в це вікно й нажати кнопку «Enter».

Пошук документів

Пошук документів здійснюється за допомогою кнопки «Знайти» і розташованого поруч із ним текстового вікна. У текстове вікно вводиться значення для пошуку. Це значення шукається в полях «Назва документа», «Автор», «Вхідний номер документа» і «Тема». Якщо шуканий рядок знайдений хоча б в одному з перерахованих полів, то документ додається в результуючий список. Для того, щоб вивести весь список літератури необхідно здійснити пошук порожнього рядка. Запуск пошуку здійснюється натисканням кнопки «Знайти».

Звіти

Для роботи зі звітами необхідно вибрати вкладку «Звіти». Вид вікна наведений на рисунку 5.3.

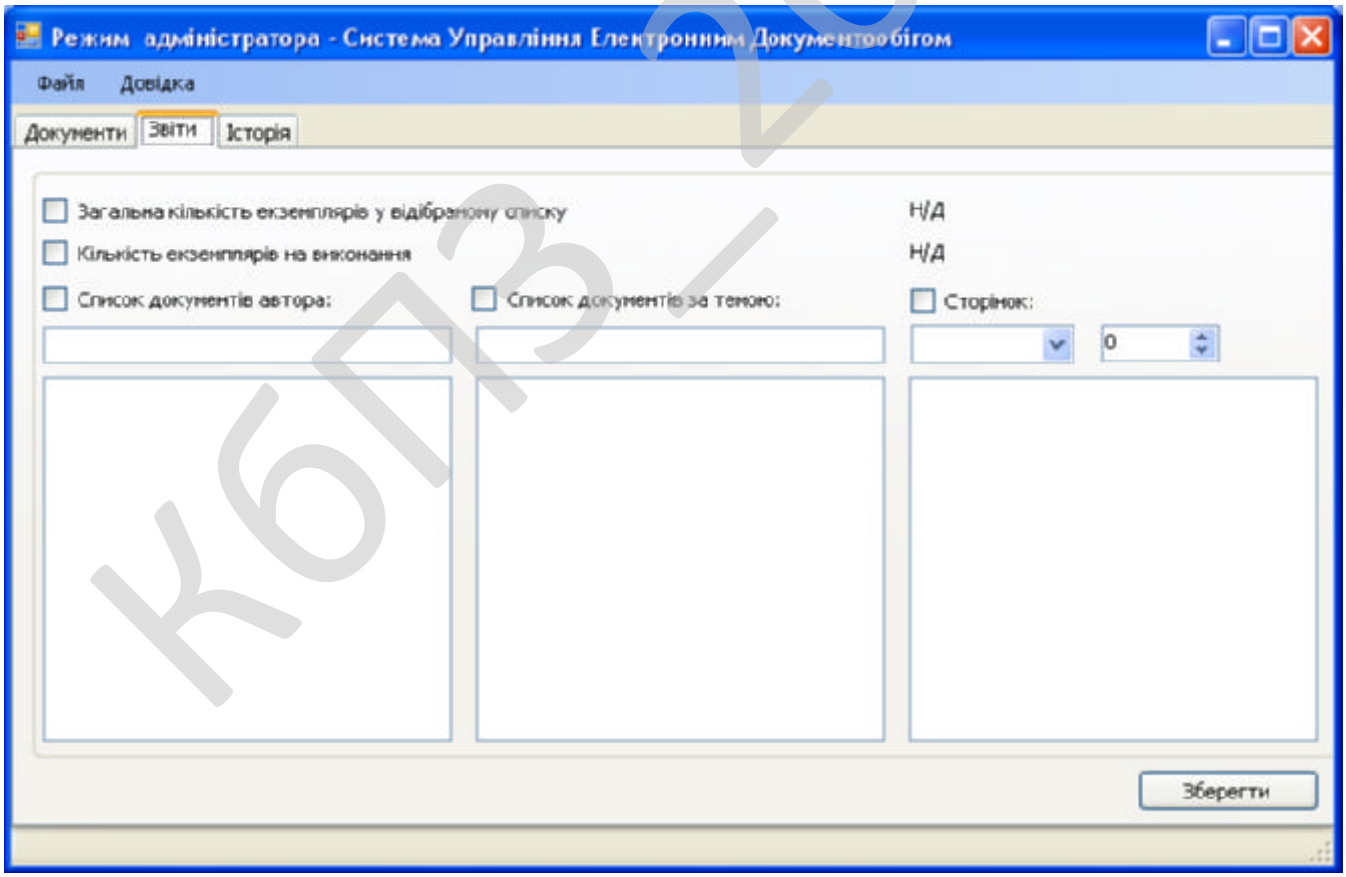

Рисунок 5.3 – Вікно перегляду статистики

Збір даних здійснюється в тому відібраному списку документів. Для того, що б почати збір даних необхідно нажати кнопку «Обновити». Параметри, по яких здійснюється збір даних, наведені нижче.

Загальна кількість і кількість вільних екземплярів

Для одержання загальної кількості й кількості вільних екземплярів необхідно встановити прапори напроти написів «Загальна кількість екземплярів у відібраному списку» і «Вільна кількість екземплярів у відібраному списку» відповідно.

Список документів автора

Є можливість виводу списку документів заданого автора. Для цього необхідно встановити прапор напроти напису «Список документів автора» і в розташоване поруч поле ввести ім'я необхідного автора (або будь-яку частину рядка, що може втримуватися в поле документи «Автор»). Регістр рядка не враховується.

Список документів по темі

Є можливість виводу списку документів по заданій темі. Для цього необхідно встановити прапор напроти напису «Список документів по темі» і в розташоване поруч поле ввести тему, що цікавить (або будь-яку частину рядка, що може втримуватися в поле документи «Теми»). Регістр рядка не враховується.

Кількість сторінок

Є можливість виводу списку документів по кількості сторінок. Для цього необхідно встановити прапор напроти напису «Список документів автора», у розташованому праворуч списку, що випадає, вибрати операцію порівняння:

– «Менш», ніж задана кількість сторінок

– «Більш», ніж задана кількість сторінок

– «Дорівнює» заданій кількості сторінок

У текстове поле, розташоване праворуч списку, що випадає, уводиться кількість сторінок.

Історія

Для перегляду історії необхідно вибрати вкладку «Історія». Вид вікна наведений на рисунку 5.4.

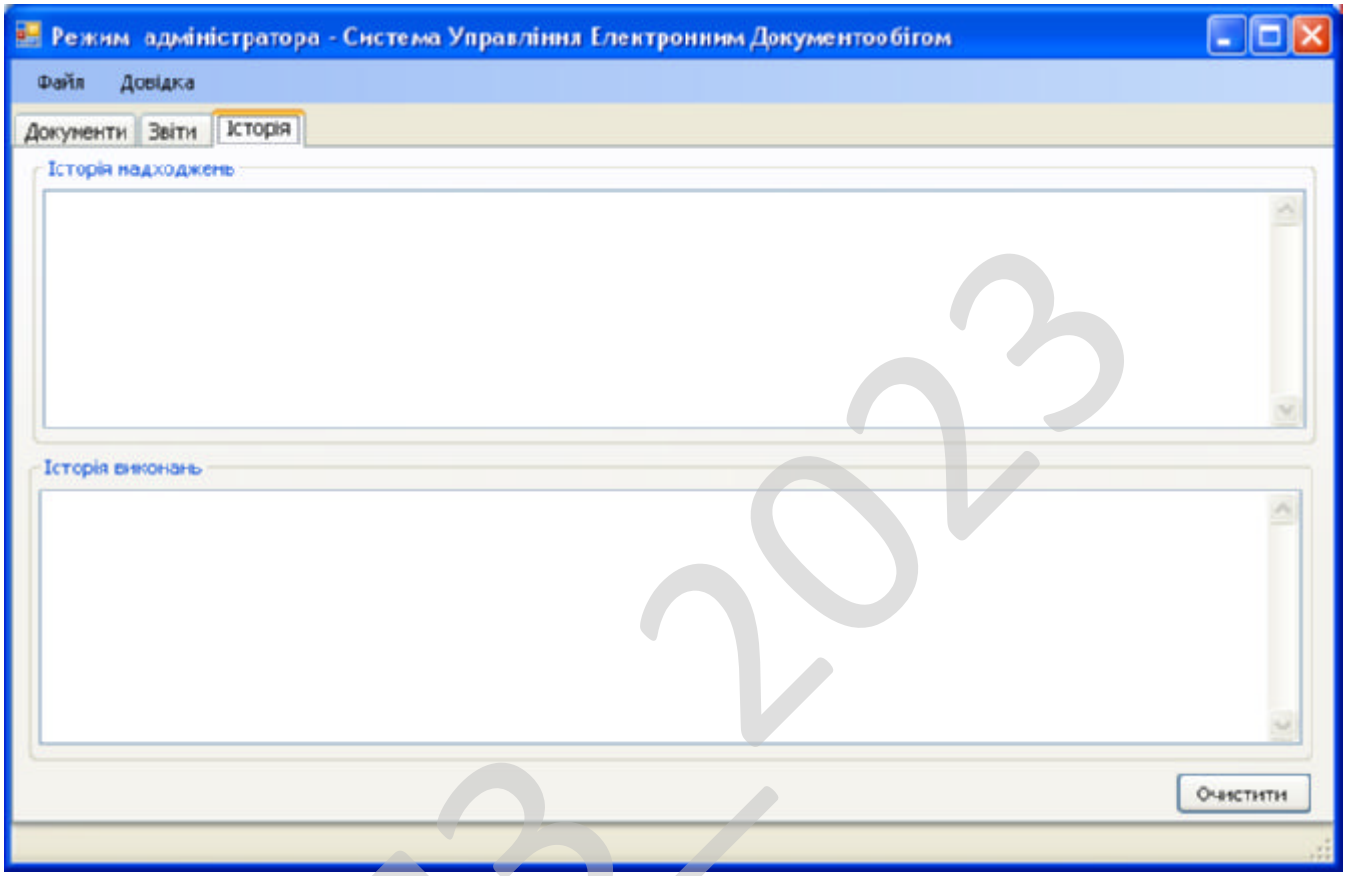

Рисунок 5.4 – Вікно історії надходжень і списань

Історія надходжень відображається в текстовому вікні, розташованому в рамці «Історія надходжень», історія списань – в «Історія списань».

Очищення історії

Є можливість очищення історії. Для цього необхідно нажати кнопку «Очистити».

Робота із програмою в режимі «Користувач»

При вході в систему в режимі користувача відображається вікно, наведене на рисунку 5.5.

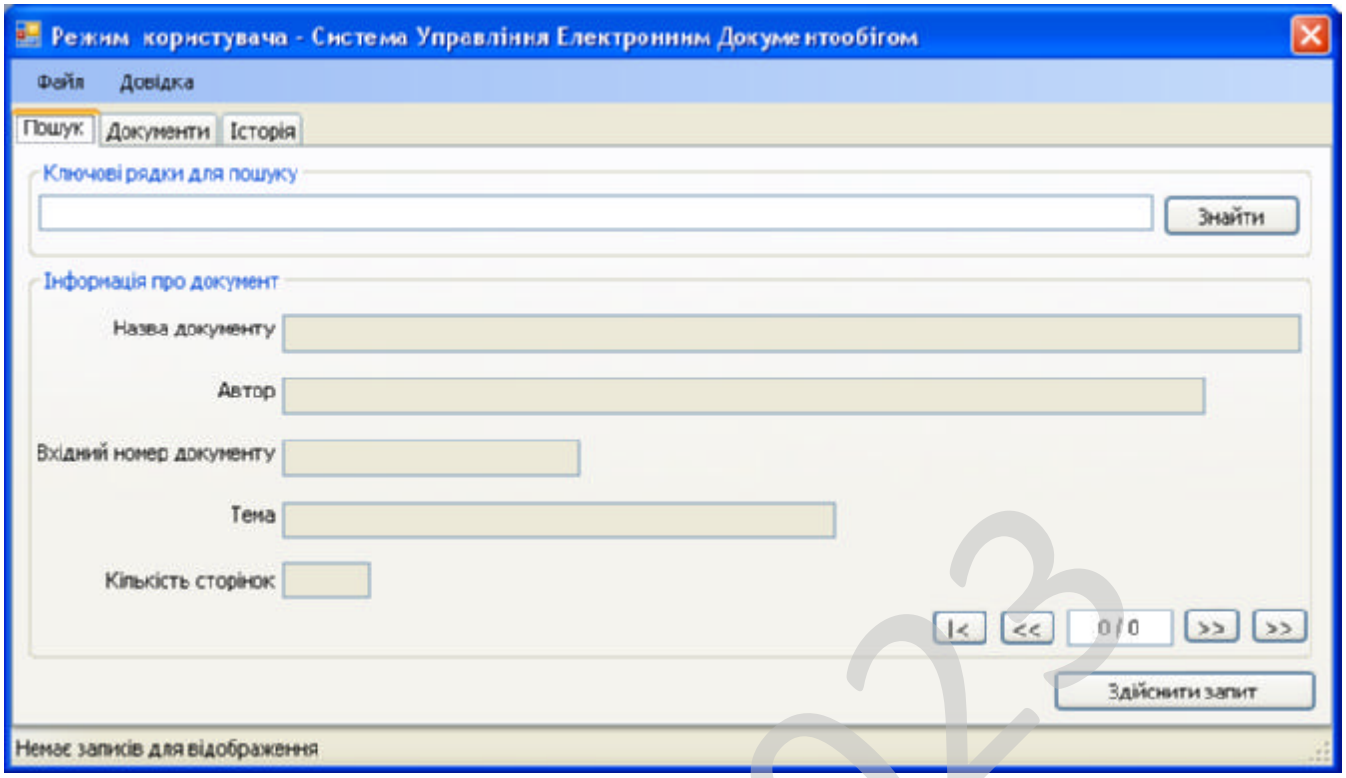

Рисунок 5.5 – Пошук літератури

Робота зі списком документів системи

Пошук документів

Пошук документів здійснюється за допомогою кнопки «Знайти» і розташованого поруч із ним текстового вікна. У текстове вікно вводиться значення для пошуку. Це значення шукається в полях «Назва документа», «Автор», «Вхідний номер документа» і «Тема». Якщо шуканий рядок знайдений хоча б в одному з перерахованих полів, то документ додається в результуючий список. Для того, щоб вивести весь список літератури необхідно здійснити пошук порожнього рядка. Запуск пошуку здійснюється натисканням кнопки «Знайти».

Запит документа

Для того, щоб взяти документ, що цікавить, із системи необхідно нажати кнопку «Запросити». Якщо вільні екземпляри документи є в системі, то обраний документ додається в список документів користувача. Одночасно доданий документ убирається зі списку документів, що переглядаються.

Навігація

Для перегляду списку документів необхідно вибрати вкладку «Пошук». Навігація за списком здійснюється в такий спосіб:

– Натискання кнопки «|<<» приводить до зсуву поточної позиції на перший елемент.

– Натискання кнопки «<<» приводить до зсуву поточної позиції на попередній елемент.

– Натискання кнопки «>>» приводить до зсуву поточної позиції на наступний елемент.

– Натискання кнопка «>>|» приводить до зміни поточної позиції на останній елемент.

Поточна позиція відображається у вікні між кнопками «<<» і «>>». За допомогою цього вікна теж може виробляється позиціонування, для це досить увести номер необхідного елемента в це вікно й нажати кнопку «Enter».

Робота зі списком документів користувача

Для перегляду списку документів користувача необхідно вибрати вкладку «Мої документи». Приклад виду вікна наведений на рисунку 5.7.

Повернення документа

Для повернення непотрібного документа в систему необхідно вибрати його зі списку, а потім нажати кнопку «Повернути». Повернутий документ буде вилучена зі списку.

Перегляд історії

Для перегляду історії користування документом необхідно вибрати вкладку «Історія». Приклад виду вікна наведений на рисунку 5.8.

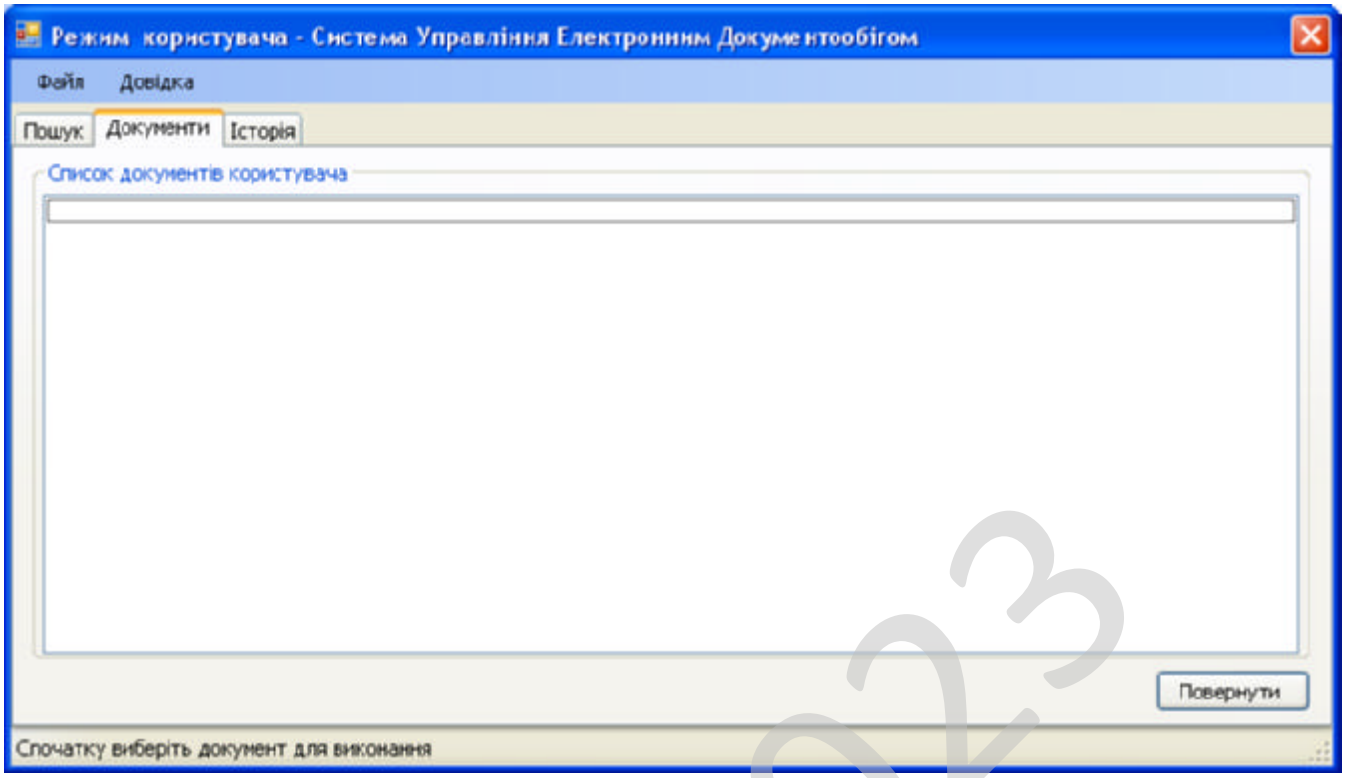

Рисунок 5.7 – Робота зі списком документів користувача

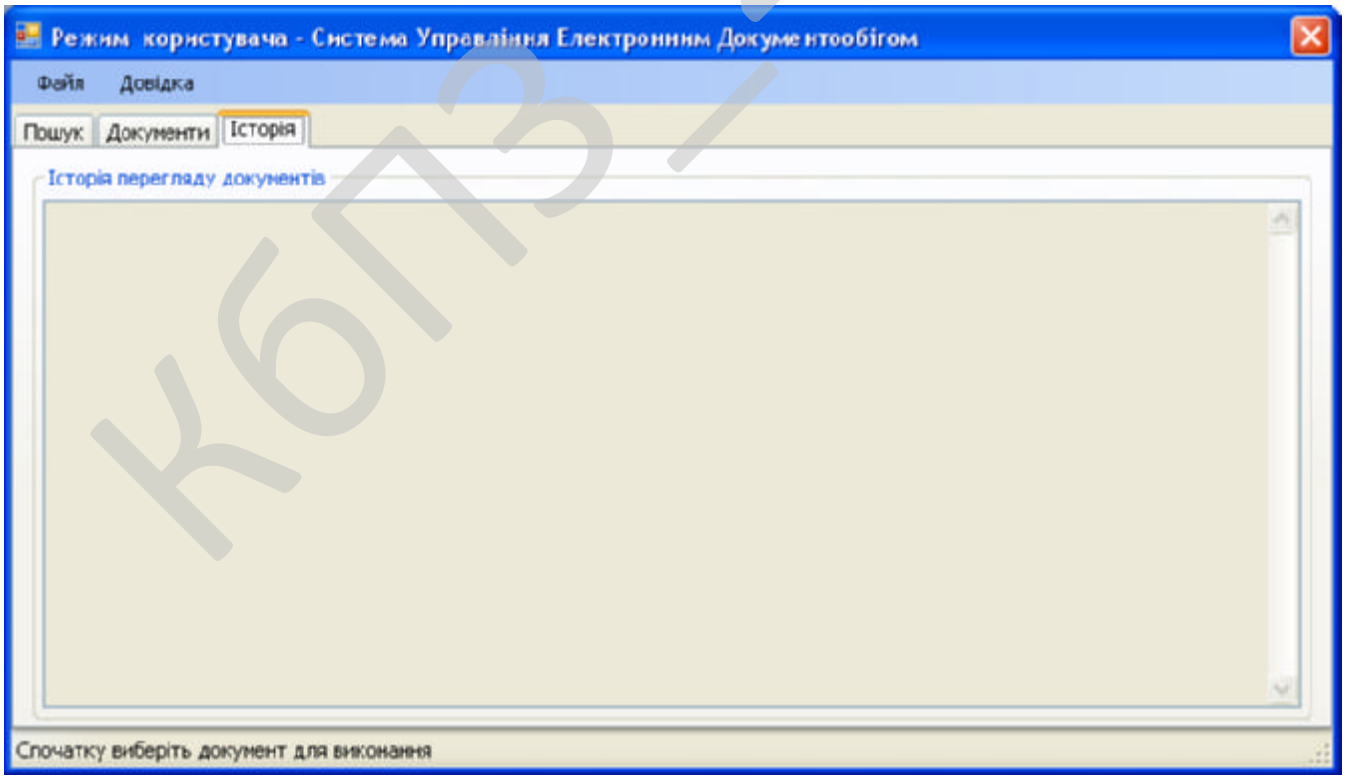

Рисунок 5.8 – Історія користування документом

Завершення роботи із програмою

Збереження результатів роботи

Для збереження результатів роботи програми необхідно вибрати пункт меню «Файл Зберегти». Тільки після цього вся пророблена робота буде збережена у файл на диску.

Вихід із програми

Для виходу із програми необхідно вибрати пункт меню «Файл – Вийти». Всі незбережені зміни будуть загублені.

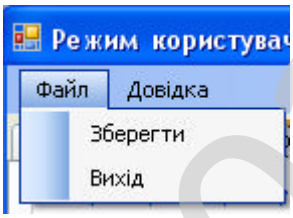

Рисунок 5.9 – Основне меню програми

На рисунку 5.10 наведене вікно з даними про розроблювача програми, наукового керівника й місце виконання магістерської роботи.

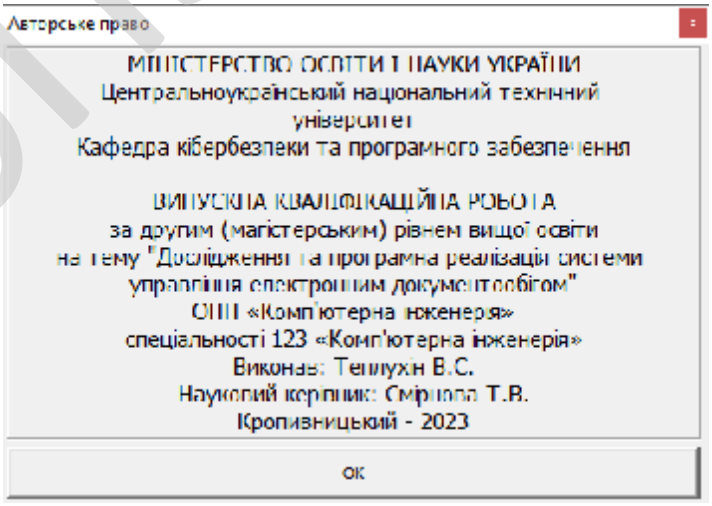

Рисунок 5.10 – Довідка

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи управління електронним документообігом.

Метою розробки є дослідження та програмна реалізація системи управління електронним документообігом.

Об'єктом дослідження є процес управління електронним документообігом.

Предметом дослідження є методи управління електронним документообігом.

Методи дослідження базуються на методах цифровізації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод управління електронним документообігом.

– Розроблено вітчизняний продукт управління електронним документообігом, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 48 днів (два місяці).

В магістерській роботі було проведено дослідження та виконана програмна реалізація системи управління електронним документообігом.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп’ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	17
3. Запланований термін розробки, днів	Fpq	48 (2 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	7
8. Кількість форм вихідної інформації.	–	8
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	2
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	1
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	3
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	1
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	1
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	1
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	3
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	3
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	1
29. Досвід роботи з програмними інструментами розробки (1-6)	–	2
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	3
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	17000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(3{,}24+3{,}64+3{,}38+4{,}94+2{,}73) = 1{,}028.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}028} = 6{,}8 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}8 \cdot (0{,}88 \cdot 1 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}89 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1 \cdot 1{,}1 \cdot 1 \cdot$$

$$\cdot 1{,}12 \cdot 1{,}22 \cdot 1{,}12 \cdot 1 \cdot 1{,}1) = 8{,}38 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4). S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%. Приймаємо S = 120 %

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 8{,}38^{0{,}33+0{,}2(1{,}028-1{,}01)} \cdot 67 = 109 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	15	Д7
Робочий проект	109	Ф 7.1-7.4
Впровадження	17	Д13
Всього	160	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{160 \cdot 1}{48\text{-}5} = 3,75 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	7	630	10,5
Монітор	60	7	420	7
Клавіатура	30	7	210	3,5
Маніпулятор «мишка»	30	7	210	3,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	3	90	1,5
Кабельні господарства ЛОМ на 1 м.п.	2,5	200	500	8,33
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	41,99

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф^{с}_{др} = \frac{З_ч \cdot n_{міс}}{1,2}, \quad (7.6)$$

$$Ф^{с}_{др} = \frac{42 \cdot 2}{1,2} = 70 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$$Ч_{ел} = 70/(48 \cdot 8) = 0{,}2 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	18293	36586
Продакт-менеджер	0,25	14000	7000
Інженер-програміст	3,75	18000	135000
Інженер-електронщик	0,2	14000	5600
Інженер-системотехнік	0,25	14000	7000
Адміністратор мережі	0,5	14000	14000
Системний програміст	0,25	14000	7000
Дизайнер WEB	0,25	14000	7000
Інженер-верстальник	0,25	14000	7000
Бухгалтер-економіст	0,25	14000	7000
Всього за період розробки	$R_{сп}$ = 6,95	-	$Ф_{роб}$ = 233186

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{233186}{6,95 \cdot 48} = 699 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 400…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 37 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 9 \cdot 8 \cdot 20000 = 1440000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 144000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$I_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$I_{нв} = 9 \cdot 3500 = 31500 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми supercomp.kiev.ua за 29.10.23 – джерело http: //supercomp.kiev.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		12721
Системний блок		7721
Процесор	AMD A8-9600 (AD9600AGM44AB) AM4, 4 ядра, 4 потоки, 3.1 GHz, 3.4 GHz, TDP – 65 Вт, 28nm	
Системна плата	ASRock A520M-HDV сокет – AM4, DDR4, 64 ГБ, 4733 MHz, LAN – 1 Гбіт/с, D-Sub (VGA), DVI, HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s, Micro-ATX	
Відеокарта	Інтегрована AMD Radeon R7	
Жорсткий диск	SSD M.2 2280 240GB Apacer (AP240GAS2280P4-1) 240 GB, 3D TLC, M.2, PCI Express 3.0 x4	
Оперативна пам’ять	DDR4 8GB 2400 MHz Patriot (PSD48G240081) DDR4, 8 ГБ, В наборі – 1	
Система охолодження	Vinga CL3011 Для процесорів – INTEL, AMD, сокет – 1151, AM4, 1150, 1155, AM2, AM3, AM3+, 775, 1156,	
Корпус	AeroCool Talon-G-BK-v1 Black (ACCM-PV43013.11), Miditower, ATX, Micro – ATX, Mini – ITX, 1 x 120 мм, Front 2x12 см	
Блок живлення	Casecom 450W (CM 450 ATX) ATX, 450 Вт, 24 pin, CPU – 4pin, SATA – 2, Peripheral – 4, +12V1 – 18A, 1x120 мм	220
інше	Клавіатура, мишка	-

Продовження таблиці 7.5

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	Монітор BenQ GL2450HM Black	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37	2800
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	Powercom BNT-600AP USB	1400

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межа до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складем таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	9	12721	11448,9	125937,9
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2800	280	3080
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	147569,4

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1440000	-	-
2. Передавальні пристрої	144000	-	-
Всього по групі	1584000	5	79200
Група 4			
3. Обчислювальна техніка	147570	-	-
Всього по групі	147570	40	59028
Нематеріальні активи			
4. Нематеріальні активи	17000	10	1700
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	121875	20	24375
7. Господарський інвентар	31500	25	7875
Всього по групі	162406	-	34507,75
Разом	$К_р$ = 1910976		$А_р$ = 174435,75

Примітка: вартість автомобіля ГАЗ Газель взята по даним електронного ресурсу http://www.avtopoisk.ua, що складає 121875 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 699 \cdot 160/17 = 6579 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 6579 \cdot 10 \cdot 0{,}01 = 658 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(6579+658) = 2678 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 6579 \cdot 15 \cdot 0{,}01 = 987 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картриджів, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно виданих норм приймаємо 2/3 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п = 210$ грн., визначаємо вартість паперу за період розробки $N_м = 2$ міс:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 210 \cdot 2/3 \cdot 2 = 70 \text{ грн.}$$

Згідно виданих норм до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків в кількості одного диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 36 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 36 грн./шт.

$$З_{М2} = 36 \cdot 1 = 36 \text{ грн.}$$

Згідно виданих норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_{з.}, \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 500+508+1155 = 2163 \text{ грн.}$$

$$З_М = (70 + 36 + 2163)/17 = 133 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п = 15\%$) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 6579 \cdot 15 \cdot 0{,}01 = 987 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е = 17$ прим.):

$$A_м = \frac{A_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 174436 \cdot 2/(17 \cdot 12) = 1710 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	6579
2. Додаткова зарплата виконавців	$З_д$	658
3. Відрахування на соціальні потреби	$С_{оц}$	2678
4. Загальногосподарські витрати	$Г_{осп}$	987
5. Витрати на матеріали	$З_М$	133
6. Освоєння нових операційних систем, мов програмування	$О_п$	987
7. Амортизація основних фондів	$А_м$	1710
8. Повна собівартість програмного забезпечення	$С_п$	13732
9. Плановий прибуток	$П_р$	6866
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	20598
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	4119,6
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	24717,6

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$C_п = 6579+658+2678+987+133+987+1710 = 13732 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 13732 = 6866 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	24718
Всього капітальних витрат	–	24718

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	51533	17178
2. Витрати на електроенергію	$З_{ел}$	562	389
3. Витрати на амортизацію	$З_{ам}$	0	6180
Всього витрат за рік	$І$	52095	23747

Витрати на обслуговування системи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \quad (7.23)$$

де: $Т_р$ – кількість годин обслуговування за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 240 годин на рік до 80 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 240 \cdot 160 \cdot 1{,}1 \cdot 1{,}22 = 51533 \text{ грн},$$

до:

$$З_{р\,нов} = 80 \cdot 160 \cdot 1{,}1 \cdot 1{,}22 = 17178 \text{ грн}.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\,баз} = 9 \cdot 0{,}15 \cdot 130 \cdot 3{,}2 = 562 \text{ грн}.$$

$$З_{ел\,нов} = 9 \cdot 0{,}15 \cdot 90 \cdot 3{,}2 = 389 \text{ грн}.$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	24718	–	6179,5
Всього відрахувань	-	–	24718	–	6179,5

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.24)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (20598 - 13732) \cdot 17 - (0,05 \cdot 1584000 + 0,4 \cdot 147570 + \\ +0,2 \cdot 121875 + 0,25 \cdot 40531 + 0,1 \cdot 17000) \cdot 2/12 = 87649 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.25)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{326976}{(20598 - 13732) \cdot 17 \cdot 12 / 2} = 0{,}5 \text{ років}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	17
2. Повна собівартість розробленої програми	Грн.	13732
3. Ціна розробленої програми	Грн.	20589
4. Плановий прибуток від реалізації розробленої програми	Грн.	6857
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1910976
7. Загальний прибуток від реалізації програмної продукції	Грн.	116569
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	87649
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,5
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	24718
11.Величина економічного ефекту у користувача програмної продукції	Грн.	22169
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,87

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \qquad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (52095\text{-}23747)\text{-}0{,}25 \cdot 24718 = 22169 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \qquad (7.28)$$

$$T_{сп} = \frac{24718}{52095 - 23747} = 0{,}87 \text{ року}.$$

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням [1]. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98, та «Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста вулючають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення впливу комп'ютера на організм програміста визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста,

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальни машини (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом.

На робочому місці програміста виникають небезпечні та шкідливі фактори: підвищений рівень шуму, несприятливі мікрокліматичні умови, недостатній рівень освітленості, шкідливі речовини, підвищений рівень електромагнітних випромінювань радіочастот, висока напруга електричної мережі, статична електрика та інші.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м’язовий апарат.

Вивчення умов праці на робочому місці програміста є необхідною умовою запобігання впливу небезпечних та шкідливих факторів на організм персоналу.

Дослідження санітарно-гігієнічних умов у приміщенні звичайно проводять з використанням пристроїв вимірювання параметрів мікроклімату – вологи повітря (психрометр), освітленості (люксметр) та шуму (шумомір), а також відомих розрахункових методик.

8.3 Пожежна безпека

Пожежна безпека на підприємстві it-індустрії являє собою комплекс організаційних заходів щодо забезпечення нормального протипожежного стану об’єкта нерухомості – житлового, виробничого, складського, офісного або торгово-розважальної споруди, приміщення або комплексу приміщень.

Нормативною основою цього питання на підприємствах усіх форм власності є Кодекс цивільного захисту України та його 13-й розділ, а також Правила пожежної безпеки України. Згідно цих нормативно-правових актів керівник і власник підприємства, компанії або організації несе повну адміністративну та кримінальну відповідальність за своєчасне і правильне введення і підтримання протипожежного режиму. Згідно Державного стандарту України 2272:2006 «Пожежна безпека. Терміни та визначення основних понять» [8] під протипожежним режимом “розуміється звід правил поведінки людей, виконання робіт і експлуатації об’єкта нерухомості, призначений для забезпечення пожежної безпеки”.

Керівник в свою чергу може делегувати частину своїх обов'язків і повноважень спеціально призначеному наказом по підприємству співробітнику. У більшості випадків призначається одна відповідальна посадова особа по підприємству або компанії в цілому і окремо – в кожному з окремих будівель, філій, відділів, дивізійних і структурних підрозділів.

Призначений співробітник приступає до виконання своїх обов'язків на підставі наказу по підприємству. Відповідальну особу в обов'язковому порядку знайомлять з цим документом, який він запевняє своїм підписом. У кожному з окремих приміщень, що охороняються повинна бути розташована інформаційна бирка з прізвищем, ім'ям, по батькові та номером контактного телефону особи, відповідальної за протипожежну безпеку. Конкретний перелік обов'язків щодо забезпечення режиму на об'єкті, комплектації, збереження та експлуатації засобів та інструментів протипожежного захисту вказується в посадових інструкціях, положеннях і наказах.

Чинними нормами і законодавчими актами України не передбачений точний список вимог, що висуваються до відповідальної особи – вік, гендерна приналежність та інші фактори значення не мають. Найчастіше компанії та організації висувають на цю посаду висококваліфікованого фахівця з вищою або середньою технічною професійною освітою, що має безперервний стаж роботи в галузі діяльності підприємства не менше ніж три роки. Крім того, бажано, що б здобувач мав мінімальний досвід діяльності в сфері пожежної безпеки, деякі технічні вміння та навички, Досвід роботи з комп'ютером. Відповідальна особа повинна пройти навчання з питань пожежної безпеки з відривом від виробництва, а в подальшому проходити курси в навчальному центрі не рідше, ніж один раз на три роки.

До кола обов'язків особи, відповідальної за протипожежну безпеку в компанії, організації або на підприємстві входять:

– розробка і ведення документації – планів з евакуації персоналу в разі пожежі, ведення журналів обліку, наказів, і спеціальних інструкцій щодо захисту

об’єктів і використання первинних засобів гасіння, інструкцій з техніки безпеки, технічного обслуговування вогнегасників різних марок і типів, систем пожежної сигналізації та інше;

– організація навчання працівників, проведення інструктажів;

– матеріально-технічне забезпечення підприємства протипожежним інвентарем – первинними засобами пожежогасіння, включаючи переносні і пересувні вогнегасники, інформаційними стендами з актуальними матеріалами, пожежними щитами і знаками пожежної безпеки та іншими;

– забезпечення технічної справності протипожежного обладнання, інструментів та спеціального інвентарю;

– періодичні перевірки стану ввіреного об’єкта;

– здача звітності до контролюючих органів, допомога у здійсненні планових та позапланових перевірок;

– організація безпечної евакуації співробітників і персоналу, матеріальних цінностей, організація гасіння вогнищ загоряння або задимлення в разі виникнення пожежі.

– Крім евакуаційного плану в будівлях і приміщеннях з масовим скупченням людей повинні бути інструкції з евакуації. Плани та інструкції повинні бути розвішені в приміщеннях на добре видимих місцях. Не рідше, ніж один раз на календарний рік на підприємствах повинні проводитися тренування на випадок виникнення надзвичайних ситуацій.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

Таким чином можна припустити, що основною причиною можнивого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга).

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Для захисного штучного заземлення застосовуються вертикальні електроди: металевий куток 80·50·6 мм., (згідно з ДСТУ 8769:2018 «Кутики сталеві гарячекатані нерівнополичні. Сортамент») довжиною L=1,7 м., та горизонтальний електрод – метелева полоса з перетином 60·5 мм. Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – у ряд.

Розрахунок проведемо за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір ρ_2 = 40 Ом·м). Умовна товщина верхнього шару грунта: H=0,5 м. Відстань між вертикальними

заземлювачами (електродами) А=1,7 м. Глибина закладення горизонтального контура заземлення t=0,6 м. Опір заземлювача, який нормується: $R_{ЗН}$ = 4 Ом. Необхідно визначити необхідну кількість вертикальних заземлювачів та длину полоси (горизонтального заземлювача).

Розрахунок

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,6+1,7/2=1,45 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi \rho = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де ψ = 1,36 – табличне значення коефіцієнта сезонності для відповідної климатичной зони у багатошаровому грунті [10];

ρ_2= 40 Ом·м. – табличне значення питомого опору нижнього шару грунта (глина) [10].

Еквівалентний діаметр ветикальньного електрода (кутка) [10]:

$$Dв=0,95 \cdot К= 0,95 \cdot 65 = 59,85 \text{ мм.} = 0,62 \text{ м.}$$

де К= (80+50)/2=65 мм. – середній розмір полиці метелевого кутка.

Відношення А/L=1,7/1,7=1

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [10]:

$$R_O=0,366(\rho/L)[\lg(2L/Dв)+(1/2)\lg((4T+L)/(4T-L))]=$$

$$=0,366(54,5/1,7)[(\lg(2 \cdot 1,7/0,62)+(1/2)\lg((4 \cdot 1,45+1,7)/(4 \cdot 1,45-1,7))]=$$

$$=21,9 \text{ Ом.}$$

Визначаємо коефіцієнт екранування вертикальних електродів Кев=0,8 при орієтовній кількості вертикальних електродів, яке дорівнює 5 [10].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без врахування горизонтального заземлювача), при $R_{ЗН}$ = 4 Ом:

$$N=R_O /(Кев \, R_{ЗН})= 21,9 / (0,8 \cdot 4)= 6,85 \approx 7 \text{ шт.}$$

Визначаємо длину з’єднуючої полоси:

$$L_П = 1{,}05 \cdot A \cdot N = 1{,}05 \cdot 1{,}7 \cdot 7 = 12{,}22 \approx 12 \text{ м.}$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [10]:

$$R_П = 0{,}366(\rho \cdot K_П / L_П) \lg(2 \cdot L_П^2 / (B \cdot t)) =$$

$$= 0{,}366(40 \cdot 5/12) \cdot \lg((2 \cdot 12^2)/(0{,}06 \cdot 0{,}6)) = 23{,}5 \text{ Ом.}$$

де $K_П$=5 – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідной климатичной зони для з'єднуючої полоси [10]:

B = 60 мм. = 0,06 м.- ширина з'єднуючої полоси (задана).

Загальньний опір розтіканню електричного струму заземлювача [10]:

$$R = (R_O \cdot R_П)/(R_O \cdot \eta_П + N \cdot R_П \cdot Кев) =$$

$$= (21{,}9 \cdot 23{,}5)/(21{,}9 \cdot 0{,}75 + 9 \cdot 23{,}5 \cdot 0{,}8) = 3{,}48 \text{ Ом.}$$

де $\eta_П$ = 0,75 – табличне значення коефіцієнта екранування з'єднуючої полоси [10].

Умова $R \le R_{ЗН}$ виконується (3,48 ≤ 4).

Так як при 7 вертикальних електродах R суттево менше $R_{ЗН}$, зменшимо кількість вертикальних електродів N до 6 і виконаємо перерахунок. У результаті остаточно отримали: R =3,98 Ом. при кількості вертикальних електродів N=6.

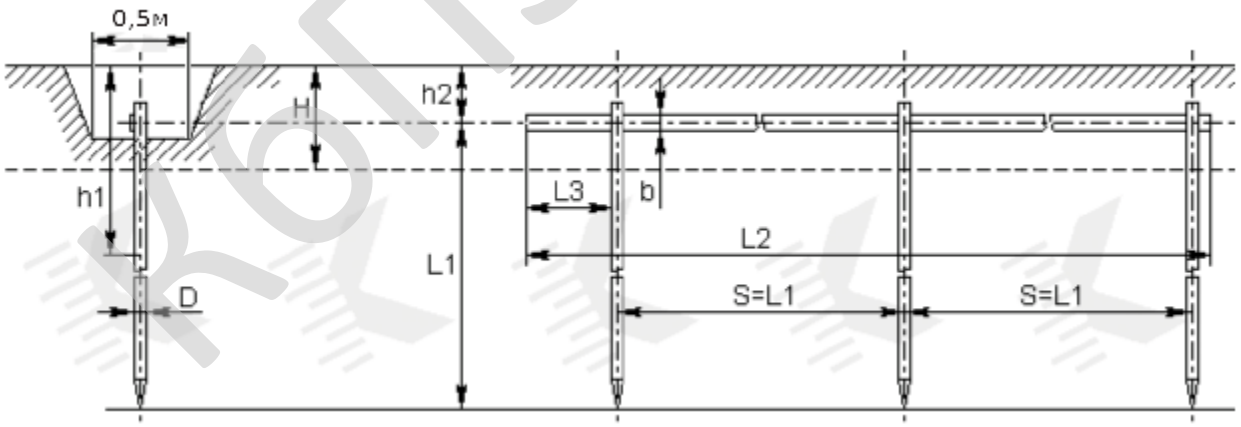

Рисунок 8.1 – Схема штучного заземлення

Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз умов праці на робочому місці програміста, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи, розроблені заходи з умов поліпшення охорони прац.

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

Список використаних джерел інформації

1. Державні будівельні норми України: ДБН В.2.5-28:2018. – Режим доступу до ресурсу: https://goo.su/9AkQ (дата звернення 19.10.23).

2. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98 (дата звернення 19.10.23).

3. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення 19.10.23).

4. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

5. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508 (дата звернення 19.10.23).

6. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення 19.09.23).

7. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.23).

8. Оришака О.В. Охорона праці в галузі та цивільний захист / О.В Оришака, Г.П. Горбачова, О.М. Мезенцева , К.М. Марченко, К.О. Буравченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький: Видавець Лисенко В.Ф., 2019. − 226 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/9258 (дата звернення 19.09.23).

9. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти для здобувачів вищої освіти спеціальностей 123 "Комп'ютерна інженерія" та 122 "Комп'ютерні науки" / М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програм. забезпечення; [укл. О.В. Оришака, К.М. Марченко]. – Кропивницький: ЦНТУ, 2022. – 19 с. [Електронний ресурс]. – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12240 (дата звернення 19.09.23).

10. Охорона праці. Ч. 1. Захисне заземлення : метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд. ; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград : КІСМ, 1997. – 20 с. – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358 (дата звернення 19.09.23).

11. Наказ Міністерства регіонального розвитку, будівництва та житлово-комунального господарства України 31.10.2016 «Про затвердження ДБН В.1.1-702016» – Режим доступу до ресурсу: ttps://ips.ligazakon.net/document/fn025551 (дата звернення 19.09.23).

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи управління електронним документообігом.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів управління електронним документообігом.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем управління електронним документообігом.

– Досліджена система управління електронним документообігом.

– На основі отриманих результатів досліджень створена програмна реалізація системи управління електронним документообігом.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання управління електронним документообігом.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Visual C++. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Camellia.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 22169 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,87 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Теплухін В.С. Дослідження та програмна реалізація системи управління електронним документообігом // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.
2. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.
3. Aaron Torres. Go Programming Cookbook Second Edition. Packt Publishing Ltd. 2019. 427 p.
4. Мелешко Є.В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: Видавець – Лисенко В.Ф., 2019. – 156 с.
5. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.
6. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.
7. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.
8. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.
9. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.
10. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

11. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w.

12. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143

13. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». Communications in Computer and Information Science, 2021, vol 1486. Springer, Cham. pp 169-184.

14. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

15. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

16. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

17. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». International Journal of Computer Network and Information Security (IJCNIS). Vol. 12, No. 3, 2020. PP.33-43.

18. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated

with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

19. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

20. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

21. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

22. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

23. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P. 707-712.

24. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P.701-706.

25. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation

Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

26. Вінтенко Б.Ю., Смірнов О.А., Коваленко А.С., Смірнов С.А., Буравченко К.О. «Дослідження вимог міжнародних стандартів IEC60880 та IEC62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 3(73), С. 155-166.

27. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 2(72), С. 170-178.

28. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». Сучасні інформаційні системи, 2023, том 7, № 2, С. 49-56.

29. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А. «Дослідження нормативної документації та стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». VI міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 20-21 квітня 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 35-36.

30. Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.

31. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у

комп'ютерних системах операторів стільникового зв'язку» Вісник Черкаського державного технологічного університету. Технічні науки. №4. С. 103-110. 2020.

32. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

33. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

34. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

35. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

36. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

37. O. Smirnov, O. Kovalenko, A. Kovalenko, S. Smirnov, V. Vialkova. The mathematical model of the testing technology for DOM XSS vulnerabilities. Scientific & practical cyber security journal (SPCSJ) Vol 2 Issue 1, 22-28 pp. [Электронный Журнал]. Georgia. Tbilisi: SCSA – 2018.

38. Oleksii Smirnov, Oleksandr Kovalenko, Jamil Al-Azzeh, Anna Kovalenko, Serhii Smirnov. Qualitative risk analysis of software development. Asian Journal of Information Technology. – Volume 17(3). – Medwell Journals. – 2018. – P. 218-230.

39. Смірнов О.А., Коваленко О.В., Коваленко А.С., Смірнов С.А. Розробка методу передтестової компіляції й розподілу доступу. Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”, м. Кропивницький. 19-20 квітня 2018р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215

40. Smirnov Oleksii, Kovalenko Oleksandr, Kovalenko Anna, Smirnov Serhii. Method of testing the DOM XSS vulnerability. International Conference «Information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces “Alexandru cel Bun”. 2017. P7.

41. Смірнов О.А., Смірнов С.А., Коваленко О.В., Коваленко А.С. Технологія тестування DOM XSS уразливості. Науково-практичний журнал кібер безпеки (SPCSJ) № 1. [Електронний журнал]. Грузія. Тбілісі: SCSA – 2017.

42. Смірнов О.А., Лисенко І.А. Інформаційна технологія проектування тестових наборів з урахуванням вимог до програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 4 (44). – Полтава: ПолтНТУ. – 2017. – С. 112-115.

43. Смірнов О.А., Смірнов С.А., Рябой Д.К., Рябая О.В. Модель вузла комутації з відносними пріоритетами, резервуванням ресурсів і обліком реальної надійності обслуговуючих приладів .Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп’ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. – Кропивницький: ЦНТУ. – 2017. – С. 198-199.

44. Смірнов О.А., Коваленко О.В. Використання псевдобулевих методів бівалентного програмування для управління ризиками розробки програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 1 (37). – Полтава: ПолтНТУ. – 2016. – С. 98-103.

45. Смірнов О.А., Лисенко І.А. Формалізація процесу проектування тестових наборів. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 3 (48). – Харків: ХУПС. – 2016. – С.96-100.

46. Смірнов О.А., Лисенко І.А. Удосконалення методу перевірки коректності таблиць рішень для подання тестових наборів. Збірник наукових праць "Системи обробки інформації". – Випуск 8 (145). – Х.: ХУПС – 2016. – С. 77-80.

47. Смірнов О.А., Лисенко І.А. Розробка впорядкованих каскадних таблиць рішень із використанням матриць слідування. Збірник наукових праць "Системи обробки інформації". – Випуск 6 (143). – Х.: ХУПС – 2016. – С. 216-220.

48. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Метод кількісної оцінки ризиків розроблення програмного забезпечення. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 128-133.

49. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Метод якісного аналізу ризиків розроблення програмного забезпечення. Наука і техніка Збройних Сил України. – Випуск 2(23). – Харків: ХУПС. – 2016. – С. 150-158.

50. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Проблеми аналізу та оцінки ризиків інформаційної діяльності. Збірник наукових праць "Системи обробки інформації". – Випуск 3 (140). – Х.: ХУПС – 2016. – С. 40-42.

51. Смірнов О.А., Коваленко А.С., Коваленко О.В., Доренський О.П. Удосконалення методу технічного обслуговування об’єктів інтегрованої інформаційної системи. Системи озброєння і військова техніка. – Випуск 2(46) – Х.: ХУПС – 2016. – С. 103-107.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....4
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....5
7 Економічні вимоги.....5
8 Вимоги щодо охорони праці.....5
9 Перелік документів, що розробляються.....6
10 Етапи розробки.....6
11 Порядок контролю та приймання.....6

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи управління електронним документообігом.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 35-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи управління електронним документообігом.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи управління електронним документообігом;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Visual C++.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинна бути розглянута пожежна безпека.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 117 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 19.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Смірнова Т.В.

Дослідження та програмна реалізація системи управління електронним документообігом

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 71

Літера: РП

Кропивницький – 2023 року

Файл Worker.cpp - вікно адміністратора

```
  /*****************************************************************************/
#include "StdAfx.h"
#include "Worker.h"
#include "Doc.h"
#include "Logger.h"
/*****************************************************************************/
using namespace System;
using namespace System::IO;
using namespace System::Collections;
/*****************************************************************************/
#define DOCS_FILE "DATA.LMB"
#define LOG_INPUT "INPUT.LML"
#define LOG_OUTPUT      "OUTPUT.LML"
/*****************************************************************************/
// NAME:          CWorker
// DESCRIPTION:   Конструктор класу
// INPUT:         N/D
/*****************************************************************************/
CWorker::CWorker(void)
{
  listDoc = gcnew ArrayList();
  logInput = gcnew CLogger();
  logOutput = gcnew CLogger();
}
/*****************************************************************************/
// NAME:          GetIndexByISBNHash
// DESCRIPTION:   Функція одержання індексу документа в загальному списку по
вхідному номеру документа
// INPUT:         ddHashValue - значення хеша вхідного номера документа
/*****************************************************************************/
Int32 CWorker::GetIndexByISBNHash(Int32 HashValue)
{
  // Цикл пошуку документа із заданим вхідним номером документа
  for(Int32 i = 0; i < listDoc->Count; i++)
  {
      // Хеш-значення вхідного номера документа заданого документа збігається з
      // хеш-     значенням вхідного номера документа знайденого документа?
      if(HashValue == ((CDoc^)listDoc[i])->GetISBNHash())
      {
            // Документ знайдений, повернути індекс
           return i;
      }
  }

  // Документ не знайдений, повернути -1
  return -1;
}
/*****************************************************************************/
// NAME:          AddDoc
// DESCRIPTION:   Функція додавання нового документа
// INPUT:         SourceDoc - об'єкт "документ" для включення в список
// OUTPUT:        TRUE - документ успішно доданий
//                FALSE - такий документ уже існує в системі
/*****************************************************************************/
Boolean CWorker::AddDoc(CDoc^ SourceDoc)
{
  // Документи з таким вхідним номером документа не існує?
  if(this->GetIndexByISBNHash(SourceDoc->GetISBNHash()) == -1)
  {
      // Додати заданній документ у загальний список
      listDoc->Add(SourceDoc);
      // Записати подія в лог
      logInput->WriteEvent(
            "Новий документ '" + ((CDoc^)SourceDoc)->GetName() +
            "', Автор '" + ((CDoc^)SourceDoc)->GetAuthor() +
```

```
            "', Вхідний номер документа " + ((CDoc^)SourceDoc)->GetISBN());
      // Функція відробила успішно
      return true;
  }
  else
  {
      // Документ із таким же вхідним номером документа існує, повернути помилку
      return false;
  }
}
/*************************************************************************/
// NAME:          RemoveDoc
// DESCRIPTION:   Функція видалення документів із системи
// INPUT:         ddISBNHash - значення хеша вхідного номера документа
// OUTPUT:              0 - видалення зроблене
//                      1 - видалення неможливо, не всі екземпляри перебувають у
системі
//                      2 - документ із заданим вхідним номером документа не
знайдений
/*************************************************************************/
Int32 CWorker::RemoveDoc(Int32 ddISBNHash)
{
  // Знайти документ по вхідному номеру документа
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);

  // Документ із таким вхідним номером документа існує?
  if(ddIndex != -1)
  {
      // Перевірити, що жодного документа немає в користувача
      if(((CDoc^)listDoc[ddIndex])->GetTotalNumber() ==
            ((CDoc^)listDoc[ddIndex])->GetFreeNumber())
      {

            // Записати подія в лог
            logOutput->WriteEvent(
                  "Виконаний документ '" + ((CDoc^)listDoc[ddIndex])->GetName()
+
                  "', Автор '" + ((CDoc^)listDoc[ddIndex])->GetAuthor() +
                  "', Вхідний номер документа " + ((CDoc^)listDoc[ddIndex])-
>GetISBN());
            // Видалити знайдений документ зі списку
            listDoc->RemoveAt(ddIndex);
            // Функція відробила успішно
            return 0;
      }
      else
      {
            // Повернути помилку, не всі документи перебувають у системі
            return 1;
      }
  }
  else
  {
      // Повернути помилку, документ із таким вхідним номером документа не був
знайдений
      return 2;
  }
}
/*************************************************************************/
// NAME:          GiveDoc
// DESCRIPTION:   Функція видачі документів користувачеві
// INPUT:         listDoc - список документів користувача (для виконання
додавання)
//                      ddISBNHash - хеш вхідного номера документа, що потрібно
одержати
// OUTPUT:        TRUE - операція пройшла успішно
//                      FALSE - у наявності немає жодного екземпляра заданого
документа
/*************************************************************************/
```

```
Boolean CWorker::GiveDoc(ArrayList^ listReader, Int32 ddISBNHash)
{
  // Визначити індекс по вхідному номеру документа
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);

  // Зменшити кількість вільних документів
  if((ddIndex != -1) && ((CDoc^)listDoc[ddIndex])->DecFreeNumber())
  {
      // Додати документ у список користувача
      listReader->Add(listDoc[ddIndex]);
      // Функція відробила успішно
      return true;
  }
  else
  {
      // Такого документа не є в наявності
      return false;
  }
}
/*********************************************************************/
// NAME:          TakeDoc
// DESCRIPTION:   Функція прийняття документів від користувача назад у систему
// INPUT:         // listDoc - список документів користувача (для виконання
видалення)
//                      ddISBNHash - хеш вхідного номера документа, що потрібно
повернути
/*********************************************************************/
void CWorker::TakeDoc(ArrayList^ listReader, Int32 ddISBNHash)
{
  // Визначити індекс по вхідному номеру документа (документ існує, тому що його
вхідний номер документа був повідомлений користувачеві)
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);

  // Видалити документ зі списку користувача
  // listReader->RemoveAt(ddIndex);

  // Збільшити кількість вільних документів
  ((CDoc^)listDoc[ddIndex])->IncFreeNumber();
}
/*********************************************************************/
// NAME:          LoadDocList
// DESCRIPTION:   Функція завантаження документів системи з файлу
// INPUT:         N/D
/*********************************************************************/
void CWorker::LoadDocList()
{
  // Перевірити існування файлу
  if(!File::Exists(DOCS_FILE))
  {
      // Файл не знайдений, закінчити виконання функції
      return;
  }

  // Відкрити файл
  StreamReader^ srDoc = gcnew StreamReader(DOCS_FILE);

  // Продовжувати, поки не буде знайдений кінець файлу
  while(srDoc->EndOfStream == false)
  {
      // Створити новий об'єкт "документ"
      CDoc^ NewDoc = gcnew CDoc();

      // Завантажити з файлу й установити параметри документа
      NewDoc->SetName(srDoc->ReadLine());
      NewDoc->SetAuthor(srDoc->ReadLine());
      NewDoc->SetISBN(srDoc->ReadLine());
      NewDoc->SetTheme(srDoc->ReadLine());
      NewDoc->SetPages(Convert::ToInt32(srDoc->ReadLine()));
      NewDoc->SetTotalNumber(Convert::ToInt32(srDoc->ReadLine()));
```

```
        NewDoc->SetFreeNumber(Convert::ToInt32(srDoc->ReadLine()));

        // Включити документ у загальний список документів
        listDoc->Add(NewDoc);
    }

    // Закрити файл
    srDoc->Close();
}
/*************************************************************************/
// NAME:          SaveDocList
// DESCRIPTION:   Функція збереження документів системи у файл
// INPUT:         N/D
/*************************************************************************/
void CWorker::SaveDocList()
{
    // Перевірити існування файлу
    if(File::Exists(DOCS_FILE))
    {
        // Видалити файл
        File::Delete(DOCS_FILE);
    }

    // Створити й відкрити файл
    StreamWriter^ swDoc = gcnew StreamWriter(DOCS_FILE);

    // Цикл запису по одному документу
    for(Int32 i = 0; i < listDoc->Count; i++)
    {
        // Одержати параметри документа й записати їх у файл
        swDoc->WriteLine(((CDoc^)listDoc[i])->GetName());
        swDoc->WriteLine(((CDoc^)listDoc[i])->GetAuthor());
        swDoc->WriteLine(((CDoc^)listDoc[i])->GetISBN());
        swDoc->WriteLine(((CDoc^)listDoc[i])->GetTheme());
        swDoc->WriteLine(Convert::ToString(((CDoc^)listDoc[i])->GetPages()));
        swDoc->WriteLine(Convert::ToString(((CDoc^)listDoc[i])-
>GetTotalNumber()));
        swDoc->WriteLine(Convert::ToString(((CDoc^)listDoc[i])->GetFreeNumber()));
    }

    // Закрити файл
    swDoc->Close();
}
/*************************************************************************/
// NAME:          FindDoc
// DESCRIPTION:   Функція пошуку документа по заданих параметрах
// INPUT:         strFindDoc - рядок для пошуку
/*************************************************************************/
ArrayList^ CWorker::FindDoc(String^ strFindValue)
{
    // Список для результату
    ArrayList^ listResult = gcnew ArrayList();

    // Шукане значення без обліку регістра
    String^ strValue = strFindValue->ToLower();

    // Задана порожній рядок?
    if(String::IsNullOrEmpty(strValue->Trim()))
    {
        //Вивести весь список
        for(Int32 i = 0; i < listDoc->Count; i++)
        {
            // Додати документ у результуючий список
            listResult->Add(listDoc[i]);
        }
    }
    else
    {
        // Вибрати з умовою
```

```
      for(Int32 i = 0; i < listDoc->Count; i++)
      {
            // Пошук рядка в кожному полі без обліку регістра
            if(((((CDoc^)listDoc[i])->GetName())->ToLower())->IndexOf(strValue)
!= -1 ||
               ((((CDoc^)listDoc[i])->GetAuthor())->ToLower())-
>IndexOf(strValue) != -1 ||
               ((((CDoc^)listDoc[i])->GetISBN())->ToLower())->IndexOf(strValue)
!= -1 ||
               ((((CDoc^)listDoc[i])->GetTheme())->ToLower())->IndexOf(strValue)
!= -1 ||
              Convert::ToString((((CDoc^)listDoc[i])->GetPages()))-
>IndexOf(strValue) != -1)
            {
                  // Додати документ у результуючий список
                  listResult->Add(listDoc[i]);
            }
      }
  }

  // Повернути список збігів
  return listResult;
}
/*************************************************************************/
// NAME:          ViewDoc
// DESCRIPTION:   Функція перегляду інформації про заданий документ
// INPUT:         ddISBNHash - хеш вхідного номера документа, інформацію про яку
треба переглянути
/*************************************************************************/
CDoc^ CWorker::ViewDoc(Int32 ddISBNHash)
{
  // Визначити індекс по вхідному номеру документа (документ існує, тому що його
вхідний номер документа був повідомлений користувачеві)
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);

  // Повернути елемент "документ" із загальне списку
  return (CDoc^)listDoc[ddIndex];
}
/*************************************************************************/
// NAME:          ReadLogs
// DESCRIPTION:   Функція зчитування історії з файлу
// INPUT:         N/D
/*************************************************************************/
void CWorker::ReadLogs()
{
  // Файл із історією надходжень існує?
  if(File::Exists(LOG_INPUT))
  {
      // Зчитування історії надходжень
      logInput->strLog = File::ReadAllText(LOG_INPUT);
  }
  else
  {
      logInput->strLog = "";
  }

  // Файл із історією списань існує?
  if(File::Exists(LOG_INPUT))
  {
      // Зчитування історії списань
      logOutput->strLog = File::ReadAllText(LOG_OUTPUT);
  }
  else
  {
      logOutput->strLog = "";
  }
}
/*************************************************************************/
// NAME:          WriteLogs
```

```
// DESCRIPTION:   Функція збереження історії у файл
// INPUT:         N/D
/*****************************************************************************/
void CWorker::WriteLogs()
{
  // Запис історії
  File::WriteAllText(LOG_INPUT, logInput->strLog);
  File::WriteAllText(LOG_OUTPUT, logOutput->strLog);
}
/*****************************************************************************/
// NAME:          ClearLogs
// DESCRIPTION:   Функція очищення історії
// INPUT:         N/D
/*****************************************************************************/
void CWorker::ClearLogs()
{
  logInput->strLog = "";
  logOutput->strLog = "";
}
/*****************************************************************************/
// NAME:          ViewInputLog
// DESCRIPTION:   Функція перегляду історії надходжень
// INPUT:         N/D
/*****************************************************************************/
String^ CWorker::ViewInputLog()
{
  return logInput->strLog;
}
/*****************************************************************************/
// NAME:          ViewOutputLog
// DESCRIPTION:   Функція перегляду історії списань
// INPUT:         N/D
/*****************************************************************************/
String^ CWorker::ViewOutputLog()
{
  return logOutput->strLog;
}
/*****************************************************************************/
```

Файл Worker.resx - xml опис вікна адміністратора

```
<?xml version="1.0" encoding=" utf-8"?>
<root>
  <xsd:schema id="root" xmlns="" xmlns:xsd=" " xmlns:msdata="urn: schemas-
microsoft-com: xml-msdata">
    <xsd:import namespace=" " />
    <xsd:element name="root" msdata:IsDataSet="true">
      <xsd:complexType>
        <xsd:choice maxOccurs="unbounded">
          <xsd:element name="metadata">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0" />
              </xsd:sequence>
              <xsd:attribute name="name" use="required" type="xsd:string" />
              <xsd:attribute name="type" type="xsd:string" />
              <xsd:attribute name="mimetype" type="xsd:string" />
              <xsd:attribute ref="xml:space" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="assembly">
            <xsd:complexType>
              <xsd:attribute name="alias" type="xsd:string" />
              <xsd:attribute name="name" type="xsd:string" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="data">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
                <xsd:element name="comment" type="xsd:string" minOccurs="0"
msdata:Ordinal="2" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required"
msdata:Ordinal="1" />
              <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
              <xsd:attribute name="mimetype" type="xsd:string"
msdata:Ordinal="4" />
              <xsd:attribute ref="xml:space" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="resheader">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required" />
            </xsd:complexType>
          </xsd:element>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
  <resheader name="resmimetype">
    <value>text/ microsoft-resx</value>
  </resheader>
  <resheader name="version">
    <value>2.0</value>
  </resheader>
  <resheader name="reader">
    <value>System.Resources.ResXResourceReader, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <resheader name="writer">
```

```
    <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <metadata name="menuStrip.TrayLocation" type="System.Drawing.Point,
System.Drawing, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a">
    <value>230, 17</value>
  </metadata>
  <metadata name="statusStrip.TrayLocation" type="System.Drawing.Point,
System.Drawing, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a">
    <value>332, 17</value>
  </metadata>
  <metadata name="groupInfo.Locked" type="System.Boolean, mscorlib,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">
    <value>True</value>
  </metadata>
  <metadata name="groupInfo.Locked" type="System.Boolean, mscorlib,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">
    <value>True</value>
  </metadata>
</root>
```

Файл Worker.h - бібліотека для файлу frmAbout.cpp

```
    #pragma once
/*************************************************************************/
#include "Worker.h"
#include "Doc.h"
#include "Counter.h"
#include "frmAbout.h"
/*************************************************************************/
#define PAGES_LESS                          "менше"
#define PAGES_GREATER                       "більше"
#define PAGES_EQUAL                         "дорівнює"
/*************************************************************************/
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
/*************************************************************************/
namespace UsrsManager
{
    public ref class frmWorker : public System::Windows::Forms::Form
    {
/*************************************************************************/
    private:
      CWorker^ Worker;              // Об'єкт "Адміністратор"

      Boolean IsCreateClicked;      // Прапор натискання по кнопці "Створити"
      Boolean IsEditClicked;        // Прапор натискання по кнопці "Редагувати"

      ArrayList^ listView;          // Поточний список документів для
відображення

      Int32 ddCurrentIndex;         // Індекс поточного відображуваного елемента
/*************************************************************************/
private: System::Windows::Forms::Button^  btnClearLogs;
/*************************************************************************/
public:     frmWorker(void)
      {
            // Ініціалізація компонентів форми
            InitializeComponent();

            // Кнопки "Створити" і "Редагувати" не минулого натиснуті жодного
разу
            IsCreateClicked = false;
            IsEditClicked = false;

            // Створити список відображуваних документів
            listView = gcnew ArrayList();

            // Поточний індекс документа для відображення не визначений
            ddCurrentIndex = -1;
      }
/*************************************************************************/
protected: ~frmWorker()
      {
            if (components)
            {
                  delete components;
            }
      }
/*************************************************************************/
    private: System::Windows::Forms::ListBox^  lstPagesDocList;
    private: System::Windows::Forms::NumericUpDown^  nmrPagesNumber;
    private: System::Windows::Forms::ComboBox^  cmbPagesDirect;
    private: System::Windows::Forms::ListBox^  lstThemeDocList;
```

```
    private: System::Windows::Forms::ListBox^  lstAuthorDocList;
    private: System::Windows::Forms::CheckBox^  chkPagesFlag;
    private: System::Windows::Forms::MenuStrip^  menuStrip;
    private: System::Windows::Forms::ToolStripMenuItem^  menuFile;
    private: System::Windows::Forms::ToolStripMenuItem^  menuFileSave;
    private: System::Windows::Forms::ToolStripMenuItem^  menuFileExit;
    private: System::Windows::Forms::ToolStripMenuItem^  menuHelp;

    private: System::Windows::Forms::ToolStripMenuItem^  menuHelpAbout;
    private: System::Windows::Forms::GroupBox^  groupBox2;
    private: System::Windows::Forms::TextBox^  txtThemeFilter;
    private: System::Windows::Forms::TextBox^  txtAuthorFilter;
    private: System::Windows::Forms::Label^  lblFreeInstanceNumber;
    private: System::Windows::Forms::Label^  lblTotalInstanceNumber;
    private: System::Windows::Forms::CheckBox^  chkThemeDocFlag;
    private: System::Windows::Forms::CheckBox^  chkAuthorDocFlag;
    private: System::Windows::Forms::CheckBox^  chkFreeInstanceFlag;
    private: System::Windows::Forms::CheckBox^  chkTotalInstaceFlag;
    private: System::Windows::Forms::NumericUpDown^  nmrFreeNumber;
    private: System::Windows::Forms::Button^  btnRefresh;
    private: System::Windows::Forms::NumericUpDown^  nmrTotalNumber;
    private: System::Windows::Forms::TabPage^  tabHistory;
    private: System::Windows::Forms::GroupBox^  groupOutput;
    private: System::Windows::Forms::TextBox^  txtOutputHistory;
    private: System::Windows::Forms::GroupBox^  groupInput;
    private: System::Windows::Forms::TextBox^  txtInputHistory;
    private: System::Windows::Forms::NumericUpDown^  nmrPageNumber;
    private: System::Windows::Forms::TextBox^  txtTheme;
    private: System::Windows::Forms::TextBox^  txtISBN;
    private: System::Windows::Forms::TextBox^  txtAuthor;
    private: System::Windows::Forms::TextBox^  txtDocName;
    private: System::Windows::Forms::TabPage^  tabReports;
    private: System::Windows::Forms::Label^  lblFreeNumber;
    private: System::Windows::Forms::Label^  lblTotalNumber;
    private: System::Windows::Forms::Label^  lblPagesNumber;
    private: System::Windows::Forms::Label^  lblTheme;
    private: System::Windows::Forms::StatusStrip^  statusStrip;
    private: System::Windows::Forms::ToolStripStatusLabel^  lblStatus;
    private: System::Windows::Forms::TextBox^  txtFindString;
    private: System::Windows::Forms::TabControl^  tabControl;
    private: System::Windows::Forms::TabPage^  tabDocs;
    private: System::Windows::Forms::Button^  btnEdit;
    private: System::Windows::Forms::Button^  btnDelete;
    private: System::Windows::Forms::Button^  btnCreate;
    private: System::Windows::Forms::GroupBox^  groupInfo;
    private: System::Windows::Forms::Label^  lblISBN;
    private: System::Windows::Forms::Label^  lblAuthor;
    private: System::Windows::Forms::Label^  lblDocName;
    private: System::Windows::Forms::GroupBox^  groupFinding;
    private: System::Windows::Forms::Button^  btnFind;
    private: System::Windows::Forms::Button^  btnFirst;
    private: System::Windows::Forms::Button^  btnPrev;
    private: System::Windows::Forms::Button^  btnNext;
    private: System::Windows::Forms::Button^  btnLast;
    private: System::Windows::Forms::TextBox^  txtPosition;
    private: System::ComponentModel::Container ^components;
/*****************************************************************************/
#pragma region Windows Form Designer generated code
      /// <summary>
      /// Необхідний метод для підтримки Розроблювача - не модифікувати
      /// зміст цього методу з кодовим редактором.
      /// </summary>
      void InitializeComponent(void)
      {
            this->lstPagesDocList = (gcnew System::Windows::Forms::ListBox());
            this->nmrPagesNumber = (gcnew
System::Windows::Forms::NumericUpDown());
            this->cmbPagesDirect = (gcnew System::Windows::Forms::ComboBox());
            this->lstThemeDocList = (gcnew System::Windows::Forms::ListBox());
```

```
            this->lstAuthorDocList = (gcnew System::Windows::Forms::ListBox());
            this->chkPagesFlag = (gcnew System::Windows::Forms::CheckBox());
            this->menuStrip = (gcnew System::Windows::Forms::MenuStrip());
            this->menuFile = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuFileSave = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuFileExit = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuHelp = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuHelpAbout = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->groupBox2 = (gcnew System::Windows::Forms::GroupBox());
            this->txtThemeFilter = (gcnew System::Windows::Forms::TextBox());
            this->txtAuthorFilter = (gcnew System::Windows::Forms::TextBox());
            this->lblFreeInstanceNumber = (gcnew
System::Windows::Forms::Label());
            this->lblTotalInstanceNumber = (gcnew
System::Windows::Forms::Label());
            this->chkThemeDocFlag = (gcnew System::Windows::Forms::CheckBox());
            this->chkAuthorDocFlag = (gcnew System::Windows::Forms::CheckBox());
            this->chkFreeInstanceFlag = (gcnew
System::Windows::Forms::CheckBox());
            this->chkTotalInstaceFlag = (gcnew
System::Windows::Forms::CheckBox());
            this->nmrFreeNumber = (gcnew
System::Windows::Forms::NumericUpDown());
            this->btnRefresh = (gcnew System::Windows::Forms::Button());
            this->nmrTotalNumber = (gcnew
System::Windows::Forms::NumericUpDown());
            this->tabHistory = (gcnew System::Windows::Forms::TabPage());
            this->btnClearLogs = (gcnew System::Windows::Forms::Button());
            this->groupOutput = (gcnew System::Windows::Forms::GroupBox());
            this->txtOutputHistory = (gcnew System::Windows::Forms::TextBox());
            this->groupInput = (gcnew System::Windows::Forms::GroupBox());
            this->txtInputHistory = (gcnew System::Windows::Forms::TextBox());
            this->nmrPageNumber = (gcnew
System::Windows::Forms::NumericUpDown());
            this->txtTheme = (gcnew System::Windows::Forms::TextBox());
            this->txtISBN = (gcnew System::Windows::Forms::TextBox());
            this->txtAuthor = (gcnew System::Windows::Forms::TextBox());
            this->txtDocName = (gcnew System::Windows::Forms::TextBox());
            this->tabReports = (gcnew System::Windows::Forms::TabPage());
            this->lblFreeNumber = (gcnew System::Windows::Forms::Label());
            this->lblTotalNumber = (gcnew System::Windows::Forms::Label());
            this->lblPagesNumber = (gcnew System::Windows::Forms::Label());
            this->lblTheme = (gcnew System::Windows::Forms::Label());
            this->statusStrip = (gcnew System::Windows::Forms::StatusStrip());
            this->lblStatus = (gcnew
System::Windows::Forms::ToolStripStatusLabel());
            this->txtFindString = (gcnew System::Windows::Forms::TextBox());
            this->tabControl = (gcnew System::Windows::Forms::TabControl());
            this->tabDocs = (gcnew System::Windows::Forms::TabPage());
            this->btnEdit = (gcnew System::Windows::Forms::Button());
            this->btnDelete = (gcnew System::Windows::Forms::Button());
            this->btnCreate = (gcnew System::Windows::Forms::Button());
            this->groupInfo = (gcnew System::Windows::Forms::GroupBox());
            this->txtPosition = (gcnew System::Windows::Forms::TextBox());
            this->btnFirst = (gcnew System::Windows::Forms::Button());
            this->btnPrev = (gcnew System::Windows::Forms::Button());
            this->btnNext = (gcnew System::Windows::Forms::Button());
            this->btnLast = (gcnew System::Windows::Forms::Button());
            this->lblISBN = (gcnew System::Windows::Forms::Label());
            this->lblAuthor = (gcnew System::Windows::Forms::Label());
            this->lblDocName = (gcnew System::Windows::Forms::Label());
            this->groupFinding = (gcnew System::Windows::Forms::GroupBox());
            this->btnFind = (gcnew System::Windows::Forms::Button());
```

```
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrPagesNumber))->BeginInit();
            this->menuStrip->SuspendLayout();
            this->groupBox 2-2->SuspendLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrFreeNumber))->BeginInit();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrTotalNumber))->BeginInit();
            this->tabHistory->SuspendLayout();
            this->groupOutput->SuspendLayout();
            this->groupInput->SuspendLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrPageNumber))->BeginInit();
            this->tabReports->SuspendLayout();
            this->statusStrip->SuspendLayout();
            this->tabControl->SuspendLayout();
            this->tabDocs->SuspendLayout();
            this->groupInfo->SuspendLayout();
            this->groupFinding->SuspendLayout();
            this->SuspendLayout();
            //
            // lstPagesDocList
            //
            this->lstPagesDocList->FormattingEnabled = true;
            this->lstPagesDocList->HorizontalScrollbar = true;
            this->lstPagesDocList->Location = System::Drawing::Point(478, 117);
            this->lstPagesDocList->Name = L"lstPagesDocList";
            this->lstPagesDocList->Size = System::Drawing::Size(223, 199);
            this->lstPagesDocList->TabIndex = 13;
            //
            // nmrPagesNumber
            //
            this->nmrPagesNumber->Location = System::Drawing::Point(583, 89);
            this->nmrPagesNumber->Maximum = System::Decimal(gcnew cli::array<
System::Int32 >(4) {65535, 0, 0, 0});
            this->nmrPagesNumber->Name = L"nmrPagesNumber";
            this->nmrPagesNumber->Size = System::Drawing::Size(64, 20);
            this->nmrPagesNumber->TabIndex = 6;
            //
            // cmbPagesDirect
            //
            this->cmbPagesDirect->FormattingEnabled = true;
            this->cmbPagesDirect->Location = System::Drawing::Point(479, 89);
            this->cmbPagesDirect->Name = L"cmbPagesDirect";
            this->cmbPagesDirect->Size = System::Drawing::Size(89, 21);
            this->cmbPagesDirect->TabIndex = 5;
            //
            // lstThemeDocList
            //
            this->lstThemeDocList->FormattingEnabled = true;
            this->lstThemeDocList->HorizontalScrollbar = true;
            this->lstThemeDocList->Location = System::Drawing::Point(242, 117);
            this->lstThemeDocList->Name = L"lstThemeDocList";
            this->lstThemeDocList->Size = System::Drawing::Size(223, 199);
            this->lstThemeDocList->TabIndex = 9;
            //
            // lstAuthorDocList
            //
            this->lstAuthorDocList->FormattingEnabled = true;
            this->lstAuthorDocList->HorizontalScrollbar = true;
            this->lstAuthorDocList->Location = System::Drawing::Point(6, 117);
            this->lstAuthorDocList->Name = L"lstAuthorDocList";
            this->lstAuthorDocList->Size = System::Drawing::Size(223, 199);
            this->lstAuthorDocList->TabIndex = 8;
            //
            // chkPagesFlag
            //
            this->chkPagesFlag->AutoSize = true;
            this->chkPagesFlag->Location = System::Drawing::Point(479, 68);
```

```
            this->chkPagesFlag->Name = L"chkPagesFlag";
            this->chkPagesFlag->Size = System::Drawing::Size(75, 17);
            this->chkPagesFlag->TabIndex = 4;
            this->chkPagesFlag->Text = L"Сторінок:";
            this->chkPagesFlag->UseVisualStyleBackColor = true;
            //
            // menuStrip
            //
            this->menuStrip->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(2) {this->menuFile, this->menuHelp});
            this->menuStrip->Location = System::Drawing::Point(0, 0);
            this->menuStrip->Name = L"menuStrip";
            this->menuStrip->Size = System::Drawing::Size(722, 24);
            this->menuStrip->TabIndex = 9;
            this->menuStrip->Text = L"menuStrip1";
            //
            // menuFile
            //
            this->menuFile->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(2) {this->menuFileSave,
                this->menuFileExit});
            this->menuFile->Name = L"menuFile";
            this->menuFile->Size = System::Drawing::Size(45, 20);
            this->menuFile->Text = L"Файл";
            //
            // menuFileSave
            //
            this->menuFileSave->Name = L"menuFileSave";
            this->menuFileSave->Size = System::Drawing::Size(152, 22);
            this->menuFileSave->Text = L"Зберегти";
            this->menuFileSave->Click += gcnew System::EventHandler(this,
&frmWorker::menuFileSave_Click);
            //
            // menuFileExit
            //
            this->menuFileExit->Name = L"menuFileExit";
            this->menuFileExit->Size = System::Drawing::Size(152, 22);
            this->menuFileExit->Text = L"Вихід";
            this->menuFileExit->Click += gcnew System::EventHandler(this,
&frmWorker::menuFileExit_Click);
            //
            // menuHelp
            //
            this->menuHelp->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(1) {this->menuHelpAbout});
            this->menuHelp->Name = L"menuHelp";
            this->menuHelp->Size = System::Drawing::Size(60, 20);
            this->menuHelp->Text = L"Довідка";
            //
            // menuHelpAbout
            //
            this->menuHelpAbout->Name = L"menuHelpAbout";
            this->menuHelpAbout->Size = System::Drawing::Size(166, 22);
            this->menuHelpAbout->Text = L"Про програму...";
            this->menuHelpAbout->Click += gcnew System::EventHandler(this,
&frmWorker::menuHelpAbout_Click);
            //
            // groupBox2
            //
            this->groupBox 2-2->Controls->Add(this->lstPagesDocList);
            this->groupBox 2-2->Controls->Add(this->nmrPagesNumber);
            this->groupBox 2-2->Controls->Add(this->cmbPagesDirect);
            this->groupBox 2-2->Controls->Add(this->chkPagesFlag);
            this->groupBox 2-2->Controls->Add(this->lstThemeDocList);
            this->groupBox 2-2->Controls->Add(this->lstAuthorDocList);
            this->groupBox 2-2->Controls->Add(this->txtThemeFilter);
            this->groupBox 2-2->Controls->Add(this->txtAuthorFilter);
            this->groupBox 2-2->Controls->Add(this->lblFreeInstanceNumber);
            this->groupBox 2-2->Controls->Add(this->lblTotalInstanceNumber);
```

```
            this->groupBox 2-2->Controls->Add(this->chkThemeDocFlag);
            this->groupBox 2-2->Controls->Add(this->chkAuthorDocFlag);
            this->groupBox 2-2->Controls->Add(this->chkFreeInstanceFlag);
            this->groupBox 2-2->Controls->Add(this->chkTotalInstaceFlag);
            this->groupBox 2-2->Location = System::Drawing::Point(8, 6);
            this->groupBox 2-2->Name = L"groupBox2";
            this->groupBox 2-2->Size = System::Drawing::Size(712, 325);
            this->groupBox 2-2->TabIndex = 0;
            this->groupBox 2-2->TabStop = false;
            //
            // txtThemeFilter
            //
            this->txtThemeFilter->Location = System::Drawing::Point(242, 90);
            this->txtThemeFilter->Name = L"txtThemeFilter";
            this->txtThemeFilter->Size = System::Drawing::Size(223, 20);
            this->txtThemeFilter->TabIndex = 7;
            //
            // txtAuthorFilter
            //
            this->txtAuthorFilter->Location = System::Drawing::Point(6, 90);
            this->txtAuthorFilter->Name = L"txtAuthorFilter";
            this->txtAuthorFilter->Size = System::Drawing::Size(223, 20);
            this->txtAuthorFilter->TabIndex = 6;
            //
            // lblFreeInstanceNumber
            //
            this->lblFreeInstanceNumber->AutoSize = true;
            this->lblFreeInstanceNumber->Location = System::Drawing::Point(476,
42);
            this->lblFreeInstanceNumber->Name = L"lblFreeInstanceNumber";
            this->lblFreeInstanceNumber->Size = System::Drawing::Size(26, 13);
            this->lblFreeInstanceNumber->TabIndex = 5;
            this->lblFreeInstanceNumber->Text = L"Н/Д";
            //
            // lblTotalInstanceNumber
            //
            this->lblTotalInstanceNumber->AutoSize = true;
            this->lblTotalInstanceNumber->Location = System::Drawing::Point(476,
19);
            this->lblTotalInstanceNumber->Name = L"lblTotalInstanceNumber";
            this->lblTotalInstanceNumber->Size = System::Drawing::Size(26, 13);
            this->lblTotalInstanceNumber->TabIndex = 4;
            this->lblTotalInstanceNumber->Text = L"Н/Д";
            //
            // chkThemeDocFlag
            //
            this->chkThemeDocFlag->AutoSize = true;
            this->chkThemeDocFlag->Location = System::Drawing::Point(240, 67);
            this->chkThemeDocFlag->Name = L"chkThemeDocFlag";
            this->chkThemeDocFlag->Size = System::Drawing::Size(176, 17);
            this->chkThemeDocFlag->TabIndex = 3;
            this->chkThemeDocFlag->Text = L"Список документів за темою:";
            this->chkThemeDocFlag->UseVisualStyleBackColor = true;
            //
            // chkAuthorDocFlag
            //
            this->chkAuthorDocFlag->AutoSize = true;
            this->chkAuthorDocFlag->Location = System::Drawing::Point(6, 67);
            this->chkAuthorDocFlag->Name = L"chkAuthorDocFlag";
            this->chkAuthorDocFlag->Size = System::Drawing::Size(165, 17);
            this->chkAuthorDocFlag->TabIndex = 2;
            this->chkAuthorDocFlag->Text = L"Список документів автора:";
            this->chkAuthorDocFlag->UseVisualStyleBackColor = true;
            //
            // chkFreeInstanceFlag
            //
            this->chkFreeInstanceFlag->AutoSize = true;
            this->chkFreeInstanceFlag->Location = System::Drawing::Point(6, 42);
            this->chkFreeInstanceFlag->Name = L"chkFreeInstanceFlag";
```

```
            this->chkFreeInstanceFlag->Size = System::Drawing::Size(208, 17);
            this->chkFreeInstanceFlag->TabIndex = 1;
            this->chkFreeInstanceFlag->Text = L"Кількість екземплярів на
виконання";
            this->chkFreeInstanceFlag->UseVisualStyleBackColor = true;
            //
            // chkTotalInstaceFlag
            //
            this->chkTotalInstaceFlag->AutoSize = true;
            this->chkTotalInstaceFlag->Location = System::Drawing::Point(6, 19);
            this->chkTotalInstaceFlag->Name = L"chkTotalInstaceFlag";
            this->chkTotalInstaceFlag->Size = System::Drawing::Size(293, 17);
            this->chkTotalInstaceFlag->TabIndex = 0;
            this->chkTotalInstaceFlag->Text = L"Загальна кількість екземплярів у
відібраному списку";
            this->chkTotalInstaceFlag->UseVisualStyleBackColor = true;
            //
            // nmrFreeNumber
            //
            this->nmrFreeNumber->Location = System::Drawing::Point(372, 196);
            this->nmrFreeNumber->Maximum = System::Decimal(gcnew cli::array<
System::Int32 >(4) {65535, 0, 0, 0});
            this->nmrFreeNumber->Name = L"nmrFreeNumber";
            this->nmrFreeNumber->ReadOnly = true;
            this->nmrFreeNumber->Size = System::Drawing::Size(80, 20);
            this->nmrFreeNumber->TabIndex = 8;
            //
            // btnRefresh
            //
            this->btnRefresh->Location = System::Drawing::Point(611, 337);
            this->btnRefresh->Name = L"btnRefresh";
            this->btnRefresh->Size = System::Drawing::Size(99, 23);
            this->btnRefresh->TabIndex = 7;
            this->btnRefresh->Text = L"Зберегти";
            this->btnRefresh->UseVisualStyleBackColor = true;
            this->btnRefresh->Click += gcnew System::EventHandler(this,
&frmWorker::btnRefresh_Click);
            //
            // nmrTotalNumber
            //
            this->nmrTotalNumber->Location = System::Drawing::Point(139, 196);
            this->nmrTotalNumber->Maximum = System::Decimal(gcnew cli::array<
System::Int32 >(4) {65535, 0, 0, 0});
            this->nmrTotalNumber->Name = L"nmrTotalNumber";
            this->nmrTotalNumber->Size = System::Drawing::Size(80, 20);
            this->nmrTotalNumber->TabIndex = 7;
            this->nmrTotalNumber->ValueChanged += gcnew
System::EventHandler(this, &frmWorker::nmrTotalNumber_ValueChanged);
            //
            // tabHistory
            //
            this->tabHistory->Controls->Add(this->btnClearLogs);
            this->tabHistory->Controls->Add(this->groupOutput);
            this->tabHistory->Controls->Add(this->groupInput);
            this->tabHistory->Location = System::Drawing::Point(4, 22);
            this->tabHistory->Name = L"tabHistory";
            this->tabHistory->Size = System::Drawing::Size(717, 366);
            this->tabHistory->TabIndex = 2;
            this->tabHistory->Text = L"Історія";
            this->tabHistory->UseVisualStyleBackColor = true;
            //
            // btnClearLogs
            //
            this->btnClearLogs->Location = System::Drawing::Point(631, 339);
            this->btnClearLogs->Name = L"btnClearLogs";
            this->btnClearLogs->Size = System::Drawing::Size(75, 23);
            this->btnClearLogs->TabIndex = 0;
            this->btnClearLogs->Text = L"Очистити";
            this->btnClearLogs->UseVisualStyleBackColor = true;
```

```
            this->btnClearLogs->Click += gcnew System::EventHandler(this,
&frmWorker::btnClearLogs_Click);
            //
            // groupOutput
            //
            this->groupOutput->Controls->Add(this->txtOutputHistory);
            this->groupOutput->Location = System::Drawing::Point(6, 166);
            this->groupOutput->Name = L"groupOutput";
            this->groupOutput->Size = System::Drawing::Size(700, 167);
            this->groupOutput->TabIndex = 1;
            this->groupOutput->TabStop = false;
            this->groupOutput->Text = L"Історія виконань";
            //
            // txtOutputHistory
            //
            this->txtOutputHistory->Location = System::Drawing::Point(6, 19);
            this->txtOutputHistory->Multiline = true;
            this->txtOutputHistory->Name = L"txtOutputHistory";
            this->txtOutputHistory->ScrollBars =
System::Windows::Forms::ScrollBars::Both;
            this->txtOutputHistory->Size = System::Drawing::Size(688, 139);
            this->txtOutputHistory->TabIndex = 2;
            //
            // groupInput
            //
            this->groupInput->Controls->Add(this->txtInputHistory);
            this->groupInput->Location = System::Drawing::Point(8, 3);
            this->groupInput->Name = L"groupInput";
            this->groupInput->Size = System::Drawing::Size(700, 157);
            this->groupInput->TabIndex = 0;
            this->groupInput->TabStop = false;
            this->groupInput->Text = L"Історія надходжень";
            //
            // txtInputHistory
            //
            this->txtInputHistory->Location = System::Drawing::Point(6, 19);
            this->txtInputHistory->Multiline = true;
            this->txtInputHistory->Name = L"txtInputHistory";
            this->txtInputHistory->ScrollBars =
System::Windows::Forms::ScrollBars::Both;
            this->txtInputHistory->Size = System::Drawing::Size(688, 130);
            this->txtInputHistory->TabIndex = 1;
            //
            // nmrPageNumber
            //
            this->nmrPageNumber->Location = System::Drawing::Point(139, 162);
            this->nmrPageNumber->Maximum = System::Decimal(gcnew cli::array<
System::Int32 >(4) {65535, 0, 0, 0});
            this->nmrPageNumber->Name = L"nmrPageNumber";
            this->nmrPageNumber->Size = System::Drawing::Size(80, 20);
            this->nmrPageNumber->TabIndex = 6;
            //
            // txtTheme
            //
            this->txtTheme->Location = System::Drawing::Point(139, 128);
            this->txtTheme->Name = L"txtTheme";
            this->txtTheme->Size = System::Drawing::Size(302, 20);
            this->txtTheme->TabIndex = 5;
            //
            // txtISBN
            //
            this->txtISBN->Location = System::Drawing::Point(139, 94);
            this->txtISBN->Name = L"txtISBN";
            this->txtISBN->Size = System::Drawing::Size(163, 20);
            this->txtISBN->TabIndex = 4;
            //
            // txtAuthor
            //
            this->txtAuthor->Location = System::Drawing::Point(139, 60);
```

```
            this->txtAuthor->Name = L"txtAuthor";
            this->txtAuthor->Size = System::Drawing::Size(505, 20);
            this->txtAuthor->TabIndex = 3;
            //
            // txtDocName
            //
            this->txtDocName->Location = System::Drawing::Point(139, 26);
            this->txtDocName->Name = L"txtDocName";
            this->txtDocName->Size = System::Drawing::Size(557, 20);
            this->txtDocName->TabIndex = 2;
            //
            // tabReports
            //
            this->tabReports->Controls->Add(this->btnRefresh);
            this->tabReports->Controls->Add(this->groupBox2);
            this->tabReports->Location = System::Drawing::Point(4, 22);
            this->tabReports->Name = L"tabReports";
            this->tabReports->Padding = System::Windows::Forms::Padding(3);
            this->tabReports->Size = System::Drawing::Size(717, 366);
            this->tabReports->TabIndex = 1;
            this->tabReports->Text = L"Звіти";
            this->tabReports->UseVisualStyleBackColor = true;
            //
            // lblFreeNumber
            //
            this->lblFreeNumber->AutoSize = true;
            this->lblFreeNumber->Location = System::Drawing::Point(263, 196);
            this->lblFreeNumber->Name = L"lblFreeNumber";
            this->lblFreeNumber->Size = System::Drawing::Size(103, 13);
            this->lblFreeNumber->TabIndex = 6;
            this->lblFreeNumber->Text = L"Вільно екземплярів";
            //
            // lblTotalNumber
            //
            this->lblTotalNumber->AutoSize = true;
            this->lblTotalNumber->Location = System::Drawing::Point(34, 196);
            this->lblTotalNumber->Name = L"lblTotalNumber";
            this->lblTotalNumber->Size = System::Drawing::Size(105, 13);
            this->lblTotalNumber->TabIndex = 5;
            this->lblTotalNumber->Text = L"Всього екземплярів";
            //
            // lblPagesNumber
            //
            this->lblPagesNumber->AutoSize = true;
            this->lblPagesNumber->Location = System::Drawing::Point(40, 162);
            this->lblPagesNumber->Name = L"lblPagesNumber";
            this->lblPagesNumber->Size = System::Drawing::Size(99, 13);
            this->lblPagesNumber->TabIndex = 4;
            this->lblPagesNumber->Text = L"Кількість сторінок";
            //
            // lblTheme
            //
            this->lblTheme->AutoSize = true;
            this->lblTheme->Location = System::Drawing::Point(108, 128);
            this->lblTheme->Name = L"lblTheme";
            this->lblTheme->Size = System::Drawing::Size(31, 13);
            this->lblTheme->TabIndex = 3;
            this->lblTheme->Text = L"Тема";
            //
            // statusStrip
            //
            this->statusStrip->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(1) {this->lblStatus});
            this->statusStrip->Location = System::Drawing::Point(0, 417);
            this->statusStrip->Name = L"statusStrip";
            this->statusStrip->Size = System::Drawing::Size(722, 22);
            this->statusStrip->TabIndex = 10;
            this->statusStrip->Text = L"statusStrip1";
            //
```

```
            // lblStatus
            //
            this->lblStatus->Name = L"lblStatus";
            this->lblStatus->Size = System::Drawing::Size(109, 17);
            this->lblStatus->Text = L"toolStripStatusLabel1";
            //
            // txtFindString
            //
            this->txtFindString->Location = System::Drawing::Point(6, 19);
            this->txtFindString->Name = L"txtFindString";
            this->txtFindString->Size = System::Drawing::Size(609, 20);
            this->txtFindString->TabIndex = 0;
            //
            // tabControl
            //
            this->tabControl->Controls->Add(this->tabDocs);
            this->tabControl->Controls->Add(this->tabReports);
            this->tabControl->Controls->Add(this->tabHistory);
            this->tabControl->Location = System::Drawing::Point(0, 27);
            this->tabControl->Name = L"tabControl";
            this->tabControl->SelectedIndex = 0;
            this->tabControl->Size = System::Drawing::Size(725, 392);
            this->tabControl->TabIndex = 11;
            //
            // tabDocs
            //
            this->tabDocs->Controls->Add(this->btnEdit);
            this->tabDocs->Controls->Add(this->btnDelete);
            this->tabDocs->Controls->Add(this->btnCreate);
            this->tabDocs->Controls->Add(this->groupInfo);
            this->tabDocs->Controls->Add(this->groupFinding);
            this->tabDocs->Location = System::Drawing::Point(4, 22);
            this->tabDocs->Name = L"tabDocs";
            this->tabDocs->Padding = System::Windows::Forms::Padding(3);
            this->tabDocs->Size = System::Drawing::Size(717, 366);
            this->tabDocs->TabIndex = 0;
            this->tabDocs->Text = L"Документи";
            this->tabDocs->UseVisualStyleBackColor = true;
            //
            // btnEdit
            //
            this->btnEdit->Location = System::Drawing::Point(402, 337);
            this->btnEdit->Name = L"btnEdit";
            this->btnEdit->Size = System::Drawing::Size(99, 23);
            this->btnEdit->TabIndex = 14;
            this->btnEdit->Text = L"Редагувати";
            this->btnEdit->UseVisualStyleBackColor = true;
            this->btnEdit->Click += gcnew System::EventHandler(this,
&frmWorker::btnEdit_Click);
            //
            // btnDelete
            //
            this->btnDelete->Location = System::Drawing::Point(507, 337);
            this->btnDelete->Name = L"btnDelete";
            this->btnDelete->Size = System::Drawing::Size(99, 23);
            this->btnDelete->TabIndex = 15;
            this->btnDelete->Text = L"Видалити";
            this->btnDelete->UseVisualStyleBackColor = true;
            this->btnDelete->Click += gcnew System::EventHandler(this,
&frmWorker::btnDelete_Click);
            //
            // btnCreate
            //
            this->btnCreate->Location = System::Drawing::Point(612, 337);
            this->btnCreate->Name = L"btnCreate";
            this->btnCreate->Size = System::Drawing::Size(99, 23);
            this->btnCreate->TabIndex = 16;
            this->btnCreate->Text = L"Створити";
            this->btnCreate->UseVisualStyleBackColor = true;
```

```
            this->btnCreate->Click += gcnew System::EventHandler(this,
&frmWorker::btnCreate_Click);
            //
            // groupInfo
            //
            this->groupInfo->Controls->Add(this->txtPosition);
            this->groupInfo->Controls->Add(this->btnFirst);
            this->groupInfo->Controls->Add(this->btnPrev);
            this->groupInfo->Controls->Add(this->btnNext);
            this->groupInfo->Controls->Add(this->btnLast);
            this->groupInfo->Controls->Add(this->nmrFreeNumber);
            this->groupInfo->Controls->Add(this->nmrTotalNumber);
            this->groupInfo->Controls->Add(this->nmrPageNumber);
            this->groupInfo->Controls->Add(this->txtTheme);
            this->groupInfo->Controls->Add(this->txtISBN);
            this->groupInfo->Controls->Add(this->txtAuthor);
            this->groupInfo->Controls->Add(this->txtDocName);
            this->groupInfo->Controls->Add(this->lblFreeNumber);
            this->groupInfo->Controls->Add(this->lblTotalNumber);
            this->groupInfo->Controls->Add(this->lblPagesNumber);
            this->groupInfo->Controls->Add(this->lblTheme);
            this->groupInfo->Controls->Add(this->lblISBN);
            this->groupInfo->Controls->Add(this->lblAuthor);
            this->groupInfo->Controls->Add(this->lblDocName);
            this->groupInfo->Location = System::Drawing::Point(6, 65);
            this->groupInfo->Name = L"groupInfo";
            this->groupInfo->Size = System::Drawing::Size(702, 266);
            this->groupInfo->TabIndex = 9;
            this->groupInfo->TabStop = false;
            this->groupInfo->Text = L"Інформація про документ";
            //
            // txtPosition
            //
            this->txtPosition->Location = System::Drawing::Point(564, 241);
            this->txtPosition->Name = L"txtPosition";
            this->txtPosition->Size = System::Drawing::Size(58, 20);
            this->txtPosition->TabIndex = 11;
            this->txtPosition->TextAlign =
System::Windows::Forms::HorizontalAlignment::Center;
            this->txtPosition->KeyDown += gcnew
System::Windows::Forms::KeyEventHandler(this, &frmWorker::txtPosition_KeyDown);
            //
            // btnFirst
            //
            this->btnFirst->Location = System::Drawing::Point(490, 241);
            this->btnFirst->Name = L"btnFirst";
            this->btnFirst->Size = System::Drawing::Size(31, 20);
            this->btnFirst->TabIndex = 9;
            this->btnFirst->Text = L"|<<";
            this->btnFirst->UseVisualStyleBackColor = true;
            this->btnFirst->Click += gcnew System::EventHandler(this,
&frmWorker::btnFirst_Click);
            //
            // btnPrev
            //
            this->btnPrev->Location = System::Drawing::Point(527, 241);
            this->btnPrev->Name = L"btnPrev";
            this->btnPrev->Size = System::Drawing::Size(31, 20);
            this->btnPrev->TabIndex = 10;
            this->btnPrev->Text = L"<<";
            this->btnPrev->UseVisualStyleBackColor = true;
            this->btnPrev->Click += gcnew System::EventHandler(this,
&frmWorker::btnPrev_Click);
            //
            // btnNext
            //
            this->btnNext->Location = System::Drawing::Point(628, 240);
            this->btnNext->Name = L"btnNext";
            this->btnNext->Size = System::Drawing::Size(31, 20);
```

```
            this->btnNext->TabIndex = 12;
            this->btnNext->Text = L">>";
            this->btnNext->UseVisualStyleBackColor = true;
            this->btnNext->Click += gcnew System::EventHandler(this,
&frmWorker::btnNext_Click);
            //
            // btnLast
            //
            this->btnLast->Location = System::Drawing::Point(665, 240);
            this->btnLast->Name = L"btnLast";
            this->btnLast->Size = System::Drawing::Size(31, 20);
            this->btnLast->TabIndex = 13;
            this->btnLast->Text = L">>|";
            this->btnLast->UseVisualStyleBackColor = true;
            this->btnLast->Click += gcnew System::EventHandler(this,
&frmWorker::btnLast_Click);
            //
            // lblISBN
            //
            this->lblISBN->AutoSize = true;
            this->lblISBN->Location = System::Drawing::Point(2, 94);
            this->lblISBN->Name = L"lblISBN";
            this->lblISBN->Size = System::Drawing::Size(137, 13);
            this->lblISBN->TabIndex = 2;
            this->lblISBN->Text = L"Вхідний номер документу";
            //
            // lblAuthor
            //
            this->lblAuthor->AutoSize = true;
            this->lblAuthor->Location = System::Drawing::Point(101, 60);
            this->lblAuthor->Name = L"lblAuthor";
            this->lblAuthor->Size = System::Drawing::Size(38, 13);
            this->lblAuthor->TabIndex = 1;
            this->lblAuthor->Text = L"Автор";
            //
            // lblDocName
            //
            this->lblDocName->AutoSize = true;
            this->lblDocName->Location = System::Drawing::Point(44, 26);
            this->lblDocName->Name = L"lblDocName";
            this->lblDocName->Size = System::Drawing::Size(95, 13);
            this->lblDocName->TabIndex = 0;
            this->lblDocName->Text = L"Назва документу";
            //
            // groupFinding
            //
            this->groupFinding->Controls->Add(this->btnFind);
            this->groupFinding->Controls->Add(this->txtFindString);
            this->groupFinding->Location = System::Drawing::Point(6, 6);
            this->groupFinding->Name = L"groupFinding";
            this->groupFinding->Size = System::Drawing::Size(702, 53);
            this->groupFinding->TabIndex = 8;
            this->groupFinding->TabStop = false;
            this->groupFinding->Text = L"Ключові рядки для пошуку";
            //
            // btnFind
            //
            this->btnFind->Location = System::Drawing::Point(621, 16);
            this->btnFind->Name = L"btnFind";
            this->btnFind->Size = System::Drawing::Size(75, 23);
            this->btnFind->TabIndex = 1;
            this->btnFind->Text = L"Знайти";
            this->btnFind->UseVisualStyleBackColor = true;
            this->btnFind->Click += gcnew System::EventHandler(this,
&frmWorker::btnFind_Click);
            //
            // frmWorker
            //
            this->AutoScaleDimensions = System::Drawing::Size(6, 13);
```

```
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(722, 439);
            this->Controls->Add(this->menuStrip);
            this->Controls->Add(this->statusStrip);
            this->Controls->Add(this->tabControl);
            this->Name = L"frmWorker";
            this->Text = L"Режим  адміністратора - Система Управління
Електронним Документообігом ";
            this->Load += gcnew System::EventHandler(this,
&frmWorker::frmWorker_Load);
            this->FormClosed += gcnew
System::Windows::Forms::FormClosedEventHandler(this,
&frmWorker::frmWorker_FormClosed);
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrPagesNumber))->EndInit();
            this->menuStrip->ResumeLayout(false);
            this->menuStrip->PerformLayout();
            this->groupBox 2-2->ResumeLayout(false);
            this->groupBox 2-2->PerformLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrFreeNumber))->EndInit();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrTotalNumber))->EndInit();
            this->tabHistory->ResumeLayout(false);
            this->groupOutput->ResumeLayout(false);
            this->groupOutput->PerformLayout();
            this->groupInput->ResumeLayout(false);
            this->groupInput->PerformLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrPageNumber))->EndInit();
            this->tabReports->ResumeLayout(false);
            this->statusStrip->ResumeLayout(false);
            this->statusStrip->PerformLayout();
            this->tabControl->ResumeLayout(false);
            this->tabDocs->ResumeLayout(false);
            this->groupInfo->ResumeLayout(false);
            this->groupInfo->PerformLayout();
            this->groupFinding->ResumeLayout(false);
            this->groupFinding->PerformLayout();
            this->ResumeLayout(false);
            this->PerformLayout();

      }
#pragma endregion
/*********************************************************************/
private: System::Void LockingForm()
      {
            // Блокування всіх елементів керування
            btnFind->Enabled = false;
            btnEdit->Enabled = false;
            btnDelete->Enabled = false;
            btnCreate->Enabled = false;
            btnFirst->Enabled = false;
            btnLast->Enabled = false;
            btnNext->Enabled = false;
            btnPrev->Enabled = false;
            txtPosition->Enabled = false;
            menuStrip->Enabled = false;
            tabControl->TabPages[1]->Enabled = false;
            tabControl->TabPages[2]->Enabled = false;
      }
/*********************************************************************/
private: System::Void UnlockingForm()
      {
            // Розблокування всіх елементів керування
            btnFind->Enabled = true;
            btnEdit->Enabled = true;
            btnDelete->Enabled = true;
            btnCreate->Enabled = true;
```

```
			btnFirst->Enabled = true;
			btnLast->Enabled = true;
			btnNext->Enabled = true;
			btnPrev->Enabled = true;
			txtPosition->Enabled = true;
			menuStrip->Enabled = true;
			tabControl->TabPages[1]->Enabled = true;
			tabControl->TabPages[2]->Enabled = true;
		}
/*************************************************************************/
private: System::Void UpdateView()
		{
			// Вивести історію
			txtInputHistory->Text = Worker->ViewInputLog();
			txtOutputHistory->Text = Worker->ViewOutputLog();

			// Список не порожній?
			if(listView->Count != 0)
			{
				// Перевірити індекс поточного елемента
				if(ddCurrentIndex >= listView->Count)
				{
					// Установити покажчик на останній елемент
					ddCurrentIndex = listView->Count - 1;
				}

				// Значення індексу негативно?
				if(ddCurrentIndex < 0)
				{
					// Установити індекс на перший елемент
					ddCurrentIndex = 0;
				}

				// Одержати характеристики документа й заповнити поля форми
				txtDocName->Text = ((CDoc^)listView[ddCurrentIndex])-
>GetName();
				txtAuthor->Text = ((CDoc^)listView[ddCurrentIndex])-
>GetAuthor();
				txtISBN->Text = ((CDoc^)listView[ddCurrentIndex])->GetISBN();
				txtTheme->Text = ((CDoc^)listView[ddCurrentIndex])-
>GetTheme();
				nmrPageNumber->Value = ((CDoc^)listView[ddCurrentIndex])-
>GetPages();
				nmrTotalNumber->Value = ((CDoc^)listView[ddCurrentIndex])-
>GetTotalNumber();
				nmrFreeNumber->Value = ((CDoc^)listView[ddCurrentIndex])-
>GetFreeNumber();

				// Установити поточну позицію
				txtPosition->Text = Convert::ToString(ddCurrentIndex + 1) + "
/ " +
					Convert::ToString(listView->Count);

				// Розблокувати кнопки редагування й видалення
				btnDelete->Enabled = true;
				btnEdit->Enabled = true;
			}
			else
			{
				// Очистити всі поля
				txtDocName->Text = "";
				txtAuthor->Text = "";
				txtISBN->Text = "";
				txtTheme->Text = "";
				nmrPageNumber->Value = 0;
				nmrTotalNumber->Value = 0;
				nmrFreeNumber->Value = 0;
				// Поточна позиція - 0
				txtPosition->Text = "0/0";
```

```
                    // Установити індекс поточного елемента
                    ddCurrentIndex = 0;
                    // Заблокувати кнопки редагування й видалення
                    btnDelete->Enabled = false;
                    btnEdit->Enabled = false;
                    // Вивести стан
                    lblStatus->Text = "Немає записів для відображення";
              }
       }
/************************************************************************/
private: System::Void frmWorker_Load(System::Object^  sender, System::EventArgs^
e)
       {
              // Створити об'єкт "Адміністратор"
              Worker = gcnew CWorker();

              // Завантажити список документів
              Worker->LoadDocList();

              // Завантажити історію
              Worker->ReadLogs();

              // Обновити форму
             UpdateView();

             // Вивести стан
             lblStatus->Text = "Для відображення існуючого списку скористайтеся
пошуком";

             // Заповнення випадаючого списку вибору документів по кількості
сторінок
             cmbPagesDirect->Items->Add(PAGES_LESS);
             cmbPagesDirect->Items->Add(PAGES_GREATER);
             cmbPagesDirect->Items->Add(PAGES_EQUAL);
       }
 /************************************************************************/
private: System::Void btnCreate_Click(System::Object^  sender,
System::EventArgs^  e)
       {
              // Кнопка "Створити" була натиснута раніше?
              if(IsCreateClicked)
              {
                    // Забрати зайві пробіли у всіх полях
                    txtDocName->Text = txtDocName->Text->Trim();
                    txtAuthor->Text = txtAuthor->Text->Trim();
                    txtISBN->Text = txtISBN->Text->Trim();
                    txtTheme->Text = txtTheme->Text->Trim();

                    // Перевірити значимі поля на коректність
                    if(txtDocName->Text == "" || txtISBN->Text == "")
                    {
                          // Обов'язкові поля не заповнені
                          MessageBox::Show("Поля 'Назва документа' і 'вхідний
номер документа' обов'язкові для заповнення!",
                                "Usrs Manager", MessageBoxButtons::OK,
MessageBoxIcon::Error);
                          return;
                    }

                    // Перевірити кількість екземплярів
                    if(Convert::ToInt32(nmrTotalNumber->Value) <= 0)
                    {
                          // Вивести повідомлення про помилку
                          MessageBox::Show("Некоректне значення кількості
екземплярів",
                                "Usrs Manager", MessageBoxButtons::OK,
MessageBoxIcon::Error);
                          return;
                    }
```

```
                    // Створити новий об'єкт "Документ"
                    CDoc^ NewDoc = gcnew CDoc;
                    // Заповнити поля об'єкта
                    NewDoc->SetName(txtDocName->Text);
                    NewDoc->SetAuthor(txtAuthor->Text);
                    NewDoc->SetISBN(txtISBN->Text);
                    NewDoc->SetTheme(txtTheme->Text);
                    NewDoc->SetPages(Convert::ToInt32(nmrPageNumber->Value));
                    NewDoc->SetTotalNumber(Convert::ToInt32(nmrTotalNumber-
>Value));
                    NewDoc->SetFreeNumber(Convert::ToInt32(nmrTotalNumber-
>Value));
                    // Додати документ у загальний список (якщо документ має
унікальний вхідний номер документа)
                    if(!Worker->AddDoc(NewDoc))
                    {
                          MessageBox::Show("Документ із таким вхідним номером
документа вже існує!",
                                "Usrs Manager", MessageBoxButtons::OK,
MessageBoxIcon::Error);
                          return;
                    }

                    // Додати документ у список для відображення
                    listView->Add(NewDoc);

                    // Обновити поля на формі
                    ddCurrentIndex ++;
                    UpdateView();

                    // Розблокувати елементи керування
                    UnlockingForm();

                    // Перемінити напис на кнопці
                    btnCreate->Text = "Створити";

                    // Вивід стану
                    lblStatus->Text = "Запис доданий";

                   // Змінити значення прапора
                   IsCreateClicked = false;
              }
             else
             {
                    // Перемінити напис на кнопці
                    btnCreate->Text = "Зберегти";

                    // Заблокувати елементи керування
                    LockingForm();

                    // Розблокувати кнопку збереження
                    btnCreate->Enabled = true;

                    // Розблокувати поля для введення
                    txtDocName->Enabled = true;
                    txtAuthor->Enabled = true;
                    txtISBN->Enabled = true;
                    txtTheme->Enabled = true;
                    nmrPageNumber->Enabled = true;
                    nmrTotalNumber->Enabled = true;

                    // Установити початкові значення полів для введення
                    txtDocName->Text = "";
                    txtAuthor->Text = "";
                    txtISBN->Text = "";
                    txtTheme->Text = "";
                    nmrPageNumber->Value = 0;
                    nmrTotalNumber->Value = 1;
```

```
                nmrFreeNumber->Value = 1;

                // Вивести стан
                lblStatus->Text = "Введіть значення полів";

                // Змінити значення прапора натискання по кнопці "Створити"
                IsCreateClicked = true;
            }
        }
/*************************************************************************/
private: System::Void btnDelete_Click(System::Object^  sender,
System::EventArgs^  e)
        {
            // Видалити документ із загального списку
            Int32 ddResult = Worker->RemoveDoc(
                ((CDoc^)listView[ddCurrentIndex])->GetISBNHash());

            // Перевірити код повернення
            switch(ddResult)
            {
            case 0:
                {
                    // Документ вилучений успішно
                    // Видалити документ зі списку для виводу
                    listView->RemoveAt(ddCurrentIndex);
                    // Обновити форму
                    UpdateView();

                    break;
                }
            case 1:
                {
                    // Документ не був вилучений, не всі екземпляри в
системі
                    MessageBox::Show("Видалення неможливо, тому що не всі
документи перебувають у системі",
                        "Usrs Manager", MessageBoxButtons::OK,
MessageBoxIcon::Error);

                    break;
                }
            case 2:
                {
                    // Документ не знайдений
                    MessageBox::Show("Видалення неможливо, документ у
системі відсутній",
                            "Usrs Manager", MessageBoxButtons::OK,
MessageBoxIcon::Error);
                    // Видалити документ зі списку для виводу
                    listView->RemoveAt(ddCurrentIndex);
                    // Обновити форму
                    UpdateView();

                    break;
                }
            }
        }
/*************************************************************************/
private: System::Void btnFind_Click(System::Object^  sender, System::EventArgs^
e)
        {
            // Забрати в рядку пошуку зайві пробіли й перетворити до нижнього
регістра
            String^ strValue = txtFindString->Text->Trim()->ToLower();

            // Перевірити рядок
            if(String::IsNullOrEmpty(strValue))
            {
                // Рядок порожня, запропонувати вивести весь список
```

```
                  if(MessageBox::Show("Рядок пошуку не заданий, вивести весь
список?",
                                                "Usrs Manager",
MessageBoxButtons::YesNo,
                                                MessageBoxIcon::Question) ==
::DialogResult::Yes)
                  {
                        // Перейти на вивід списку
                        goto OUT_LIST;
                  }
                  // Завершити роботу функції
                  return;
            }

OUT_LIST:   // Здійснити пошук документа по заданих параметрах
            listView = Worker->FindDoc(strValue);

            // Вивести стан
            lblStatus->Text = "Пошук завершений";
            // Обновити форму
            UpdateView();
      }
/*********************************************************************/
private: System::Void btnFirst_Click(System::Object^  sender, System::EventArgs^
e)
      {
            // Установити індекс на перший елемент
            ddCurrentIndex = 0;
            // Обновити форму
            UpdateView();
      }
/*********************************************************************/
private: System::Void btnPrev_Click(System::Object^  sender, System::EventArgs^
e)
      {
            // Установити індекс на попередній елемент
            ddCurrentIndex ---i;
            // Обновити форму
            UpdateView();
      }
/*********************************************************************/
private: System::Void btnNext_Click(System::Object^  sender, System::EventArgs^
e)
      {
            // Установити індекс на наступний елемент
            ddCurrentIndex ++;
            // Обновити форму
            UpdateView();
      }
/*********************************************************************/
private: System::Void btnLast_Click(System::Object^  sender, System::EventArgs^
e)
      {
            // Установити індекс на останній елемент
            ddCurrentIndex = listView->Count - 1;
            // Обновити форму
            UpdateView();
      }
/*********************************************************************/
private: System::Void btnEdit_Click(System::Object^  sender, System::EventArgs^
e)
      {
            // Кнопка "Редагувати" була натиснута раніше?
            if(IsEditClicked)
            {
                  // Забрати зайві пробіли
                  txtDocName->Text = txtDocName->Text->Trim();
                  txtAuthor->Text = txtAuthor->Text->Trim();
                  txtTheme->Text = txtTheme->Text->Trim();
```

```
                    // Перевірити значимі поля на коректність
                    if(txtDocName->Text == "")
                    {
                          // Обов'язкові поля не заповнені
                          MessageBox::Show("Поле 'Назва документа' обов'язково
для заповнення!",
                                "Usrs Manager", MessageBoxButtons::OK,
MessageBoxIcon::Error);
                          return;
                    }

                    // Відредагувати об'єкт у списках
                    ((CDoc^)listView[ddCurrentIndex])->SetName(txtDocName->Text);
                    ((CDoc^)listView[ddCurrentIndex])->SetAuthor(txtAuthor-
>Text);
                    ((CDoc^)listView[ddCurrentIndex])->SetTheme(txtTheme->Text);
                    ((CDoc^)listView[ddCurrentIndex])-
>SetPages(Convert::ToInt32(nmrPageNumber->Value));

                    // (???) Видалити документ зі списку перегляду
                    //Worker->RemoveDoc(((CDoc^)listView[ddCurrentIndex])-
>GetISBNHash());
                    // (???) Додати документ у список перегляду
                    //Worker->AddDoc((CDoc^)listView[ddCurrentIndex]);

                    // Обновити форму
                    UpdateView();

                    // Розблокувати елементи керування
                    UnlockingForm();

                    // Розблокувати некоректируємі поля
                    txtISBN->Enabled = true;
                    nmrTotalNumber->Enabled = true;
                    nmrFreeNumber->Enabled = true;

                    // Перемінити напис на кнопці
                    btnEdit->Text = "Редагувати";

                   // Змінити значення прапора натискання кнопки "Редагувати"
                   IsEditClicked = false;
              }
              else
              {
                    // Перемінити напис на кнопці
                    btnEdit->Text = "Зберегти";

                    // Заблокувати елементи керування
                    LockingForm();

                    // Розблокувати кнопку збереження
                    btnEdit->Enabled = true;

                    // Заблокувати некоректируємі поля
                    txtISBN->Enabled = false;
                    nmrTotalNumber->Enabled = false;
                    nmrFreeNumber->Enabled = false;

                    // Змінити значення прапора натискання кнопки "Редагувати"
                    IsEditClicked = true;
              }
        }
/*********************************************************************/
private: System::Void menuFileSave_Click(System::Object^  sender,
System::EventArgs^  e)
        {
              // Зберегти список документів у файл
              Worker->SaveDocList();
```

```
            // Зберегти історію
            Worker->WriteLogs();

            // Вивести стан
            lblStatus->Text = "Збереження виконане";
      }
/*************************************************************************/
private: System::Void menuFileExit_Click(System::Object^  sender,
System::EventArgs^  e)
      {
            // Вийти з додатка
            Application::Exit();
      }
/*************************************************************************/
private: System::Void nmrTotalNumber_ValueChanged(System::Object^  sender,
System::EventArgs^  e)
      {
            // Можливе натискання тільки при створенні документа
            // Кількість вільних екземплярів дорівнює загальній кількості
екземплярів
            nmrFreeNumber->Value = nmrTotalNumber->Value;
      }
/*************************************************************************/
   private: System::Void frmWorker_FormClosed(System::Object^  sender,
System::Windows::Forms::FormClosedEventArgs^  e)
     {
            // Вийти з додатка
            Application::Exit();
     }
/*************************************************************************/
private: System::Void btnClearLogs_Click(System::Object^  sender,
System::EventArgs^  e)
      {
            // Очистити історію
            Worker->ClearLogs();

            // Обновити форму
            UpdateView();

            // Вивести стан
            lblStatus->Text = "Очищення виконане";
      }
/*************************************************************************/
private: System::Void btnRefresh_Click(System::Object^  sender,
System::EventArgs^  e)
      {
            // Створити об'єкт "Лічильник"
            CCounter^ Counter = gcnew CCounter();

            // Установлений прапор підрахунку сумарної кількості екземплярів?
            if(chkTotalInstaceFlag->Checked)
            {
                  // Обчислити
                  lblTotalInstanceNumber->Text =
                        Convert::ToString(Counter-
>GetTotalInstanceNumber(listView));
            }

            // Установлений прапор підрахунку сумарної кількості вільних
екземплярів?
            if(chkFreeInstanceFlag->Checked)
            {
                  // Обчислити
                  lblFreeInstanceNumber->Text =
                        Convert::ToString(Counter-
>GetFreeInstanceNumber(listView));
            }
```

```
            // Встановлений прапор підрахунку документів заданого автора?
            if(chkAuthorDocFlag->Checked)
            {
                  // Створити тимчасовий список
                  ArrayList^ listTemp = gcnew ArrayList();

                  // Одержати список документів заданого автора
                  listTemp = Counter->GetDocListOfAuthor(listView,
txtAuthorFilter->Text);

                  // Очистити список на формі
                  lstAuthorDocList->Items->Clear();
                  // Вивести список знайдених документів
                  for(Int32 i = 0; i < listTemp->Count; i++)
                  {
                        // Зчитати параметри документа й додати в список на
формі
                        lstAuthorDocList->Items->Add(
                             Convert::ToString(i + 1) + ". '" +
                             ((CDoc^)listTemp[i])->GetName() + "', '" +
                             ((CDoc^)listTemp[i])->GetAuthor() + "', що входить
номер документа " +
                             ((CDoc^)listTemp[i])->GetISBN());
                  }

                  // Очистити тимчасовий список
                  listTemp->Clear();
            }

            // Установлений прапор виводу списку документів по заданій темі?
            if(chkThemeDocFlag->Checked)
            {
                  // Створити тимчасовий список
                  ArrayList^ listTemp = gcnew ArrayList();

                  // Одержить список документів по заданій темі
                  listTemp = Counter->GetDocListOnTheme(listView,
txtThemeFilter->Text);

                  // Очистити список для виводу на формі
                  lstThemeDocList->Items->Clear();
                  // Вивести список знайдених документів
                  for(Int32 i = 0; i < listTemp->Count; i++)
                  {
                        // Завантажити параметри документа й додати в список на
формі
                        lstThemeDocList->Items->Add(
                             Convert::ToString(i + 1) + ". '" +
                             ((CDoc^)listTemp[i])->GetName() + "', '" +
                             ((CDoc^)listTemp[i])->GetTheme() + "', що входить
номер документа " +
                             ((CDoc^)listTemp[i])->GetISBN());
                  }
                  // Очистити тимчасовий список
                  listTemp->Clear();
            }
            // Установлений прапор виводу списку документів із заданою
кількістю сторінок?
            if(chkPagesFlag->Checked)
            {
                  // Створити тимчасовий список
                  ArrayList^ listTemp = gcnew ArrayList();

                  // Одержати список документів, у яких кількість сторінок
задовольняє
                  //  заданій умові
                  listTemp = Counter->GetDocListByPages(listView,
                        cmbPagesDirect->Text, Convert::ToInt32(nmrPagesNumber-
>Value));
```

```
                    // Очистити список для виводу на формі
                    lstPagesDocList->Items->Clear();
                    // Вивести список знайдених документів
                    for(Int32 i = 0; i < listTemp->Count; i++)
                    {
                          // Завантажити параметри документа й додати в список на
формі
                          lstPagesDocList->Items->Add(
                               Convert::ToString(i + 1) + ". '" +
                               ((CDoc^)listTemp[i])->GetName() + "', " +
                               Convert::ToString(((CDoc^)listTemp[i])-
>GetPages())
                               + " стор., що входить номер документа " +
((CDoc^)listTemp[i])->GetISBN());
                    }
                    // Очистити тимчасовий список
                    listTemp->Clear();
              }
              // Вивести стан
              lblStatus->Text = "Відновлення необхідних звітів закінчене";
        }
/************************************************************************/
private: System::Void txtPosition_KeyDown(System::Object^  sender,
System::Windows::Forms::KeyEventArgs^  e)
        {
              // Натиснута клавіша "Enter"?
              if( e-e->KeyCode == Keys::Enter)
              {
                    try
                    {
                          // Спробувати одержати введений номер документа для
перегляду
                          ddCurrentIndex = Convert::ToInt32(txtPosition->Text) -
1;
                    }
                    catch(...)
                    {
                          // Якщо в процесі перетворення номера з рядка в число
                          //  виникла помилка, те ніяк на це не реагувати
                    }
                    // Обновити форму
                    UpdateView();
              }
              else
              {
                    // Текстове поле містить символ "/"?
                    if(txtPosition->Text->IndexOf("/") != -1)
                    {
                          // Для початку введення нового номера позиції,
необхідно
                          //  очистити текстове поле
                          txtPosition->Text = "";
                    }
              }
        }
/************************************************************************/
private: System::Void menuHelpAbout_Click(System::Object^  sender,
System::EventArgs^  e)
        {
              // Відобразити форму з інформацією про програму
              UsrsManager::frmAbout^ frmNew = gcnew UsrsManager::frmAbout();
              frmNew->Show();
        }
/************************************************************************/
};
}
```

Файл Reader.cpp - вікно користувача

```
/*****************************************************************************/
#include "StdAfx.h"
#include "Reader.h"
#include "Worker.h"
#include "Logger.h"
/*****************************************************************************/
using namespace System;
using namespace System::IO;
using namespace System::Collections;
using namespace System::Windows::Forms;
/*****************************************************************************/
#define READER_FILE_EXTENSION ".LMR"
/*****************************************************************************/
// NAME:          CReader
// DESCRIPTION:   Конструктор класу
// INPUT:         strUserName - ім'я облікового запису користувача
/*****************************************************************************/
CReader::CReader(String^ strUserName)
{
      // Сформувати ім'я файлу облікового запису
      strFileName = strUserName + READER_FILE_EXTENSION;
      // Створити список для документів користувача
      listMyDoc = gcnew ArrayList();
      // Створити об'єкт для ведення історії
      logUsing = gcnew CLogger();
}
/*****************************************************************************/
// NAME:          Load
// DESCRIPTION:   Функція завантаження списку документів користувача з файлу
// INPUT:         Worker - об'єкт, через який здійснюється одержання інформації
про
//                         книзі по вхідному номеру документа
/*****************************************************************************/
Boolean CReader::Load(CWorker^ Worker)
{
      // Перевірка існування файлу користувача
      if(!File::Exists(strFileName))
      {
            // Видати запит на створення нового файлу
            if(MessageBox::Show("Файл користувача не знайдений, створити
новий?",
                  "Система УЕД", MessageBoxButtons::YesNo,
                  MessageBoxIcon::Warning) == DialogResult::Yes)
            {
                  // Створити файловий потік
                  FileStream^ fsReaderFile = gcnew FileStream(strFileName,
                        FileMode::CreateNew);
                  // Відкрити файл користувача
                  StreamWriter^ swReaderFile = gcnew StreamWriter(fsReaderFile);
                  // Кількість документів у користувача дорівнює 0
                  swReaderFile->WriteLine("0");
                  // Закрити файл
                  swReaderFile->Close();
                  // Закрити файловий потік
                  fsReaderFile->Close();
                  // Закінчити виконання функції, але виконати вхід
                  return true;
            }
            else
            {
                  // Закінчити виконання функції й не виконувати вхід
                  return false;
            }
      }
      // Відкрити файл користувача
```

```
	StreamReader^ srReaderFile = gcnew StreamReader(strFileName);
	// Завантажити кількість документів
	Int32 ddDocsNumber = Convert::ToInt32(srReaderFile->ReadLine());
	// Цикл зчитування інформації про документи
	for(Int32 i = 0; i < ddDocsNumber; i++)
	{
		// Зчитування вхідний номер документа з файлу
		Int32 ddISBNHash = Convert::ToInt32(srReaderFile->ReadLine());

		// Одержання інформації й додавання документи в список документів
користувача
		listMyDoc->Add(Worker->ViewDoc(ddISBNHash));
	}

	// Кінець файлу не досягнуть?
	if(!srReaderFile->EndOfStream)
	{
		// Завантажити історію одержань і повернень документів
		logUsing->strLog = srReaderFile->ReadToEnd();
	}
	else
	{
		// Обнулити рядок історії
		logUsing->strLog = "";
	}

	// Закрити файл користувача
	srReaderFile->Close();

	// Закінчити виконання функції й виконати вхід
	return true;
}
/*************************************************************************/
// NAME:          Save
// DESCRIPTION:   Функція збереження списку документів користувача у файл
// INPUT:         N/D
/*************************************************************************/
void CReader::Save()
{
	// Перевірка існування файлу користувача
	if(File::Exists(strFileName))
	{
		// Видалення старого файлу
		File::Delete(strFileName);
	}
	// Відкрити новий файл користувача
	StreamWriter^ swReaderFile = gcnew StreamWriter(strFileName);

	// Записати кількість документів
	Int32 ddDocsNumber = listMyDoc->Count;
	swReaderFile->WriteLine(Convert::ToString(ddDocsNumber));
	// Цикл запису інформації про документи
	for(Int32 i = 0; i < ddDocsNumber; i++)
	{
		// Запис вхідний номер документа у файл
		swReaderFile->WriteLine(((CDoc^)listMyDoc[i])->GetISBNHash());
	}
	// Записати лог
	swReaderFile->WriteLine(logUsing->strLog);
	// Закрити файл користувача
	swReaderFile->Close();
}
/*************************************************************************/
// NAME:          GetMyDocsList
// DESCRIPTION:   Функція одержання списку документів користувача
// INPUT:         N/D
/*************************************************************************/
ArrayList^ CReader::GetMyDocsList()
{
```

```
	return listMyDoc;
}
/*************************************************************************/
// NAME:          RequireDoc
// DESCRIPTION:   Функція запиту заданого документа
// INPUT:         Worker - об'єкт "Адміністратор", через який здійснюється
одержання
//                      ddHash - хеш-значення вхідного номера документа, що
потрібно одержати
/*************************************************************************/
Boolean CReader::RequireDoc(CWorker^ Worker, Int32 ddHash)
{
	// Запит на одержання документи до "адміністратора"
	Boolean fResult = Worker->GiveDoc(listMyDoc, ddHash);
	// Результат одержання документів позитивний?
	if(fResult)
	{
		// Індекс останньої документи
		Int32 ddIndex = listMyDoc->Count - 1;
		// Записати операцію в лог
		logUsing->WriteEvent("Отриманий документ '" +
((CDoc^)listMyDoc[ddIndex])->GetName() +
			"', Автор '" + ((CDoc^)listMyDoc[ddIndex])->GetAuthor() +
			"', що входить номер документа " +
((CDoc^)listMyDoc[ddIndex])->GetISBN());

		// Операція пройшла успішно
		return true;
	}
	else
	{
		// Такого документа в наявності не є
		return false;
	}
}
/*************************************************************************/
// NAME:          ReleaseDoc
// DESCRIPTION:   Функція повернення заданого документа
// INPUT:         Worker - об'єкт "Адміністратор", через який здійснюється
повернення
//                      ddIndex – індекс документа в списку користувача
/*************************************************************************/
void CReader::ReleaseDoc(CWorker^ Worker, Int32 ddIndex)
{
	// Повернути документ
	Worker->TakeDoc(listMyDoc, ((CDoc^)listMyDoc[ddIndex])->GetISBNHash());
	// Записати операцію в лог
	logUsing->WriteEvent(
		"Повернутий документ '" + ((CDoc^)listMyDoc[ddIndex])->GetName() +
		"', Автор '" + ((CDoc^)listMyDoc[ddIndex])->GetName() +
		"', що входить номер документа " + ((CDoc^)listMyDoc[ddIndex])-
>GetISBN());
	// Видалити документ зі списку користувача
	listMyDoc->RemoveAt(ddIndex);
}
/*************************************************************************/
// NAME:          ViewLog
// DESCRIPTION:   Функція перегляду історії
// INPUT:         N/D
/*************************************************************************/
String^ CReader::ViewLog()
{
	return logUsing->strLog;
}
/*************************************************************************/
```

Файл Reader.resx - xml опис вікна користувача

```
 <?xml version="1.0" encoding=" utf-8"?>
<root>
  <xsd:schema id="root" xmlns="" xmlns:xsd=" " xmlns:msdata="urn: schemas-
microsoft-com: xml-msdata">
    <xsd:import namespace=" " />
    <xsd:element name="root" msdata:IsDataSet="true">
      <xsd:complexType>
        <xsd:choice maxOccurs="unbounded">
          <xsd:element name="metadata">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0" />
              </xsd:sequence>
              <xsd:attribute name="name" use="required" type="xsd:string" />
              <xsd:attribute name="type" type="xsd:string" />
              <xsd:attribute name="mimetype" type="xsd:string" />
              <xsd:attribute ref="xml:space" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="assembly">
            <xsd:complexType>
              <xsd:attribute name="alias" type="xsd:string" />
              <xsd:attribute name="name" type="xsd:string" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="data">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
                <xsd:element name="comment" type="xsd:string" minOccurs="0"
msdata:Ordinal="2" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required"
msdata:Ordinal="1" />
              <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
              <xsd:attribute name="mimetype" type="xsd:string"
msdata:Ordinal="4" />
              <xsd:attribute ref="xml:space" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="resheader">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required" />
            </xsd:complexType>
          </xsd:element>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
  <resheader name="resmimetype">
    <value>text/ microsoft-resx</value>
  </resheader>
  <resheader name="version">
    <value>2.0</value>
  </resheader>
  <resheader name="reader">
    <value>System.Resources.ResXResourceReader, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <resheader name="writer">
```

```
    <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <metadata name="statusStrip.TrayLocation" type="System.Drawing.Point,
System.Drawing, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a">
    <value>436, 17</value>
  </metadata>
  <metadata name="menuStrip.TrayLocation" type="System.Drawing.Point,
System.Drawing, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a">
    <value>540, 17</value>
  </metadata>
</root>
```

Файл Reader.h - бібліотека для файлу frmAbout.cpp

```
#pragma once
/*************************************************************************/
#include "Worker.h"
#include "Doc.h"
#include "Reader.h"

#include "frmAbout.h"
/*************************************************************************/
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
/*************************************************************************/
namespace UsrsManager
{
	public ref class frmReader : public System::Windows::Forms::Form
	{
/*************************************************************************/
	private:
	CWorker^ Worker;		// Об'єкт класу "Адміністратор"
	CReader^ Reader;		// Об'єкт класу "Користувач"

	ArrayList^ listView;	// Поточний відображуваний список документів

	Int32 ddCurrentIndex;	// Індекс поточного відображуваного елемента

	String^ strUserName;	// Поточне ім'я користувача
/*************************************************************************/
	public:
	frmReader(String^ strReaderName)
	{
		// Ініціалізація елементів керування на формі
		InitializeComponent();

		// Створити список відображуваних документів
		listView = gcnew ArrayList();

		// Поточний індекс не визначений
		ddCurrentIndex = -1;

		// Зберегти ім'я користувача
		strUserName = strReaderName;
	}
/*************************************************************************/
protected: ~frmReader()
	{
		if (components)
		{
			delete components;
		}
	}
/*************************************************************************/
	private: System::Windows::Forms::Button^  btnRequest;
	private: System::Windows::Forms::ToolStripStatusLabel^  lblStatus;
	private: System::Windows::Forms::StatusStrip^  statusStrip;
	private: System::Windows::Forms::TabPage^  tabFinding;
	private: System::Windows::Forms::GroupBox^  groupBox1;
	private: System::Windows::Forms::Label^  lblPagesNumber;
	private: System::Windows::Forms::Label^  lblTheme;
	private: System::Windows::Forms::Label^  lblISBN;
	private: System::Windows::Forms::Label^  lblAuthor;
	private: System::Windows::Forms::Label^  lblDocName;
	private: System::Windows::Forms::GroupBox^  groupFinding;
```

```
        private: System::Windows::Forms::Button^  btnFind;
        private: System::Windows::Forms::TextBox^  txtFindString;
        private: System::Windows::Forms::TabControl^  tabControl;
        private: System::Windows::Forms::TabPage^  tabMyDocs;
        private: System::Windows::Forms::Button^  btnReturn;
        private: System::Windows::Forms::GroupBox^  groupBox2;
        private: System::Windows::Forms::ListBox^  lstMyDocs;
        private: System::Windows::Forms::TabPage^  tabHistory;
        private: System::Windows::Forms::GroupBox^  groupBox3;
        private: System::Windows::Forms::TextBox^  txtHistory;
        private: System::Windows::Forms::ToolStripMenuItem^  menuHelp;

        private: System::Windows::Forms::ToolStripMenuItem^  menuHelpAbout;
        private: System::Windows::Forms::ToolStripMenuItem^  menuFileExit;
        private: System::Windows::Forms::MenuStrip^  menuStrip;
        private: System::Windows::Forms::ToolStripMenuItem^  menuFile;
        private: System::Windows::Forms::ToolStripMenuItem^  menuFileSave;
        private: System::Windows::Forms::TextBox^  txtPosition;
        private: System::Windows::Forms::Button^  btnFirst;
        private: System::Windows::Forms::Button^  btnPrev;
        private: System::Windows::Forms::Button^  btnNext;
        private: System::Windows::Forms::Button^  btnLast;
        private: System::Windows::Forms::TextBox^  txtPagesNumber;
        private: System::Windows::Forms::TextBox^  txtTheme;
        private: System::Windows::Forms::TextBox^  txtISBN;
        private: System::Windows::Forms::TextBox^  txtAuthor;
        private: System::Windows::Forms::TextBox^  txtDocName;

        private:
        /// <summary>
        /// Required designer variable.
        /// </summary>
        System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
        /// <summary>
        /// Необхідний метод для підтримки Розроблювача - не модифікувати
        /// зміст цього методу з кодовим редактором.
        /// </summary>
        void InitializeComponent(void)
        {
                this->btnRequest = (gcnew System::Windows::Forms::Button());
                this->lblStatus = (gcnew
System::Windows::Forms::ToolStripStatusLabel());
                this->statusStrip = (gcnew System::Windows::Forms::StatusStrip());
                this->tabFinding = (gcnew System::Windows::Forms::TabPage());
                this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());
                this->txtPosition = (gcnew System::Windows::Forms::TextBox());
                this->btnFirst = (gcnew System::Windows::Forms::Button());
                this->btnPrev = (gcnew System::Windows::Forms::Button());
                this->btnNext = (gcnew System::Windows::Forms::Button());
                this->btnLast = (gcnew System::Windows::Forms::Button());
                this->txtPagesNumber = (gcnew System::Windows::Forms::TextBox());
                this->txtTheme = (gcnew System::Windows::Forms::TextBox());
                this->txtISBN = (gcnew System::Windows::Forms::TextBox());
                this->txtAuthor = (gcnew System::Windows::Forms::TextBox());
                this->txtDocName = (gcnew System::Windows::Forms::TextBox());
                this->lblPagesNumber = (gcnew System::Windows::Forms::Label());
                this->lblTheme = (gcnew System::Windows::Forms::Label());
                this->lblISBN = (gcnew System::Windows::Forms::Label());
                this->lblAuthor = (gcnew System::Windows::Forms::Label());
                this->lblDocName = (gcnew System::Windows::Forms::Label());
                this->groupFinding = (gcnew System::Windows::Forms::GroupBox());
                this->btnFind = (gcnew System::Windows::Forms::Button());
                this->txtFindString = (gcnew System::Windows::Forms::TextBox());
                this->tabControl = (gcnew System::Windows::Forms::TabControl());
                this->tabMyDocs = (gcnew System::Windows::Forms::TabPage());
                this->btnReturn = (gcnew System::Windows::Forms::Button());
                this->groupBox2 = (gcnew System::Windows::Forms::GroupBox());
```

```
            this->lstMyDocs = (gcnew System::Windows::Forms::ListBox());
            this->tabHistory = (gcnew System::Windows::Forms::TabPage());
            this->groupBox3 = (gcnew System::Windows::Forms::GroupBox());
            this->txtHistory = (gcnew System::Windows::Forms::TextBox());
            this->menuHelp = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuHelpAbout = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuFileExit = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuStrip = (gcnew System::Windows::Forms::MenuStrip());
            this->menuFile = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuFileSave = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->statusStrip->SuspendLayout();
            this->tabFinding->SuspendLayout();
            this->groupBox 1-1->SuspendLayout();
            this->groupFinding->SuspendLayout();
            this->tabControl->SuspendLayout();
            this->tabMyDocs->SuspendLayout();
            this->groupBox 2-2->SuspendLayout();
            this->tabHistory->SuspendLayout();
            this->groupBox 3-3->SuspendLayout();
            this->menuStrip->SuspendLayout();
            this->SuspendLayout();
            //
            // btnRequest
            //
            this->btnRequest->Location = System::Drawing::Point(567, 285);
            this->btnRequest->Name = L"btnRequest";
            this->btnRequest->Size = System::Drawing::Size(144, 23);
            this->btnRequest->TabIndex = 12;
            this->btnRequest->Text = L"Здійснити запит";
            this->btnRequest->UseVisualStyleBackColor = true;
            this->btnRequest->Click += gcnew System::EventHandler(this,
&frmReader::btnRequest_Click);
            //
            // lblStatus
            //
            this->lblStatus->Name = L"lblStatus";
            this->lblStatus->Size = System::Drawing::Size(109, 17);
            this->lblStatus->Text = L"toolStripStatusLabel1";
            this->lblStatus->Click += gcnew System::EventHandler(this,
&frmReader::lblStatus_Click);
            //
            // statusStrip
            //
            this->statusStrip->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(1) {this->lblStatus});
            this->statusStrip->Location = System::Drawing::Point(0, 365);
            this->statusStrip->Name = L"statusStrip";
            this->statusStrip->Size = System::Drawing::Size(724, 22);
            this->statusStrip->TabIndex = 13;
            this->statusStrip->Text = L"statusStrip1";
            //
            // tabFinding
            //
            this->tabFinding->Controls->Add(this->btnRequest);
            this->tabFinding->Controls->Add(this->groupBox1);
            this->tabFinding->Controls->Add(this->groupFinding);
            this->tabFinding->Location = System::Drawing::Point(4, 22);
            this->tabFinding->Name = L"tabFinding";
            this->tabFinding->Padding = System::Windows::Forms::Padding(3);
            this->tabFinding->Size = System::Drawing::Size(717, 314);
            this->tabFinding->TabIndex = 0;
            this->tabFinding->Text = L"Пошук";
            this->tabFinding->UseVisualStyleBackColor = true;
            //
```

```
            // groupBox1
            //
            this->groupBox 1-1->Controls->Add(this->txtPosition);
            this->groupBox 1-1->Controls->Add(this->btnFirst);
            this->groupBox 1-1->Controls->Add(this->btnPrev);
            this->groupBox 1-1->Controls->Add(this->btnNext);
            this->groupBox 1-1->Controls->Add(this->btnLast);
            this->groupBox 1-1->Controls->Add(this->txtPagesNumber);
            this->groupBox 1-1->Controls->Add(this->txtTheme);
            this->groupBox 1-1->Controls->Add(this->txtISBN);
            this->groupBox 1-1->Controls->Add(this->txtAuthor);
            this->groupBox 1-1->Controls->Add(this->txtDocName);
            this->groupBox 1-1->Controls->Add(this->lblPagesNumber);
            this->groupBox 1-1->Controls->Add(this->lblTheme);
            this->groupBox 1-1->Controls->Add(this->lblISBN);
            this->groupBox 1-1->Controls->Add(this->lblAuthor);
            this->groupBox 1-1->Controls->Add(this->lblDocName);
            this->groupBox 1-1->Location = System::Drawing::Point(6, 65);
            this->groupBox 1-1->Name = L"groupBox1";
            this->groupBox 1-1->Size = System::Drawing::Size(702, 214);
            this->groupBox 1-1->TabIndex = 9;
            this->groupBox 1-1->TabStop = false;
            this->groupBox 1-1->Text = L"Інформація про документ";
            //
            // txtPosition
            //
            this->txtPosition->Location = System::Drawing::Point(568, 187);
            this->txtPosition->Name = L"txtPosition";
            this->txtPosition->Size = System::Drawing::Size(58, 20);
            this->txtPosition->TabIndex = 9;
            this->txtPosition->TextAlign =
System::Windows::Forms::HorizontalAlignment::Center;
            this->txtPosition->KeyDown += gcnew
System::Windows::Forms::KeyEventHandler(this, &frmReader::txtPosition_KeyDown);
            //
            // btnFirst
            //
            this->btnFirst->Location = System::Drawing::Point(494, 187);
            this->btnFirst->Name = L"btnFirst";
            this->btnFirst->Size = System::Drawing::Size(31, 20);
            this->btnFirst->TabIndex = 7;
            this->btnFirst->Text = L"|<<";
            this->btnFirst->UseVisualStyleBackColor = true;
            this->btnFirst->Click += gcnew System::EventHandler(this,
&frmReader::btnFirst_Click);
            //
            // btnPrev
            //
            this->btnPrev->Location = System::Drawing::Point(531, 187);
            this->btnPrev->Name = L"btnPrev";
            this->btnPrev->Size = System::Drawing::Size(31, 20);
            this->btnPrev->TabIndex = 8;
            this->btnPrev->Text = L"<<";
            this->btnPrev->UseVisualStyleBackColor = true;
            this->btnPrev->Click += gcnew System::EventHandler(this,
&frmReader::btnPrev_Click);
            //
            // btnNext
            //
            this->btnNext->Location = System::Drawing::Point(632, 186);
            this->btnNext->Name = L"btnNext";
            this->btnNext->Size = System::Drawing::Size(31, 20);
            this->btnNext->TabIndex = 10;
            this->btnNext->Text = L">>";
            this->btnNext->UseVisualStyleBackColor = true;
            this->btnNext->Click += gcnew System::EventHandler(this,
&frmReader::btnNext_Click);
            //
            // btnLast
```

```
            //
            this->btnLast->Location = System::Drawing::Point(669, 186);
            this->btnLast->Name = L"btnLast";
            this->btnLast->Size = System::Drawing::Size(31, 20);
            this->btnLast->TabIndex = 11;
            this->btnLast->Text = L">>|";
            this->btnLast->UseVisualStyleBackColor = true;
            this->btnLast->Click += gcnew System::EventHandler(this,
&frmReader::btnLast_Click);
            //
            // txtPagesNumber
            //
            this->txtPagesNumber->Location = System::Drawing::Point(139, 161);
            this->txtPagesNumber->Name = L"txtPagesNumber";
            this->txtPagesNumber->ReadOnly = true;
            this->txtPagesNumber->Size = System::Drawing::Size(48, 20);
            this->txtPagesNumber->TabIndex = 6;
            //
            // txtTheme
            //
            this->txtTheme->Location = System::Drawing::Point(139, 128);
            this->txtTheme->Name = L"txtTheme";
            this->txtTheme->ReadOnly = true;
            this->txtTheme->Size = System::Drawing::Size(302, 20);
            this->txtTheme->TabIndex = 5;
            //
            // txtISBN
            //
            this->txtISBN->Location = System::Drawing::Point(139, 94);
            this->txtISBN->Name = L"txtISBN";
            this->txtISBN->ReadOnly = true;
            this->txtISBN->Size = System::Drawing::Size(163, 20);
            this->txtISBN->TabIndex = 4;
            //
            // txtAuthor
            //
            this->txtAuthor->Location = System::Drawing::Point(139, 60);
            this->txtAuthor->Name = L"txtAuthor";
            this->txtAuthor->ReadOnly = true;
            this->txtAuthor->Size = System::Drawing::Size(505, 20);
            this->txtAuthor->TabIndex = 3;
            //
            // txtDocName
            //
            this->txtDocName->Location = System::Drawing::Point(139, 26);
            this->txtDocName->Name = L"txtDocName";
            this->txtDocName->ReadOnly = true;
            this->txtDocName->Size = System::Drawing::Size(557, 20);
            this->txtDocName->TabIndex = 2;
            //
            // lblPagesNumber
            //
            this->lblPagesNumber->AutoSize = true;
            this->lblPagesNumber->Location = System::Drawing::Point(40, 164);
            this->lblPagesNumber->Name = L"lblPagesNumber";
            this->lblPagesNumber->Size = System::Drawing::Size(99, 13);
            this->lblPagesNumber->TabIndex = 4;
            this->lblPagesNumber->Text = L"Кількість сторінок";
            //
            // lblTheme
            //
            this->lblTheme->AutoSize = true;
            this->lblTheme->Location = System::Drawing::Point(108, 128);
            this->lblTheme->Name = L"lblTheme";
            this->lblTheme->Size = System::Drawing::Size(31, 13);
            this->lblTheme->TabIndex = 3;
            this->lblTheme->Text = L"Тема";
            //
            // lblISBN
```

```
            //
            this->lblISBN->AutoSize = true;
            this->lblISBN->Location = System::Drawing::Point(2, 94);
            this->lblISBN->Name = L"lblISBN";
            this->lblISBN->Size = System::Drawing::Size(137, 13);
            this->lblISBN->TabIndex = 2;
            this->lblISBN->Text = L"Вхідний номер документу";
            //
            // lblAuthor
            //
            this->lblAuthor->AutoSize = true;
            this->lblAuthor->Location = System::Drawing::Point(101, 60);
            this->lblAuthor->Name = L"lblAuthor";
            this->lblAuthor->Size = System::Drawing::Size(38, 13);
            this->lblAuthor->TabIndex = 1;
            this->lblAuthor->Text = L"Автор";
            //
            // lblDocName
            //
            this->lblDocName->AutoSize = true;
            this->lblDocName->Location = System::Drawing::Point(44, 26);
            this->lblDocName->Name = L"lblDocName";
            this->lblDocName->Size = System::Drawing::Size(95, 13);
            this->lblDocName->TabIndex = 0;
            this->lblDocName->Text = L"Назва документу";
            //
            // groupFinding
            //
            this->groupFinding->Controls->Add(this->btnFind);
            this->groupFinding->Controls->Add(this->txtFindString);
            this->groupFinding->Location = System::Drawing::Point(6, 6);
            this->groupFinding->Name = L"groupFinding";
            this->groupFinding->Size = System::Drawing::Size(702, 53);
            this->groupFinding->TabIndex = 8;
            this->groupFinding->TabStop = false;
            this->groupFinding->Text = L"Ключові рядки для пошуку";
            //
            // btnFind
            //
            this->btnFind->Location = System::Drawing::Point(621, 19);
            this->btnFind->Name = L"btnFind";
            this->btnFind->Size = System::Drawing::Size(75, 23);
            this->btnFind->TabIndex = 1;
            this->btnFind->Text = L"Знайти";
            this->btnFind->UseVisualStyleBackColor = true;
            this->btnFind->Click += gcnew System::EventHandler(this,
&frmReader::btnFind_Click);
            //
            // txtFindString
            //
            this->txtFindString->Location = System::Drawing::Point(6, 19);
            this->txtFindString->Name = L"txtFindString";
            this->txtFindString->Size = System::Drawing::Size(609, 20);
            this->txtFindString->TabIndex = 0;
            //
            // tabControl
            //
            this->tabControl->Controls->Add(this->tabFinding);
            this->tabControl->Controls->Add(this->tabMyDocs);
            this->tabControl->Controls->Add(this->tabHistory);
            this->tabControl->Location = System::Drawing::Point(0, 27);
            this->tabControl->Name = L"tabControl";
            this->tabControl->SelectedIndex = 0;
            this->tabControl->Size = System::Drawing::Size(725, 340);
            this->tabControl->TabIndex = 14;
            //
            // tabMyDocs
            //
            this->tabMyDocs->Controls->Add(this->btnReturn);
```

```
            this->tabMyDocs->Controls->Add(this->groupBox2);
            this->tabMyDocs->Location = System::Drawing::Point(4, 22);
            this->tabMyDocs->Name = L"tabMyDocs";
            this->tabMyDocs->Padding = System::Windows::Forms::Padding(3);
            this->tabMyDocs->Size = System::Drawing::Size(717, 314);
            this->tabMyDocs->TabIndex = 1;
            this->tabMyDocs->Text = L"Документи";
            this->tabMyDocs->UseVisualStyleBackColor = true;
            //
            // btnReturn
            //
            this->btnReturn->Location = System::Drawing::Point(622, 285);
            this->btnReturn->Name = L"btnReturn";
            this->btnReturn->Size = System::Drawing::Size(85, 23);
            this->btnReturn->TabIndex = 1;
            this->btnReturn->Text = L"Повернути";
            this->btnReturn->UseVisualStyleBackColor = true;
            this->btnReturn->Click += gcnew System::EventHandler(this,
&frmReader::btnReturn_Click);
            //
            // groupBox2
            //
            this->groupBox 2-2->Controls->Add(this->lstMyDocs);
            this->groupBox 2-2->Location = System::Drawing::Point(8, 6);
            this->groupBox 2-2->Name = L"groupBox2";
            this->groupBox 2-2->Size = System::Drawing::Size(699, 273);
            this->groupBox 2-2->TabIndex = 0;
            this->groupBox 2-2->TabStop = false;
            this->groupBox 2-2->Text = L"Список документів користувача";
            this->groupBox 2-2->Enter += gcnew System::EventHandler(this,
&frmReader::groupBox2_Enter);
            //
            // lstMyDocs
            //
            this->lstMyDocs->FormattingEnabled = true;
            this->lstMyDocs->HorizontalScrollbar = true;
            this->lstMyDocs->Location = System::Drawing::Point(6, 19);
            this->lstMyDocs->Name = L"lstMyDocs";
            this->lstMyDocs->Size = System::Drawing::Size(687, 251);
            this->lstMyDocs->TabIndex = 0;
            //
            // tabHistory
            //
            this->tabHistory->Controls->Add(this->groupBox3);
            this->tabHistory->Location = System::Drawing::Point(4, 22);
            this->tabHistory->Name = L"tabHistory";
            this->tabHistory->Size = System::Drawing::Size(717, 314);
            this->tabHistory->TabIndex = 2;
            this->tabHistory->Text = L"Історія";
            this->tabHistory->UseVisualStyleBackColor = true;
            //
            // groupBox3
            //
            this->groupBox 3-3->Controls->Add(this->txtHistory);
            this->groupBox 3-3->Location = System::Drawing::Point(8, 8);
            this->groupBox 3-3->Name = L"groupBox3";
            this->groupBox 3-3->Size = System::Drawing::Size(700, 303);
            this->groupBox 3-3->TabIndex = 2;
            this->groupBox 3-3->TabStop = false;
            this->groupBox 3-3->Text = L"Історія перегляду документів";
            //
            // txtHistory
            //
            this->txtHistory->Location = System::Drawing::Point(6, 19);
            this->txtHistory->Multiline = true;
            this->txtHistory->Name = L"txtHistory";
            this->txtHistory->ReadOnly = true;
            this->txtHistory->ScrollBars =
System::Windows::Forms::ScrollBars::Both;
```

```
            this->txtHistory->Size = System::Drawing::Size(688, 278);
            this->txtHistory->TabIndex = 0;
            //
            // menuHelp
            //
            this->menuHelp->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(1) {this->menuHelpAbout});
            this->menuHelp->Name = L"menuHelp";
            this->menuHelp->Size = System::Drawing::Size(60, 20);
            this->menuHelp->Text = L"Довідка";
            //
            // menuHelpAbout
            //
            this->menuHelpAbout->Name = L"menuHelpAbout";
            this->menuHelpAbout->Size = System::Drawing::Size(166, 22);
            this->menuHelpAbout->Text = L"Про програму...";
            this->menuHelpAbout->Click += gcnew System::EventHandler(this,
&frmReader::menuHelpAbout_Click);
            //
            // menuFileExit
            //
            this->menuFileExit->Name = L"menuFileExit";
            this->menuFileExit->Size = System::Drawing::Size(152, 22);
            this->menuFileExit->Text = L"Вихід";
            this->menuFileExit->Click += gcnew System::EventHandler(this,
&frmReader::menuFileExit_Click);
            //
            // menuStrip
            //
            this->menuStrip->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(2) {this->menuFile, this->menuHelp});
            this->menuStrip->Location = System::Drawing::Point(0, 0);
            this->menuStrip->Name = L"menuStrip";
            this->menuStrip->Size = System::Drawing::Size(724, 24);
            this->menuStrip->TabIndex = 12;
            this->menuStrip->Text = L"menuStrip1";
            //
            // menuFile
            //
            this->menuFile->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(2) {this->menuFileSave,
                this->menuFileExit});
            this->menuFile->Name = L"menuFile";
            this->menuFile->Size = System::Drawing::Size(45, 20);
            this->menuFile->Text = L"Файл";
            //
            // menuFileSave
            //
            this->menuFileSave->Name = L"menuFileSave";
            this->menuFileSave->Size = System::Drawing::Size(152, 22);
            this->menuFileSave->Text = L"Зберегти";
            this->menuFileSave->Click += gcnew System::EventHandler(this,
&frmReader::menuFileSave_Click);
            //
            // frmReader
            //
            this->AutoScaleDimensions = System::Drawing::Size(6, 13);
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(724, 387);
            this->Controls->Add(this->statusStrip);
            this->Controls->Add(this->tabControl);
            this->Controls->Add(this->menuStrip);
            this->MaximizeBox = false;
            this->MinimizeBox = false;
            this->Name = L"frmReader";
            this->Text = L"Режим  користувача - Система Управління Електронним
Документообігом ";
            this->Load += gcnew System::EventHandler(this,
&frmReader::frmReader_Load);
```

```
            this->FormClosed += gcnew
System::Windows::Forms::FormClosedEventHandler(this,
&frmReader::frmReader_FormClosed);
            this->statusStrip->ResumeLayout(false);
            this->statusStrip->PerformLayout();
            this->tabFinding->ResumeLayout(false);
            this->groupBox 1-1->ResumeLayout(false);
            this->groupBox 1-1->PerformLayout();
            this->groupFinding->ResumeLayout(false);
            this->groupFinding->PerformLayout();
            this->tabControl->ResumeLayout(false);
            this->tabMyDocs->ResumeLayout(false);
            this->groupBox 2-2->ResumeLayout(false);
            this->tabHistory->ResumeLayout(false);
            this->groupBox 3-3->ResumeLayout(false);
            this->groupBox 3-3->PerformLayout();
            this->menuStrip->ResumeLayout(false);
            this->menuStrip->PerformLayout();
            this->ResumeLayout(false);
            this->PerformLayout();

      }
#pragma endregion
 /*************************************************************************/
 private: System::Void UpdateView()
       {
             // Список не порожній?
             if(listView->Count != 0)
             {
                   // Перевірити індекс поточного елемента
                   if(ddCurrentIndex >= listView->Count)
                   {
                         // Установити покажчик на останній елемент
                         ddCurrentIndex = listView->Count - 1;
                   }

                   // Поточний індекс менше нуля?
                   if(ddCurrentIndex < 0)
                   {
                         // Установити індекс на перший елемент
                         ddCurrentIndex = 0;
                   }

                   // Одержати характеристики документа й заповнити поля форми
                   txtDocName->Text = ((CDoc^)listView[ddCurrentIndex])-
>GetName();
                   txtAuthor->Text = ((CDoc^)listView[ddCurrentIndex])-
>GetAuthor();
                   txtISBN->Text = ((CDoc^)listView[ddCurrentIndex])->GetISBN();
                   txtTheme->Text = ((CDoc^)listView[ddCurrentIndex])-
>GetTheme();
                   txtPagesNumber->Text =
Convert::ToString(((CDoc^)listView[ddCurrentIndex])->GetPages());

                   // Документ є в наявності?
                   if(((CDoc^)listView[ddCurrentIndex])->GetFreeNumber() > 0)
                   {
                         // Розблокувати кнопку запросу
                         btnRequest->Enabled = true;
                   }
                   else
                   {
                         // Заблокувати кнопку запросу
                         btnRequest->Enabled = false;
                   }

                   // Установити й вивести поточну позицію
                   txtPosition->Text = Convert::ToString(ddCurrentIndex + 1) + "
/ " +
```

```
                        Convert::ToString(listView->Count);
                }
                else
                {
                        // Очистити всі поля
                        txtDocName->Text = "";
                        txtAuthor->Text = "";
                        txtISBN->Text = "";
                        txtTheme->Text = "";
                        txtPagesNumber->Text = "";
                        // Поточна позиція - 0
                        txtPosition->Text = "0/0";
                        // Установити індекс поточного елемента
                        ddCurrentIndex = 0;
                        // Вивести стан
                        lblStatus->Text = "Немає записів для відображення";
                }

                // Вивести історію
                txtHistory->Text = Reader->ViewLog();

                // Одержати список документів користувача
                ArrayList^ listTemp = gcnew ArrayList();
                listTemp = Reader->GetMyDocsList();

                // Очистити список на формі
                lstMyDocs->Items->Clear();
                // Кількість документів не дорівнює нулю?
                if(listTemp->Count != 0)
                {
                        // Заповнити список документів на формі
                        for(Int32 i = 0; i < listTemp->Count; i++)
                        {
                                // Одержати характеристики документа й вивести їх у
список
                                lstMyDocs->Items->Add(Convert::ToString(i + 1) + ". '"
+
                                        ((CDoc^)listTemp[i])->GetName() + "', '" +
                                        ((CDoc^)listTemp[i])->GetAuthor() + "', вхідний
номер документа " +
                                        ((CDoc^)listTemp[i])->GetISBN());
                        }
                }
        }
/*********************************************************************/
private: System::Void frmReader_Load(System::Object^  sender, System::EventArgs^
e)
        {
                // Створити об'єкт "Адміністратор"
                Worker = gcnew CWorker();
                // Завантажити список документів
                Worker->LoadDocList();

                // Створити об'єкт класу "Користувач"
                Reader = gcnew CReader(strUserName);
                // Завантажити список документів, що перебувають у користувача
                Reader->Load(Worker);

               // Вивести стан
               lblStatus->Text = "Для відображення списку потрібних документів
скористайтеся пошуком";
               // Обновити форму
               UpdateView();
        }
 /*********************************************************************/
private: System::Void btnFind_Click(System::Object^  sender, System::EventArgs^
e)
        {
```

```
            // Забрати з рядку пошуку пробіли, перетворити до нижнього регістра
й зберегти
            String^ strValue = txtFindString->Text->Trim()->ToLower();

            // Перевірка рядка
            if(String::IsNullOrEmpty(strValue))
            {
                  // Рядок порожня, запропонувати вивести весь список
                  if(MessageBox::Show("Рядок пошуку не завдань, вивести весь
список?",
                                                "Система УЕД",
MessageBoxButtons::YesNo,
                                                MessageBoxIcon::Question) ==
::DialogResult::Yes)
                  {
                        // Перейти на вивід списку
                        goto OUT_LIST;
                  }
                  // Завершити роботу функції
                  return;
            }

OUT_LIST:   // Виконати функцію пошуку
            listView = Worker->FindDoc(strValue);

            // Вивести стан
            lblStatus->Text = "Пошук завершено";
            // Обновити форму
            UpdateView();
      }
/*************************************************************************/
private: System::Void btnFirst_Click(System::Object^  sender, System::EventArgs^
e)
      {
            // Установити індекс на перший елемент
            ddCurrentIndex = 0;
            // Обновити форму
            UpdateView();
      }
/*************************************************************************/
private: System::Void btnPrev_Click(System::Object^  sender, System::EventArgs^
e)
      {
            // Установити індекс на попередній елемент
            ddCurrentIndex ---i;
            // Обновити форму
            UpdateView();
      }
/*************************************************************************/
private: System::Void btnNext_Click(System::Object^  sender, System::EventArgs^
e)
      {
            // Установити індекс на наступний елемент
            ddCurrentIndex ++;
            // Обновити форму
            UpdateView();
      }
/*************************************************************************/
private: System::Void btnLast_Click(System::Object^  sender, System::EventArgs^
e)
      {
            // Установити індекс на останній елемент
            ddCurrentIndex = listView->Count - 1;
            // Обновити форму
            UpdateView();
      }
/*************************************************************************/
private: System::Void txtPosition_KeyDown(System::Object^  sender,
System::Windows::Forms::KeyEventArgs^  e)
```

```
	{
		// Натиснута клавіша "Enter"?
		if( e->KeyCode == Keys::Enter)
		{
			try
			{
				// Спробувати одержати введений номер документа для
перегляду
				ddCurrentIndex = Convert::ToInt32(txtPosition->Text) -
1;
			}
			catch(...)
			{
			// Якщо в процесі перетворення номера з рядка в число
			//  виникла помилка, те ніяк на це не реагувати
			}

			// Обновити форму
			UpdateView();
		}
		else
		{
			// Текстове поле містить символ "/"?
			if(txtPosition->Text->IndexOf("/") != -1)
			{
				// Для початку введення нового номера позиції,
необхідно
				//  очистити текстове поле
				txtPosition->Text = "";
			}
		}
	}
/*************************************************************************/
private: System::Void btnRequest_Click(System::Object^  sender,
System::EventArgs^  e)
	{
		// Запросити документ, попередньо перетворивши значення вхідного
номера документа в текстовому
		//  поле до нижнього регістра й забравши бічні пробіли
		if(Reader->RequireDoc(Worker, txtISBN->Text->ToLower()->Trim()-
>GetHashCode()))
		{
			// Запит пройшов вдало, видалити документ із поточного списку
перегляду
			listView->RemoveAt(ddCurrentIndex);

			// Обновити форму
			UpdateView();
			// Вивести стан
			lblStatus->Text = "Документ успішно отримано";
		}
		else
		{
			// Неможливо одержати документ, вивести повідомлення про
цьому
			lblStatus->Text = "Вибачте, але на даний момент немає жодного
документу по даному запиту";
		}
	}
/*************************************************************************/
private: System::Void btnReturn_Click(System::Object^  sender,
System::EventArgs^  e)
	{
		// У списку документів є обраний елемент?
		if(lstMyDocs->SelectedIndex != -1)
		{
			// Повернути обрану документ
			Reader->ReleaseDoc(Worker, lstMyDocs->SelectedIndex);
```

```
                // Обновити форму
                UpdateView();
                // Вивести стан
                lblStatus->Text = "Документ виконано";
            }
            else
            {
                // Вивести повідомлення про помилку
                lblStatus->Text = "Спочатку виберіть документ для виконання";
            }
        }
/*************************************************************************/
private: System::Void menuFileSave_Click(System::Object^  sender,
System::EventArgs^  e)
        {
             // Збереження модифікованого списку документів у файл
            Worker->SaveDocList();

            // Збереження списку документів користувача
            Reader->Save();
        }
/*************************************************************************/
private: System::Void menuFileExit_Click(System::Object^  sender,
System::EventArgs^  e)
        {
            // Закрити додаток
            Application::Exit();
        }
/*************************************************************************/
private: System::Void frmReader_FormClosed(System::Object^  sender,
System::Windows::Forms::FormClosedEventArgs^  e)
        {
            // Закрити додаток
            Application::Exit();
        }
/*************************************************************************/
private: System::Void menuHelpAbout_Click(System::Object^  sender,
System::EventArgs^  e)
        {
            // Відобразити форму з інформацією про програму
            UsrsManager::frmAbout^ frmNew = gcnew UsrsManager::frmAbout();
            frmNew->Show();
        }
/*************************************************************************/
private: System::Void groupBox2_Enter(System::Object^  sender,
System::EventArgs^  e) {
        }
private: System::Void lblStatus_Click(System::Object^  sender,
System::EventArgs^  e) {
        }
};
}
```

Файл frmLogin.resx - xml опис форми «Вхід у систему УЕД»

```
<?xml version="1.0" encoding=" utf-8"?>
<root>
    <xsd:schema id="root" xmlns="" xmlns:xsd=" " xmlns:msdata="urn: schemas-
microsoft-com: xml-msdata">
    <xsd:import namespace=" " />
    <xsd:element name="root" msdata:IsDataSet="true">
      <xsd:complexType>
        <xsd:choice maxOccurs="unbounded">
          <xsd:element name="metadata">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0" />
              </xsd:sequence>
              <xsd:attribute name="name" use="required" type="xsd:string" />
              <xsd:attribute name="type" type="xsd:string" />
              <xsd:attribute name="mimetype" type="xsd:string" />
              <xsd:attribute ref="xml:space" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="assembly">
            <xsd:complexType>
              <xsd:attribute name="alias" type="xsd:string" />
              <xsd:attribute name="name" type="xsd:string" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="data">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
                <xsd:element name="comment" type="xsd:string" minOccurs="0"
msdata:Ordinal="2" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required"
msdata:Ordinal="1" />
              <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
              <xsd:attribute name="mimetype" type="xsd:string"
msdata:Ordinal="4" />
              <xsd:attribute ref="xml:space" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="resheader">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required" />
            </xsd:complexType>
          </xsd:element>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
  <resheader name="resmimetype">
    <value>text/ microsoft-resx</value>
  </resheader>
  <resheader name="version">
    <value>2.0</value>
  </resheader>
  <resheader name="reader">
    <value>System.Resources.ResXResourceReader, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <resheader name="writer">
    <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
</root>
```

Файл frmLogin.h - бібліотека для файлу frmLogin.resx

```
/*************************************************************************/
#pragma once
/*************************************************************************/
#include "frmWorker.h"
#include "frmReader.h"
#include "Usrs.h"
/*************************************************************************/
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
/*************************************************************************/
#define WORKER_LOGIN    "worker"
/*************************************************************************/
namespace UsrsManager
{
      public ref class frmLogin : public System::Windows::Forms::Form
      {
/*************************************************************************/
      private:
            CUsrs^ Usrs;                  // Керуючий об'єкт класу "Система"
/*************************************************************************/
public:     frmLogin(void)
            {
                  InitializeComponent();
                  //
                  //Додавання цього коду в конструктор
                  //
            }
/*************************************************************************/
protected: ~frmLogin()
            {
                  if (components)
                  {
                        delete components;
                  }
            }
/*************************************************************************/
      private: System::Windows::Forms::GroupBox^  groupBox;
      private: System::Windows::Forms::TextBox^  txtPassword;
      private: System::Windows::Forms::TextBox^  txtLogin;
      private: System::Windows::Forms::Label^  lblPassword;
      private: System::Windows::Forms::Label^  lblLogin;
      private: System::Windows::Forms::Button^  btnEnter;
      private: System::Windows::Forms::Button^  btnExit;
      private: System::Windows::Forms::Button^  btnDelete;
      private: System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
            /// <summary>
            /// Необхідний метод для підтримки Розроблювача - не модифікувати
            /// зміст цього методу з кодовим редактором.
            /// </summary>
            void InitializeComponent(void)
            {
                  this->groupBox = (gcnew System::Windows::Forms::GroupBox());
                  this->txtPassword = (gcnew System::Windows::Forms::TextBox());
                  this->txtLogin = (gcnew System::Windows::Forms::TextBox());
                  this->lblPassword = (gcnew System::Windows::Forms::Label());
                  this->lblLogin = (gcnew System::Windows::Forms::Label());
                  this->btnEnter = (gcnew System::Windows::Forms::Button());
                  this->btnExit = (gcnew System::Windows::Forms::Button());
                  this->btnDelete = (gcnew System::Windows::Forms::Button());
                  this->groupBox->SuspendLayout();
                  this->SuspendLayout();
```

```
                    //
                    // groupBox
                    //
                    this->groupBox->Controls->Add(this->txtPassword);
                    this->groupBox->Controls->Add(this->txtLogin);
                    this->groupBox->Controls->Add(this->lblPassword);
                    this->groupBox->Controls->Add(this->lblLogin);
                    this->groupBox->Location = System::Drawing::Point(12, 11);
                    this->groupBox->Name = L"groupBox";
                    this->groupBox->Size = System::Drawing::Size(237, 80);
                    this->groupBox->TabIndex = 8;
                    this->groupBox->TabStop = false;
                    this->groupBox->Text = L"Параметри облікового запису";
                    //
                    // txtPassword
                    //
                    this->txtPassword->Location = System::Drawing::Point(57, 45);
                    this->txtPassword->Name = L"txtPassword";
                    this->txtPassword->PasswordChar = '*';
                    this->txtPassword->Size = System::Drawing::Size(174, 20);
                    this->txtPassword->TabIndex = 1;
                    //
                    // txtLogin
                    //
                    this->txtLogin->Location = System::Drawing::Point(57, 19);
                    this->txtLogin->Name = L"txtLogin";
                    this->txtLogin->Size = System::Drawing::Size(174, 20);
                    this->txtLogin->TabIndex = 0;
                    //
                    // lblPassword
                    //
                    this->lblPassword->AutoSize = true;
                    this->lblPassword->Location = System::Drawing::Point(6, 48);
                    this->lblPassword->Name = L"lblPassword";
                    this->lblPassword->Size = System::Drawing::Size(44, 13);
                    this->lblPassword->TabIndex = 1;
                    this->lblPassword->Text = L"Пароль";
                    //
                    // lblLogin
                    //
                    this->lblLogin->AutoSize = true;
                    this->lblLogin->Location = System::Drawing::Point(13, 19);
                    this->lblLogin->Name = L"lblLogin";
                    this->lblLogin->Size = System::Drawing::Size(33, 13);
                    this->lblLogin->TabIndex = 0;
                    this->lblLogin->Text = L"Логін";
                    //
                    // btnEnter
                    //
                    this->btnEnter->Location = System::Drawing::Point(12, 98);
                    this->btnEnter->Name = L"btnEnter";
                    this->btnEnter->Size = System::Drawing::Size(75, 23);
                    this->btnEnter->TabIndex = 2;
                    this->btnEnter->Text = L"Вхід";
                    this->btnEnter->UseVisualStyleBackColor = true;
                    this->btnEnter->Click += gcnew System::EventHandler(this,
&frmLogin::btnEnter_Click);
                    //
                    // btnExit
                    //
                    this->btnExit->Location = System::Drawing::Point(174, 98);
                    this->btnExit->Name = L"btnExit";
                    this->btnExit->Size = System::Drawing::Size(75, 23);
                    this->btnExit->TabIndex = 3;
                    this->btnExit->Text = L"Вихід";
                    this->btnExit->UseVisualStyleBackColor = true;
                    this->btnExit->Click += gcnew System::EventHandler(this,
&frmLogin::btnExit_Click);
                    //
```

```
                // btnDelete
                //
                this->btnDelete->Location = System::Drawing::Point(93, 98);
                this->btnDelete->Name = L"btnDelete";
                this->btnDelete->Size = System::Drawing::Size(75, 23);
                this->btnDelete->TabIndex = 4;
                this->btnDelete->Text = L"Видалення";
                this->btnDelete->UseVisualStyleBackColor = true;
                this->btnDelete->Click += gcnew System::EventHandler(this,
&frmLogin::btnDelete_Click);
                //
                // frmLogin
                //
                this->AutoScaleDimensions = System::Drawing::Size(6, 13);
                this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                this->ClientSize = System::Drawing::Size(260, 129);
                this->Controls->Add(this->groupBox);
                this->Controls->Add(this->btnEnter);
                this->Controls->Add(this->btnExit);
                this->Controls->Add(this->btnDelete);
                this->MaximizeBox = false;
                this->MinimizeBox = false;
                this->Name = L"frmLogin";
                this->Text = L"Вхід у систему УЕД";
                this->groupBox->ResumeLayout(false);
                this->groupBox->PerformLayout();
                this->ResumeLayout(false);

        }
#pragma endregion
/*************************************************************************/
private: System::Boolean PrepareLoginPassword()
        {
                // Перетворити ім'я до нижнього регістра
                txtLogin->Text = txtLogin->Text->ToLower();
                // Забрати в поле ім'я бічні пробіли
                txtLogin->Text = txtLogin->Text->Trim();

                // Поле для введення ім'я не містить значення?
                if(String::IsNullOrEmpty(txtLogin->Text))
                {
                        // Видати повідомлення про те, поле не містить значення
                        MessageBox::Show("Введіть ім'я користувача.", "Usrs
Manager",
                                MessageBoxButtons::OK, MessageBoxIcon::Warning);
                        // Вийти
                        return false;
                }

                // Перевірити введені символи
                for(Int32 i = 0; i < txtLogin->Text->Length; i++)
                {
                        // Перевіряється символ
                        Char chTest = txtLogin->Text[i];

                        // Перевірка на коректність
                        if(!(chTest >= 'A' && chTest <= 'Z') &&
                             !(chTest >= 'a' && chTest <= 'z') &&
                             !(chTest >= 'A' && chTest <= 'Я') &&
                             !(chTest >= 'a' && chTest <= 'я') &&
                             !(chTest == '_'))
                        {
                             // Видати повідомлення про те, поле містить
неприпустимий символ
                             MessageBox::Show("Логін містить неприпустимі
символи!",
                                    "Usrs Manager", MessageBoxButtons::OK,
                                    MessageBoxIcon::Warning);
```

```
                        // Вийти
                        return false;
                    }
                }

                return true;
            }
/*************************************************************************/
private: System::Void btnEnter_Click(System::Object^  sender, System::EventArgs^
e)
            {
                // Підготовка значень
                if(!PrepareLoginPassword())
                {
                    return;
                }

                // Уміст поля ім'я - логін адміністратора?
                if(txtLogin->Text == WORKER_LOGIN)
                {
                    // Увійти в обліковий запис адміністратора
                    if(Usrs->WorkerLogin(txtPassword->Text))
                    {
                        // Відобразити форму адміністратора
                        UsrsManager::frmWorker^ frmNew = gcnew
UsrsManager::frmWorker;
                        frmNew->Show();
                    }
                    else
                    {
                        // Вхід неможливий, закінчити роботу функції
                        return;
                    }
                }
                else
                {
                    // Увійти в обліковий запис користувача
                    if(Usrs->ReaderLogin(txtLogin->Text, txtPassword-
>Text))
                    {
                        // Відобразити форму для користувача
                        UsrsManager::frmReader^ frmNew =
                            gcnew UsrsManager::frmReader(txtLogin-
>Text);
                        frmNew->Show();
                    }
                    else
                    {
                        // Вхід неможливий, закінчити роботу функції
                        return;
                    }
                }

                // Сховати форму входу
                this->Hide();
            }
/*************************************************************************/
private: System::Void btnExit_Click(System::Object^  sender, System::EventArgs^
e)
            {
                // Закрити додаток
                Application::Exit();
            }
/*************************************************************************/
private: System::Void btnDelete_Click(System::Object^  sender,
System::EventArgs^  e)
            {
                // Одержати подтвержение на видалення
                if(MessageBox::Show("Ви дійсно хочете зробити видалення?",
```

```
                        "Usrs Manager", MessageBoxButtons::YesNo,
                        MessageBoxIcon::Information) == ::DialogResult::Yes)
                  {
                        // Підготовка значень логіна й пароля
                        if(!PrepareLoginPassword())
                        {
                              // Якщо виникли помилки, то завершити роботу
функції
                              return;
                        }

                        // Уміст поля ім'я - логін адміністратора?
                        if(txtLogin->Text == WORKER_LOGIN)
                        {
                                    // Обліковий запис адміністратора видалити
не можна
                                    MessageBox::Show("Неможливо видалити
обліковий запис адміністратора!",
                                          "Usrs Manager", MessageBoxButtons::OK,
                                          MessageBoxIcon::Error);
                        }
                        else
                        {
                              // Видалити обліковий запис користувача
                              if(Usrs->RemoveReader(txtLogin->Text,
txtPassword->Text))
                              {
                                    // Видалення пройшло успішно
                                    MessageBox::Show("Обліковий запис успішно
вилучений",
                                          "Usrs Manager",
MessageBoxButtons::OK,
                                          MessageBoxIcon::Information);

                              }
                              else
                              {
                                    // При видаленні відбулася помилка
                                    MessageBox::Show("Неможливо видалити
обліковий запис",
                                          "Usrs Manager",
MessageBoxButtons::OK,
                                          MessageBoxIcon::Error);
                              }
                        }
                  }
            }
/*************************************************************************/
};
}
```

Файл Doc.cpp - робота з документами

```
/*****************************************************************************/
#include "StdAfx.h"
#include "Doc.h"
/*****************************************************************************/
using namespace System;
/*****************************************************************************/
// NAME:          CDoc
// DESCRIPTION:   Конструктор класу
// INPUT:         N/D
// OUTPUT:        N/D
/*****************************************************************************/
CDoc::CDoc(void)
{
      ddTotalNumber = 0;      // Загальна кількість екземплярів дорівнює 0
      ddFreeNumber = 0; // Кількість вільних екземплярів дорівнює 0
}
/*****************************************************************************/
// NAME:          GetName
// DESCRIPTION:   Функція одержання назви документів
// INPUT:         N/D
// OUTPUT:        Назва документа
/*****************************************************************************/
String^ CDoc::GetName()
{
      return strName;
}
/*****************************************************************************/
// NAME:          SetName
// DESCRIPTION:   Функція установки назви документів
// INPUT:         strValue - назва документа
// OUTPUT:        TRUE - Назва документа установлена
//                      FALSE - Назва не встановлена, рядок-параметр порожня
/*****************************************************************************/
Boolean CDoc::SetName(String^ strValue)
{
      // Перевірити, чи не порожній рядок
      if(!String::IsNullOrEmpty(strValue))
      {
            // Установити значення
            strName = strValue;
            return true;
      }
      else
      {
            return false;
      }
}
/*****************************************************************************/
// NAME:          GetAuthor
// DESCRIPTION:   Функція одержання автора документа
// INPUT:         N/D
// OUTPUT:        Автор документи
/*****************************************************************************/
String^ CDoc::GetAuthor()
{
      // Повернути значення
      return strAuthor;
}
/*****************************************************************************/
// NAME:          SetAuthor
// DESCRIPTION:   Функція установки автора документа
// INPUT:         strValue - ім'я автора документа
// OUTPUT:        N/D
/*****************************************************************************/
void CDoc::SetAuthor(String^ strValue)
```

```
{
      // Установити значення
      strAuthor = strValue;
}
/*************************************************************************/
// NAME:          GetISBN
// DESCRIPTION:   Функція одержання вхідний номер документа
// INPUT:         N/D
// OUTPUT:        вхідний номер документа
/*************************************************************************/
String^ CDoc::GetISBN()
{
      // Повернути значення
      return strISBN;
}
/*************************************************************************/
// NAME:          SetISBN
// DESCRIPTION:   Функція установки вхідний номер документа
// INPUT:         strValue - вхідний номер документа
// OUTPUT:        TRUE - вхідний номер документа встановлений
//                      FALSE - вхідний номер документа не встановлений, рядок-
параметр порожня
/*************************************************************************/
Boolean CDoc::SetISBN(String^ strValue)
{
      // Перевірити, чи не порожній рядок
      if(!String::IsNullOrEmpty(strValue))
      {
            // Установити значення
            strISBN = strValue;
            // Обчислити й установити хеш-код вхідний номер документа
            ddISBNHash = strISBN->GetHashCode();
            return true;
      }
      else
      {
            return false;
      }
}
/*************************************************************************/
// NAME:          GetTheme
// DESCRIPTION:   Функція одержання теми
// INPUT:         N/D
// OUTPUT:        Тема документи
/*************************************************************************/
String^ CDoc::GetTheme()
{
      // Повернути значення
      return strTheme;
}
/*************************************************************************/
// NAME:          SetTheme
// DESCRIPTION:   Функція установки теми
// INPUT:         strValue - тема документи
// OUTPUT:        N/D
/*************************************************************************/
void CDoc::SetTheme(String^ strValue)
{
      // Установити значення
      strTheme = strValue;
}
/*************************************************************************/
// NAME:          GetPages
// DESCRIPTION:   Функція одержання кількості сторінок
// INPUT:         N/D
// OUTPUT:        Кількість сторінок у книзі
/*************************************************************************/
Int32 CDoc::GetPages()
{
```

```
	// Повернути значення
	return ddPages;
}
/*************************************************************************/
// NAME:		SetPages
// DESCRIPTION:	Функція установки кількості сторінок
// INPUT:		ddValue - кількість сторінок
// OUTPUT:		TRUE - кількість сторінок установлена
//			FALSE - кількість сторінок установлена, неприпустиме
значення аргументу
/*************************************************************************/
Boolean	CDoc::SetPages(Int32 ddValue)
{
	// Параметр має припустиме значення?
	if(ddValue >= 0)
	{
		// Установити значення
		ddPages = ddValue;
		return true;
	}
	else
	{
		return false;
	}
}
/*************************************************************************/
// NAME:		GetTotalNumber
// DESCRIPTION:	Функція одержання загальної кількості екземплярів
// INPUT:		N/D
// OUTPUT:		Загальна кількість екземплярів документи
/*************************************************************************/
Int32 CDoc::GetTotalNumber()
{
	// Повернути значення
	return ddTotalNumber;
}
/*************************************************************************/
// NAME:		SetTotalNumber
// DESCRIPTION:	Функція установки загальне кількості екземплярів
// INPUT:		ddValue - загальна кількість екземплярів
// OUTPUT:		TRUE - кількість екземплярів установлена
//			FALSE - кількість екземплярів установлена, негативне
значення
//			аргумента або аргумент перевищує кількість екземплярів,
що перебуває в
//			користувачів
/*************************************************************************/
Boolean	CDoc::SetTotalNumber(Int32 ddValue)
{
	// Перевірка параметра
	if(ddValue >= 0 && ddValue >= ddTotalNumber - ddFreeNumber)
	{
		// Установити значення
		ddTotalNumber = ddValue;
		return true;
	}
	else
	{
		return false;
	}
}
/*************************************************************************/
// NAME:		GetFreeNumber
// DESCRIPTION:	Функція одержання кількості вільних екземплярів
// INPUT:		N/D
// OUTPUT:		Кількість вільних екземплярів
/*************************************************************************/
Int32 CDoc::GetFreeNumber()
{
```

```
      // Повернути значення
      return ddFreeNumber;
}
/*************************************************************************/
// NAME:          SetFreeNumber
// DESCRIPTION:   Функція установки кількості вільних екземплярів
// INPUT:         ddValue - кількість вільних екземплярів
// OUTPUT:        TRUE - кількість вільних екземплярів установлене
//                      FALSE - значення параметра перевищує загальна кількість
екземплярів
/*************************************************************************/
Boolean     CDoc::SetFreeNumber(Int32 ddValue)
{
      // Порівняння параметра із загальною кількістю екземплярів
      if(ddTotalNumber >= ddValue)
      {
            // Установити значення
            ddFreeNumber = ddValue;
            return true;
      }
      else
      {
            // ddFreeNumber = ddTotalNumber;
            return false;
      }
}
/*************************************************************************/
// NAME:          GetISBNHash
// DESCRIPTION: Функція одержання хеш-коду від вхідного номера документа
// INPUT:         N/D
// OUTPUT: Хеш-         Код вхідний номер документа
/*************************************************************************/
Int32 CDoc::GetISBNHash()
{
      // Повернути хеш-значення вхідного номера документа
      return ddISBNHash;
}
/*************************************************************************/
// NAME:          DecFreeNumber
// DESCRIPTION:   Функція зменшення кількості вільних екземплярів
// INPUT:         N/D
// OUTPUT:        TRUE - кількість вільних екземплярів декрементовано
//                FALSE - зменшення неприпустимо, кількість вільних екземплярів
дорівнює 0
/*************************************************************************/
Boolean     CDoc::DecFreeNumber()
{
     // Кількість вільних екземплярів не дорівнює 0?
     if(ddFreeNumber != 0)
     {
            // Зменшити значення
            ddFreeNumber ---i;
            return true;
      }
      else
      {
            return false;
      }
}
/*************************************************************************/
// NAME:          IncFreeNumber
// DESCRIPTION:   Функція збільшення кількості вільних екземплярів
// INPUT:         N/D
// OUTPUT:        TRUE - кількість вільних екземплярів інкрементовано
//                FALSE - значення неприпустимо, кількість вільних екземплярів
дорівнює
//                      загальній кількості екземплярів
/*************************************************************************/
Boolean     CDoc::IncFreeNumber()
```

```
{
	// Кількість вільних екземплярів не дорівнює загальній кількості
екземплярів?
	if(ddFreeNumber != ddTotalNumber)
	{
		// Збільшити значення
		ddFreeNumber ++;
		return true;
	}
	else
	{
		return false;
	}
}
/*****************************************************************************/
```

Файл Doc.h - бібліотека для файлу Doc. cpp

```
     #pragma once
/*************************************************************************/
using namespace System;
/*************************************************************************/
ref class CDoc                                                  // Клас "Документ"
{
private:
     String^ strName;                                   // Назва
     String^ strAuthor;                                 // Автор
     String^ strISBN;                                   // вхідний номер документа
     String^ strTheme;                                  // Тема
     Int32 ddPages;                                     // Кількість сторінок
     Int32 ddTotalNumber;                               // Загальна кількість
екземплярів
     Int32 ddFreeNumber;                                // Кількість вільних
(перебувають в
                                                        // системі) екземплярів
     Int32 ddISBNHash;                                  // Значення хеш-функції від
вхідний номер документа

public:
     CDoc(void);                                        // Конструктор класу

     String^ GetName();                                 // Функція одержання назви
документів
     String^ GetAuthor();                               // Функція одержання автора
документа
     String^ GetISBN();                                 // Функція одержання
вхідний номер документа
     String^ GetTheme();                                // Функція одержання теми
     Int32  GetPages();                                 // Функція одержання
кількості сторінок
     Int32  GetTotalNumber();                           // Функція одержання
загальне кількості екземплярів
     Int32  GetFreeNumber();                            // Функція одержання
кількості вільних екземплярів
     Int32  GetISBNHash();                              // Функція одержання хеша-
коду від вхідний номер документа

     Boolean SetName(String^ strValue);         // Функція установки назви
документів
     void SetAuthor(String^ strValue);          // Функція установки автора
документа
     Boolean SetISBN(String^ strValue);         // Функція установки вхідний
номер документа
     void SetTheme(String^ strValue);           // Функція установки теми
     Boolean      SetPages(Int32 ddValue);            // Функція установки
кількості сторінок
     Boolean      SetTotalNumber(Int32 ddValue);      // Функція установки
загальне кількості екземплярів
     Boolean      SetFreeNumber(Int32 ddValue); // Функція установки кількості
вільних екземплярів

     Boolean      DecFreeNumber();                    // Функція зменшення
кількості вільних екземплярів
     Boolean      IncFreeNumber();                    // Функція збільшення
кількості вільних екземплярів
};
/*************************************************************************/
```

Файл Usrs.cpp - робота з обліковими записами

```
    /*****************************************************************************/
#include "StdAfx.h"
#include "Usrs.h"
/*****************************************************************************/
using namespace System;
using namespace System::IO;
using namespace System::Windows::Forms;
using namespace System::Collections;
/*****************************************************************************/
#define USER_LIST_FILE              "USERLIST.TXT"
#define WORKER_NAME                       "ADMINISTRATOR"
#define WORKER_FILE_EXTENSION ".LMW"
#define READER_FILE_EXTENSION ".LMR"
/*****************************************************************************/
// NAME:          CUsrs
// DESCRIPTION:   Конструктор класу
// INPUT:         N/D
/*****************************************************************************/
CUsrs::CUsrs(void)
{
}
/*****************************************************************************/
// NAME:          WorkerLogin
// DESCRIPTION:   Функція входу й завантаження даних про адміністратора
// INPUT:         strPassword - пароль адміністратора
/*****************************************************************************/
Boolean CUsrs::WorkerLogin(System::String ^strPassword)
{
    // Перевірити існування файлу зі списком імен і паролів
    if(File::Exists(USER_LIST_FILE))
    {
     // Відкрити файл
     StreamReader^ srUserList = gcnew StreamReader(USER_LIST_FILE);

     // Завантажити перший рядок з ім'ям адміністратора
     String^ strLogin = srUserList->ReadLine();

     // Зрівняти введене ім'я з ім'ям у файлі
     if(String::Compare(strLogin, WORKER_NAME) != 0)
     {
      // Логіни не збігаються
      MessageBox::Show("Обліковий запис не знайдено.");
      // Повернути помилку
      return false;
     }

     // Завантажити другий рядок з паролем адміністратора
     String^ strPasswordInFile = srUserList->ReadLine();

     // Закрити файл
     srUserList->Close();

     // Зрівняти введений пароль із паролем у файлі
     if(String::Compare(strPassword, strPasswordInFile) != 0)
     {
      // Паролі не збігаються
      MessageBox::Show("Невірний пароль!", "Система УЕД",
            MessageBoxButtons::OK, MessageBoxIcon::Error);
      // Повернути помилку
      return false;
     }

     // Вхід виконаний
     return true;
    }
    else
```

```
    {
     // Створити файл зі списком користувачів
     StreamWriter^ swUserList = gcnew StreamWriter(USER_LIST_FILE);

     // Зберегти ім'я користувача й пароль
     swUserList->WriteLine(WORKER_NAME);
     swUserList->WriteLine(strPassword);

     // Закрити файл
     swUserList->Close();
    }

    return true;
}
/*************************************************************************/
// NAME:          ReaderLogin
// DESCRIPTION:   Функція входу й завантаження даних про користувача
// INPUT:         strName - ім'я облікового запису користувача
//          strPassword - пароль для входу
/*************************************************************************/
Boolean CUsrs::ReaderLogin(System::String^ strName, System::String^ strPassword)
{
    // Прапор удалого входу
    Boolean IsLoginValid = false,
      IsPasswordValid = false;

    // Перевірка існування файлу зі списком користувачів
    if(!File::Exists(USER_LIST_FILE))
    {
     // Видати повідомлення про помилку
     MessageBox::Show("Помилка! Для початку роботи необхідно створити обліковий
запис.",
      "Система УЕД", MessageBoxButtons::OK, MessageBoxIcon::Error);
     // Повернути помилку
     return false;
    }

    // Відкрити файл
    StreamReader^ srUserList = gcnew StreamReader(USER_LIST_FILE);

    do
    {
     // Завантажити ім'я й пароль
     String^ strNameInFile = srUserList->ReadLine();
     String^ strPasswordInFile = srUserList->ReadLine();

     if(String::Compare(strName, strNameInFile) == 0)
     {
     // Ім'я користувача знайдене
      IsLoginValid = true;

      // Зрівняти введений пароль із паролем у файлі
      if(String::Compare(strPassword, strPasswordInFile) == 0)
      {
            // Пароль збігається
            IsPasswordValid = true;
            // Закінчити цикл пошуку
            break;
      }
      else
      {
            // Ім'я знайдене, але пароль не збігається
            IsPasswordValid = false;
            // Закінчити цикл пошуку
            break;
      }

     }
    }
```

```
    while(srUserList->EndOfStream == false);

    // Закрити файл
    srUserList->Close();

    // Якщо логін збігається, а пароль не збігається, то перервати виконання
    if(IsLoginValid && !IsPasswordValid)
    {
     // Повідомити користувача
     MessageBox::Show("Невірний пароль!", "Система УЕД",
      MessageBoxButtons::OK, MessageBoxIcon::Error);
     // Повернути помилку
     return false;
    }

    // Якщо ні логін, ні пароль не збігаються, то створити новий обліковий запис
    if(!IsLoginValid && !IsPasswordValid)
    {
     // Запропонувати створити пользоватею новий обліковий запис
     if (MessageBox::Show("Облікового запису з таким ім'ям користувача не
знайдено, створити новий обліковий запис?",
                          "Система УЕД", MessageBoxButtons::YesNo,
                          MessageBoxIcon::Question) == DialogResult::Yes)
     {
      // Створити новий обліковий запис
      if(!AddReader(strName, strPassword))
      {
            MessageBox::Show("При створенні нового облікового запису виникла
помилка", "Система УЕД", MessageBoxButtons::OK,
                                    MessageBoxIcon::Error);
            // Не виконувати вхід
            return false;
      }
     }
     else
     {
      // Не виконувати вхід
      return false;
     }
    }

    return true;
}
/*********************************************************************/
// NAME:          AddReader
// DESCRIPTION:   Функція додавання нового облікового запису користувача
// INPUT:         strName - ім'я користувача
//          strPassword - пароль
/*********************************************************************/
Boolean CUsrs::AddReader(System::String ^strName, System::String ^strPassword)
{
    // Створити файловий потік
    FileStream^ fsUserList = gcnew FileStream(USER_LIST_FILE, FileMode::Append);
    // Відкрити файл зі списком користувачів
    StreamWriter^ swUserList = gcnew StreamWriter(fsUserList);

    // Записати ім'я й пароль
    swUserList->WriteLine(strName + Convert::ToChar(13) + Convert::ToChar(10) +
strPassword);

    // Закрити файл
    swUserList->Close();
    // Закрити файловий потік
    fsUserList->Close();

    // Створити файловий потік
    FileStream^ fsReaderFile = gcnew FileStream(strName->ToUpper() +
READER_FILE_EXTENSION,
                                                FileMode::CreateNew);
```

```
    // Відкрити файл користувача
    StreamWriter^ swReaderFile = gcnew StreamWriter(fsReaderFile);
    // Кількість документів у користувача дорівнює 0
    swReaderFile->WriteLine("0");
    // Закрити файл
    swReaderFile->Close();
    // Закрити файловий потік
    fsReaderFile->Close();

    return true;
}
/*************************************************************************/
// NAME:          RemoveReader
// DESCRIPTION:   Функція видалення облікового запису користувача
// INPUT:         strName - ім'я облікового запису
//                strPassword - пароль
/*************************************************************************/
Boolean CUsrs::RemoveReader(System::String ^strName, System::String
^strPassword)
{
    // Створити файловий потік
    FileStream^ fsUserList = gcnew FileStream(USER_LIST_FILE, FileMode::Open);

    // Пошук позиції рядка із заданим ім'ям користувача

    // Тимчасовий рядок
    String^ strTmp = "";
    // Лічильник символів
    Int64 dqCounter = 0;
    // Непарні рядки - рядка з іменами
    Boolean IsLoginString = true;
    // Цикл пошуку
    do
    {
     // Завантажити байт
     Char dbChar = fsUserList->ReadByte();
     // Якщо байт службовий або досягнуть кінець файлу
     if(dbChar == 0x0D || fsUserList->Position == fsUserList->Length)
     {
      // Якщо рядок ім'я
      if(IsLoginString)
      {
            // Зрівняти ім'я користувача з виділеним рядком
            if(String::Compare(strTmp, strName) == 0)
            {
                  // Скорегувати покажчик
                  dqCounter -= strTmp->Length;
                  // Перервати виконання циклу
                  break;
            }
      }

      // Обнулити рядок
      strTmp = "";
      // Інвертувати прапор рядка ім'я
      IsLoginString = !IsLoginString;
      // Якщо не кінець файлу
      if(fsUserList->Position + 1 != fsUserList->Length)
      {
            // Пропустити службовий символ
            fsUserList->ReadByte();
            dqCounter ++;
      }
     }
     else
     {
      // Додати лічений символ до тимчасового рядка
      strTmp += dbChar;
     }
```

```
        // Збільшить покажчик
        dqCounter ++;
      }
      // Продовжувати, поки не кінець файлу
      while(fsUserList->Position != fsUserList->Length);

      // Переміститися у файлі до знайденого рядка
      fsUserList->Position = dqCounter;
      // Заповнити ім'я пробілами
      for(Int32 i = 0; i < strTmp->Length; i++)
      {
        fsUserList->WriteByte(' ');
      }

      // Закрити файловий потік
      fsUserList->Close();
      // Видалити файл користувача
      File::Delete(strName + READER_FILE_EXTENSION);

      return true;
}
/*****************************************************************************/
```

Файл Usrs.h - бібліотека для файлу Usrs.cpp

```
     /*************************************************************************
/
#pragma once
/*************************************************************************/
using namespace System;
/*************************************************************************/
ref class CUsrs   // Клас реєстрації, видалення й авторизації користувачів
{
public:
     // Конструктор класу
     CUsrs(void);

     // Функція додавання нового облікового запису користувача
     Boolean AddReader(System::String^ strName, System::String^ strPassword);
     // Функція видалення облікового запису користувача
     Boolean RemoveReader(System::String^ strName, System::String^ strPassword);

     // Функція входу й завантаження даних про користувача
     Boolean ReaderLogin(System::String^ strName, System::String^ strPassword);
     // Функція входу й завантаження даних про адміністратора
     Boolean WorkerLogin(System::String^ strName);
};
/*************************************************************************/
```

Файл frmAbout.resx - xml опис форми «про програму...»

```
 <?xml version="1.0" encoding=" utf-8" ?>
- <root>
- <xsd:schema id="root" xmlns="" xmlns:xsd=" " xmlns:msdata="urn: schemas-
microsoft-com: xml-msdata">
  <xsd:import namespace=" " />
- <xsd:element name="root" msdata:IsDataSet="true">
- <xsd:complexType>
- <xsd:choice maxOccurs="unbounded">
- <xsd:element name="metadata">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element name="value" type="xsd:string" minOccurs="0" />
  </xsd:sequence>
  <xsd:attribute name="name" use="required" type="xsd:string" />
  <xsd:attribute name="type" type="xsd:string" />
  <xsd:attribute name="mimetype" type="xsd:string" />
  <xsd:attribute ref="xml:space" />
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="assembly">
- <xsd:complexType>
  <xsd:attribute name="alias" type="xsd:string" />
  <xsd:attribute name="name" type="xsd:string" />
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="data">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element name="value" type="xsd:string" minOccurs="0" msdata:Ordinal="1"
/>
  <xsd:element name="comment" type="xsd:string" minOccurs="0" msdata:Ordinal="2"
/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required" msdata:Ordinal="1"
/>
  <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
  <xsd:attribute name="mimetype" type="xsd:string" msdata:Ordinal="4" />
  <xsd:attribute ref="xml:space" />
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="resheader">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element name="value" type="xsd:string" minOccurs="0" msdata:Ordinal="1"
/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required" />
  </xsd:complexType>
  </xsd:element>
  </xsd:choice>
  </xsd:complexType>
  </xsd:element>
  </xsd:schema>
- <resheader name="resmimetype">
  <value>text/ microsoft-resx</value>
  </resheader>
- <resheader name="version">
  <value>2.0</value>
  </resheader>
- <resheader name="reader">
  <value>System.Resources.ResXResourceReader, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
- <resheader name="writer">
```

```
  <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
- <data name="txtInfo.Text" xml:space="preserve">

  <value>МАГІСТЕРСЬКА РОБОТА На тему: Дослідження та програмна реалізація
системи управління електронним документообігом Керівник: Смірнова Т.В. Розробив:
студент Теплухін Віктор Сергійович. гр. КІ-22М-2 ЦНТУ м. Кропивницький
2023</value>
  </data>
  </root>
```

Файл frmAbout.h - бібліотека для файлу frmAbout.resx

```
#pragma once

using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;


namespace UsrsManager
{

	/// <summary>
	/// Summary for frmAbout
	///

	/// </summary>
	public ref class frmAbout : public System::Windows::Forms::Form
	{
/*************************************************************************/
public:	frmAbout(void)
		{
			InitializeComponent();
			//
			//Додавання цього коду в конструктор
			//
		}
/*************************************************************************/
protected: ~frmAbout()
		{
			if (components)
			{
				delete components;
			}
		}
/*************************************************************************/
	private: System::Windows::Forms::Button^  btnOk;
	private: System::Windows::Forms::TextBox^  txtInfo;

	private: System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Необхідний метод для підтримки Розроблювача - не модифікувати
		/// зміст цього методу з кодовим редактором.
		/// </summary>
		void InitializeComponent(void)
		{
			System::ComponentModel::ComponentResourceManager^  resources =
(gcnew System::ComponentModel::ComponentResourceManager(frmAbout::typeid));
			this->btnOk = (gcnew System::Windows::Forms::Button());
			this->txtInfo = (gcnew System::Windows::Forms::TextBox());
			this->SuspendLayout();
			//
			// btnOk
			//
			this->btnOk->Location = System::Drawing::Point(205, 216);
			this->btnOk->Name = L"btnOk";
			this->btnOk->Size = System::Drawing::Size(75, 23);
			this->btnOk->TabIndex = 0;
			this->btnOk->Text = L"OK";
			this->btnOk->UseVisualStyleBackColor = true;
			this->btnOk->Click += gcnew System::EventHandler(this,
&frmAbout::btnOk_Click);
			//
```

```
                    // txtInfo
                    //
                    this->txtInfo->Location = System::Drawing::Point(7, 12);
                    this->txtInfo->Multiline = true;
                    this->txtInfo->Name = L"txtInfo";
                    this->txtInfo->ReadOnly = true;
                    this->txtInfo->Size = System::Drawing::Size(273, 198);
                    this->txtInfo->TabIndex = 3;
                    this->txtInfo->Text = resources->GetString(L"txtInfo.Text");
                    this->txtInfo->TextAlign =
System::Windows::Forms::HorizontalAlignment::Center;
                    //
                    // frmAbout
                    //
                    this->AutoScaleDimensions = System::Drawing::Size(6, 13);
                    this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(287, 243);
                    this->Controls->Add(this->btnOk);
                    this->Controls->Add(this->txtInfo);
                    this->MaximizeBox = false;
                    this->MinimizeBox = false;
                    this->Name = L"frmAbout";
                    this->Text = L"Про програму...";
                    this->ResumeLayout(false);
                    this->PerformLayout();

            }
#pragma endregion
/*************************************************************************/
private: System::Void btnOk_Click(System::Object^  sender, System::EventArgs^
e)
             {
                    // Закрити форму
                    frmAbout::Close();
             }
/*************************************************************************/
      };
}
```