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 першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи управління електронним документообігом кафедри КбПЗ”

Виконав здобувач вищої освіти
IV курсу, групи КМ-19
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Пануш В.Д.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Буравченко К.О.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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
« 17 » січня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Панушу Валентину Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи управління електронним документообігом кафедри КбПЗ*

2. Керівник роботи *Буравченко Костянтин Олегович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 10-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи управління електронним документообігом кафедри КбПЗ*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Буравченко К.О.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Пануш В.Д.
(прізвище та ініціали)

АНОТАЦІЯ

Пануш В.Д. Програмне забезпечення системи управління електронним документообігом кафедри КбПЗ. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи управління електронним документообігом кафедри КбПЗ.

Метою розробки є програмне забезпечення системи управління електронним документообігом кафедри КбПЗ.

Результат роботи – програмна реалізація системи управління електронним документообігом кафедри КбПЗ.

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

Panush V.D. Software support of the electronic document flow management system of the department of CsSw. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the first (bachelor) level of higher education, software was developed, which is intended for the electronic document flow management system of the CsSw department.

The purpose of the development is the software of the electronic document flow management system of the department of CsSw.

The result of the work is the software implementation of the electronic document flow management system of the Department of Medical Sciences.

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

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 кафедри КбПЗ
ТЕ	–	типові елементи
ФФО	–	формат файлових об'єктів
EDMS	–	Electronic Document Management Systems

ВСТУП

Актуальність теми. Характерною рисою постіндустріального суспільства на сучасному етапі є його глибока інформатизація – впровадження в усі сфери життя засобів обчислювальної техніки й комунікацій. Рівень інформатизації визначається рівнем застосовуваних інформаційних технологій. Одним з основних напрямків розвитку інформаційних технологій, є системи управління.

Системи управління застосуються для автоматизації таких завдань, як управління виробництвом, обліком, збутом, кадрами, фінансами й інформаційними ресурсами, зокрема широке застосування одержали організаційно-технічні системи, які забезпечують процес створення, управління доступом і поширення електронних документів у комп'ютерних мережах, що забезпечують контроль над потоками документів в організації. Такі системи одержали назву систем управління електронним документообігом (СУЕД). Важливість СУЕД для організації підкреслює той факт, що СУЕД є основою для розгортання систем класу управління підприємством, систем управління продажами й інших важливих для сучасного підприємства інформаційних систем, тому від функціонування СУЕД всіх інформаційних систем організації залежить робота всієї організації.

Зараз СУЕД незалежно друг від друга роблять більше сотні компаній, тому через непогодженість роботи СУЕД виникають проблеми при взаємодії систем документообігу різних виробників. Як правило, ці проблеми пов’язанні з несумісністю використовуваних у різних системах форматів подання даних, різних карток документів, не стандартизованих систем і підходів, використовуваних при проектуванні систем. Використовувані в цих СУЕД формати файлових об'єктів є надлишковими й створюють високе навантаження на обчислювальні й комунікаційні ресурси, що в Україні є серйозною

перешкодою для поширення СУЕД, оскільки для територіально – розподілених організацій у Україні є характерним відсутність стійких каналів передачі даних.

Необхідно відзначити, що більшість використовуваних методик оцінки ефективності роботи СУЕД засновані на оцінці економічного ефекту від впровадження СУЕД і не дозволяють повною мірою оцінити ефективність використання СУЕД обчислювальних і комунікаційних ресурсів організації.

Все це приводить як до зниження ефективності роботи систем управління електронним документообігом кафедри КбПЗ, так і найчастіше до повної їхньої неспроможності, оскільки доводитися імпортувати документи з електронної в паперову форму для того, щоб передати документи з одного підрозділу/організації.

З обліком вищесказаного, актуальність теми визначається необхідністю підвищення ефективності роботи елементів СУЕД, що є малопроробленим науково-технічним завданням. Можливі шляхи рішення поставленого завдання з використанням розроблених до теперішнього часу технології відкритих систем і методів функціональної стандартизації дозволяють сформулювати наступні основні вимоги до СУЕД:

– забезпечення масштабованості, інтероперабельності;

– зниження вимог до обчислювальних, комунікаційних ресурсів, що дозволить підвищити ефективності роботи СУЕД у цілому.

Під інтероперабельністю мається на увазі здатність до взаємодії двох і більше систем або компонентів для обміну інформацією й використанню цієї інформації.

Інтероперабельність програмного забезпечення (функціональність програмного забезпечення) – здатність програмного продукту виконувати набір функцій визначених у його зовнішньому описі й задовольняючих заданим потребам користувачів або тим які маються на увазі.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи управління електронним документообігом кафедри КбПЗ.

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

– Огляд існуючих систем управління електронним документообігом кафедри КбПЗ.

– Дослідження системи управління електронним документообігом кафедри КбПЗ.

– Програмна реалізація системи управління електронним документообігом кафедри КбПЗ.

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

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи управління електронним документообігом кафедри КбПЗ, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Ціль дослідження архітектури – установлення об'єктів (компонентів) СУЕД і взаємозв'язків між компонентами для виявлення слабких місць на рівні компонентної архітектури СУЕД, усунення яких дозволить підвищити ефективність роботи СУЕД.

Управління інформаційними ресурсами має для діяльності будь-якої установи особливе значення. У сучасному світі установи зіштовхуються з необхідністю обробки колосального обсягу інформації. Незалежно від правового статусу або організаційних форм діяльності установи покликані активно взаємодіяти з органами виконавчої й законодавчої влади, структурами, що беруть участь у регулюванні економіки. Все це у свою чергу породжує специфічний документообіг.

Таким чином, установи гостро відчувають необхідність організації ефективного управління інформаційними ресурсами й уживають активних заходів по використанню комп'ютерних технологій у сфері управління документаційними потоками (і інформацією в широкому змісті).

На жаль, застосування комп'ютерних програм у сфері роботи з управлінської (організаційно-розпорядницької) документацією в ряді випадків не супроводжується структурною перебудовою роботи з документацією, що істотно знижує ефект від застосування навіть самих багатофункціональних спеціалізованих (і відповідно дорогих) програмних комплексів. Нерідкі ситуації, коли впровадження комп'ютерних систем носить формальний характер і не супроводжується скільки-небудь істотною оптимізацією й уніфікацією документаційних процесів.

У більшості установ зі складною структурою важливе значення має рівень організації взаємодії підрозділів і порядок обміну інформацією. Більша частина

інформації передається у вигляді документів на паперовому носії (обмін службовою документацією й звітністю).

Слід також зазначити, що протягом ряду років досить чітко простежується тенденція збільшення обсягів інформаційних потоків, що проходять через сучасні установи. Характерно, що відбувається ріст не тільки електронного документообігу на традиційних носіях, але й інформації, що проходить по електронних каналах, а також документів, пов'язаних з функціонуванням комп'ютерних систем.

Робота з автоматизації діловодних процесів почалася в Україні ще на початку 90-х років. Істотним кроком в упорядкуванні роботи зі службовою документацією стало впровадження в 1997 році Системи автоматизації електронного документообігу й діловодства – САДД. Програмний комплекс був розроблений на базі типового програмного продукту однієї з фірм, що спеціалізується на автоматизації роботи з документами.

Система управління електронного документообігу (СУЕД) або EDMS (Electronic Document Management Systems) – це система автоматизації роботи з документами протягом усього їхнього життєвого циклу (створення, зміна, зберігання, пошук, класифікація та ін.), а також процесів взаємодії між співробітниками. При цьому під документами в першу чергу маються на увазі неструктуровані документи (файли Word, Excel та ін.). Як правило, СУЕД містить у собі електронний архів документів і систему автоматизації ділових процесів.

Ефективне управління документацією на основі СУЕД засновано на трьох складових системи:

– технологія (на основі сучасних комп'ютерних комплексів);

– корпоративні правила створення й використання інформаційних ресурсів (і їхнє закріплення в розпорядницьких документах);

– психологія користувачів і їхнє навчання (при необхідності індивідуальне).

У системах електронного документообігу здійснюється реєстрація нормативних документів, розпорядницьких документів (наказів, розпоряджень), переписки з органами влади, установами, кредитними організаціями, іншими установами й підприємствами, а також громадянами. Крім цього в СУЕД ведеться робота із внутрішньою службовою перепискою й проектами організаційно-розпорядницьких документів, створюваних структурними установ.

СУЕД забезпечує контроль за рухом і виконанням документів, містить повну інформацію про доручення, даних керівництвом і діях виконавців. Важливим елементом СУЕД є система формальних і семантичних посилань на взаємозалежні документи й доручення. Пошук у СУЕД крім традиційних діловодних реквізитів базується на системі класифікаторів (у тому числі тематичних), що дозволяють здійснювати контекстний відбір документів.

Процедура ведення класифікаторів строго регламентована. Частина довідників формується фахівцями, що ведуть реєстрацію (наприклад, класифікатор організацій – кореспондентів). Внесення змін в інші – прерогатива адміністратора, крім того, адміністратор здійснює постійний моніторинг нових позицій класифікаторів і при необхідності коректує їх. Таким чином, усувається можливе дублювання позицій довідників, і усуваються помилки при реєстрації документів. Ефективність роботи системи може бути забезпечена тільки за умови регулярного навчання користувачів і “м'якого” контролю за їхніми діями в системі (виправлення помилок, дотримання вимог по заповнення обов'язкових інформаційних реквізитів і т.п.).

Створення СУЕД зв'язане й зі зміною ролі служби документаційного забезпечення: визначаються єдині технологічні вимоги до організації електронного документообігу з використанням СУЕД, установлює систему “адміністративних рамок” у роботі з документами, що фактично є методологічною базою для організації роботи з інформацією.

У цілому встановлення чіткого порядку використання системи й правил роботи з інформацією є одним з основних факторів, що забезпечили успішне впровадження СУЕД і її повноцінне використання.

Таким чином розробка програмного забезпечення системи управління електронним документообігом кафедри КбПЗ є актуальною задачею, яка потребує вирішення у даній бакалаврській роботі.

1.2 Область застосування

Визначимо область застосування системи, яка розробляється. Для цього введемо декілька понять.

Електронний документ – це поняття більше широке, ніж просто електронний образ паперового документа. Сюди включають якийсь набір даних (текст, графічне й відеозображення, аудіозапис), створений за допомогою комп'ютера або збережений на ньому. Цей набір супроводжується карткою з атрибутами (подібно картотеці книг у бібліотеці), по яких документ можна швидко знайти (назва, автор, дата створення й т.п.).

Під електронним документообігом (або ЕДО) розуміється спосіб організації роботи з документами, при якому основна маса документів організації (підприємства) використовується в електронному виді й зберігається централізовано в так званих електронних архівах (ЕА), своєрідних інформаційних складах, або сховищах даних. Електронний документообіг може бути внутрішнім і зовнішнім, і це накладає певну специфіку на інформаційний обмін.

Відповідно під системою управління електронного документообігу (СУЕД) у вузькому змісті розуміється програмне забезпечення (комп'ютерна програма, система), що дозволяє організувати роботу з електронними документами (створення, зміна, пошук, зберігання), а також взаємодія між співробітниками: передачу документів, видачу завдань (розпоряджень, доручень) і контроль за ними, відправлення повідомлень і т.п. У більше широкому змісті під

СУЕД розуміється сучасна організаційно-технологічна структура, що пронизує весь виробничий організм, що включає в себе й програмну, і технічну, і методологічну складову, а також організаційні й нормативно-правові аспекти.

Але в нинішніх умовах конкуренції й ринку інформація є одним з найважливіших ресурсів, основним капіталом компанії. Чим більше різноманітних «робітників даних» (документів, файлів та ін.) по історії роботи компанії містить у своїх «засіках» інформаційна система й чим вони краще організовані й керовані, тим сама компанія виявляється більше конкурентоспроможною, захищеною і незалежною при втраті людських ресурсів (звільненнях, переходах) і більшою мірою застрахована від втрати клієнтів, партнерів, частини бізнесу, потенційно важливих документів та ін.

Таким чином, можна визначити ефективність використання системи управління електронного документообігу (СУЕД) у вузькому змісті для окремих виробництв і користувачів. У цьому випадку розумно розглядати наступні види ефектів:

– економічний – його показники враховують у вартісному вираженні всі види результатів і витрат, обумовлених реалізацією СУЕД. Автоматизація діловодства, автоматизація потоків документів, автоматизація контролю виконання документів і доручень сприяє скороченню помилок, що є присутнім при ручній праці, прискоренню процесу документообігу, що дає безсумнівний виграш у часі й економію у витратах електроенергії, витрат на оплату машинного часу й т.д.;

– фінансовий – розрахунок показників цього виду ефекту базується на фінансових результатах використання СУЕД, що знаходить висвітлення в скорочення часу на операції з документами, тим самим знижує тривалість роботи з документами;

– ресурсний – його показниками відбивають вплив використання СУЕД на обсяг виробництва й споживання того або іншого виду матеріального ресурсу

(електроенергії, трудових ресурсів і ін.). Як уже було сказано, виграш у часі при роботі з документами знизить витрата електроресурсів, трудових ресурсів і т.д.;

– науково-технічний – включає новизну, простоту, корисність СУЕД;

– соціальний – його показники враховують соціальні результати реалізації СУЕД, що виражаються в зменшенні трудомісткості підготовки й переробки одиниці даних в автоматизованій системі управління документообігом.

Для рішення, поставлених завдань по забезпеченню інтеграції й взаємодії СУЕД різних виробників існує кілька підходів:

1. Вибір єдиної платформи для СУЕД і її повсюдне впровадження.

2. Вибір двох систем і організація обміну даними між ними з використанням спеціально розроблених адаптерів.

3. Довільне рішення зі стандартизованими інтерфейсами.

Доказано, що третій підхід до рішення проблеми інтеграції й взаємодії СУЕД є найбільш ефективним.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи управління електронним документообігом кафедри КбПЗ, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

У рамках виконання бакалаврської роботи, дослідимо основні сучасні СУЕД, що мають широке комерційне використання: Hummingbird Enterprise, Documentum, LanDocs, Microsoft SharePoint Portal Server, Optima Workflow, Бос-Референт, Справа і Євфрат, на їхню відповідність типовим вимогам, сформульованим у роботі.

Сучасний бізнес припускає усе більш складні взаємодії. Необхідно налагодити бездоганну взаємодію й обмін інформацією між людьми, бізнес-системами підприємства з метою забезпечення оптимальної продуктивності, зміцнення ділових зв'язків і збільшення швидкості прийняття рішень. Однак компанії будь-якого розміру, що працюють у всіх галузях, стикаються із труднощами при спробах побудувати автоматизовані системи управління корпоративною інформацією, набудовуються легко у відповідності з мінливими умовами й ефективно використовують доступні й необхідні ресурси.

Hummingbird Enterprise

Hummingbird Enterprise являє собою всеосяжний програмний інструментарій для управління інформаційними потоками в масштабах організації.

Основні переваги.

Для кінцевих користувачів:

– Запуск процесів з популярних додатків, таких як Microsoft Outlook, Word і інших програм Office.

– Повідомлення виконавців про завдання через звичний інтерфейс – по електронній пошті, через браузер, систему миттєвого обміну повідомленнями, мобільний пристрій або модуль Webtop.

– Попередження про прострочені, незакінчені або розширені завдання, делегування одним клацанням миші (при наявності відповідних повноважень у користувача в цьому процесі).

Для адміністраторів:

– Наочний графічний інтерфейс, для простого завдання бізнес-потоків, використовуючи стандартну процедуру «drag and drop», завдання логіки умовних розгалужень, правил ескалації й бізнес-правил.

– Швидке моделювання методом проб і помилок на стадії проектування.

– Ефективний моніторинг статусу за допомогою графічних «моментальних знімків» всіх процесів безпосередньо з Web-браузера.

– Залучення партнерів, постачальників і інших зовнішніх учасників без необхідності додаткового навчання.

– Просте донастроювання й розширення бізнес-процесів при необхідності.

Для компаній:

– Швидке й погоджене просування документації на етапах створення, схвалення й публікації.

– Зберігання всієї інформації, документів і метаданих, пов'язаних із процесом, у захищеному репозитарії.

– Швидка реакція у відповідь на нові запити споживачів і мінливі умови ринку.

Основні компоненти Hummingbird Enterprise:

– Hummingbird DM – інтегрована система управління документами в масштабі підприємства, що дозволяє перетворювати електронну інформацію з документів у знання й яка є основою для рішень в області управління вмістом і колективною роботою. Надає великі можливості організації, пошуку й

поширення за допомогою інтуїтивно-зрозумілого середовища, що забезпечує простий і безпечний доступ до будь-яких сховищ корпоративних даних.

– Hummingbird RM – сумісна з основними стандартами система управління записами для створення захищеного, організованого робітничого середовища для контролю над життєвим циклом корпоративних даних від створення до знищення.

– Hummingbird WorkFlow – являє собою всеосяжний програмний інструментарій для управління інформаційними потоками в масштабах організації. Основні характеристики:

– просте графічне проектування процесів документообігу без допомоги IT– фахівців;

– контроль, управління й моніторинг всіх поточних процесів;

– залучення зовнішніх учасників при збереженні контролю над процесом;

– автоматичний запуск процесів, з огляду на різні етапи життєвого циклу інформації, таких як зворотне копіювання в сховище, публікація й знищення;

– розширення процесів Hummingbird Enterprise шляхом інтеграції зі сторонніми системами;

– призначення, моніторинг і завершення завдань і пов'язаної з ними документації через обраний користувачем інтерфейс – електронну пошту, браузер, IM або мобільний пристрій.

– Hummingbird Collaboration – надійно захищене мережне середовище колективної роботи й взаємодії для розподілених робочих груп для організації спільної роботи як внутрішніх (співробітників компанії), так і зовнішніх (замовників, партнерів) учасників на основі проектів / документів.

– Hummingbird KM – сполучення потужних технологій, що забезпечують пошук, категоризацію й добування інформації для найбільш швидкого й коректного доступу до корпоративних даних і накопиченого досвіду.

– Hummingbird Portal – робітниче середовище на основі Web, що погоджує воєдино додатки й інформацію, надаючи повний огляд накопичених корпоративних даних.

– Hummingbird BI – комплексний пакет аналізу, що дозволяє задавати питання й одержувати відповіді для наступного використання при побудові актуальних і точних звітів. Це інтегроване й масштабоване рішення дозволяє одержувати й аналізувати дані з CRM і ERP систем, сховищ даних, успадкованих додатків і нестандартних архівів даних.

– Hummingbird ETL – потужний засіб інтеграції даних, для виконання добування, перетворення й завантаження (ETL) даних між додатками, сховищами й архівами даних. Це рішення дозволяє здійснювати оперативний і коректний обмін даними таким чином, що критично важливі структуровані дані будуть доступні для інтелектуальних працівників організації для побудови запитів і аналізу.

Documentum

Платформа Documentum 6 є основою широкого спектра продуктів EMC Documentum, що включає крім платформної інфраструктури програмне забезпечення, згруповане у відповідні набори продуктів для кожної з областей керування змістом.

– Knowledge Worker Suite призначений для підтримки процесів створення вмісту й процесів взаємодії співробітників у ході його узгодження й обговорення. Найкраще застосування Knowledge Worker Suite знаходить у колективних процесах створення вмісту – таких, як керування технічною документацією, розробка нового продукту, розробка договорів і комерційних пропозицій й т.і.

– Transactional Content Management вирішує завдання автоматизації високопродуктивних операційних процесів. Основу Transactional Content Management становить EMC Documentum Business Process Suite, що забезпечує керування повним життєвим циклом бізнес-процесів – від проектування й

моделювання, до аналізу й оптимізації.

– Interactive Content Management забезпечує процеси створення, керування, публікації й доставки різних видів цифрового медіа-змісту по різних каналах. Рішення поєднує технології Digital Asset Management і Web Content Management.

– Compliance & Archiving – це набір продуктів, що забезпечують можливості збору й архівування змісту для довгострокового зберігання відповідно до корпоративних регламентів і вимогами регулювання. На основі продуктів Compliance & Archiving будуються рішення для довгострокового зберігання електронних документів, що відповідають таким стандартам керування записами, як Do 5015.2, DOMEA, MoReq, ISO 15489.

Гнучка, з найширшим діапазоном можливостей, платформа Documentum 6, є основою для додатків по керуванню змістом і архівуванню, надаючи єдині інструменти й сервіси для керування змістом, процесами й сховищами. Ключові інновації Documentum 6 включають:

– Спрощення розробки й інтеграції додатків завдяки сервісно-орієнтованому підходу до керування змістом.

– Прискорення розробки додатків завдяки новим засобам розробки, у яких застосовується технологія Eclipse.

– Розширення можливостей настроювання, що дозволяють прискорити розробку й багаторазово використовувати наявні результати.

– Нові засоби масштабування, істотно підвищувальна продуктивність у розподілених конфігураціях.

На сьогоднішній день EMC Documentum 6 є єдиною на ринку уніфікованою ECM-платформою корпоративного класу. Documentum 6 надає кращу інфраструктуру для будь-яких додатків по керуванню змістом у корпоративному масштабі й надає самий широкий набір сервісів по керуванню змістом, що дозволяє замовникам працювати з максимальною продуктивністю за рахунок оптимального використання інформаційних активів.

LanDocs

Система LanDocs являє собою пакет прикладних програм-компонентів.

LanDocs: ДІЛОВОДСТВО – базова система автоматизації процесів діловодства й ведення архіву електронних документів реалізована в архітектурі «клієнт-сервер» на базі промислової СУБД, функціонує в локальній мережі на комп'ютерах з ОС Windows. Поставляється для роботи із СУБД Oracle і MS SQL Server.

LanDocs: СЕРВЕР ДОКУМЕНТІВ – серверне програмне забезпечення для централізованого керування зберіганням змісту документів (файлів документів) в електронному архіві. Взаємодіє із системою LanDocs: ДІЛОВОДСТВО й здійснює підтримку операцій читання, запису, видалення, передачі файлів документів на довгострокове зберігання, протоколювання всіх цих операцій на спеціалізованому сервері під керуванням ОС Windows.

LanDocs: ПОВНОТЕКСТОВИЙ ПОШУК – серверне програмне забезпечує можливість пошуку по текстах документів з використанням морфологічної нормалізації.

LanDocs: СКАНУВАННЯ ДОКУМЕНТІВ – система сканування паперових документів і перегляду їхніх електронних образів дозволяє працювати з електронними копіями паперових документів, виводити зображення в різних масштабах, розглядати деталі в режимі «збільшувального скла», панорамувати, видаляти із зображення плями й багато чого іншого.

LanDocs: ПАКЕТНЕ СКАНУВАННЯ ДОКУМЕНТІВ – програма пакетного сканування являє собою розширення програми LanDocs: СКАНУВАННЯ ДОКУМЕНТІВ. На відміну від останньої, орієнтована на автоматизацію процесів масового (потокового) сканування багатосторінкових документів.

LanDocs: ПОШТОВА ПІДСИСТЕМА – спеціалізоване серверне програмне забезпечення LanDocs: ПОШТОВИЙ СЕРВЕР забезпечує можливість розсилання повідомлень, завдань і документів системи LanDocs користувачам

корпоративної електронної пошти.

Клієнтське програмне забезпечення LanDocs: ПОШТОВИЙ КЛІЄНТ, інтегроване із програмами електронної пошти MS Outlook або Lotus Notes, дозволяє користувачеві:

– одержувати завдання й документи з контуру регламентного діловодства;

– перенаправляти завдання й документи й здійснювати контроль їхнього виконання; робити узгодження документів;

– готовити й направляти відповідні документи й звітувати про виконання завдань;

– при використанні компонента LanDocs: ПІДСИСТЕМА БЕЗПЕКИ – також підписувати документи електронним цифровим підписом, робити перевірку дійсності підписів під документами й, при необхідності, шифрування повідомлень і документів.

LanDocs: ІНТЕРНЕТ ДОСТУП – спеціалізований WEB-сервер, що забезпечує користувачам можливість роботи із системою автоматизації діловодства й корпоративного архіву електронних документів через мережу Internet дозволяє виконувати реєстрацію, пошук, розсилання й інші операції з документами, використовуючи як клієнт стандартний інтернет-броузер.

LanDocs: ПІДСИСТЕМА БЕЗПЕКИ – спеціалізоване серверне й клієнтське програмне забезпечення, що реалізує функції захисту інформації за допомогою використання електронного цифрового підпису й шифрування даних. Забезпечує підтримку інфраструктури відкритих ключів (PKI) відповідно до RFC 2459 (Internet X.509 Public Key Infrastructure), підтримує функції центра сертифікації (certification authority) і централізованого сховища сертифікатів користувачів. Шифрування документів виробляється в прозорому для користувача режимі на індивідуальних ключах. Події, пов'язані з роботою підсистеми протоколюються в спеціальному журналі безпеки.

Підтримується одночасне використання декількох засобів криптографічного захисту інформації (криптопровайдерів). Поставляється у

варіантах:

– на основі базового криптопровайдера Microsoft (CSP).

LanDocs: МАРШРУТИЗАЦІЯ (WorkFlow) – програмне забезпечення, що розширює можливості базових компонентів LanDocs у частині керування маршрутами обробки документів, дозволяє наочним образом (за допомогою побудови графічної карти) задати процес руху документа. Карта визначає маршрут руху документа й задає умови переходу від одного етапу обробки до іншого. Підсистема виконує моніторинг активних робіт, ініційованих по маршрутних картах, переводить роботу з етапу на етап, формує, відправляє й контролює завдання, відслідковує строки виконання етапів. Функції аналізу дозволяють одержувати статистичні дані по фактичних параметрах процесів обробки документів.

LanDocs: XML-ПІДСИСТЕМА ОБМІНУ – дозволяє організувати обмін документами й здійснити синхронізацію довідників (класифікаторів) між двома локальними системами документообігу на основі LanDocs або між системою LanDocs і зовнішньою програмною системою. Обмін даними здійснюється у форматі XML. Центральним компонентом підсистеми є ядро, що забезпечує гарантовану керовану доставку пакетів даних. Підсистема надає стороннім розроблювачам документований інтерфейс прикладного програмування (API), що забезпечує можливості її настроювання на зовнішні джерела й приймачі XML-пакетів.

LanDocs: СЕРВЕР КОНТЕНТА – спеціалізований сервер додатків, що відкриває функціональність керування документами LanDocs для доступу із предметно-орієнтованих додатків. Побудований з використанням технології Microsoft .NET і за допомогою WEB-сервісів забезпечує доступ до функцій і об'єктів системи з WEB-додатків, використовується для інтеграції LanDocs з портальними додатками.

LanDocs: АДМІНІСТРАТОР – цей програмний модуль служить для настроювання на певну структуру діловодства й адміністрування системи

LanDocs.

LanDocs: ДИЗАЙНЕР ФОРМ – призначений для створення й настроювання екранних форм гнучких реєстраційних карток системи LanDocs.

LanDocs: ПЛАНУВАЛЬНИК – являє собою додатковий модуль, призначений для підтримки періодично виконуваних завдань (наприклад, підготовки щомісячних або щоквартальних звітів), а також генерації повідомлень у рамках цих завдань. Адміністраторові надається інструмент, що дозволяє створювати завдання, які будуть запускатися із заданою періодичністю.

LanDocs: OLE-SDK – документований OLE-інтерфейс для вбудовування сервісів керування документами LanDocs в Windows-додатки сторонніх розроблювачів.

Microsoft SharePoint Portal Server

SharePoint – веб-орієнтована платформа для спільної роботи й система керування документами, розроблена й продавана компанією Microsoft. Це рішення може використовуватися для створення корпоративного веб-порталу, на якому розміщаються спільно використовувані документи або спеціалізовані додатки, такі як вікі або блогі. Дані в SharePoint організовані у вигляді списків (наприклад, завдання, обговорення, календарі) і бібліотек документів. Функціональність SharePoint представляється користувачеві за допомогою веб-частин – елементів керування, що показують списки й що дозволяють редагувати їх. Такі веб-частини розміщаються на сторінках, які, у свою чергу, розміщаються на порталі й доступні користувачеві через браузер. Насправді, SharePoint є додатком ASP.NET 2.0, що використовує IIS для відображення веб-сторінок і SQL Server для зберігання даних.

SharePoint представлений у вигляді двох основних продуктів – Windows SharePoint Services (WSS) і Microsoft Office SharePoint Server (MOSS). Крім цього, пропонується інструментальний засіб Microsoft Office SharePoint Designer (SPD).

Windows SharePoint Services (WSS) – безкоштовний додаток до Windows Server. WSS надає базову інфраструктуру для спільної роботи – редагування,

зберігання документів, контроль версій і т.д. Так само він містить у собі таку функціональність, як «маршрути» руху документів (зачатки документообігу), списки завдань, нагадування, онлайн-дискусії. Раніше WSS був відомий як SharePoint Team Services.

Microsoft Office SharePoint Server (MOSS) – платний компонент для інтеграції функціональності SharePoint у роботу додатків MS Office. Він є надбудовою над WSS і розширює його можливості. Містить у собі інструменти для бізнес аналітики – Excel Services, Business Data Catalog. MOSS дозволяє одержати доступ до Microsoft Project Server і до форм Microsoft Office InfoPath через браузер, централізовано, відповідно до концепції багатомодульного порталу. Підтримує спеціальні бібліотеки, такі як PowerPoint Template Libraries. MOSS раніше був відомий як SharePoint Server і SharePoint Portal Server.

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 першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи управління електронним документообігом кафедри КбПЗ.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) 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

Опис розробленої системи управління електронної документації

Для роботи з електронними документами в Україні протягом багатьох років найбільше активно використовуються системи, які можна віднести в основному до двох класів: системи електронних архівів (СЕА) і системи управління електронним документообігом кафедри КбПЗ (СУЕД). Деякі автори пропонують виділяти ще один клас, цікавий саме для українських користувачів, – системи електронного діловодства, – і, щоб не плутати з останніми, називають їх СДЕ – системи діловодства електронні. Але в принципі вітчизняні СУЕД, як правило, містять у собі й ці функції.

Незважаючи на завали західних термінів у цьому IT-сегменті, багато фахівців в Україні воліють дотримуватися традиційного терміна «електронний документообіг», хоча він сьогодні не зовсім точно (повно) визначає суть питання. На практиці даний термін убирає в себе куди більше широкий зміст. Наприклад, у поняття «електронний документообіг» в українській компанії DIRECTUM вкладають весь спектр технологій: управління електронними документами (електронний архів); управління діловими процесами (workflow); управління інтернет-даними; управління знаннями; традиційне діловодство.

Українські системи більшою мірою орієнтовані на підтримку документоорієнтованого управління, а не на управління довільним контентом (тобто найрізноманітнішим інформаційним наповненням, змістом), як на Заході.

Названі вище умовні класи споконвічно націлювалися на різні завдання й сценарії використання.

СЕА створювалися, щоб протягом тривалого часу зберігати й ураховувати документи як такі своєрідні інформаційні склади. Документ супроводжувався

відповідними реквізитами (карткою, аналогічної бібліотечної), щоб його потім можна було знайти й переглянути. Для реалізації аналогічних завдань з'являються різні, у тому числі апаратні, системи зберігання й пошуку паперових документів, мультимедиа– і фотоматеріалів та ін., уніфікуються формати документів.

Що відбувається з документом поза архівом, дану систему не цікавить. У процесі розвитку в СЕА з'являються інструменти й функції для реалізації політики зберігання й міграції документів. Наприклад, виділяють документи оперативного й довгострокового зберігання й переклад з одного виду в інший. Таким чином, і тут починають з'являтися елементи руху й стадії життєвого циклу документа, найпростіші docflow і workflow.

Якщо СЕА споконвічно створювалися для зберігання й обліку документів, то СУЕД призначалися для управління документами.

У західних джерелах зустрічається кілька варіантів термінів, що мають відношення до управління документами, які, з одного боку, відбивають тенденції розвитку даного напрямку, а з іншогобоку – маркетингові хитрування деяких виробників. От ці терміни: Document Management – управління документами; Electronic Document Management System (EDMS) – система управління електронними документами, електронним документообігом; Enterprise Document Management Systems (теж EDMS) – системи управління корпоративними документами.

У різних авторів (і розроблювачів) у ці терміни може вкладатися різний зміст. Деякі наші експерти затверджують, що EDMS-системи відповідають скоріше за управління зберіганням документів, а не за документообіг (рух документів). Іншими словами, виходить, що якісь системи EDMS ближче до СЕА. Це свідчить про те, що в кожному конкретному випадку (незважаючи на заяви постачальників) потрібно більш глибоко вивчати функціональне наповнення й реальні можливості систем, перш ніж відносити їх до того або іншого класу.

Будь-який документ проходить різні фази свого життєвого циклу, і його, для зручності й прискорення роботи, на всіх цих стадіях можна зробити доступним іншим співробітникам, з якими необхідна взаємодія, у тому числі ще в процесі підготовки документа. Звідси виникають і такі поняття, як версіоність документів, стадії життєвого циклу, стану документів, маршрути руху, спільна (групова) робота, процес, docflow і workflow та ін.

У свою чергу, СУЕД розвиваються в напрямку посилення архівних функцій: з'являються різноманітні сховища документів, подання документа в різних форматах.

У результаті з'являється новий клас систем, що одержав назву ECM, у якому представлені функції обох вищезгаданих напрямків (СЕА й СУЕД).

Enterprise Content Management (ECM) – це управління корпоративними інформаційними ресурсами, їхнім змістом і наповненням. Поняття ECM-системи трохи ширше, ніж СУЕД. Під першою розуміють набір технологій, інструментів і методів, використовуваних для збору, управління, нагромадження, зберігання й доставки інформації всім споживачам усередині організації. Наприклад, для того, щоб мати право називатися ECM-системою, СУЕД повинна містити засоби сканування документів, гарантувати схоронність документів, підтримувати регламенти доступу до них і їхнього зберігання, управління інтернет-даними й «динамічним контентом» при організації взаємодії багатьох користувачів і т.д.

Комплексна автоматизація підприємств, компаній, як правило, будується шляхом інтеграції декількох систем, кожна з яких вирішує певне коло завдань. Тому дуже важливо правильно визначити, що саме повинне реалізовуватися в рамках кожної системи, і забезпечити їхня раціональна взаємодія.

Відповідно до набору реалізованих функцій і областей застосування в СУЕД укрупнено можна виділити наступні напрямки:

– системи діловодства;

– електронні архіви;

– workflow-системи (для стислості будемо називати їх WF-системами);

– комплексні, або ECM-системи (про останні ми вже говорили вище).

Якщо традиційні СУЕД усе ще тяжіють до автоматизації досить вузької сфери «канцелярсько-офісного» документообігу, то багато сучасних систем ЕДО демонструють значно більші можливості й орієнтовані також на роботу з бізнес-процесами, підтримку й управління інформаційними потоками й організацію взаємодії користувачів.

Вимоги до архітектури й устаткування для СУЕД зміщаються убік одночасної роботи багатьох користувачів з конкретними документами, процесами, проектами, більшими масивами неструктурованої інформації. Істотно підвищуються вимоги до можливостей пошуку, надійності зберігання, до обслуговування й підтримки значних обсягів збереженої інформації. Специфіка автоматизації бізнес-процесів у СУЕД виставляє свої вимоги по швидкості обробки інформаційних потоків, можливості їхнього контролю й перерозподілу, по автоматизації окремих операцій, кроків, робіт, по інструментах і засобам настроювання маршрутів та ін.

Які завдання й напрямки недоцільно реалізовувати в СУЕД? До подібних завдань можна віднести, наприклад, кадрове діловодство або облік фінансових документів, завдання аналізу та ін.

Незважаючи на те, що формально назви цих завдань начебто б прямо пов'язані з документами, перше завдання полягає не в управлінні самими документами, а в обліку й управлінні кадрами. Отут потрібно одержувати різні вибірки по персоналу (наприклад, по утворенню, підлозі, спеціальностям, даті прийому/звільнення), оперувати даними з кадрових наказів і т.п. Для цього інформація в базі даних повинна зберігатися в структурованому виді, а не як окремі неструктуровані документи. Створення кожного документа повинне відбиватися на зміні стану персоналу, тому для автоматизації кадрового діловодства доцільніше використовувати спеціалізовані системи управління

персоналом, які можуть бути інтегровані із СУЕД для зберігання й узгодження неструктурованої інформації (резюме, фотографій, кадрових наказів і т.д.).

Аналогічна ситуація зі структурованими фінансовими документами: рахунками, заявками на оплату, платіжними документами й т.п. Вони тісно пов'язані з розрахунком заборгованості, строками оплати, статтями бюджету й іншими параметрами, як правило, що враховуються в бухгалтерських й ERP-системах. Оскільки, наприклад, електронний образ рахунку-фактури однаково прийдеться формувати, уводити й зберігати в структурованому виді (у бухгалтерській або відповідній обліковій системі), то заносити його в СУЕД скануванням недоцільно. У цьому випадку найбільш правильний варіант – інтеграція ECM– і ERP-систем, при якій запису ERP-системи можуть відправлятися у вигляді вкладень у завдання ECM-системи, наприклад, для узгодження, а підсумкові звіти ERP-системи можуть зберігатися в СУЕД і підписуватися електронним цифровим підписом.

Тому, ухвалюючи рішення щодо виборі й розвитку IT-інфраструктури підприємства, керівник (або, принаймні, IT-служба) повинна вирішити нелегке завдання – оптимально наповнити інформаційне середовище компанії, відшукати найбільш ефективні для конкретної фірми сполучення продуктів і технологій, розмежувати області перетинання взаємодоповнюючих IT-систем найбільш раціональним образом. Загалом, знайти ефективне застосування для кожної. А для цього йому треба добре розбиратися в нюансах і термінології, у тонкощах систем, пропонованих сьогодні ринком, представляти й ураховувати особливості й істотні відмінності СУЕД і ERP-систем, характер даних систем кожного типу. Провести вододіл між цими системами й чітко визначити їхні технологічні границі, методологію взаємодії й завдання користувачів не тільки на сьогодні, але й хоча б на середньострокову перспективу, визначити потенційні обсяги даних, оцінити можливості масштабування та ін.

Довгострокова IT-стратегія повинна припускати наявність самостійної СУЕД, що буде вирішувати свої завдання й при необхідності інтегруватися з

ERP-системою по ряду напрямків. Приміром, від створення й руху документів, підписання їх ЕЦП до відправлення структурованих документів по маршрутах узгодження з використанням механізмів workflow СУЕД, розміщення документів тривалого зберігання й ERP-звітності в електронних архівах СУЕД та ін.

Відповідно, виходячи з вищесказаного, система управління електронного документообігу забезпечує:

1. Розмежування прав доступу до розробленої документації. Права доступу встановлюються для кожного об'єкта (залежно від ієрархії на дереві параметричної моделі), для кожного виробу (на рівні випуску певного типу документації).

2. Виключення дублювання й множинного введення тих самих характеристик у документах, що випускаються. При створенні нових документів виробляється автоматичний пошук наявності документа в древі документів, при наявності документа в древі формується запит на редагування наявного документа або заміни його новим документом. Будь-які дані мають у сховище даних своє унікальне місце розташування й дерево зв'язків цих даних з іншими об'єктами. Ці дані вводяться в сховище користувачем, що має на це відповідні права. При цьому активізуються зв'язки з іншими об'єктами й ним передаються посилання на ці дані із вказівкою, що об'єкт повинен зробити із цими даними (використовувати в якості вихідних даних, погодити документ і т.і.).

3. Автоматичну реєстрацію розроблених документів. При створенні нового документа виробляється автоматичне внесення документа в древо документів з наданням прав на даний документ, після чого він здобуває статус робочого документа.

4. Відображення інформації про документ на різних картках електронної картотеки. Картка електронної картотеки являє собою древо параметрів документів, що забезпечує швидкий пошук необхідної інформації й визначення стадії створення документа.

5. Надання списку всіх документів, що перебувають у роботі в кого-небудь із проектантів виробу. Це важливо, наприклад, для головних конструкторів і керівників відділів, які одержать можливість:

– у реальному масштабі часу, у дереві об'єктної моделі, визначити в якого об'єкта й у якому стані перебуває будь-який документ, що цікавить;

– одержати оперативну інформацію про маршрут проходження документа, що цікавить, і часу його знаходження на кожному етапі маршруту;

– визначити строки проходження етапів маршруту документа, що цікавить;

– у випадку порушення цих строків визначити зв'язку, по яких автоматично згенеруються повідомлення про порушення строків проходження етапів у маршруті документа;

– оцінити вплив порушення строків проходження документа на розробку пов'язаних з ним документів.

6. Забезпечення легкого пошуку документів і пов'язаної з ними інформації з можливістю збереження параметрів для подальшого використання. За допомогою електронної картотеки можна швидко знайти необхідний документ, або за допомогою запиту з певними параметрами знайти документи, схожі по певних атрибутах. Крім цього, можна використовувати повнотекстний пошук документації. При цьому запит може бути збережений для можливого його повторення в списку найбільше часто повторюваних запитів.

7. Автоматичну архівацію документів.

8. Маршрутизацію документів залежно від типу ділової процедури.

9. Роботу із загальрозпорядницькою документацією. За допомогою редактора загальрозпорядницької документації можна створювати накази, рішення, розпорядження; погоджувати їх, ставити на контроль, відслідковувати строки виконання й одержувати звіти.

10. Супровід процесу розробки документації з паралельним звертанням до електронних документів.

При реалізації інформаційного забезпечення на основі параметричної моделі об'єкта схема взаємодії параметричних об'єктів проектованих виробів має більш структурований вид, що різко зменшує кількість зв'язків, виключає дублювання потоків інформації, систематизує адміністрування, забезпечує контроль проходження документації й веде до істотних вигід, що дозволяє перевести процес автоматизації проектування на принципово новий рівень.

3.2 Розробка структурної схеми

На рисунку 3.1 зображена структурна схема розробленої відкритої системи обміну даними. Вона розроблена на основі проведеного дослідження існуючих еталонних моделей, за результатами яких за основу для синтезу еталонної моделі середовища відкритої системи обміну даними була прийнята модель, описана в міжнародному стандарті ISO/IEC TR 14252:1996 (E), ANSI/IEEE Std 1003/ 0-1995 Information technology – Guide to the POSIX Open System Environment (OSE). На основі даної моделі була розроблена еталонна модель середовища системи обміну даними, що складається із трьох технологічних рівнів:

– зовнішнього середовища;
– програмного забезпечення середнього рівня (посередника);
– користувальницьких додатків.

Ця модель й знайшла своє відображення у структурній схемі зображеній на рисунку 3.1.

На рисунку 3.2 зображена структурна схема системи управління електронної документації.

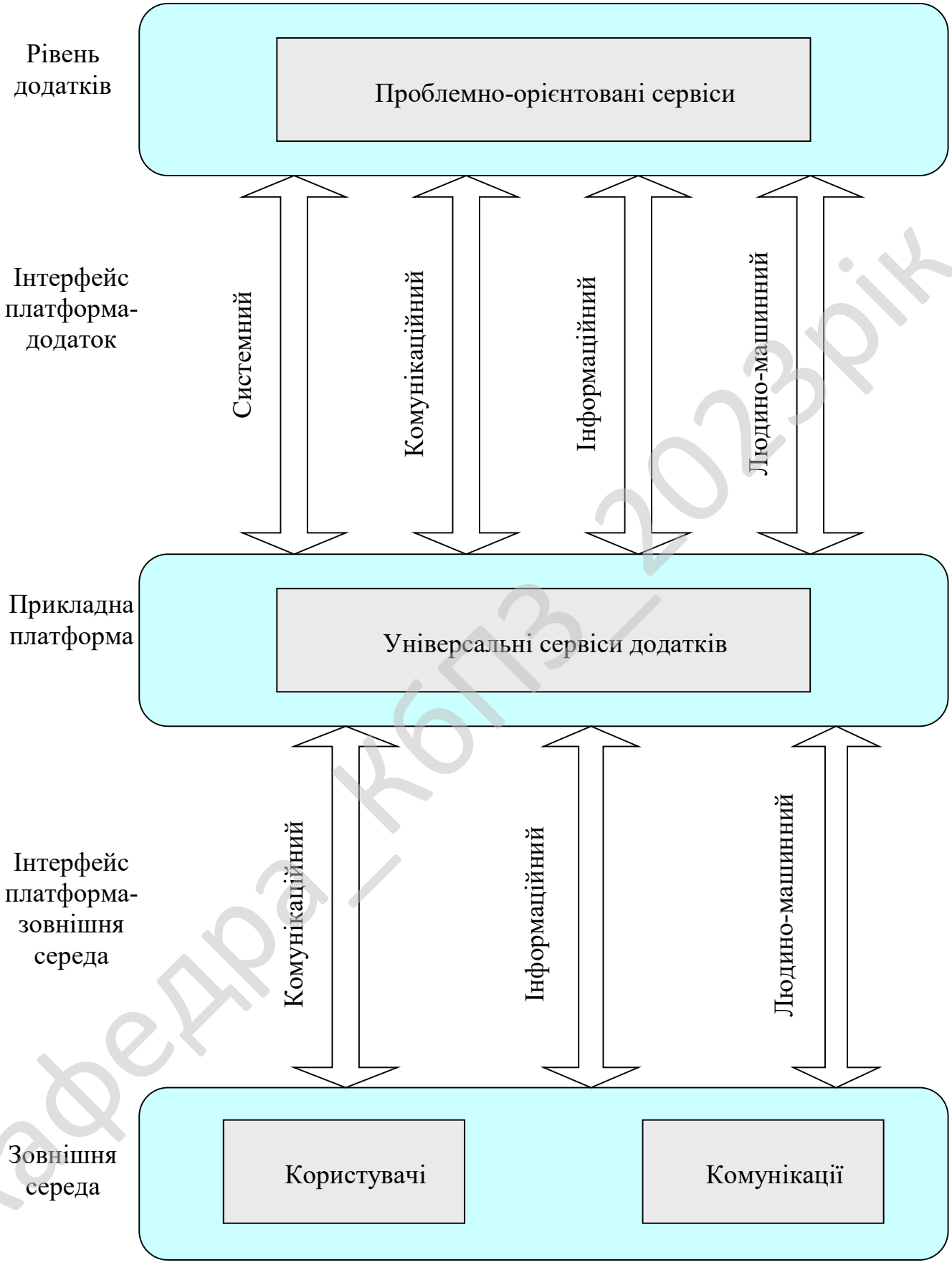

Рисунок 3.1 – Структурна схема відкритої системи обміну даними

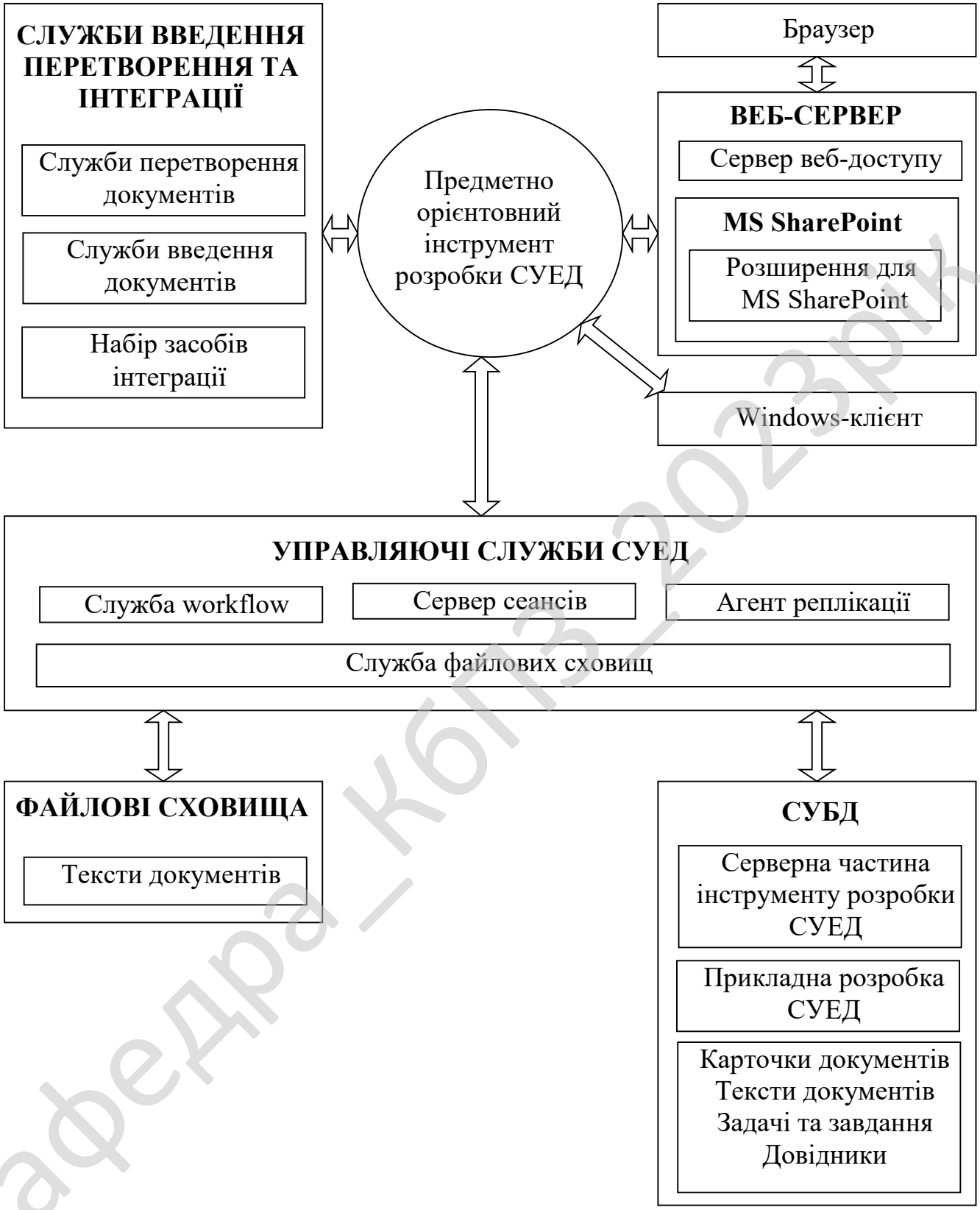

Рисунок 3.2 – Структурна схема системи управління електронної документації

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.

– 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; настроювання 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.

На базі системи СУЕД розробляється широкий набір бізнес-рішень, націлених на побудову ефективної системи керування бізнесом.

Бізнес-рішення – це програмно-консалтинговий продукт, орієнтований на досягнення заданого ефекту при рішенні певного бізнесу-завдання. Відмінні риси бізнес-рішення:

– Фокус на бізнес-завданнях. Метою бізнес-рішення є задоволення потреб конкретних бізнес-замовників і рішення конкретного бізнесу-завдання.

– Вимірюваний бізнес-ефект. Для кожного бізнес-рішення визначається перелік показників ефективності, які дозволяють виміряти ефект від його впровадження в організації.

– Методики й бізнес-консалтинг. Бізнес-рішення крім технічного рішення включає також методичні матеріали, технологію впровадження, послуги бізнесу-консалтингу.

Бізнес-рішення дають замовникам нові можливості в реалізації проектів, пов'язаних з електронним документообігом, керуванням бізнес-процесами й взаємодією. Завдяки пропрацьованості бізнес-рішень і орієнтації їх на конкретні бізнес-завдання, впровадження проходить із мінімальними ризиками.

Проекти реалізуються в цілому швидше й більш результативно. При цьому різні бізнес-рішення побудовані на одній основі – системи СУЕД. Це дозволяє забезпечити їхнє послідовне впровадження в організації й будувати з окремих бізнес-рішень цілісну інформаційну систему.

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 3.3).

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 кафедри КбПЗ 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.

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 кафедри КбПЗ і підвищити ефективність роботи ТЕ.

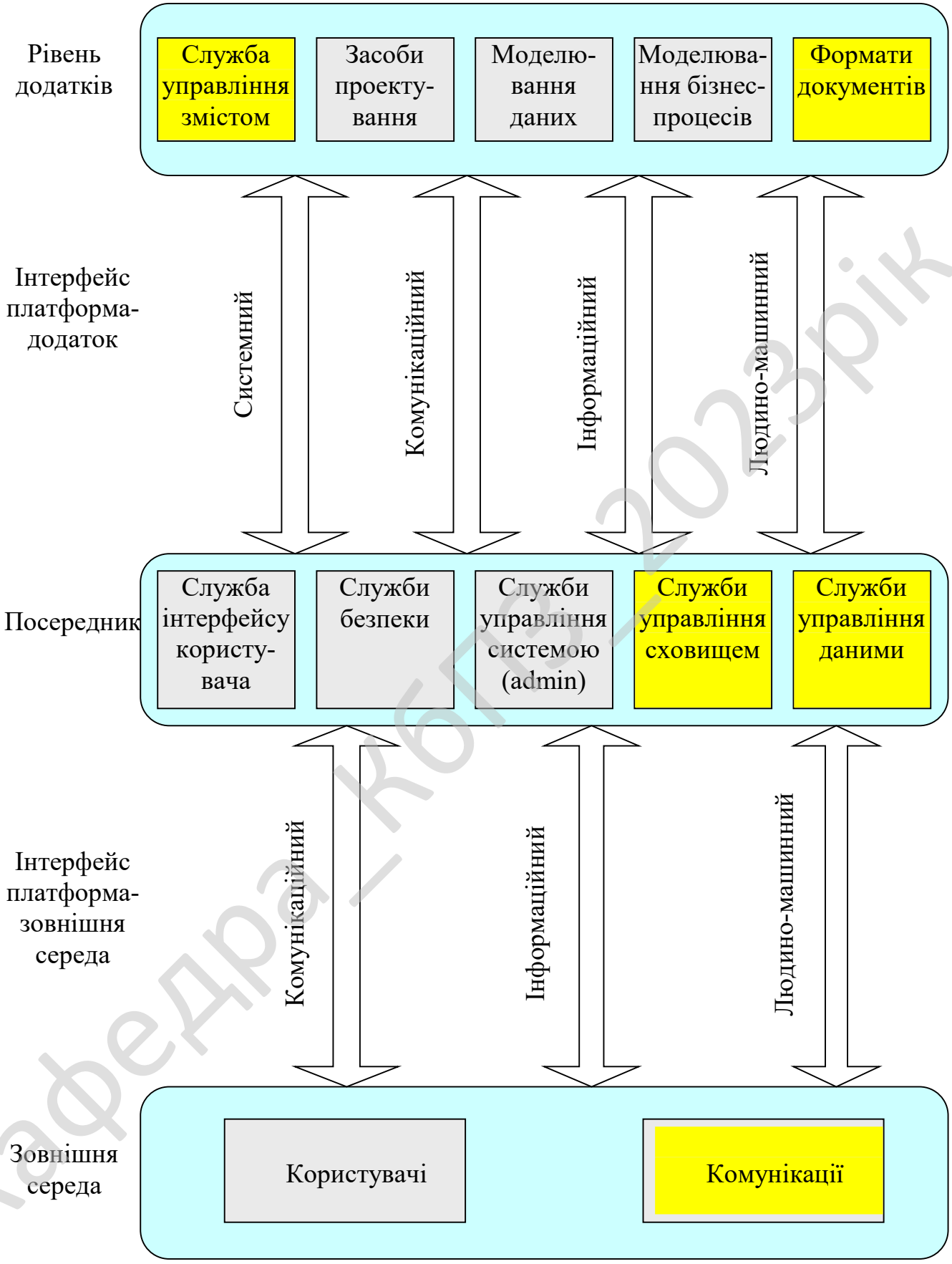

Рисунок 3.3 – Функціональна схема системи управління електронним документообігом кафедри КбПЗ

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 3.4.

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.

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.

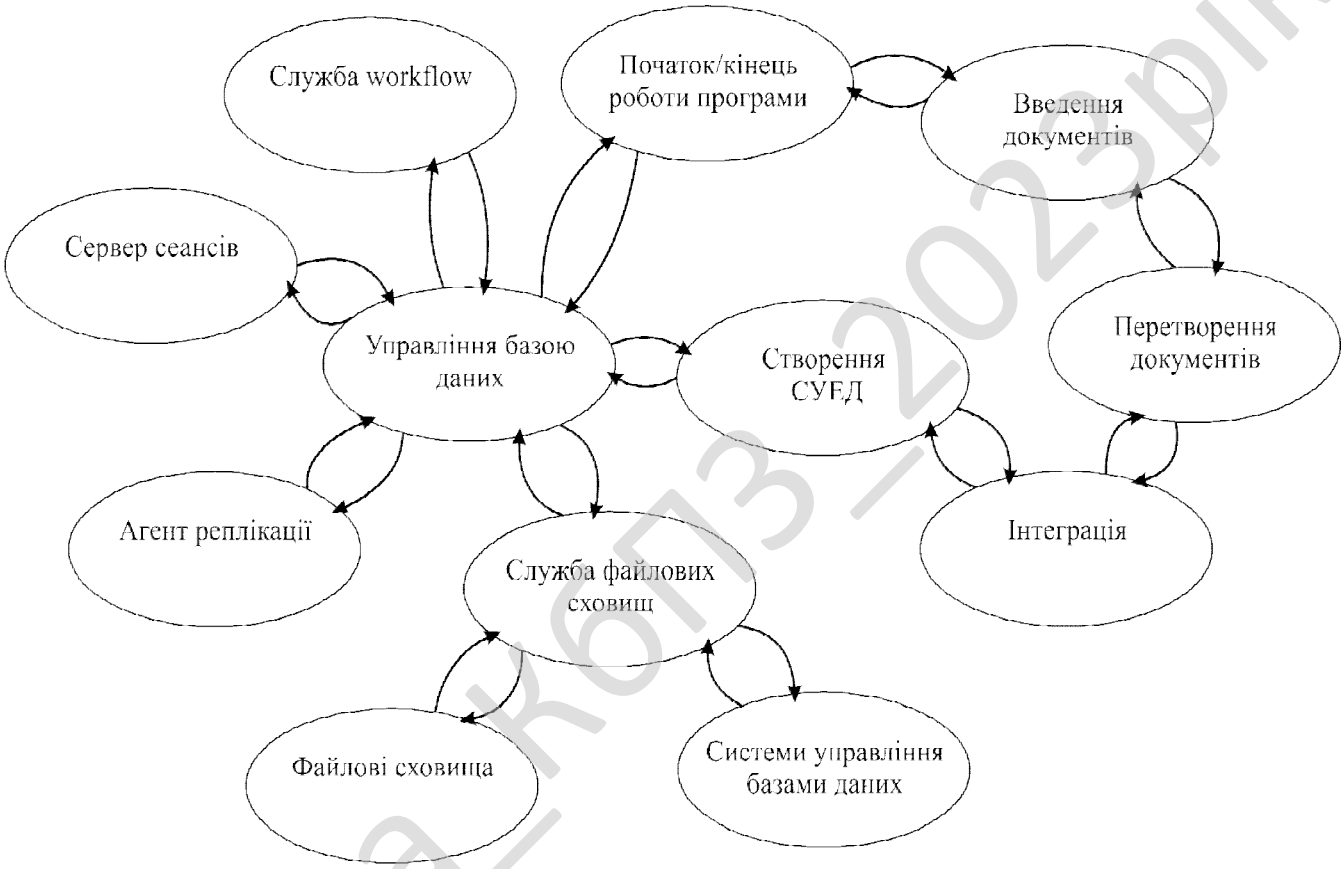

Рисунок 3.4 – Діаграма процесів системи

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 бакалаврської роботи, програмного забезпечення системи управління електронним документообігом кафедри КбПЗ.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ 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.

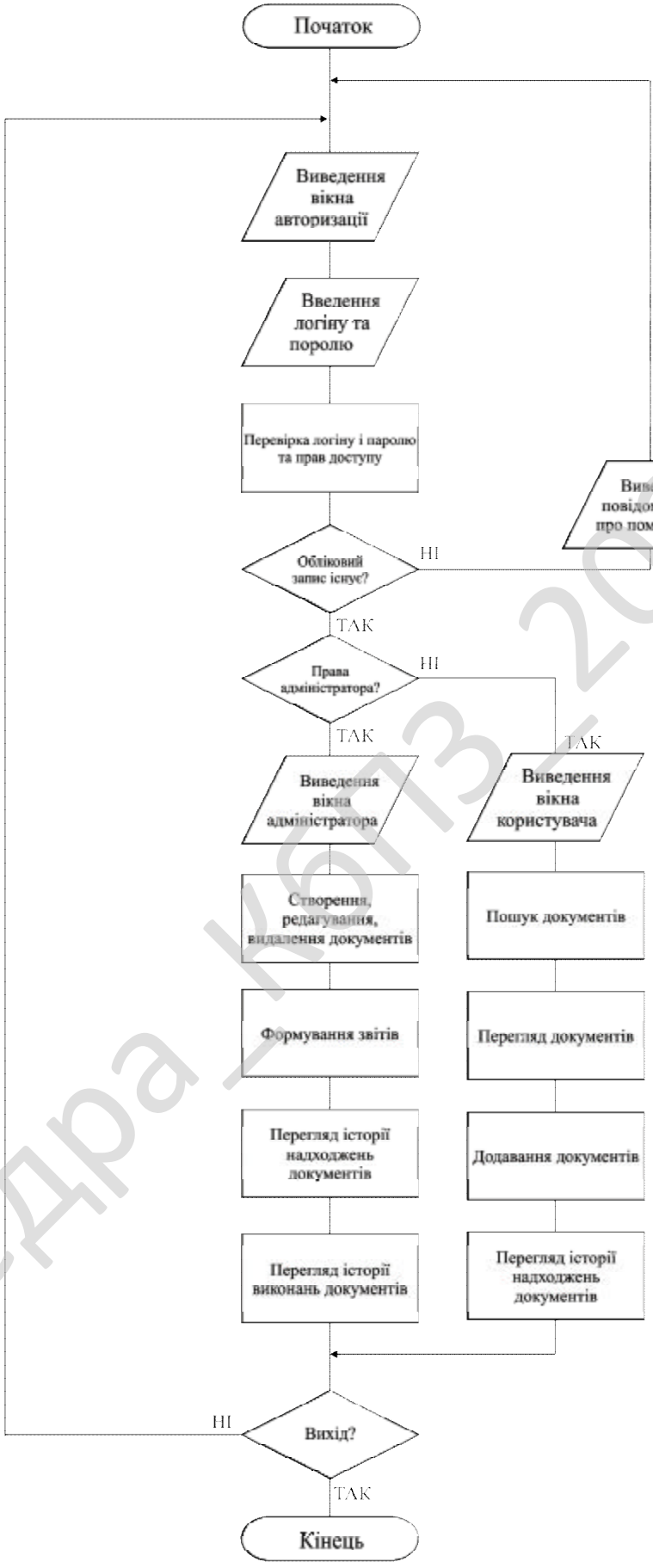

Рисунок 4.1 – Блок-схема алгоритму роботи основної програми

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.

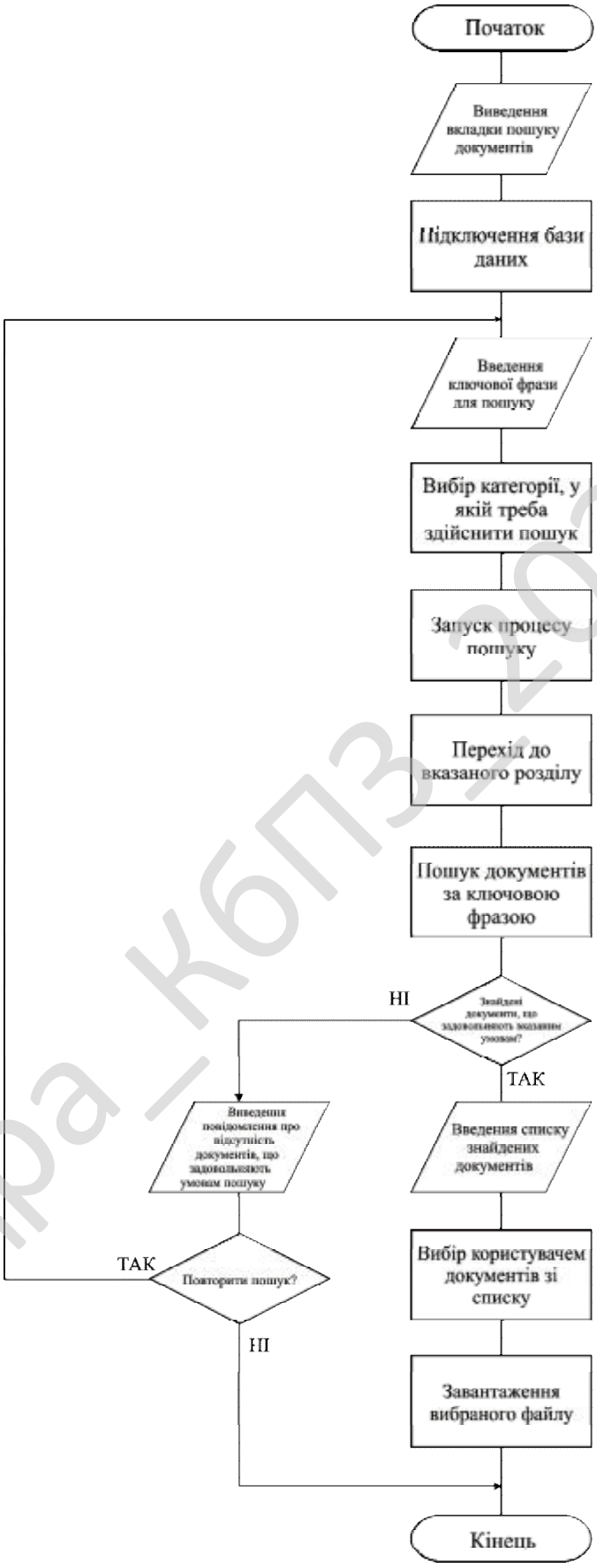

Рисунок 4.2 – Блок-схема алгоритму роботи підпрограми пошуку документів

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.

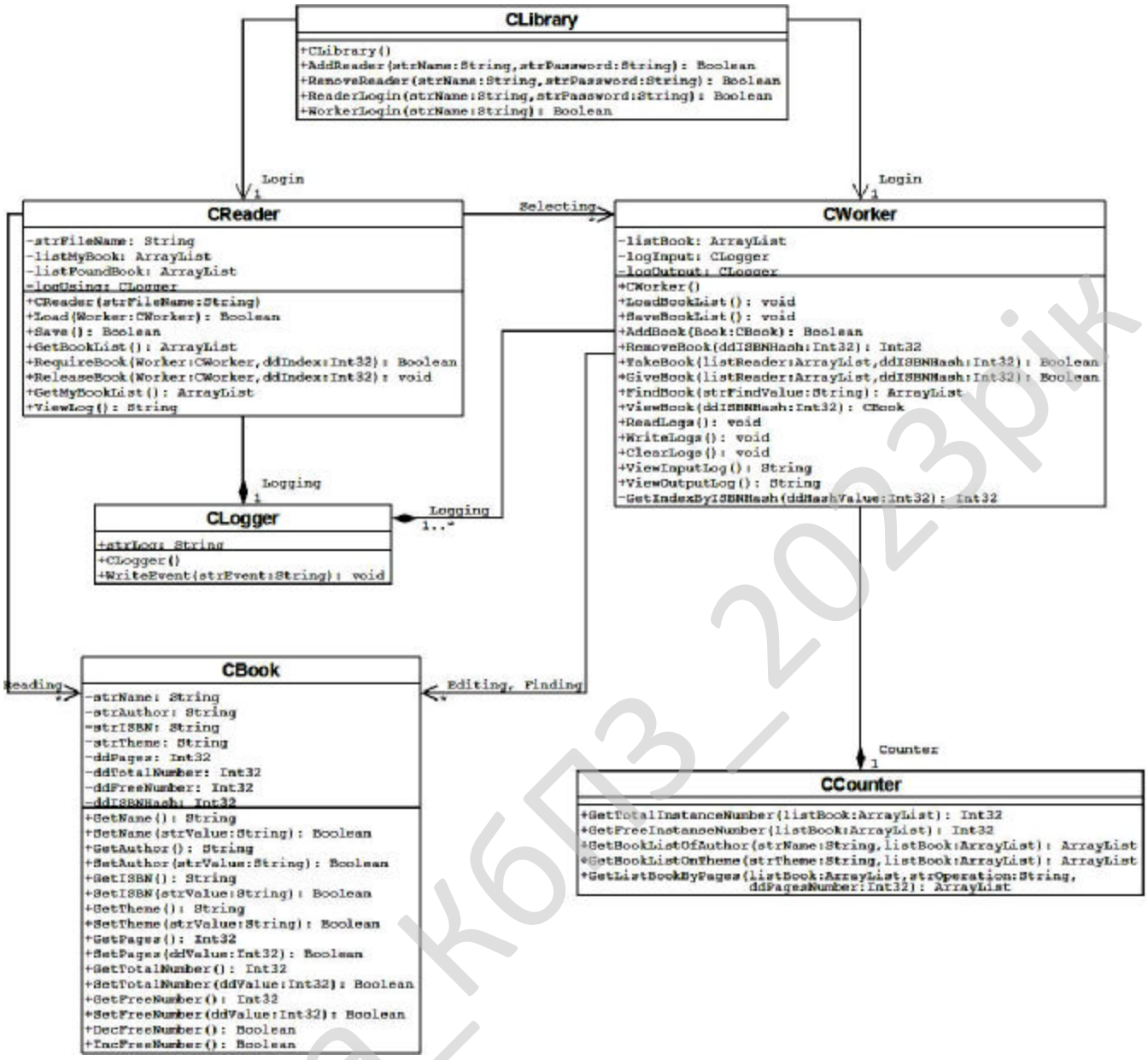

Рисунок 4.3 – Class_Digramm

Проведемо оцінку якості роботи системи управління електронним документообігом кафедри КбПЗ.

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 кафедри КбПЗ. 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.

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:

M={«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 кафедри КбПЗ 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 кафедри КбПЗ.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм Lucifer. Алгоритм Lucifer являє собою мережу перестановок і підстановок, його основні блоки нагадують блоки алгоритму DES. В DES результат функції f складається операцією XOR із входом попереднього раунду, утворюючи вхід наступного раунду. В S-блоках алгоритму Lucifer 4-бітові входи й виходи, вхід S-блоків являє собою перетасований вихід S-блоків попереднього раунду, входом S-блоків першого раунду служить відкритий текст. Для вибору використовуваного S-блоку із двох можливих використовується біт ключа. (Lucifer реалізує все це в єдиному T-блоці з 9 бітами на вході й 8 бітами на виході). На відміну від алгоритму DES, половини блоку між раундами не переставляються, та й саме поняття половини блоку в алгоритмі Lucifer не використовується. У цього алгоритму 16 раундів, 128-бітові блоки й більше проста, чим в DES, схема розгорнення ключа.

Блок тексту розглядається як ненегативне ціле число, або як кілька незалежних ненегативних цілих чисел. Довжина блоку завжди вибирається рівною ступеню двійки. У алгоритмі Lucifer використовуються наступні типи операцій:

– Таблична підстановка, при якій група біт відображається в іншу групу біт. Це так звані S-box.

– Переміщення, за допомогою якого біти повідомлення переупорядковуються.

– Операція додавання по модулю 2, позначувана XOR або ⊕.

– Операція додавання по модулю 2^{32} або по модулю 2^{16}.

– Циклічне зрушення на деяке число біт.

Ці операції циклічно повторюються в алгоритмі, створюючи так звані раунди. Входом кожного раунду є вихід попереднього раунду й ключ, що отриманий по певному алгоритму із ключа шифрування K. Ключ раунду називається підключем. Алгоритм шифрування може бути представлений у такий спосіб:

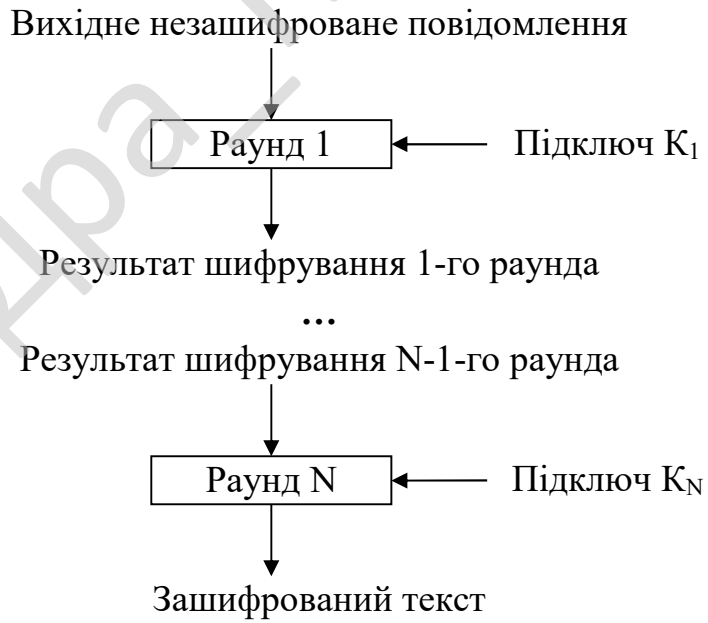

Рисунок 4.4 – Структура алгоритму алгоритмі Lucifer

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблене програмне забезпечення реалізує систему управління електронним документообігом кафедри КбПЗ.

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.

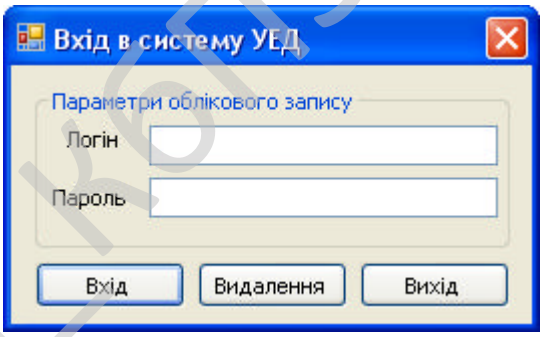

Рисунок 5.1 – Стартове вікно програми

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.

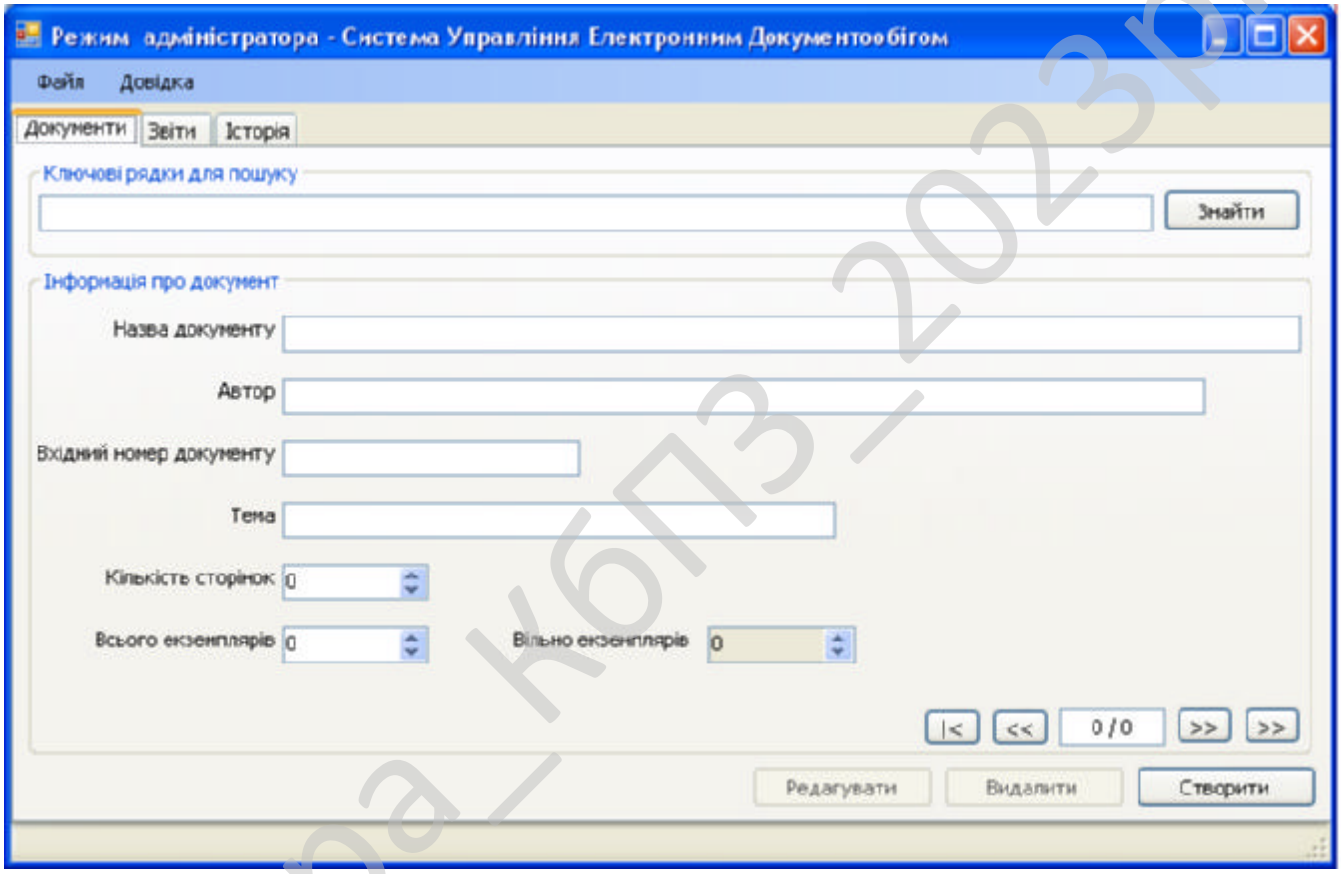

Рисунок 5.2 – Програма в режимі «Адміністратор»

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.

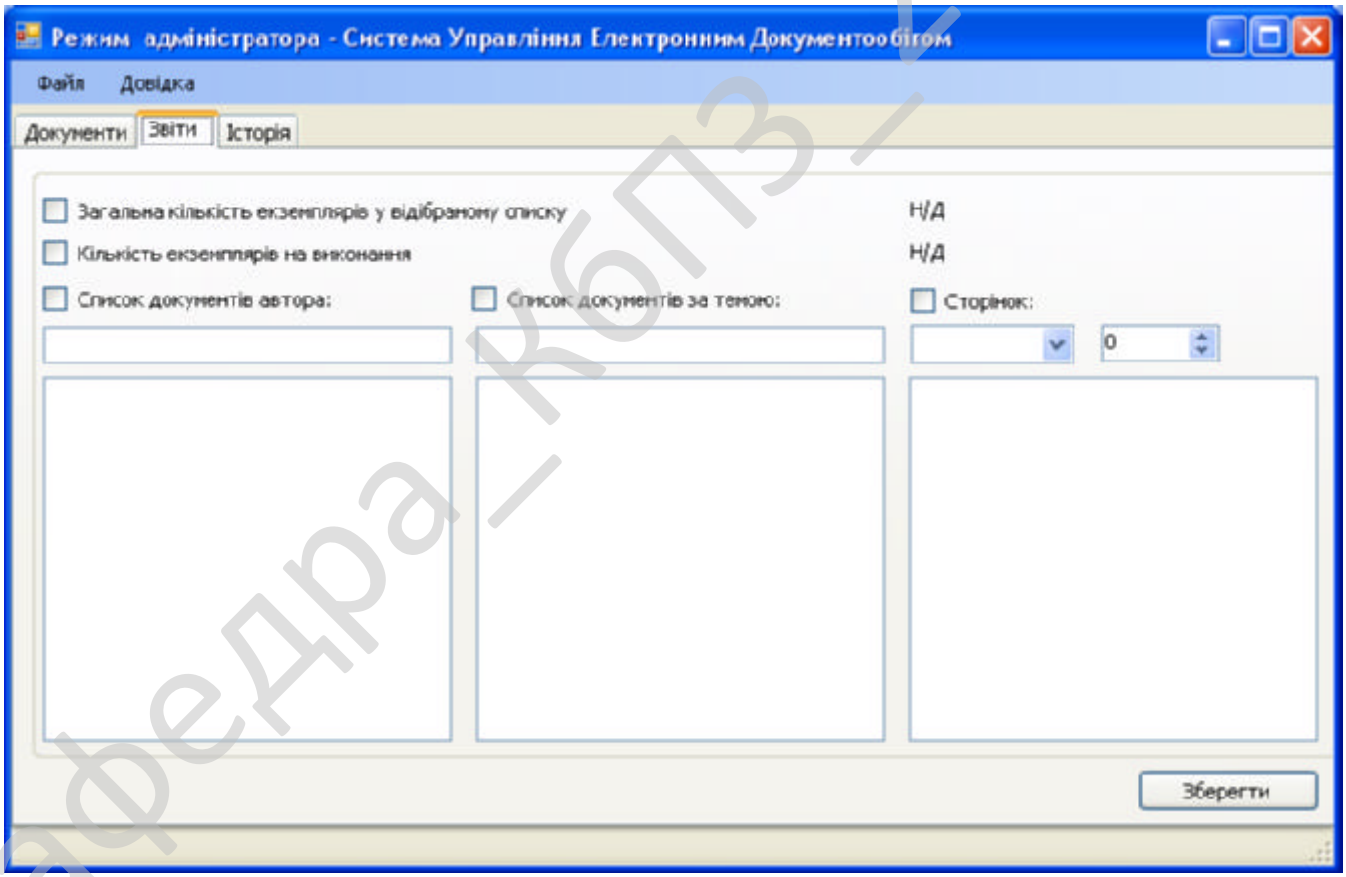

Рисунок 5.3 – Вікно перегляду статистики

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.

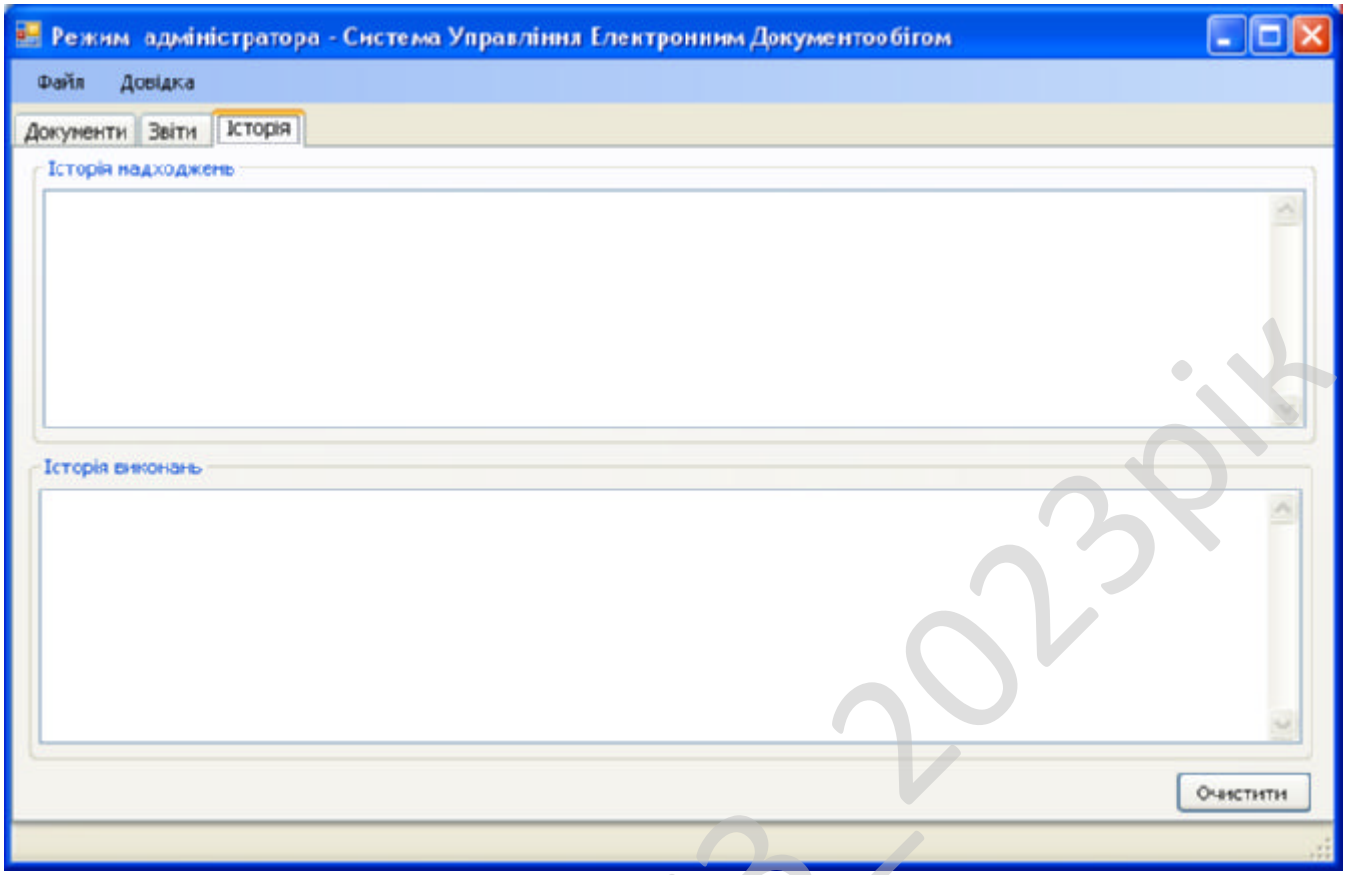

Рисунок 5.4 – Вікно історії надходжень і списань

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.

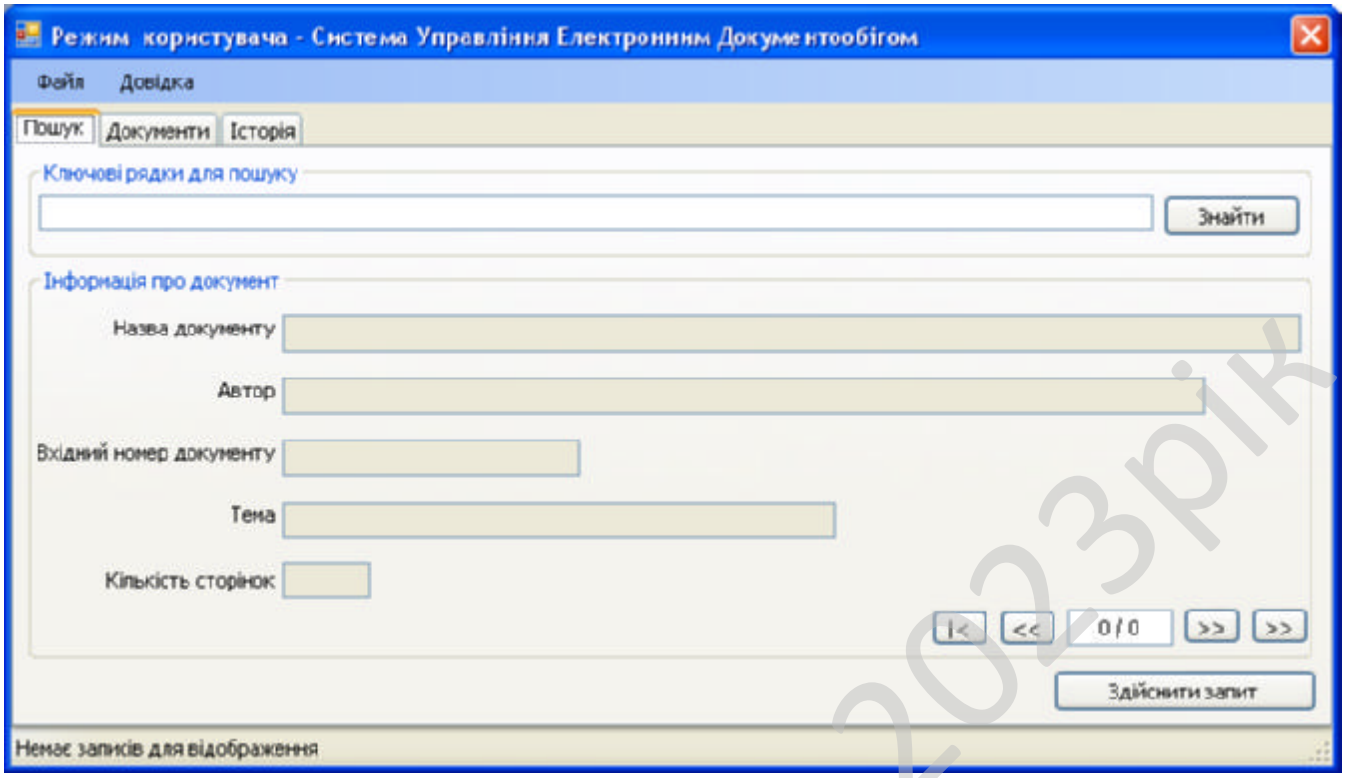

Рисунок 5.5 – Пошук літератури

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.

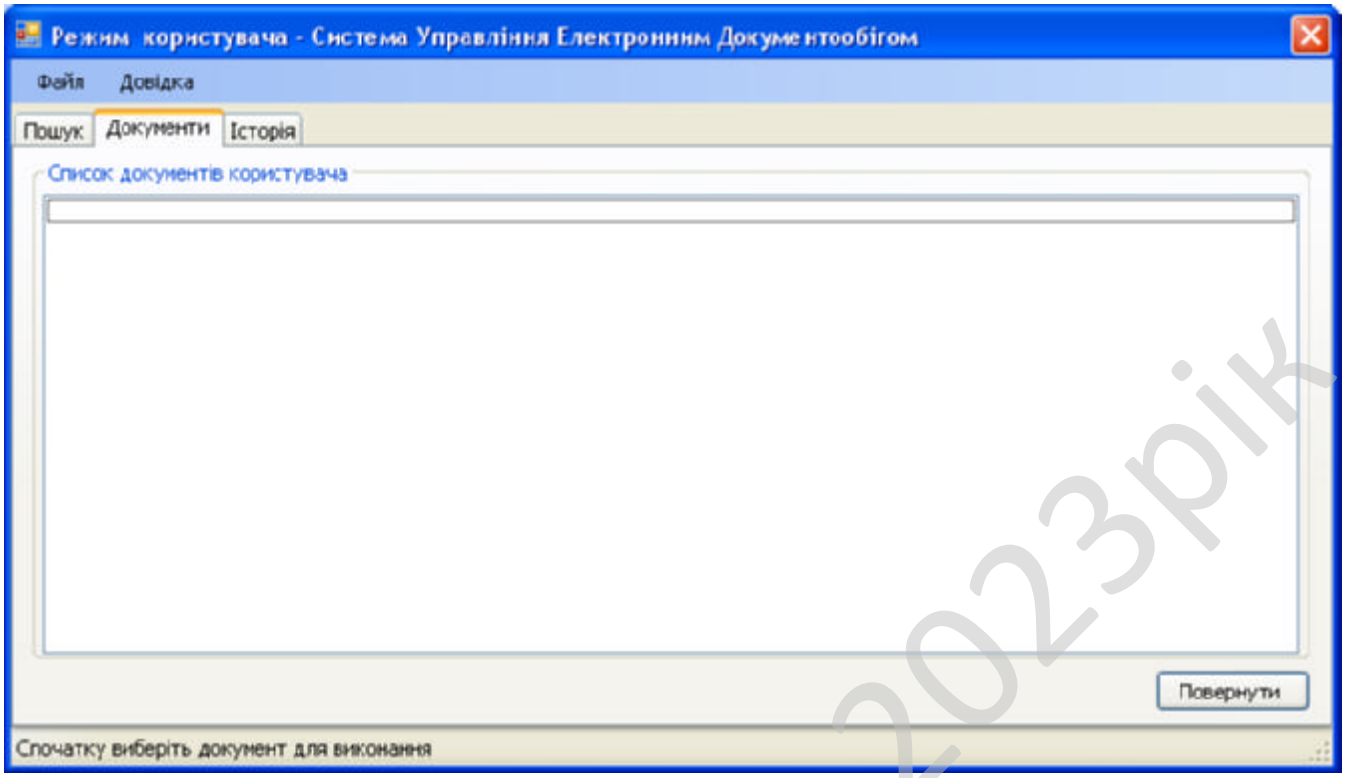

Рисунок 5.7 – Робота зі списком документів користувача

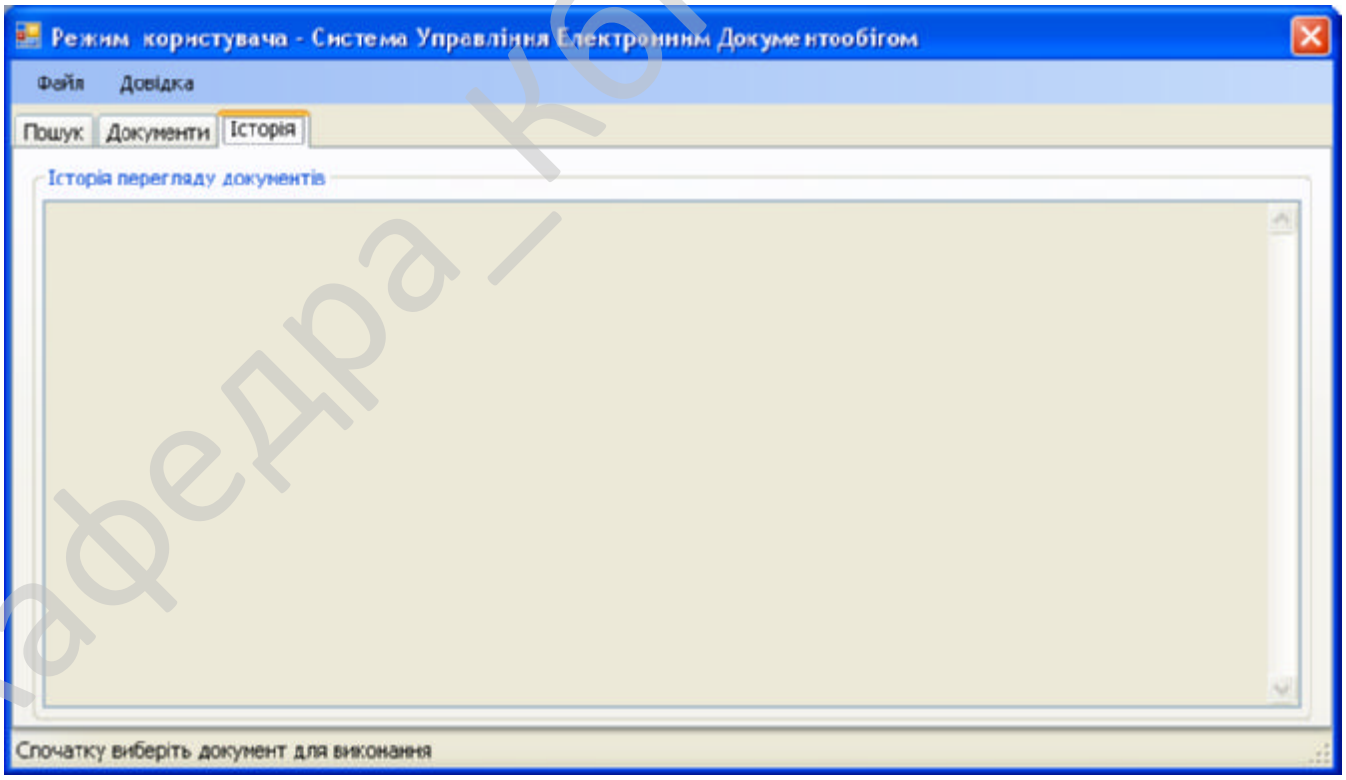

Рисунок 5.8 – Історія користування документом

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.

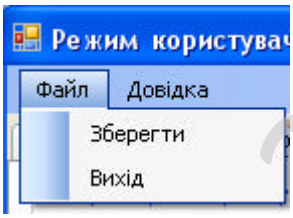

Рисунок 5.9 – Основне меню програми

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

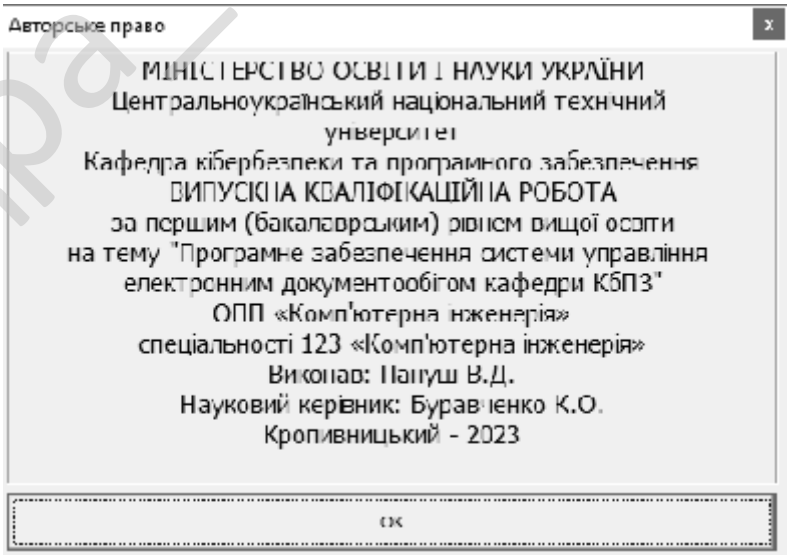

Рисунок 5.10 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи управління електронним документообігом кафедри КбПЗ.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем управління електронним документообігом кафедри КбПЗ.

– Досліджена система управління електронним документообігом кафедри КбПЗ.

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

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання управління електронним документообігом кафедри КбПЗ.

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 призначені для системи управління електронним документообігом кафедри КбПЗ. 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 Lucifer.

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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ISO/IEC TR 14252:1996 (E), ANSI/IEEE Std 1003/0-1995 Information technology – Guide to the POSIX Open System Environment (OSE)

2. Андрощук О.С. Методологічні аспекти побудови інтелектуальних систем підтримки прийняття рішень в особливих ситуаціях / О.С. Андрощук, В.В. Огурцов, О.І. Демідова // Восточно-Европейский журнал передовых технологий. – Х.: Технологический цент, 2009. – 5/2 (41). – С. 18-21.

3. Ашеров А. Т. Эргономика информационных технологий: учеб. издание / А.Т. Ашеров, С.А. Капленко, В.В. Чубук. – Х.: ХГЭУ, 2000. – 224 с.

4. Байлов В. В. Эксплуатация и сервис радиоэлектронных систем: учеб. Пособие / В.В. Байлов, В.С. Плаксиенко. – Таганрог: Изд-во ТРТУ, 2002. – 90 с.

5. Барзилович Е.Ю. Модели технического обслуживания сложных систем: учеб. пособие / Барзилович Е.Ю. – М.: Высш. школа, 1982. – 231 с.

6. Бартіш М.Я. Дослідження операцій. Лінійні моделі: підручник / М.Я. Бартіш, І.М. Дудзяни. – Львів: Видавничий центр ЛНУ імені Івана Франка, 2007. – Ч.1., 168с.

7. Бейхельт Ф. Надежность и техническое обслуживание. Математический подход / Ф. Бейхельт, П. Франкен. Пер. с нем. – М.: Радио и связь, 1988. – 392 с.

8. Беневоленский С.Б. Алгоритм идентификации процессов деградации физических свойств технических объектов / С.Б Беневоленский, А.А. Лисов // Измерительная техника. – М: Стандартинформ, 2005. – № 2. – С. 16-18.

9. Бет Э.В. Метод семантических таблиц / Э.В. Бет; перев. под ред. А.В. Идельсона и Г.Е. Минца // Математическая теория логического вывода. – М.: Наука, 1967. – С. 191-199.

10. Бланк И. А. Инвестиционный менеджмент / И.А. Бланк. – К.: Ника - Центр, 2006. – 448 с.

11. Борисюк А. О. Теоретичні основи автоматизації процесів вироблення рішень в системах управління Повітряних Сил: навч. посіб. для слухачів, курс. та студ. вищ. навч. закл. 2-ге вид. переробл. та доп. / А.О. Борисюк, М.А.Павленко, О.І.Тимочко; Мін-во освіти та науки України, ХУПС. – Х.: ХУПС, 2011. – 184 с.

12. Брюшинкин В. Н. Логика и процедуры поиска вывода / В.Н. Брюшинки; отв. за ред. Карпенко А.С // Логические исследования; Ин-т философии РАН. – М.-СПб: ЦГИ, 2010. – Вып. 16. – С. 85-106.

13. Василевський О. М. Нормування показників надійності технічних засобів: навчальний посібник / О. М. Василевський, В. О. Поджаренко. – Вінниця: ВНТУ, 2010. – 129 с.

14. Веклич В.Ф. Діагностування технічного стану тролейбусів./ В.Ф. Веклич – М.: Транспорт, 1990. – 295 с.

15. Вентцель Е.С. Теория вероятностей и ее инженерные приложения / Е.С. Вентцель, Л.А. Овчаров. – М.: Высш. шк., 2000. – 480 с.

16. Волков И.К. Исследование операций: Учебное пособие для вузов. 2-е изд. / И.К. Волков , Е.А. Загоруйко; под ред. В.С. Зарубина, А.П. Крищенко. – М.: МГТУ им. Н.Э. Баумана, 2002. – 436 с.

17. Воронин М.Я. Технико-экономическая эффективность сложных оптико-радиоэлектронных систем СВЧ: монография / М.Я. Воронин, И.Н. Карманов, М.Г. Карманова, И.В. Лесных, М.Ф. Носков, А.В. Синельников ; под общ. ред. М.Я. Воронина. – Новосибирск : СГГА, 2012. – 156 с.

18. Герасимов Б.М. Системы поддержки принятия решений: проектирование, применение, оценка эффективности / Б.М. Герасимов, М.М. Дивизинюк, И.Ю. Субач. – Севастополь: Издательский центр, 2004. – 318 с.

19. Герасимов Б.М. Человеко-машинные системы принятия решений с элементами искусственного интеллекта / Б.М. Герасимов, В.А. Тарасов, И.А. Токарев. – К.: Наукова думка, 1993. – 184 с.

20. Гладков Л.А. Генетические алгоритмы: учебн. пос. / Л.А. Гладков, В.В. Курейчик, В.М. Курейчик. – М.: Физматлит, 2006. – 320 с.

21. Головина Е.Ю. Интеллектуальные методы для создания информационных систем: учебн. пос. / Е.Ю. Головина. – М.: Издательский дом МЭИ, 2011. – 102 с.

22. Горбонос Ф.В. Економіка підприємств: підруч. / Ф.В. Горбонос, Г.В. Черевно, Н.Ф. Павленчик, А.О. Павленчик. – К.: Знання, 2010. – 463 с.

23. Горфинкель В.Я. Экономика предприятия / В.Я. Горфинкель, В.А. Швандар. – М.: ЮНИТИ-ДАНА, 2007. – 70 с.

24. Грицунов О.В. Інформаційні системи та технології. Навч. посібн. / О.В. Грицунов – Х.: ХНАМГ, 2010. – 222 с.

25. Данилевич С.Б. Влияние погрешности измерения на достоверность результатов выборочного измерительного контроля / С.Б. Данилевич, В.В. Княжевский // Измерительная техника. – М: Стандартинформ, 2004. – № 12. – С. 8-11.

26. Данилевич С.Б. Специфика измерений и допускового измерительного контроля / С.Б. Данилевич // Измерительная техника. – М: Стандартинформ, 2003. – № 8. – С. 16-18.

27. Данилов А.А. Метрологическое обеспечение измерительных систем: учеб. пособие / А.А. Данилов. – Пенза: Профессионал, 2008. – 63 с.

28. Демидов Б.А. Системная методология планирования развития, предпроектных исследований и внешнего проектирования вооружения и военной техники: монография / Б.А. Демидов, М.И. Луханин, А.Ф. Величко, М.В. Науменко. – К.: Стилос, 2011. – 464 с.

29. Демидов Б.А. Системно-концептуальные основы деятельности в военно-технической области: в 3 кн., кн. 2. Организационно-методические основы деятельности в военно-технической области / Б.А. Демидов. – К.: Технол. парк, 2006. – 1152 с.

30. Демидович Б. П. Краткий курс высшей математики: учеб. пособие для ВУЗов / Б.П. Демидович, В.А. Кудрявцев. – М.: Астрель, 2001. – 655 с.

31. Державна цільова науково-технічна програма створення державної інтегрованої інформаційної системи забезпечення управління рухомими об'єктами (зв'язок, навігація, спостереження): Постанова Кабінету Міністрів України від 17 вересня 2008 р. № 834 // Офіційний вісник України. – 2008. – №29. – С.1145.

32. Джарратано Д. Экспертные системы: принципы разработки и программирование / Д. Джарратано, Г. Райли.; пер. с англ. – М.: Вильямс, 2006. – 1152 с.

33. Евграфов В. Г. Особенности эргономического проектирования и экспертизы тренажерно-обучающих систем / В.Г. Евграфов. – Питер: СПб., 2007. – 224 с.

34. Заболотній С.В. Застосування розкладу в просторі з порідним елементом для вирішення задач ймовірнісної діагностики / С.В. Заболотній // Восточно-Европейский журнал передовых технологий. – Харьков: Технологический цент, 2014. – Вып. 4/4 (70). – С. 28-35.

35. Закон України "Про інфорацію" № 2658-XII: станом на 09 квітня 2015 р./ Верховна Рада України. – Офіц. вид. – Київ: Парлам. Ви-во, 2015. – № 26. – С. 219.

36. Зайцев Д. В. Многопозиционные радиолокационные системы. Методы и алгоритмы обработки информации в условиях помех / Д.В. Зайцев. – М: Радиотехника, 2007. – 114 с.

37. Зайцев Д. В. Багатопозиційні радіолокаційні системи / Д.В. Зайцев – М.: Радіотехніка, 2007. – 96 с.

38. Золотов С. И. Интеллектуальные информационные системы: учебн. пособ. / С.И. Золотов. – Воронеж: Научная книга, 2007. – 140 с.

39. Искусственный интеллект. Справочник / Под ред. Д.А. Поспелова. – М.: Радио и связь, 1990. – 348 с.

40. Інформаційні технології. Взаємозв'язок відкритих систем. Базова еталонна модель. Частина 1. Базова модель (ISO/IEC 7498-1:1994, IDT):ДСТУ ISO/IEC 7498-1:2004 – [Чинний від 2006–01–01]. – Київ: Держспоживстандарт України, 2007. – 67 с. – (Національний стандарт України).

41. Карманов И.Н. Измерения, испытания, контроль. Метрология и метрологическое обеспечение: учеб. пособ. / И.Н. Карманов, Н.А. Мещеряков, О.К. Ушаков. – Новосибирск: СГГА, 2006. – 184 с.

42. Каспина Т.В. Экономика и управление приборостроительным произвоздством: учебн. пособ. для высших учебных заведений / Т.В. Каспина, Н.Н. Лямина. – М.: ИЦ "Академия", 2008. – 240 с.

43. Клюев В.В. Неразрушающий контроль и диагностика. Справочник, 2-е изд., перераб. и доп. / В.В. Клюев – М: Машиностроение, 2003. – 656 с.

44. Клюня В.Л. Основы экономической теории / В.Л. Клюня, Н.В. Черченко. – Минск: Минск, 2006. – 238 с.

45. Коваленко А.С. Разработка структуры базы данных интегрированной информационной системы / А.С. Коваленко, А.В. Коваленко // Информационные технологии и защита информации в информационно-коммуникационных системах: монографія / Под редакцией профессора В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2015. – С. 54-64.

46. Кожанова А.С. Обґрунтування необхідності створення систем технічної діагностики інтегрованих інформаційних систем / О.А. Смірнов, А.С. Кожанова, О.В. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2013. – Вип. 6(113). – С. 255-257.

47. Коваленко А.С. Задачи распознавания ситуаций в ERP системах / А.В. Коваленко ., А.А. Смірнов, А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2014. – Вип. 4(120). – С. 161-164.

48. Коваленко А.С. Підсистема технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах /

А.С. Коваленко , О.А.Смірнов, О.В. Коваленко // Системи озброєння і військова техніка.– Х.: ХУПС, 2014. – № 1(37). – С. 126-129.

49. Коваленко А.С. Анализ эффективности использования экспертной системы технической диагностики с традиционной структурой / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 2(38). – С. 106-108.

50. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.

51. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.

52. Коваленко А.С. Структура системи технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Кіровоградського національного технічного університету / техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Кіровоград: Вид-во КНТУ, 2014. – Вип. 27. – С. 245-251.

53. Коваленко А.С. Дослідження будови інтегрованої інформаційної системи та її елементів / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 4(40). – С. 85-88.

54. Коваленко А.С. Розробка структури бази даних для обліку технічного стану елементів інтегрованої інформаційної системи з урахуванням вимог споживачів інформації / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2015. – Вип. 1(126). – С. 75-79.

55. Коваленко А.С. Обґрунтування набору даних для оцінки технічного стану інтегрованої інформаційної системи / А.С. Коваленко,

О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2015. – Вип. 1(42). – С.39-41.

56. Коваленко А.С. Експертна система технічного діагностування інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2015. – № 1(41). – С. 106-111.

57. Коваленко А.С. Удосконалення методу технічного обслуговування об'єктів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко, О.П. Доренський // Системи озброєння і військова техніка. – Х.: ХУПС, 2016. – № 2(46). – С. 109-114.

58. Коваленко А.С. Метод визначення оптимального комплексу робіт з відновлення працездатності інтегрованої системи технічної діагностики в умовах ресурсних обмежень / А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2016. – Вип. 3(140). – С. 69-72.

59. Kovalenko A.S. Information model and its element for displaying information on technical condition of objects of integrated information system / A.S. Kovalenko, A.A. Smirnov, A.V. Kovalenko, A.P. Dorensky // International Journal of Computational Engineering Research (IJCER). – India: Delhi, 2016. – Volume 6, Issue 1. – P. 21-27.

60. Кожанова А.С. Система технічної діагностики інтегрованих інформаційних систем – обґрунтування необхідності створення, визначення понятійного апарату та напрямів досліджень / А.С. Кожанова, О.А. Смірнов, М.П. Савченко, Д.М. Ізосімов, В.В. Мороз // Створення та модернізація озброєння і військової техніки в сучасних умовах: Тринадцята наук.-техн. конф., 5-6 вер. 2013 р., м. Феодосія: тези доп. – Феодосія: ДНВЦ, 2013. – С. 187-188.

61. Кожанова А.С. Визначення основних напрямків досліджень щодо створення системи технічної діагностики інтегрованих інформаційних систем / А.С. Кожанова, О.А. Смірнов, А.В. Челпанов // Проблемні питання розвитку

озброєння та військової техніки Збройних Сил України: IV наук.-техн. конф., 16-20 груд. 2013 р., м. Київ: зб. тез. – Київ: ЦНДІ ОВТ ЗСУ, 2013. – С. 293.

62. Коваленко А.С. Обґрунтування необхідності створення систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Інформатика та системні науки : V Всеукр. наук.-практ. конф., 13–15 бер. 2014 р., м. Полтава : зб. тез. – Полтава: ПУЕТ, 2014. – С. 292-294.

63. Коваленко А.С. Задачи распознавания ситуаций в системах организационной стратегии интеграции производства и операций / А.С. Коваленко, А.В. Коваленко // Комбінаторні конфігурації та їх застосування: XVI міжнар. наук.-практ. сем., 11-12 квіт. 2014 р., м. Кіровоград: зб. тез. – Кіровоград: КНТУ, 2014. – С. 53-55.

64. Коваленко А.С. Створення систем технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Проблеми і перспективи розвитку ІТ-індустрії: VI між нар. наук.-практ. конф., 17-18 квіт. 2014 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2014. – С. 241.

65. Коваленко А.С. Визначення понятійного апарату та напрямів досліджень для синтезу систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Комп'ютерне моделювання у наукоємних технологіях (КМНТ-2014): наук.-техн. конф. з міжнар. участю, 28-31 трав. 2014 р., м. Харків: зб. наук. праць. – Харків: ХНУ, 2014. – С. 190-193.

66. Коваленко А.С. Основні складові та функції системи технічної діагностики інтегрованих інформаційних систем / Коваленко А.С. // Інформаційні технології та комп'ютерна інженерія: наук.-практ. конф., 4 груд. 2014 р., м. Кіровоград: зб. тез доп. – Кіровоград: КНТУ, 2014. – С. 236.

67. Коваленко А.С. Розробка структури бази даних інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко //

Проблеми і перспективи розвитку ІТ-індустрії: VII міжнар. наук.-практ. конф., 17-18 квіт. 2015 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2015. – С. 15.

68. Коваленко А.С. Дослідження елементів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Комбінаторні конфігурації та їх застосування: XVII між нар. наук.-практ. сем., 17-18 квіт. 2015 р., м. Кіровоград: зб. тез – Кіровоград: КНТУ, 2015. – С. 5.

69. Коваленко А.С. Метод автоматизованої перевірки результатів вимірювання параметрів об'єкті в інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Стратегія якості у промисловості і освіті: XI міжнар. конф., 1 – 5 черв. 2015 р., м. Варна, Болгарія.: зб. матер. – Варна: ТУВ, 2015. – С. 423-426.

70. Коваленко А.С. Обґрунтування необхідності створення розподіленої бази даних для забезпечення захисту рухомих повітряних об`єктів / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Перспективні напрями захисту інформації: І всеукр. наук.-практ. конф., 07 вер. 2015 р., м. Одеса: зб. тез доп. – Одеса: ОНАЗ, 2015. – С. 35-39.

71. Коваленко А.С. Розробка інформаційної моделі автоматизованої оцінки технічного стану інтегральної інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Інформаційні технології та взаємодії (IT & I): II між нар. наук.-практ. конф., 3-5 лист. 2015 р., м. Київ: тези доп. – Київ: КНУ ім. Т. Шевченка, 2015. – С. 41-42.

72. Коваленко А.С. Разработка метода усовершенствования технического обслуживания интегрированной информационной системы / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Информационные и телекоммуникационные технологии: образование, наука, практика: II междунар. научн.-практ. конф., 3-4 дек. 2015 г., г. Алматы, Казахстан: сб. труд. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – Т.2. – С. 423-427.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРБ-123.23.0007.00.00.ТЗ			
Вим.	Арк.	№ документа	Підпис	Дата				
Розробив		Пануш В.Д.			Програмне забезпечення системи управління електронним документообігом кафедри КбПЗ	Літ.	Аркуш	Аркушів
Перевірив		Буравченко К.О.				Б	1	6
						ЦНТУ КМ-19		
Н. Контр.		Гермак В.С.						
Затв.		Смірнов О.А.						

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи управління електронним документообігом кафедри КбПЗ.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 10-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи управління електронним документообігом кафедри КбПЗ.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) 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;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи управління електронним документообігом кафедри КбПЗ;

– 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 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 84 аркуша.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

11.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 5.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Буравченко К.О.

Програмне забезпечення системи управління електронним документообігом кафедри КбПЗ

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>БАКАЛАВРСЬКА РОБОТА На тему: Програмне забезпечення системи управління
електронним документообігом кафедри КбПЗ Керівник: Буравченко К.О. Розробив:
студент Пануш Валентин Дмитровичгр. КМ-19 КНТУ 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();
             }
/*************************************************************************/
      };
}
```