

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ДЛЯ РОБОТИ З ВІДДАЛЕНИМИ ДАНИМИ

SOFTWARE FOR WORK WITH REMOTE DATA

Виконав(ла): студент(ка) 4 курсу, групи КНТ-212сп

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

СКИБА Д.В.

(ПРИЗВИЩЕ та ініціали)

Керівник КОЦУР М.І.

(ПРИЗВИЩЕ та ініціали)

Рецензент ІЛЛЯШЕНКО М.Б.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

СКИБИ Дениса Вадимовича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для роботи з віддаленими даними. Software for Work with Remote Data

керівник проєкту (роботи) к.т.н., доцент, КОЦУР Михайло Ігорович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 03 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЦУР М.І., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання “28” квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Денис СКИБА
(підпис) (Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Михайло КОЦУР
(підпис) (Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
100 с., 3 табл., 33 рис., 3 дод., 18 джерел.

C#, VISUAL STUDIO, ВЕБЗАСТОСУНОК, ДАНІ, ІНФОРМАЦІЙНА СИСТЕМА, МОВА ПРОГРАМУВАННЯ, СЕРЕДОВИЩЕ РОЗРОБКИ.

Об'єкт дослідження – процес розробки інформаційної системи для роботи з віддаленими даними.

Предмет дослідження – інформаційні системи для роботи з віддаленими даними.

Мета роботи – розробка програмного забезпечення інформаційної системи для роботи з віддаленими даними.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки Visual Studio.

Результати. Розроблено проєктні рішення для створення програмного забезпечення інформаційної системи для роботи з віддаленими даними. Створено програмне забезпечення інформаційної системи для роботи з віддаленими даними. Проведено тестування розробленого програмного забезпечення інформаційної системи для роботи з віддаленими даними.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення інформаційної системи для роботи з віддаленими даними за допомогою мови програмування C# та середовища розробки Visual Studio.

Галузь використання – зберігання та обробка даних.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 100 pages, 3 tables, 33 figures, 3 appendixes, 18 sources.

C#, VISUAL STUDIO, WEB APPLICATION, DATA, INFORMATION SYSTEM, PROGRAMMING LANGUAGE, DEVELOPMENT ENVIRONMENT.

The object of research is the process of developing information system for working with remote data.

The subject of the research is information systems for working with remote data.

The purpose of this work is to develop the software of the information system for working with remote data.

Materials, methods and technical tools: C# programming language, Visual Studio development environment.

Results. Project solutions for the creation of software for the information system for working with remote data has been designed. The software of the information system for working with remote data has been developed. The developed software of the information system for working with remote data has been tested.

Conclusions. The goal of the work has been achieved. The software tools of information system for working with remote data using the C# programming language and the Visual Studio development environment have been developed.

Scope of use – data storage and processing.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Інформаційні системи для підтримки роботи з віддаленими даними.....	11
1.2 Аналіз інформаційних систем для роботи з віддаленими даними	22
1.3 Висновки за розділом 1	32
2 Матеріали і методи	34
2.1 Вибір мови програмування	34
2.2 Вибір середовища розробки для створення програмного забезпечення інформаційної системи для роботи з віддаленими даними.....	37
2.3 Висновки за розділом 2	40
3 Опис програми	41
3.1 Структура програмного забезпечення інформаційної системи для роботи з віддаленими даними	41
3.2 Функціонування програмного забезпечення інформаційної системи для роботи з віддаленими даними	43
3.3 Опис особливостей реалізації програмного забезпечення інформаційної системи для роботи з віддаленими даними.....	45
3.4 Проектування інтерфейсу програмного забезпечення інформаційної системи для роботи з віддаленими даними.....	48
3.5 Висновки за розділом 3	52
4 Експлуатація, тестування та експериментальне дослідження програми.....	54
4.1 Призначення й умови застосування програми	54
4.2 Характеристики програмного забезпечення інформаційної системи для роботи з віддаленими даними	55
4.3 Інструкція по експлуатації програми.....	56
4.3.1 Звернення до програми.....	56
4.3.2 Вхідні й вихідні дані	56

4.3.3 Повідомлення.....	57
4.4 Виконання програмного забезпечення інформаційної системи для роботи з віддаленими даними	57
4.5 Тестування програмного забезпечення інформаційної системи для роботи з віддаленими даними	62
4.6 Висновки за розділом 4	62
Висновки	63
Перелік джерел посилань	66
Додаток А Технічне завдання	68
Додаток Б Фрагмент тексту програми	72
Додаток В Слайди презентації	93

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVVM – Model View ViewModel;

VS – Visual Studio;

БД – база даних;

ІС – інформаційна система;

ПЗ – програмне забезпечення.

ВСТУП

Програмне забезпечення для роботи з віддаленими даними залишається актуальним завдяки широкому використанню хмарних технологій. Такі інформаційні системи забезпечують зручний доступ до файлів з будь-якого пристрою, що особливо важливо в умовах віддаленої роботи та необхідності спільного редагування документів у реальному часі [1]-[3].

Одним із ключових факторів популярності таких інформаційних систем є безпека даних. Локальні носії схильні до фізичних пошкоджень та збоїв, тоді як хмарні сервіси забезпечують резервне копіювання та додаткові механізми захисту інформації. Також важливим аспектом є інтеграція з іншими програмними сервісами чи корпоративними комунікаційними платформами, що дозволяє автоматизувати бізнес-процеси та покращувати ефективність роботи з файлами [3]-[5].

Гнучкість та масштабованість таких інформаційних систем робить їх зручними як для індивідуальних користувачів, так і для великих компаній, оскільки вони дозволяють розширювати доступний простір і налаштовувати систему під власні потреби [6], [7].

Попри всі переваги програмного забезпечення для роботи з віддаленими даними, такі рішення мають і певні недоліки. Суттєвим недоліком таких інформаційних систем є їх вартість. Базові версії багатьох сервісів для роботи з віддаленими даними можуть бути безкоштовними, але при потребі у додатковому обсязі пам'яті, розширених функціях чи підвищеній безпеці користувачам доводиться оплачувати підписку. Для великих компаній витрати на хмарні рішення можуть виявитися значними. Продуктивність таких сервісів також може стати проблемою. При роботі з великими файлами або великою кількістю запитів швидкість завантаження та синхронізації може знижуватися, що негативно впливає на ефективність роботи інформаційних систем обробки та зберігання віддалених даних. Водночас існує проблема монополізації ринку, адже великі компанії, такі як

Google, Microsoft і Dropbox, домінують у цій сфері. Це створює попит на альтернативні рішення, зокрема для корпоративних клієнтів.

Тому актуальною є розробка інформаційної системи для роботи з віддаленими даними. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення інформаційної системи для роботи з віддаленими даними.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та інформаційних систем для роботи з віддаленими даними;
- здійснити проектування програмного забезпечення інформаційної системи для роботи з віддаленими даними;
- створити програмне забезпечення інформаційної системи для роботи з віддаленими даними;
- виконати тестування розробленого програмного забезпечення інформаційної системи для роботи з віддаленими даними.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Інформаційні системи для підтримки роботи з віддаленими даними

Інформаційні системи для роботи з віддаленими даними забезпечують безпечний доступ, збереження, обробку та передачу інформації незалежно від місця розташування користувача. Такі інформаційні системи використовуються в корпоративному середовищі, наукових дослідженнях, дистанційному навчанні та інших сферах, де необхідний оперативний доступ до даних через мережу [1].

Однією з основних функцій інформаційних систем для роботи з віддаленими даними є централізоване управління інформацією, що дає змогу користувачам взаємодіяти з даними через хмарні сервіси, вебінтерфейси або спеціалізовані клієнтські програми. Завдяки цьому забезпечується безперервний обмін файлами, доступ до баз даних, спільна робота над документами та підтримка операцій у розподілених середовищах [1].

Використання інформаційних систем для роботи з віддаленими даними дає можливість безпечно обмінюватися файлами та взаємодіяти з інформацією незалежно від місця перебування. Надійний захист даних є ключовим аспектом для організацій, що працюють у віддаленому режимі або використовують хмарні технології для зберігання інформації [1].

В умовах дистанційної роботи критично важливо дотримуватись заходів безпеки, щоб запобігти несанкціонованому доступу до конфіденційних матеріалів. Багато організацій упроваджують спеціальні політики та програми навчання, що дозволяють користувачам правильно використовувати хмарні сервіси та уникати потенційних ризиків, пов'язаних із передачею та обробкою інформації [1], [2].

Завдяки можливостям сучасних хмарних платформ можна отримувати доступ до даних з будь-якого пристрою, підключеного до мережі. Це суттєво спрощує робочі процеси, знижує потребу у фізичному зберіганні файлів та

забезпечує синхронізацію інформації між різними системами. Використання таких технологій дозволяє організаціям мінімізувати затримки у виконанні завдань і зробити процеси обміну даними більш ефективними [1], [2].

Безпека в хмарних сервісах реалізується за допомогою вбудованих механізмів шифрування, контролю доступу та багаторівневої автентифікації. Це допомагає запобігати витокам інформації та забезпечує захист переданих даних. Водночас користувачі також мають дотримуватись установлених правил безпеки, адже недостатньо надійні паролі чи передача облікових даних третім особам можуть створити загрозу інформаційній безпеці [1], [2].

Інформаційні системи для роботи з віддаленими даними можуть бути налаштовані відповідно до потреб бізнесу, дозволяючи автоматизувати деякі аспекти управління інформацією. У певних сферах, де зберігання даних на локальних серверах є обов'язковим, можливе використання гібридних рішень, що поєднують переваги локальної інфраструктури та хмарних технологій [1].

Розвиток хмарних сервісів відкриває нові можливості для організацій, які прагнуть забезпечити безперебійну взаємодію співробітників у гнучких робочих умовах. Віддалена робота та цифрові технології стають важливими елементами сучасного бізнесу, і їх інтеграція дозволяє компаніям оптимізувати процеси та підвищувати ефективність роботи [1], [2].

Використання хмарних технологій у таких системах забезпечує гнучкість і масштабованість, дозволяючи компаніям швидко адаптувати інфраструктуру до змінних умов роботи. Користувачі можуть отримувати доступ до своїх файлів із будь-якого пристрою, підключеного до Інтернету, а також синхронізувати інформацію між різними платформами [1], [2].

Популярні інформаційні системи для роботи з віддаленими даними включають різноманітні рішення, такі як FTP-клієнти, хмарні файлові менеджери та системи управління документами. Вони відрізняються функціональністю, рівнем безпеки та можливостями інтеграції з іншими сервісами, що дозволяє організаціям обирати оптимальні рішення для своїх потреб [1], [2].

Інформаційні системи для підтримки роботи з віддаленими даними відіграють ключову роль у сучасному цифровому середовищі, сприяючи ефективному управлінню інформацією та забезпечуючи безпечну взаємодію користувачів з даними незалежно від їх місцезнаходження [1], [2].

Інформаційні системи, що забезпечують роботу з віддаленими даними, є важливим інструментом для організацій, які використовують хмарні технології для зберігання та обміну інформацією. Використання таких рішень дозволяє працювати в розподілених командах, забезпечуючи доступ до файлів та ресурсів незалежно від місцезнаходження користувачів [1], [2].

Функціональність подібних систем визначається потребами компанії. Одні з них орієнтовані на індивідуальних користувачів, інші — на корпоративні середовища, де важливими є високий рівень безпеки та розширені можливості управління правами доступу. Надійні механізми ідентифікації та захисту інформації дозволяють зменшити ризики, пов'язані з несанкціонованим використанням даних [2], [3].

Гнучкість таких сервісів забезпечується сумісністю з різними платформами та інтеграцією з іншими інформаційними системами. Завдяки цьому обмін файлами та їхнє редагування можливі як з настільних комп'ютерів, так і з мобільних пристроїв, що дає змогу оперативно виконувати робочі завдання [2], [3].

Окрім зручності використання, такі програмні рішення мають враховувати ефективність та економічну доцільність. При виборі оптимального варіанту важливим є баланс між продуктивністю, витратами та можливостями розширення функціоналу в майбутньому. Деякі сервіси надають розширені можливості адміністрування, що дозволяє керувати процесами обміну інформацією та контролювати її збереження [2], [3].

Сучасні платформи для віддаленої роботи підтримують різноманітні формати файлів і способи їхньої передачі, дозволяючи уникнути обмежень традиційних методів, таких як використання електронної пошти або

локальних серверів. Це сприяє підвищенню продуктивності та забезпечує безперебійну взаємодію між учасниками робочого процесу [2], [3].

У сучасних умовах цифрового середовища можливість працювати з віддаленими даними відіграє важливу роль для забезпечення ефективної взаємодії між користувачами. Використання хмарних рішень дозволяє зберігати, передавати та редагувати інформацію без прив'язки до фізичних носіїв, що суттєво спрощує організацію роботи та комунікацію між різними учасниками процесу [2], [3].

Інформаційні системи, які базуються на хмарних технологіях, дають змогу зберігати цифровий контент у віддалених центрах обробки даних. Завдяки цьому користувачі отримують можливість доступу до необхідних матеріалів з будь-якого пристрою, підключеного до мережі Інтернет. Такі рішення значно підвищують мобільність і гнучкість у виконанні завдань, а також спрощують співпрацю над спільними проєктами [2], [3].

Передавання інформації через хмарні сервіси відбувається у кілька етапів. Файли спочатку завантажуються у сховище через вебінтерфейс або спеціальні програми. Далі вони шифруються та розміщуються на серверах, що гарантує їхню безпеку. Для доступу до цих файлів створюються унікальні посилання, якими можна ділитися з іншими користувачами. В залежності від налаштувань, такі посилання можуть мати певні обмеження, наприклад, за часом дії або за кількістю дозволених завантажень [2], [3].

Функціональність хмарних сервісів спрямована на забезпечення надійного зберігання та захисту інформації. Використання шифрування, багаторівневої ідентифікації та інших методів підвищує рівень безпеки даних. Крім того, такі платформи підтримують можливості редагування файлів у реальному часі, що полегшує спільну роботу [2], [3].

Гнучкість у налаштуванні параметрів доступу дає змогу контролювати, хто саме і в якому обсязі може взаємодіяти з інформацією. Це важливо як для персонального використання, так і для організацій, які працюють з великою кількістю конфіденційних даних. Додаткові функції, такі як інтеграція з

іншими сервісами або автоматизація процесів, розширюють можливості застосування хмарних платформ [2], [3].

Використання подібних технологій робить робочі процеси більш зручними та ефективними, дозволяючи зменшити залежність від локальних сховищ і фізичних носіїв інформації. Завдяки цьому можна швидко адаптуватися до змін та забезпечувати безперервний доступ до необхідних даних незалежно від місця перебування користувачів [2], [3].

У сучасному цифровому просторі хмарні сервіси для обміну файлами стали незамінним інструментом для організації роботи з даними. Вони забезпечують можливість доступу до інформації з будь-якої точки світу без прив'язки до конкретного пристрою, що значно покращує ефективність віддаленої роботи та колективної взаємодії. Передача файлів відбувається миттєво, що мінімізує затримки у спілкуванні й спрощує обмін інформацією [3], [4].

Завдяки використанню хмарних технологій значно скорочуються витрати на фізичні носії та технологічну інфраструктуру. Рівень безпеки, який пропонують такі платформи, значно перевищує традиційні методи, оскільки передбачає шифрування даних під час передавання та зберігання. Це робить хмарний обмін файлами надійною альтернативою класичним способам передачі даних, таким як вкладення в електронній пошті чи використання зовнішніх носіїв [3], [4].

Ефективна командна робота стає можливою завдяки функціоналу спільного редагування та керування доступом. Використання таких платформ дозволяє уникнути дублювання файлів і забезпечує актуальність змін у реальному часі. При виборі хмарного сервісу важливо враховувати, наскільки зручний і функціональний його інтерфейс, а також які можливості пропонуються для захисту даних та організації колективної роботи [3], [4].

Різні сервіси можуть мати свої особливості. Деякі платформи пропонують спрощений підхід до керування файлами, що допомагає уникнути хаосу та дублювання інформації. Доступні також розширені інструменти для

контролю доступу, такі як тимчасові посилання, обмеження на завантаження або додаткові заходи ідентифікації. Високий рівень захисту забезпечується комплексними механізмами шифрування, що гарантують безпечне зберігання та передавання даних [3], [4].

Хмарні інформаційні технології докорінно змінюють підхід до організації цифрового робочого процесу. Гнучкість, надійність та швидкість роботи з файлами роблять їх незамінним інструментом для бізнесу та особистого використання. Завдяки таким рішенням процес управління даними стає простішим, безпечнішим і доступнішим, що дозволяє зосередитися на головних завданнях, а не на технічних аспектах зберігання й передавання інформації [3], [4].

Сучасні програмні рішення для керування файлами надають можливість зручного доступу до інформації з будь-якої точки світу, а також дозволяють швидко передавати дані незалежно від їхнього формату. Використовуючи такі інструменти, можна не лише пересилати документи, зображення чи відео файли, а й контролювати, хто саме має можливість отримати до них доступ [3], [4].

Для ефективної віддаленої взаємодії необхідні інструменти, які спрощують спільну роботу та дають змогу швидко обмінюватися файлами, що суттєво впливає на продуктивність у сучасних умовах праці. Зручність таких систем полягає у впорядкуванні цифрового контенту та спрощеній навігації, що дозволяє оперативно знаходити необхідні дані. Використання правильно підібраної платформи дає змогу уникати дублювання, підтримувати актуальність інформації та забезпечувати зручне зберігання. При цьому можливість обмежувати доступ до файлів дозволяє уникнути несанкціонованого використання інформації [4], [5].

Інструменти для безпечного зберігання й передачі даних усувають потребу в численних електронних листах і затримках, пов'язаних із надсиланням вкладень. Декілька простих дій достатньо, щоб завантажити необхідні файли та поділитися ними з іншими користувачами. Крім того, такі

платформи забезпечують функціонал редагування, маркування та коментування, що особливо корисно для командної роботи над складними проєктами [4], [5].

Завдяки таким програмним рішенням можна безперешкодно працювати з файлами великого розміру, уникаючи обмежень електронної пошти. Багато робочих процесів потребують швидкого обміну великими обсягами даних, і можливість робити це без перевантаження поштових сервісів значно спрощує взаємодію [4], [5].

Захист інформації є ще одним важливим аспектом цифрових рішень для керування файлами. Використання розширених налаштувань дозволяє контролювати доступ до чутливих даних і забезпечувати їхню безпеку без ризику порушення конфіденційності [4], [5].

У сучасних умовах, коли дистанційна робота стає дедалі поширенішою, зручні системи для керування документами допомагають підтримувати продуктивність і зменшують необхідність фізичної присутності в офісі. Це сприяє гнучкості робочого процесу, що позитивно впливає як на ефективність працівників, так і на зручність виконання завдань [4], [5].

У контексті віддаленої роботи особливої уваги потребує питання безпечного обміну файлами. Оскільки фізичне поширення документів на всю організацію є неможливим, застосовуються цифрові технології. Однак широкий вибір доступних рішень може ускладнити процес вибору оптимального варіанту [4], [5]. Ефективна передача файлів дозволяє членам команди залишатися в курсі актуальної інформації, координувати свої дії та підтримувати відповідальність за виконані завдання, незалежно від місця перебування. Крім того, надійні методи обміну даними сприяють зміцненню довіри між колегами [4], [5].

Існує кілька ключових підходів до організації віддаленого доступу до файлів. Використання хмарних сховищ забезпечує можливість зберігання документів, мультимедійних файлів та інших ресурсів із подальшим доступом до них з будь-якого пристрою. Такі сервіси, як Google Drive чи OneDrive,

надають додаткові функції резервного копіювання та збереження історії змін. Для координації роботи часто застосовуються спеціалізовані платформи для спільної діяльності. Вони дають змогу організовувати комунікацію, керувати проєктами та здійснювати обмін файлами без необхідності використання сторонніх рішень [4], [5].

Електронна пошта також залишається одним із найзручніших способів передавання даних. Додавання вкладень до повідомлень дозволяє швидко поширювати документи, хоча слід враховувати обмеження щодо розміру файлів. Для забезпечення конфіденційності інформації під час передавання використовуються спеціальні сервіси із захищеним каналом зв'язку. Вибір таких платформ особливо актуальний при роботі з чутливими даними, адже традиційні методи не завжди гарантують належний рівень безпеки [4], [5].

Ще одним варіантом є спільні мережеві диски, які дозволяють зберігати дані у загальному доступі. Такі рішення можуть бути як фізичними, так і віртуальними, а їх перевага полягає у можливості одночасного доступу для кількох користувачів. Для налаштування такого сховища достатньо визначитися з форматом використання, надати відповідні права доступу та організувати структуру збережених матеріалів [4], [5].

Для ефективного обміну файлами під час віддаленої роботи важливо мати належну організацію та дотримуватись певних стандартів безпеки. Хоча технології є необхідним інструментом для забезпечення безперебійної роботи, вони не можуть забезпечити належний рівень безпеки без правильного застосування. Правильне управління обліковими записами та доступами до файлів дозволяє оптимізувати процес обміну даними [5], [6].

Не слід використовувати особисті електронні адреси для передачі великих файлів, оскільки це може створити проблеми через обмеження розміру файлів і недостатній рівень захисту. Корпоративні рішення для обміну файлами надають більш високий рівень шифрування і захисту даних, що значно підвищує безпеку порівняно зі споживчими сервісами [5], [6].

Управління доступом має здійснюватися за принципом мінімальних привілеїв. Це означає, що доступ до файлів має бути обмеженим лише до тих, які необхідні користувачеві. Такий підхід дозволяє забезпечити максимальний захист інформації, особливо коли це стосується загальних ресурсів чи спільних папок [5], [6].

Не менш важливо, щоб усі члени команди були ознайомлені з основами безпеки, включаючи виявлення шкідливих атак та створення надійних паролів. Навчання або надання корисних ресурсів для самостійного ознайомлення допомагає знизити ризики [5], [6].

Організація файлів та папок також є важливою частиною безпечного обміну даними. У разі віддаленої роботи систематизація документів у хмарних сховищах не лише спрощує доступ до них, а й дозволяє уникнути безладу. Це забезпечує швидкий пошук необхідних матеріалів та підтримує порядок. Вибір зручної і доступної платформи для обміну файлами також має важливе значення. Сервіси, які легко інтегруються в робочі процеси, допомагають заощадити час і зменшити кількість помилок. Користувачам стає зручніше працювати з великими файлами, а сам процес обміну стає швидким і ефективним [5], [6].

У результаті впровадження цих стандартів можна забезпечити безпечний обмін файлами для команди під час віддаленої роботи, що сприяє підтримці високої продуктивності та зменшує ризики кіберзагроз. Користування такими інструментами, як Google Drive чи Dropbox, робить передачу документів і мультимедійних файлів більш надійною, при цьому підтримуючи високий рівень захисту [5], [6].

Розширення можливостей віддаленої роботи суттєво вплинуло на організацію трудового процесу, що особливо стало помітним у період глобальних змін. Якщо раніше дистанційний формат використовувався переважно у певних сферах або для окремих посад, то за нових умов він став невід'ємною частиною діяльності багатьох компаній. Виникла необхідність оперативної адаптації до нових реалій, що спричинило масове впровадження

технологій для віддаленої взаємодії, збереження даних і спільної роботи. З переходом до віддаленої роботи підвищилася потреба у надійних інструментах для зберігання та передачі файлів. Для цього активно використовуються хмарні сервіси, які дозволяють зручно організовувати документи, відкривати до них доступ для колективного редагування та забезпечувати захист даних. Завдяки можливостям спільного використання ресурсів та історії змін, такі системи значно полегшують взаємодію в межах команди [6], [7].

Не менш важливим є використання платформ для комунікації, які інтегрують у собі функції обміну файлами. Завдяки цьому документи та інші необхідні матеріали можуть передаватися безпосередньо в робочих чатах, що прискорює процес спільної роботи. Такі рішення дають змогу не лише обмінюватися текстовими повідомленнями, а й проводити обговорення проєктів у режимі реального часу [6], [7].

Для ефективного управління процесами також застосовуються інструменти керування проєктами, які поєднують організацію завдань із можливістю централізованого зберігання важливих файлів. Це сприяє впорядкованій роботі та забезпечує швидкий доступ до необхідних ресурсів для всіх учасників команди [6], [7].

Передача файлів відіграє ключову роль у забезпеченні злагодженої взаємодії між учасниками робочого процесу, особливо в умовах віддаленої діяльності. Завдяки можливості спільного редагування документів у режимі реального часу значно спрощується координація завдань, що позитивно впливає на загальну ефективність роботи. Такий підхід дозволяє оперативно обмінюватися ідеями, переглядати матеріали, залишати коментарі та спільно працювати над проєктами незалежно від фізичного розташування учасників. Це усуває необхідність багаторівневого листування та сприяє більш швидкому прийняттю рішень [6], [7].

Раціональна організація обміну файлами допомагає створити прозору систему роботи, де кожен учасник має доступ до актуальної інформації та розуміє загальну картину процесу. Чітке структурування даних дозволяє

уникнути плутанини, підвищує рівень взаєморозуміння та сприяє ефективній координації командної діяльності. Вибір відповідного інструменту для спільного використання матеріалів залежить від особливостей робочого середовища та потреб команди [6], [7].

Забезпечення зручного доступу до файлів значно покращує взаємодію між співробітниками. Використання сучасних рішень для обміну даними дає можливість працювати над документами одночасно, оперативно вносити правки, надавати коментарі та обговорювати ідеї без затримок. Це сприяє відкритості процесів, робить комунікацію прозорішою та дозволяє всім учасникам бути обізнаними щодо поточного стану справ [6], [7].

Засоби для відео зв'язку дозволяють не лише проводити зустрічі, а й обмінюватися файлами під час спільної роботи. Функції демонстрації екрана та передача документів безпосередньо під час обговорення сприяють згадженій комунікації навіть за фізичної віддаленості між співробітниками. Коли виникає необхідність передати великі файли, використовуються спеціалізовані сервіси, що гарантують швидку та безпечну передачу даних. Завдяки цьому забезпечується збереження цілісності файлів та захист інформації під час її переміщення [6], [7].

Розвиток технологій зробив можливим ефективну організацію дистанційної роботи, надаючи широкий спектр інструментів для збереження, обміну та управління інформацією. Це сприяє підвищенню продуктивності та підтримці безперебійної співпраці незалежно від фізичного розташування членів команди [6], [7].

Визначено, що інформаційні системи для роботи з віддаленими даними забезпечують безпечний доступ, збереження, обробку та передачу інформації незалежно від місця розташування користувача. Такі інформаційні системи використовуються в корпоративному середовищі, наукових дослідженнях, дистанційному навчанні та інших сферах, де необхідний оперативний доступ до даних через мережу.

1.2 Аналіз інформаційних систем для роботи з віддаленими даними

Проаналізуємо інформаційні системи для роботи з віддаленими даними [8]-[14].

Програмне забезпечення WinSCP (рис. 1.1) є ефективним інструментом для керування віддаленими даними, що працює на операційній системі Windows [8], [12].

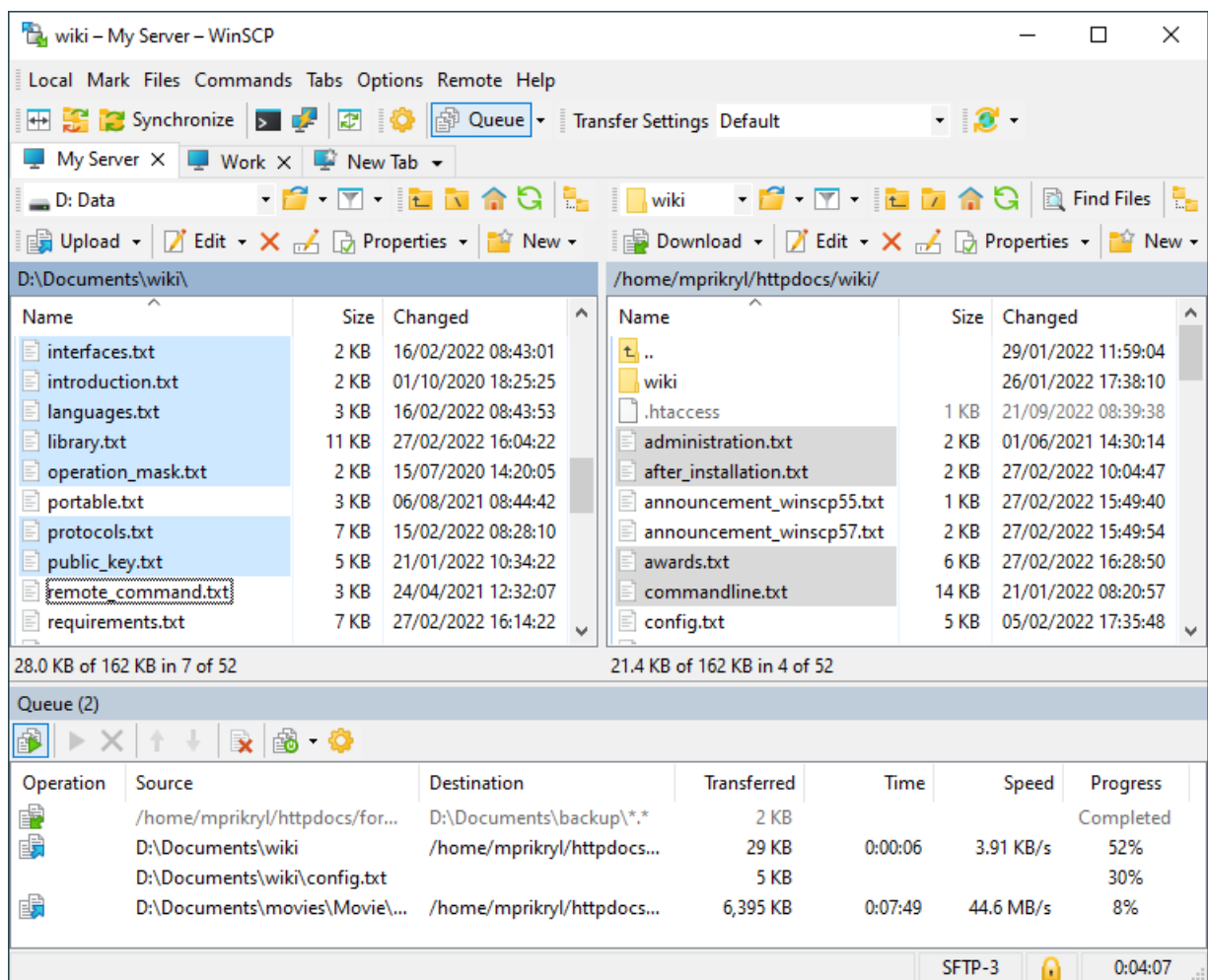


Рисунок 1.1 – Програмне забезпечення WinSCP [12]

Програмне забезпечення WinSCP забезпечує можливість підключення до віддалених серверів і здійснення операцій із файлами, використовуючи різні протоколи, такі як FTP, SFTP, SCP і WebDAV. Завдяки цьому WinSCP

дозволяє не лише передавати файли між локальним комп'ютером і сервером, а й виконувати їх синхронізацію, забезпечуючи швидкий і безпечний обмін даними [8], [12].

Графічний інтерфейс, реалізований у програмі, спрямований на максимальну зручність користувача. Використовуючи метод перетягування файлів, можна легко керувати віддаленими даними, здійснюючи копіювання, переміщення, перейменування чи видалення. Для користувачів, які потребують автоматизації процесів, передбачено можливість створення сценаріїв, що дозволяє оптимізувати роботу з файлами та виконувати повторювані завдання без необхідності ручного втручання [8], [12].

WinSCP підтримує захищені з'єднання через HTTPS, що гарантує безпечний обмін даними між локальним і віддаленим середовищем. Крім WebDAV, програма сумісна з іншими популярними протоколами, що забезпечує гнучкість у використанні та адаптацію до різних завдань. Інтеграція з Windows, а також сумісність із такими програмами, як PuTTY, робить цей інструмент зручним для адміністрування серверів і керування даними в мережі [8], [12].

Завдяки підтримці текстового та двійкового режимів передачі, можливості роботи у фоновому режимі та організації черги передач, WinSCP дозволяє ефективно оперувати великими обсягами інформації. Для зручності користувачів передбачено вбудований редактор, а також підтримку зовнішніх текстових редакторів, що спрощує редагування файлів безпосередньо в середовищі програми [8], [12].

Програмне забезпечення WinSCP адаптоване для використання з портативних носіїв, що дозволяє працювати з віддаленими даними без потреби встановлення програми на комп'ютер. Завдяки багаторазовій локалізації WinSCP є доступним для широкої аудиторії користувачів. Велика спільнота та наявність детальної документації дозволяють швидко знаходити відповіді на питання та використовувати весь потенціал програми для роботи з віддаленими серверами та сховищами даних [8], [12].

Основними характеристиками програмного забезпечення WinSCP є такі [8], [12]:

- підтримка віддалених протоколів: програмне забезпечення WinSCP підтримує з'єднання з віддаленими серверами через SFTP, SCP, FTP, FTPS і WebDAV, що дозволяє безпечно передавати файли [8], [12];

- графічний інтерфейс: програмне забезпечення WinSCP має зручне вікно з двопанельним відображенням файлів, що спрощує керування віддаленими даними [8], [12];

- автоматизація: програмне забезпечення WinSCP підтримує створення сценаріїв для виконання рутинних операцій з файлами, що підвищує ефективність роботи [8], [12];

- інтеграція з Windows: програмне забезпечення WinSCP дозволяє запускати команди безпосередньо з провідника Windows і підтримує перетягування файлів [8], [12];

- безпека: програмне забезпечення WinSCP використовує шифрування для передачі даних, а також підтримує двофакторну автентифікацію для підключень [8], [12];

- синхронізація файлів: програмне забезпечення WinSCP дозволяє автоматично оновлювати дані між локальним комп'ютером і віддаленим сервером [8], [12];

- портативний режим: програмне забезпечення WinSCP може працювати без встановлення, зберігаючи конфігурацію у файлі, що зручно для використання з флеш-носіїв [8], [12];

- підтримка зовнішніх редакторів: програмне забезпечення WinSCP дозволяє відкривати і змінювати файли безпосередньо на віддаленому сервері через сторонні редактори [8], [12];

- інтеграція з іншими програмами: програмне забезпечення WinSCP може працювати разом із PuTTY та іншими SSH-клієнтами, що розширює можливості управління серверами [8], [12].

Серед недоліків програмного забезпечення WinSCP можна відзначити обмежену кросплатформеність, оскільки програма розроблена виключно для Windows і не має офіційних версій для інших операційних систем. Крім того, інтерфейс, хоч і зручний для досвідчених користувачів, може здатися складним для тих, хто вперше працює з передачею віддалених даних. У порівнянні з деякими альтернативами, такими як FileZilla, можливості керування з'єднаннями та інтеграція з іншими програмами можуть бути менш гнучкими. Додатково, для використання розширених функцій автоматизації, таких як створення сценаріїв, потрібні певні технічні навички, що може ускладнити роботу для початківців [8], [12].

Програмне забезпечення BitKineх (рис. 1.2) представляє собою спеціалізоване програмне забезпечення, розроблене для роботи з віддаленими даними та забезпечення швидкої передачі файлів між серверами [8], [13].

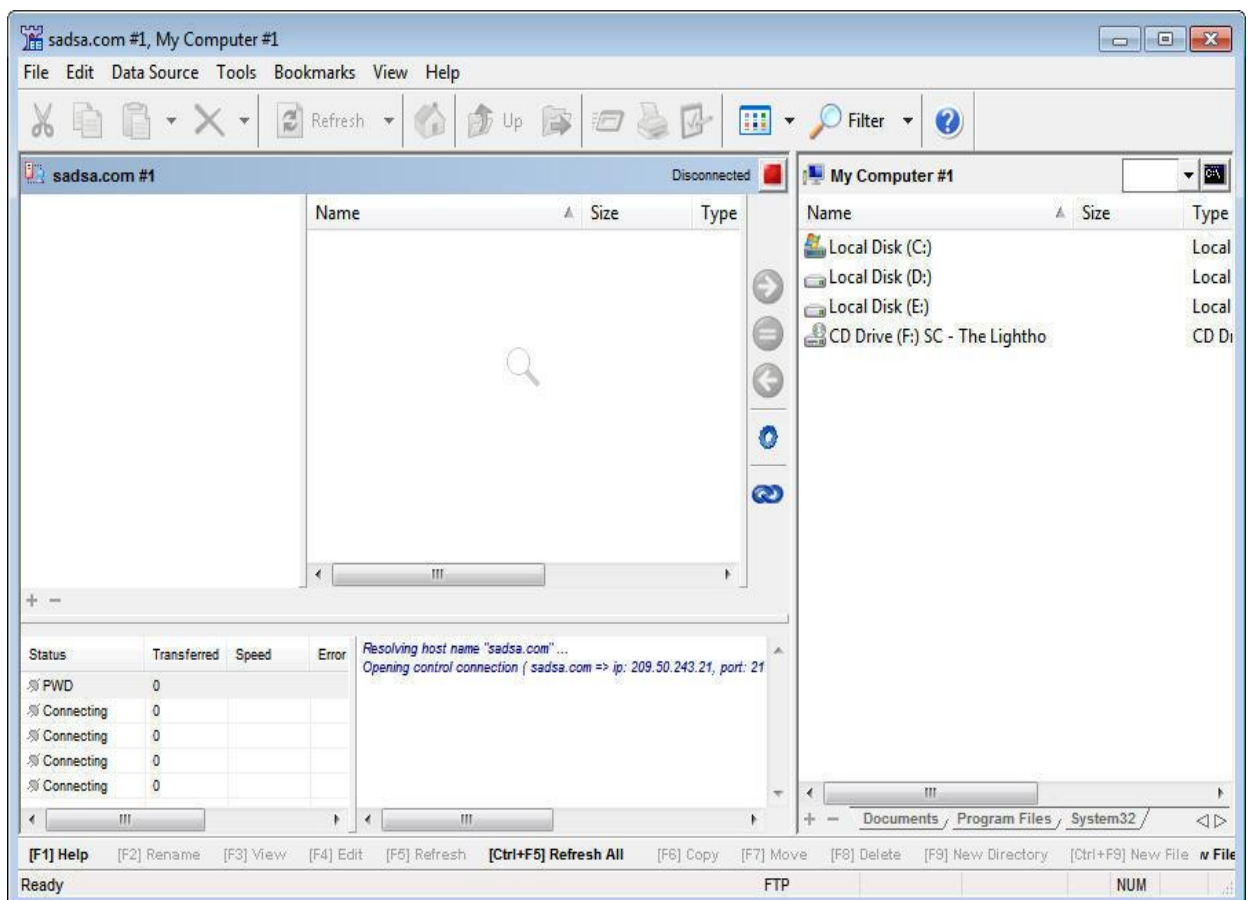


Рисунок 1.2 – Програмне забезпечення BitKineх [13]

Програмне забезпечення BitKinex дозволяє підключатися до декількох серверів одночасно, що особливо корисно для адміністраторів мереж, які мають потребу в ефективному управлінні різними віддаленими ресурсами. Завдяки підтримці кількох мережових протоколів, таких як SFTP, HTTP, WebDAV і FXP, забезпечується безперебійний процес обміну файлами, незалежно від типу підключення [8], [13].

Головною перевагою програми BitKinex є її здатність працювати з кількома підключеннями одночасно, що значно прискорює обмін даними між серверами. Це дозволяє здійснювати передачу файлів великого обсягу без значних затримок, підвищуючи загальну продуктивність. Крім того, програмне забезпечення підтримує гнучке налаштування параметрів підключень, що дає можливість оптимізувати робочий процес відповідно до конкретних потреб користувачів [8], [13].

Для забезпечення захисту переданих даних у BitKinex реалізовано систему автентифікації, яка передбачає створення облікового запису під час першого запуску програми. Кожне підключення до віддаленого сервера здійснюється з використанням збережених облікових даних, що підвищує безпеку доступу до конфіденційної інформації. Додатково передбачена можливість встановлення паролів для окремих файлів, що забезпечує контроль доступу до важливих даних [8], [13].

Інтерфейс програми BitKinex спроектовано таким чином, щоб забезпечити зручність роботи навіть для користувачів, які не мають значного досвіду у сфері адміністрування віддалених серверів. Завдяки інтуїтивному дизайну та продуманій навігації основні операції, пов'язані з передачею файлів та управлінням з'єднаннями, виконуються швидко та без зайвих ускладнень. Унікальна особливість BitKinex полягає в можливості паралельного керування запитами та передачі даних між серверами незалежно від типу використовуваного протоколу [8], [13].

Попри значну кількість переваг, слід враховувати, що процес встановлення BitKinex може зайняти більше часу, ніж у схожих програм.

Водночас, після налаштування всієї необхідної конфігурації, програма забезпечує високу ефективність та стабільність роботи, що робить її надійним інструментом для керування віддаленими даними [8], [13].

Програмне забезпечення BitKinex має такі особливості [8], [13]:

- підтримка віддалених серверів: програмне забезпечення BitKinex підтримує можливість підключення до кількох серверів одночасно, що спрощує роботу користувачів з віддаленими даними та адміністрування мережевих ресурсів [8], [13];

- сумісність з протоколами: програмне забезпечення BitKinex підтримує широкий спектр мережевих протоколів, зокрема SFTP, HTTP, WebDAV, FXP, що дозволяє працювати з різними типами серверів [8], [13];

- паралельне підключення: дозволяє одночасно здійснювати передачу файлів через кілька каналів, що підвищує швидкість обміну даними [8], [13];

- інтегроване керування запитами: програмне забезпечення BitKinex об'єднує передачу файлів та інші серверні запити в єдиному інтерфейсі, що робить робочий процес ефективнішим [8], [13];

- підвищена безпека: програмне забезпечення BitKinex передбачає можливість захисту окремих файлів паролем, що забезпечує додатковий рівень безпеки під час роботи з конфіденційними даними [8], [13];

- створення облікового запису: під час встановлення програмного забезпечення BitKinex потрібно зареєструвати обліковий запис, що використовується для всіх майбутніх підключень до серверів [8], [13];

- гнучке налаштування параметрів: програмне забезпечення BitKinex містить широкий набір налаштувань для керування процесами передачі файлів, що дозволяє адаптувати програму до потреб користувача [8], [13];

- простий інтерфейс: програмне забезпечення BitKinex надає зручний та інтуїтивно зрозумілий інтерфейс, який полегшує використання програми навіть для новачків [8], [13];

– тривалий процес встановлення: інсталяція програмного забезпечення BitKineх займає більше часу порівняно з іншими подібними клієнтами, що може бути незручним для користувачів [8], [13].

Серед недоліків програмного забезпечення BitKineх можна відзначити складний та тривалий процес встановлення, що може спричинити певні труднощі для користувачів, які вперше працюють із цією програмою. Крім того, інтерфейс, хоча й функціональний, може здатися застарілим у порівнянні з сучасними аналогами, що впливає на зручність використання. Деякі користувачі також відзначають нестабільність підключень у певних конфігураціях мережі, що може призводити до розривів з'єднання під час передачі файлів. Обмежені можливості налаштувань у порівнянні з іншими FTP-клієнтами також можуть бути мінусом для досвідчених адміністраторів, яким потрібна гнучкість у конфігурації параметрів підключення [8], [13].

Програмне забезпечення Air Explorer (рис. 1.3) призначено для ефективної роботи з віддаленими даними, надаючи можливість керувати файлами, розташованими у хмарних сховищах [8], [14].

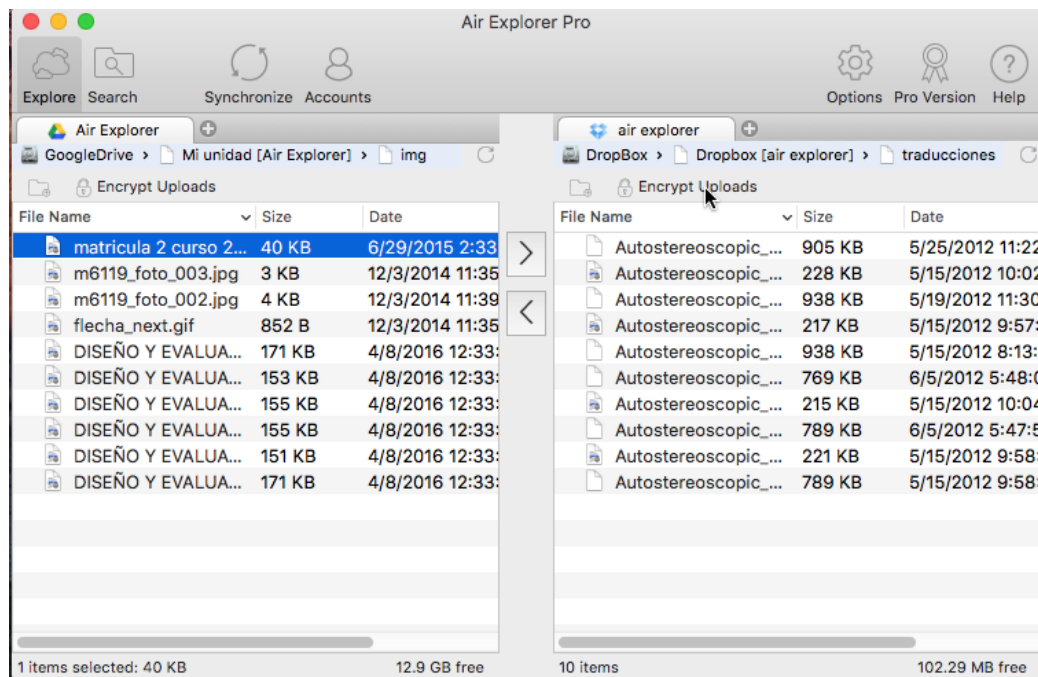


Рисунок 1.3 – Програмне забезпечення Air Explorer [14]

Програмне забезпечення Air Explorer сумісне з різними операційними системами та забезпечує централізований доступ до даних у різних хмарних сервісах [8], [14].

Основна функціональність програми Air Explorer включає підтримку підключень до різних хмарних платформ, а також можливість працювати з обліковими записами, що використовують FTP, FTPS та SFTP. Завдяки цьому користувач отримує гнучкий інструмент для передачі, синхронізації та адміністрування файлів у віддаленому середовищі [8], [14].

Air Explorer дозволяє користувачам зручно працювати з різними хмарними платформами. Це забезпечує зручний доступ до файлів на кількох хмарних сервісах без необхідності постійно перемикатися між різними програмами чи браузерами. Програма має зручний та інтуїтивно зрозумілий інтерфейс, який дозволяє швидко працювати з хмарними сервісами та файлами. Це дає змогу навіть новачкам без проблем виконувати основні операції з файлами, такі як копіювання, переміщення або видалення [8], [14].

Окрім стандартних можливостей, програмне забезпечення Air Explorer дозволяє легко інтегрувати облікові записи WebDAV, що розширює можливості доступу до хмарних даних. Для цього достатньо вибрати відповідну опцію в меню облікових записів і вказати необхідні дані для підключення [8], [14].

Програмне забезпечення Air Explorer має такі особливості [8], [14]:

- підтримка хмарних сервісів: програмне забезпечення Air Explorer підтримує роботу з популярними хмарними платформами, такими як Google Drive, OneDrive, Dropbox та іншими [8], [14];

- керування кількома обліковими записами: програмне забезпечення Air Explorer дозволяє підключати й використовувати кілька акаунтів одного сервісу одночасно [8], [14];

- робота з FTP і WebDAV: програмне забезпечення Air Explorer підтримує підключення через FTP, FTPS, SFTP та WebDAV для доступу до віддалених серверів [8], [14];

- синхронізація файлів: програмне забезпечення Air Explorer забезпечує двосторонню синхронізацію між локальними даними та хмарним сховищем [8], [14];
- можливість шифрування: програмне забезпечення Air Explorer надає інструменти для шифрування файлів перед їх передачею в хмару для підвищення безпеки [8], [14];
- функція пошуку: програмне забезпечення Air Explorer дозволяє користувачам здійснювати пошук файлів безпосередньо у підключених хмарних сховищах [8], [14];
- передача між хмарними сервісами: програмне забезпечення Air Explorer підтримує копіювання та переміщення файлів між різними хмарними платформами без необхідності завантаження на локальний пристрій [8], [14];
- простий інтерфейс: програмне забезпечення Air Explorer забезпечує зручне користування завдяки інтуїтивному дизайну та функціоналу [8], [14];
- інтеграція з операційними системами: програмне забезпечення Air Explorer доступне для операційних систем Windows і macOS, що робить програму гнучкою у використанні [8], [14].

Серед недоліків програмного забезпечення Air Explorer можна відзначити обмежені можливості безкоштовної версії, яка має обмеження на кількість одночасних підключень до хмарних сервісів. Також спостерігається залежність швидкості передачі даних від характеристик конкретного хмарного провайдера, що може впливати на продуктивність роботи. Деякі користувачі зазначають, що інтерфейс програми потребує вдосконалення для підвищення зручності використання, особливо під час виконання складних операцій із файлами. Окрім цього, можливі періодичні проблеми з автентифікацією та синхронізацією облікових записів, що може ускладнювати роботу з деякими хмарними сервісами [8], [14].

Порівняльну характеристику інформаційних систем для роботи з віддаленими даними наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння інформаційних систем для роботи з віддаленими даними

Критерій порівняння	WinSCP [12]	BitKinex [13]	Air Explorer [14]
Інтерфейс користувача	+—	+	+
Підтримка хмарних облікових записів	—	—	+
Простота налаштувань	+	+—	—
Підтримка FTP та FTPS	+	+	+—
Підтримка WebDAV	—	—	+
Синхронізація файлів	+—	+—	+

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато інформаційних систем для роботи з віддаленими даними. Попри всі переваги програмного забезпечення для роботи з віддаленими даними, такі рішення мають і певні недоліки. Суттєвим недоліком таких інформаційних систем є їх вартість. Базові версії багатьох сервісів для роботи з віддаленими даними можуть бути безкоштовними, але при потребі у додатковому обсязі пам'яті, розширених функціях чи підвищеній безпеці користувачам доводиться оплачувати підписку. Для великих компаній витрати на хмарні рішення можуть виявитися значними. Продуктивність таких сервісів також може стати проблемою. При роботі з великими файлами або великою кількістю запитів швидкість завантаження та синхронізації може знижуватися, що негативно впливає на ефективність роботи інформаційних систем обробки та зберігання віддалених даних. Водночас існує проблема монополізації ринку, адже великі компанії, такі як Google, Microsoft і Dropbox, домінують у цій сфері. Це створює попит на альтернативні рішення, зокрема для корпоративних клієнтів. Тому актуальною

є розробка програмного забезпечення інформаційної системи для роботи з віддаленими даними.

При розробці програмного забезпечення інформаційної системи для роботи з віддаленими даними необхідно забезпечити такі функціональні вимоги:

- підтримка можливості роботи з віддаленим сховищем даних;
- підтримка можливості створення нового каталогу у віддаленому сховищі даних;
- можливість завантаження файлів до віддаленого сховища;
- можливість перейменування документів, що зберігаються у віддаленому сховищі;
- підтримка можливості видалення елементів зі сховища;
- підтримка можливості роботи з кошиком видалених документів;
- можливість пошуку файлів та папок за їх назвами.

1.3 Висновки за розділом 1

Визначено, що інформаційні системи для роботи з віддаленими даними забезпечують безпечний доступ, збереження, обробку та передачу інформації незалежно від місця розташування користувача. Такі інформаційні системи використовуються в корпоративному середовищі, наукових дослідженнях, дистанційному навчанні та інших сферах, де необхідний оперативний доступ до даних через мережу.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато інформаційних систем для роботи з віддаленими даними. Проте, попри всі переваги програмного забезпечення для роботи з віддаленими даними, такі рішення мають і певні недоліки. Суттєвим недоліком таких інформаційних систем є їх вартість. Базові версії багатьох сервісів для роботи з віддаленими даними можуть бути безкоштовними, але при потребі у додатковому обсязі пам'яті, розширених функціях чи

підвищеній безпеці користувачам доводиться оплачувати підписку. Для великих компаній витрати на хмарні рішення можуть виявитися значними. Продуктивність таких сервісів також може стати проблемою. При роботі з великими файлами або великою кількістю запитів швидкість завантаження та синхронізації може знижуватися, що негативно впливає на ефективність роботи інформаційних систем обробки та зберігання віддалених даних. Водночас існує проблема монополізації ринку, адже великі компанії, такі як Google, Microsoft і Dropbox, домінують у цій сфері. Це створює попит на альтернативні рішення, зокрема для корпоративних клієнтів. Тому актуальною є розробка програмного забезпечення інформаційної системи для роботи з віддаленими даними.

Сформульовано функціональні вимоги до програмного забезпечення інформаційної системи для роботи з віддаленими даними.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення інформаційної системи для роботи з віддаленими даними було використано мову програмування C# [15], [16].

C# являє собою сучасну об'єктно-орієнтовану мову програмування, що була розроблена компанією Microsoft як частина платформи .NET. Вона поєднує в собі найкращі властивості різних мов програмування, таких як C++ і Java, що дозволяє створювати програмне забезпечення для широкого спектру завдань [15], [16].

Однією з основних характеристик мови програмування C# є підтримка об'єктно-орієнтованого програмування, що дозволяє ефективно організовувати код, використовуючи об'єкти для зручного управління різними елементами програми. Завдяки автоматичному керуванню пам'яттю через збирач сміття, C# допомагає уникати витоків пам'яті та підвищує безпеку програм [15], [16].

Основна перевага цієї мови полягає в тому, що вона працює в екосистемі .NET, що забезпечує можливість розробки різноманітних програм, таких як веб-додатки через ASP.NET Core, настільні програми з використанням WPF або WinForms, а також багатоплатформні рішення за допомогою Xamarin. Крім того, C# активно використовується для розробки ігор в Unity [15], [16].

Завдяки вбудованій підтримці асинхронного програмування за допомогою деяких конструкцій, мова C# дозволяє ефективно працювати з віддаленими даними, що особливо корисно при взаємодії з сервером. Після випуску .NET Core, мова програмування C# стала більш доступною для розробки програм, що працюють на різних операційних системах, таких як Windows, Linux і macOS [15], [16].

C# широко використовується для роботи з такими фреймворками, як Entity Framework, Dapper і ADO.NET, і є одним із найбільш популярних

виборів у сфері веброзробки, розробки корпоративних систем, хмарних служб і мобільних додатків [15], [16].

Мова C# продовжує активно розвиватися завдяки постійній підтримці з боку Microsoft і великої спільноти розробників. Це робить C# надійним вибором для створення програмного забезпечення, яке має бути продуктивним, безпечним і масштабованим [15], [16].

C# є ідеальним варіантом для розробки програмного забезпечення з багатьох причин. Він надає можливість створювати ефективні вебзастосунки та серверні рішення за допомогою .NET Framework і .NET Core, що робить його особливо корисним для хмарних сервісів і API. Завдяки сумісності з популярними хмарними платформами, такими як Azure, C# є зручним для розробки рішень з керування віддаленими файлами та даними [15], [16].

Завдяки вбудованій підтримці асинхронного програмування, мова дозволяє зручно працювати з віддаленими запитами до сервера, автоматизуючи керування потоками і дозволяючи зберігати високу продуктивність. C# має великий набір бібліотек, таких як HttpClient, System.IO, Azure.Storage.Blobs, що полегшують взаємодію з файловими системами та мережевими інтерфейсами [15], [16].

Мова також добре задокументована, що знижує ймовірність помилок у коді та підвищує його надійність, що є важливим аспектом для програм, які працюють з конфіденційними даними. Крім того, C# підтримує як розробку настільних, так і вебзастосунків, що дозволяє створювати міжплатформні програми з використанням WPF або Blazor, а для серверних рішень застосовується ASP.NET [15], [16].

У підсумку, мова програмування C# є надзвичайно потужним інструментом для розробки програмного забезпечення, завдяки своїй універсальності, підтримці хмарних технологій, асинхронному програмуванню та широким можливостям інтеграції з вебсервісами та іншими технологіями [15], [16].

Обґрунтування вибору мови програмування для розробки програмного забезпечення інформаційної системи для роботи з віддаленими даними наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення інформаційної системи для роботи з віддаленими даними

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Продуктивність	+	+—	+
Асинхронне програмування	+	+—	+—
Підтримка хмарних сервісів	+	+	+—
Зручність для створення ПЗ інформаційної системи для роботи з віддаленими даними	+	—	—
Масштабованість та ефективність при великій кількості запитів	+	+—	+

Отже, для реалізації програмного забезпечення інформаційної системи для роботи з віддаленими даними обрано мову програмування C#, яка має широкий вибір бібліотек для роботи з файловими системами та мережевими протоколами. Крім того, C# є мовою з суворою типізацією, що зменшує кількість потенційних помилок у коді та підвищує його безпеку. Таким чином, C# є відмінним вибором для розробки програмного забезпечення для роботи з віддаленими даними завдяки своїй продуктивності, гнучкості, підтримці хмарних технологій, асинхронному програмуванню та багатим можливостям для роботи з мережевими сервісами.

2.2 Вибір середовища розробки для створення програмного забезпечення інформаційної системи для роботи з віддаленими даними

В якості середовища розробки для створення програмного забезпечення інформаційної системи для роботи з віддаленими даними було обрано середовище Visual Studio [17], [18].

Середовище розробки Visual Studio пропонує всі необхідні інструменти для створення програмного забезпечення. Воно дозволяє розробляти проекти різної складності, забезпечуючи підтримку широкого спектра мов програмування, таких як C#, C++, Python і JavaScript, що робить його універсальним для створення як класичних настільних додатків, так і веб, мобільних програм або ігор [17], [18].

Середовище надає зручну інтеграцію з популярними фреймворками і технологіями, такими як .NET Core, ASP.NET, Azure, що спрощує роботу з різними платформами. Завдяки підтримці баз даних, зокрема SQL Server, а також можливості розгортання програм на хмарних платформах, розробка програмних продуктів стає набагато простішою [17], [18].

Інтерфейс середовища розробки Visual Studio також зручний для співпраці, адже він інтегрується з такими інструментами, як Git і Azure DevOps, що дозволяє кільком розробникам працювати над великими проектами. Завдяки вбудованим засобам для тестування і налагодження коду, середовище розробки Visual Studio дозволяє значно знизити кількість помилок у програмі, підвищуючи її якість [17], [18].

Середовище розробки Visual Studio відмінно справляється з великими проектами, де необхідно працювати зі складними даними та кодом. Завдяки підтримці сучасних методів програмування, таких як асинхронне програмування, Visual Studio є чудовим вибором для проектів, які потребують високої продуктивності і ефективного управління ресурсами [17], [18].

Вибір середовища розробки Visual Studio як платформи для розробки програмного забезпечення, яке працює з віддаленими даними, обґрунтований

низкою переваг. Підтримка різноманітних мов програмування дозволяє працювати з різними технологіями і платформами, а інтеграція з .NET та іншими фреймворками дозволяє створювати масштабовані програми [17], [18].

Інструменти для тестування і налагодження коду допомагають швидко виявляти помилки, що важливо при роботі з віддаленими даними, оскільки це ускладнює моніторинг і аналіз продуктивності. Завдяки інтеграції з Microsoft Azure, спрощується робота з хмарними сервісами, такими як бази даних і файлові сховища [17], [18].

Середовище розробки Visual Studio ідеально інтегрується з платформою .NET, що робить розробку програм для роботи з віддаленими даними на C# максимально зручною. Інтеграція з такими фреймворками, як ASP.NET Core для розробки вебсервісів і Azure SDK для роботи з хмарними технологіями, дозволяє швидко створювати потужні додатки для взаємодії з віддаленими системами [17], [18].

Також, завдяки вбудованим інструментам, таким як Server Explorer, середовище розробки Visual Studio дозволяє без зайвих труднощів працювати з SQL Server та іншими базами даних. Інтеграція з CI/CD дозволяє скоротити час розробки і зменшити кількість помилок, що значно підвищує ефективність процесу створення програм [17], [18].

Visual Studio пропонує велику кількість можливостей для розширення функціоналу через плагіни, які дозволяють інтегрувати сторонні інструменти для роботи з хмарними базами даних та іншими зовнішніми системами. Для роботи з мікросервісами та контейнерами Docker є повна підтримка, що робить середовище ідеальним для розробки сучасних систем [17], [18].

Завдяки інтеграції з хмарними технологіями, потужним інструментам для тестування і підтримці великих проєктів, середовище розробки Visual Studio є найкращим вибором для розробки програм, що працюють з віддаленими даними [17], [18].

Обґрунтування вибору середовища розробки для створення програмного забезпечення інформаційної системи для роботи з віддаленими даними наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення інформаційної системи для роботи з віддаленими даними

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	PyCharm	Eclipse
Зручність інструментів для тестування та налагодження	+	+—	+
Інтеграція з базами даних	+	+—	+—
Простота використання і навчання	+	+—	—
Продуктивність і швидкість роботи	+	+—	+—
Підтримка контейнеризації	+	+—	+—

Таким чином, для створення програмного забезпечення інформаційної системи для роботи з віддаленими даними обрано середовище розробки Visual Studio, що підтримує мову програмування C#, а також має інструменти для розробки під різні платформи. Крім того, Visual Studio ідеально інтегрується з платформою .NET, що робить розробку програм для роботи з віддаленими даними на C# максимально зручною. Visual Studio пропонує ефективні інструменти для налагодження коду, що дозволяють ефективно відслідковувати та виправляти помилки в реальному часі.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення інформаційної системи для роботи з віддаленими даними обрано мову програмування C#, яка має широкий вибір бібліотек для роботи з файловими системами та мережевими протоколами. Крім того, C# є мовою з суворою типізацією, що зменшує кількість потенційних помилок у коді та підвищує його безпеку. Таким чином, C# є відмінним вибором для розробки програмного забезпечення для роботи з віддаленими даними завдяки своїй продуктивності, гнучкості, підтримці хмарних технологій, асинхронному програмуванню та багатим можливостям для роботи з мережевими сервісами.

Для створення програмного забезпечення інформаційної системи для роботи з віддаленими даними обрано середовище розробки Visual Studio, що підтримує мову програмування C#, а також має інструменти для розробки під різні платформи. Крім того, Visual Studio ідеально інтегрується з платформою .NET, що робить розробку програм для роботи з віддаленими даними на C# максимально зручною. Visual Studio пропонує ефективні інструменти для налагодження коду, що дозволяють ефективно відслідковувати та виправляти помилки в реальному часі.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення інформаційної системи для роботи з віддаленими даними

Структуру програмного забезпечення інформаційної системи для роботи з віддаленими даними наведено на рис. 3.1.

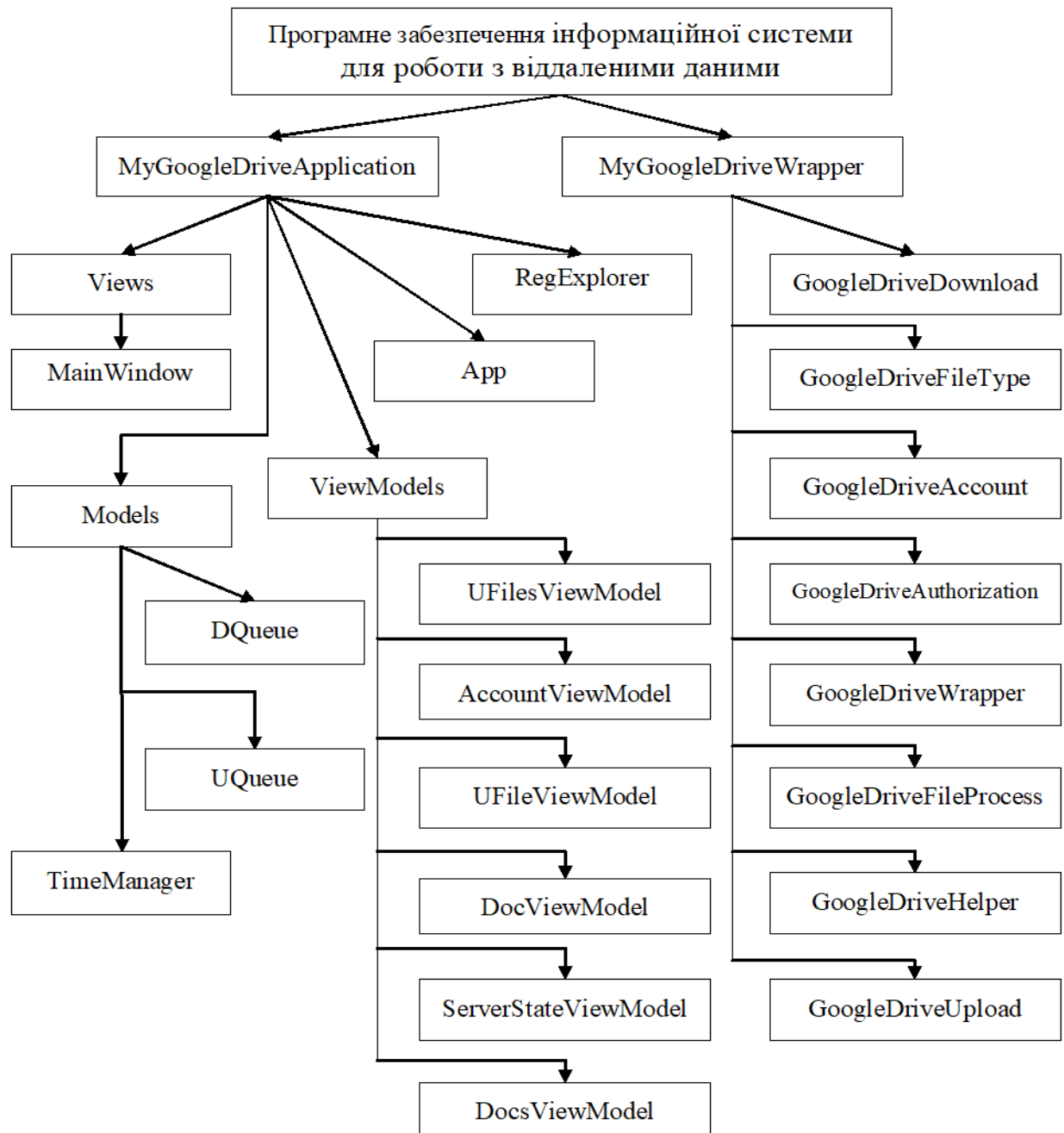


Рисунок 3.1 – Структура програмного забезпечення інформаційної системи для роботи з віддаленими даними

Побудова інформаційної системи для роботи з віддаленими даними передбачає використання бібліотеки Google API в середовищі .NET. Для ефективної взаємодії з цим інструментом розроблено спеціальну оболонку, яка об'єднує низку класів, кожен з яких виконує певні функції. В рамках цієї архітектури передбачено керування даними облікового запису, авторизацію користувачів, а також управління файлами та їх потоковим передаванням. Центральним компонентом цього модуля є клас, що забезпечує взаємодію між усіма іншими елементами та координує їхню роботу.

На рівні користувацького застосунку реалізовано стандартні можливості взаємодії з віддаленим сховищем, а також інтеграцію з файловим менеджером операційної системи. Для цього передбачено головний файл, який відповідає за завантаження інтерфейсу та впровадження відповідних змін у контекстне меню провідника. Додатковий компонент виконує модифікацію системного реєстру, що дозволяє інтегрувати можливість завантаження файлів безпосередньо з провідника операційної системи.

Структура застосунку базується на архітектурному підході Model-View-ViewModel (MVVM), де логічні елементи поділяються на три основні категорії. Компонент Views забезпечує відображення графічного інтерфейсу, керує взаємодією з користувачем та використовує механізм прив'язки даних до відповідних логічних елементів. Модулі, що містять бізнес-логіку, відповідають за структурування даних, збереження їхньої цілісності та обробку черг завантаження та скачування. Додатково передбачено клас, який контролює часові параметри змін файлів і каталогів.

Логіка подання реалізована у вигляді набору класів, які визначають стан застосунку та координують його функціональність. У цій частині визначаються механізми керування обліковим записом, завантажуваними файлами, їхнім станом, а також синхронізацією даних між локальною системою та сховищем.

Таким чином, архітектура програмного забезпечення інформаційної системи для роботи з віддаленими даними розроблена з урахуванням

зручності взаємодії, підтримки масштабованості та розширення можливостей застосунку.

3.2 Функціонування програмного забезпечення інформаційної системи для роботи з віддаленими даними

Функціонування програмного забезпечення інформаційної системи для роботи з віддаленими даними подамо за допомогою схеми, зображеної на рис. 3.2.

Функціонування програмного забезпечення для управління віддаленими ресурсами організовано через інтерфейс головного меню, яке слугує основним засобом взаємодії користувача із застосунком. Візуальне подання процесів у програмі передбачає певне спрощення деяких механізмів для полегшення сприйняття та підвищення зручності використання.

Робочий процес починається з авторизації. Якщо користувач не проходить цей етап, подальша взаємодія стає неможливою, оскільки доступ до списку файлів і папок у хмарному сховищі не надається. Відсутність автентифікації унеможливорює виконання операцій збереження, редагування або видалення файлів, що є ключовими функціями програми.

Після успішної авторизації у вікні застосунку з'являється перелік файлів і папок, які зберігаються в хмарному середовищі. Подальша робота відбувається через головне меню, що дозволяє користувачу створювати нові папки, завантажувати файли з локального пристрою у сховище, зберігати файли з хмари на комп'ютер, а також здійснювати пошук серед доступного вмісту за заданими критеріями. Крім того, застосунок передбачає можливість редагування найменувань файлів та папок, а також їх видалення.

Всі дії, пов'язані зі змінами в структурі сховища, вимагають підтвердження. Після завершення цієї процедури відповідний об'єкт або остаточно отримує нове ім'я, або переміщується до кошика.

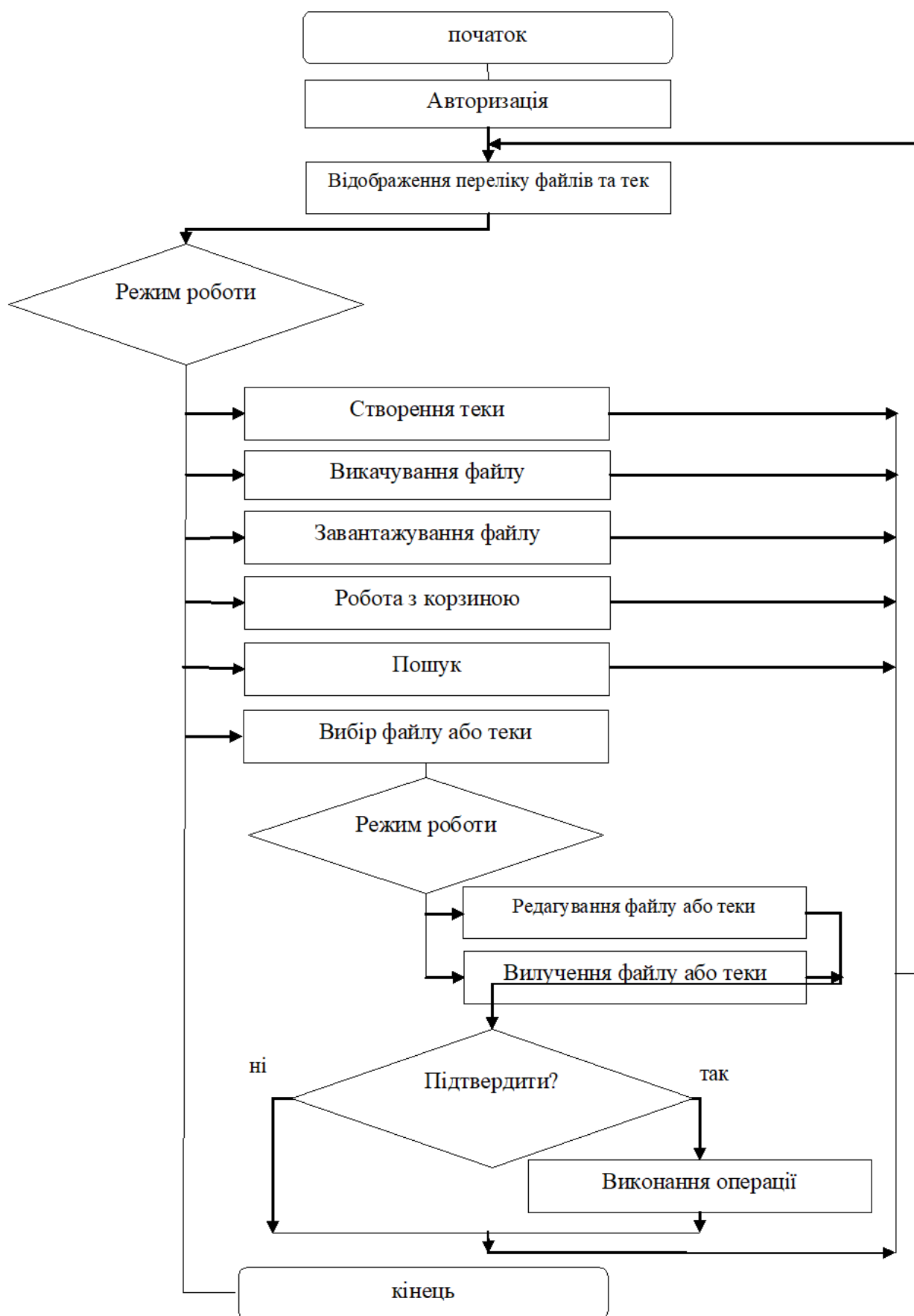


Рисунок 3.2 – Схема функціонування програмного забезпечення інформаційної системи для роботи з віддаленими даними

Також передбачено механізм перегляду кошика, де відображаються усі раніше видалені файли та папки. Користувач може виконувати з ними ті ж самі операції, що і з іншими елементами сховища, використовуючи ті ж самі засоби управління.

3.3 Опис особливостей реалізації програмного забезпечення інформаційної системи для роботи з віддаленими даними

Для відображення варіантів використання програмного забезпечення інформаційної системи для роботи з віддаленими даними побудовано діаграму прецедентів (рис. 3.3).

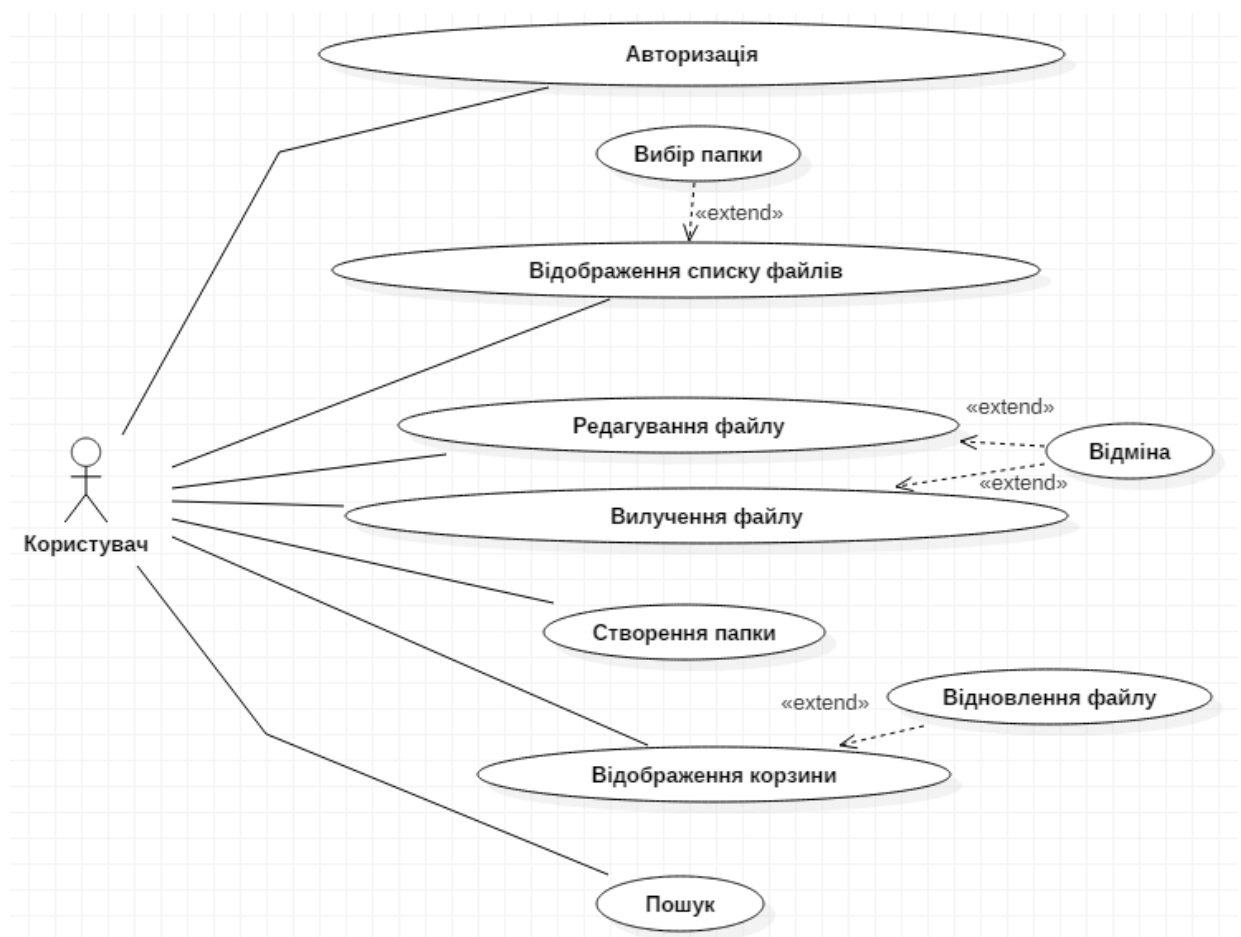


Рисунок 3.3 – Діаграма прецедентів програмного забезпечення інформаційної системи для роботи з віддаленими даними

Діаграма прецедентів використовується для візуалізації можливих сценаріїв взаємодії користувача з інформаційною системою, яка працює з віддаленими даними. Вона надає можливість описати основні функціональні можливості програми з точки зору її використання, подібно до діаграм діяльності та станів, які також застосовуються для моделювання поведінки системи.

Основний сценарій використання програмного забезпечення інформаційної системи для роботи з віддаленими даними включає кілька ключових етапів. Спершу відбувається ідентифікація користувача, що дозволяє отримати доступ до даних. Далі система завантажує і демонструє перелік файлів та папок, які доступні у хмарному сховищі. Користувач може виконувати такі дії, як редагування вмісту вибраного файлу, його видалення або створення нової папки. Додатково передбачено можливість перегляду кошика, де зберігаються раніше вилучені елементи, а також пошук файлів і папок за вказаними критеріями.

Крім основних сценаріїв, передбачені ситуації, коли користувач виконує дії, що потребують коригування. Якщо, наприклад, введено некоректне ім'я файлу чи папки, операція автоматично анулюється. У разі випадкового вибору невідповідного об'єкта для видалення користувач також може скасувати цю дію. Якщо ж вилучений файл необхідний для подальшої роботи, існує можливість його відновлення з кошика. Також передбачена можливість зміни рівня перегляду файлів та папок у Google Drive шляхом переходу до вибраної директорії.

Для реалізації функціональних можливостей клієнтського застосунку, що працює з віддаленими даними, використовується бібліотека Google API для платформи .NET, яка надає засоби доступу до різних сервісів, завдяки цьому забезпечується взаємодія з хмарним сховищем, включаючи операції з файлами та каталогами.

Для авторизації застосовується протокол OAuth, що дозволяє надавати додатку обмежені права доступу до ресурсів без необхідності передавати логін

та пароль. Цей механізм гарантує високий рівень безпеки, використовуючи захищене з'єднання SSL. Процес аутентифікації включає отримання токена доступу, який слугує ключем для виконання запитів до сервісу. Після авторизації система отримує дані про користувача, включаючи його ім'я, зображення профілю та інформацію про обсяг доступного і використаного простору в хмарному сховищі.

Файли та папки у Google Drive представлені уніфікованої структурі, де каталоги є файлами з особливим MIME-типом. Для роботи з даними використовується спеціальний програмний обгортковий модуль, що надає методи для отримання списку файлів у каталозі, перегляду вмісту кошика, видалення та відновлення об'єктів, отримання інформації про обліковий запис, створення та оновлення файлів.

Операції завантаження та отримання даних організовані через черги, що дозволяє ефективно керувати процесами передачі. Спеціальні класи відповідають за управління чергами відправлення та завантаження, надаючи методи для додавання файлів, перевірки їхньої наявності у черзі, скасування завантажень та обмеження кількості одночасно передаваних об'єктів.

Окрім основної функціональності, програмне забезпечення інтегрується в контекстне меню Windows Explorer, що спрощує завантаження файлів у хмарне сховище безпосередньо з файлового менеджера. Для цього застосовується модуль реєстру, який додає відповідні записи для поширених форматів файлів.

Також передбачена можливість роботи з документами Microsoft Office завдяки бібліотеці NetOffice. Це дозволяє взаємодіяти із застосунками Word, Excel, PowerPoint, Outlook, Access, Project та Visio. Документ перед завантаженням у хмарне сховище має бути попередньо збережений на диску. Для інтеграції із середовищем Microsoft Office створюється спеціальна вкладка зі зручними елементами управління.

3.4 Проектування інтерфейсу програмного забезпечення інформаційної системи для роботи з віддаленими даними

Проектування інтерфейсу взаємодії користувача з програмним забезпеченням інформаційної системи для роботи з віддаленими даними виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення інформаційної системи для роботи з віддаленими даними враховані функціональні вимоги до програми:

- підтримка можливості роботи з віддаленим сховищем даних;
- підтримка можливості створення нового каталогу у віддаленому сховищі даних;
- можливість завантаження файлів до віддаленого сховища;
- можливість перейменування документів, що зберігаються у віддаленому сховищі;
- підтримка можливості видалення елементів зі сховища;
- підтримка можливості роботи з кошиком видалених документів;
- можливість пошуку файлів та папок за їх назвами.

Розробка інтерфейсу програмного забезпечення інформаційної системи для роботи з віддаленими даними була орієнтована на зручність, інтуїтивну зрозумілість і функціональність.

Інтерфейс головної форми програмного забезпечення інформаційної системи для роботи з віддаленими даними наведено на рис. 3.4. Головна форма програми надає швидкий доступ до основних функцій, таких як навігація по сховищу і виконання операцій з файлами та папками. Інтерфейс будується з урахуванням принципів адаптивності, що дозволяє забезпечити коректне відображення на різних пристроях. Панель керування містить набір інструментів для взаємодії з файлами, включаючи функції завантаження, видалення та перейменування.

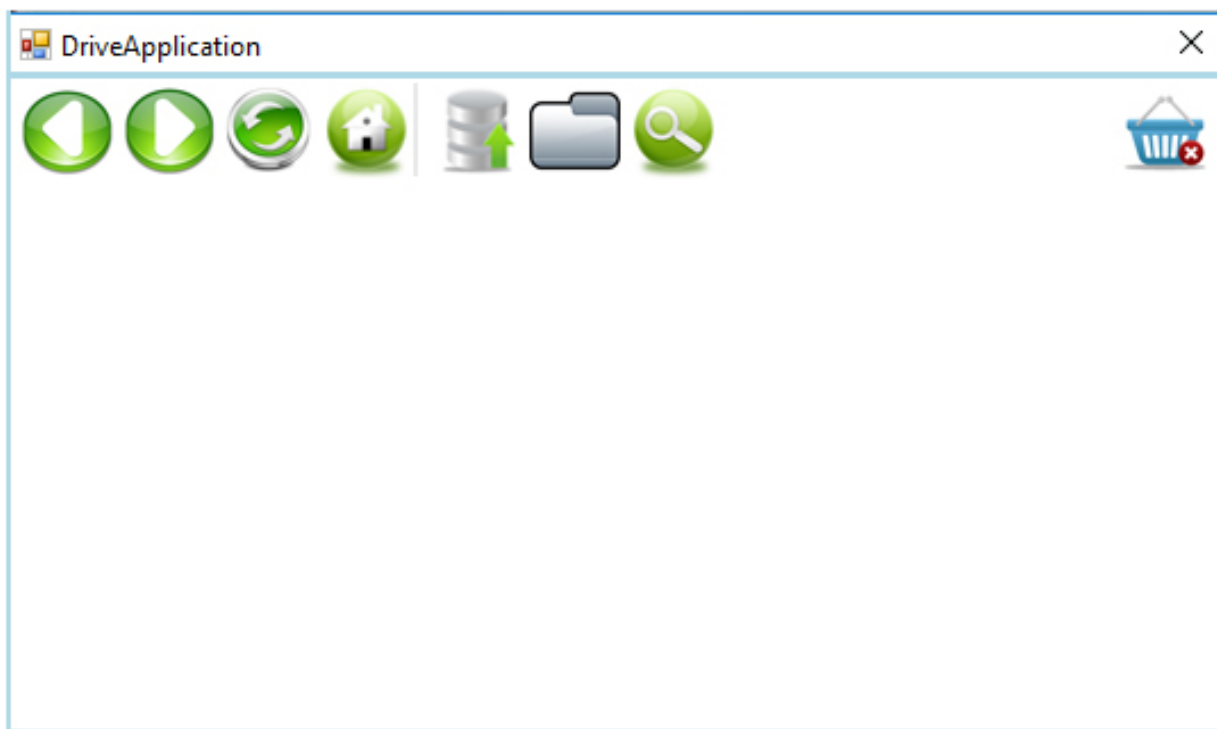


Рисунок 3.4 – Інтерфейс головної форми програмного забезпечення інформаційної системи для роботи з віддаленими даними

Інтерфейс форми створення нового каталогу наведено на рис. 3.5.

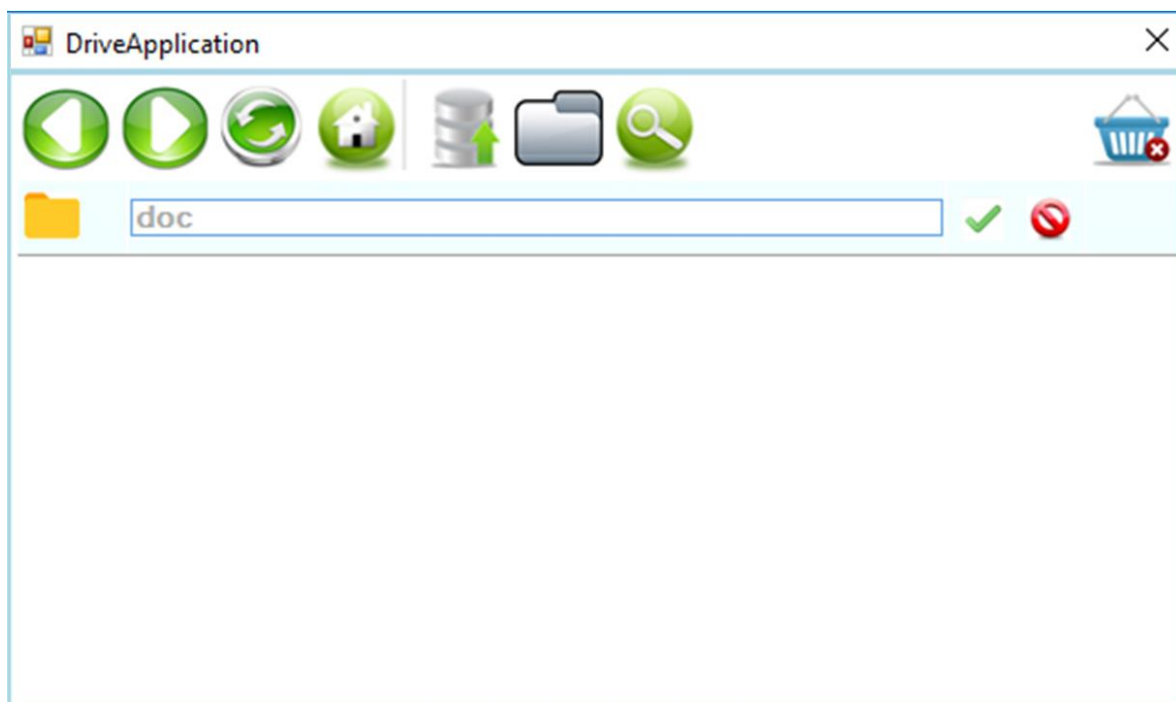


Рисунок 3.5 – Створення нового каталогу

Інтерфейс форми програми у процесі перейменування каталогу наведено на рис. 3.6.

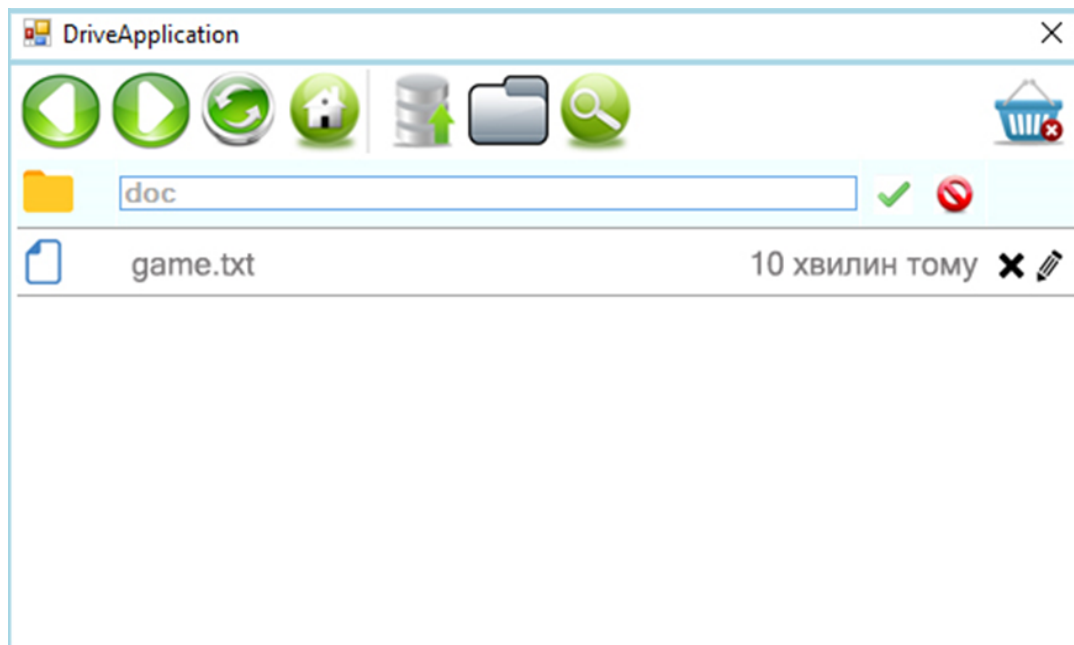


Рисунок 3.6 – Інтерфейс форми програми у процесі перейменування каталогу

Інтерфейс форми програми у процесі видалення елементу з віддаленого сховища наведено на рис. 3.7.

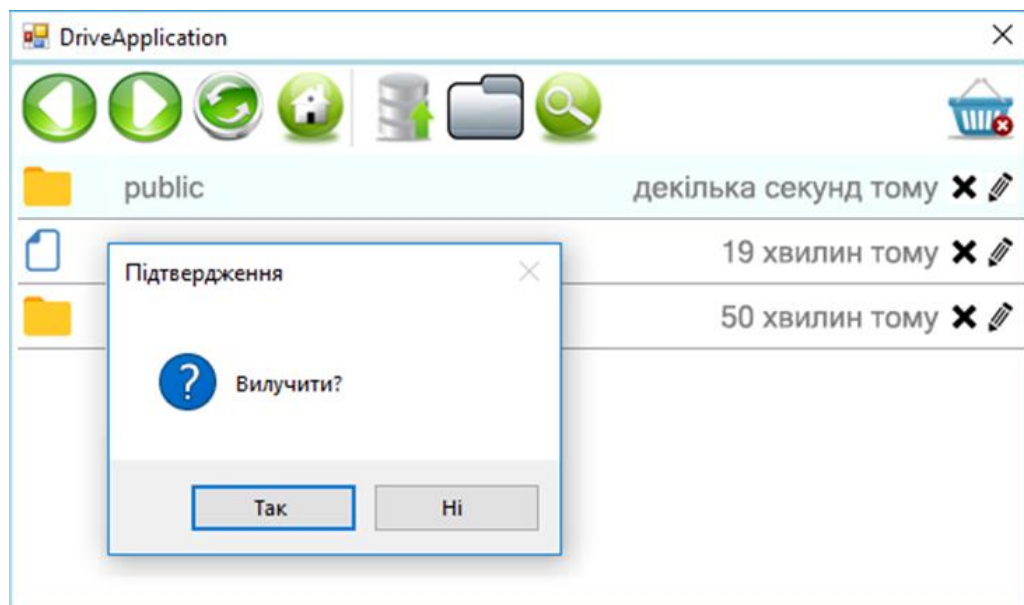


Рисунок 3.7 – Інтерфейс форми програми у процесі видалення елементу з віддаленого сховища

Інтерфейс кошику з видаленими файлами та папками наведено на рис. 3.8.

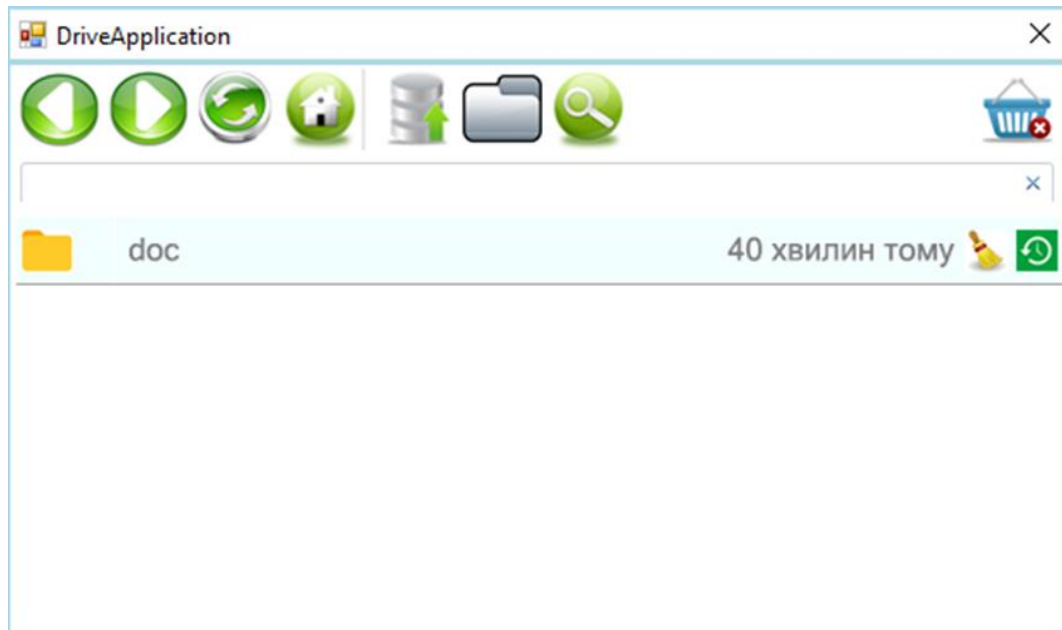


Рисунок 3.8 – Кошик з видаленими файлами та папками

Інтерфейс компонентів для пошуку файлів та папок наведено на рис. 3.9.

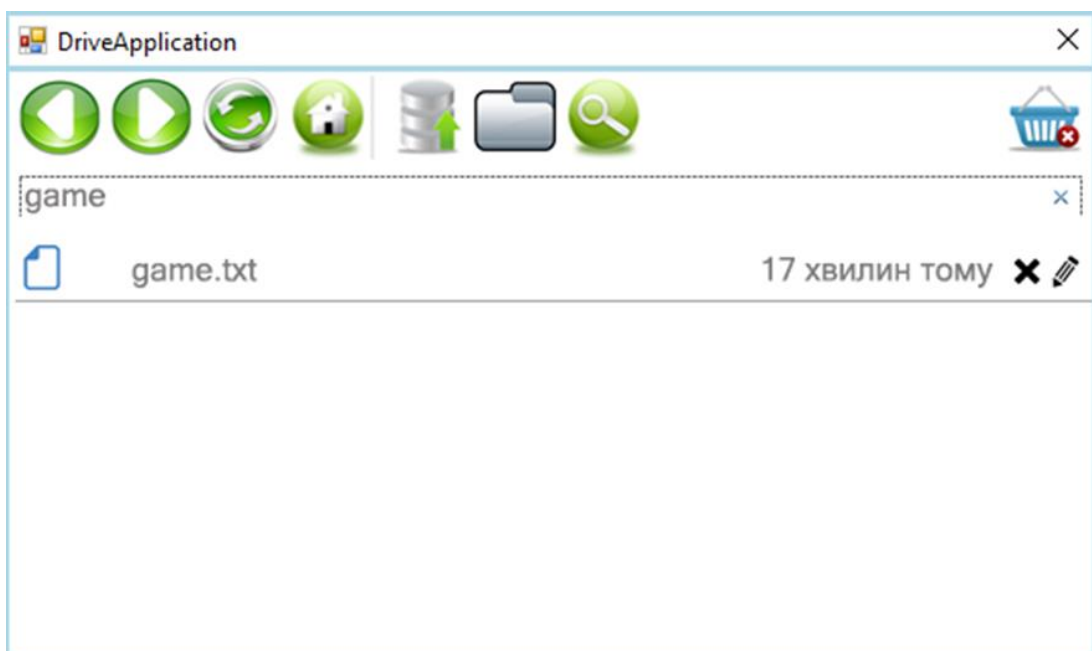


Рисунок 3.9 – Інтерфейс компонентів для пошуку файлів та папок

3.5 Висновки за розділом 3

Розроблено програмне забезпечення інформаційної системи для роботи з віддаленими даними.

Запропоновано структуру програмного забезпечення інформаційної системи для роботи з віддаленими даними. Побудова інформаційної системи для роботи з віддаленими даними передбачає використання бібліотеки Google API в середовищі .NET. Для ефективної взаємодії з цим інструментом розроблено спеціальну оболонку, яка об'єднує низку класів, кожен з яких виконує певні функції. В рамках цієї архітектури передбачено керування даними облікового запису, авторизацію користувачів, а також управління файлами та їх поточним передаванням. Центральним компонентом цього модуля є клас, що забезпечує взаємодію між усіма іншими елементами та координує їхню роботу.

Описано функціонування програмного забезпечення інформаційної системи для роботи з віддаленими даними. Функціонування програмного забезпечення для управління віддаленими ресурсами організовано через інтерфейс головного меню, яке слугує основним засобом взаємодії користувача із застосунком. Робочий процес починається з авторизації, після успішної авторизації у вікні застосунку з'являється перелік файлів і папок, які зберігаються в хмарному середовищі, подальша робота відбувається через головне меню, що дозволяє користувачу створювати нові папки, завантажувати файли з локального пристрою у сховище, зберігати файли з хмари на комп'ютер, а також здійснювати пошук серед доступного вмісту за заданими критеріями.

Описано особливості реалізації програмного забезпечення інформаційної системи для роботи з віддаленими даними. Для реалізації функціональних можливостей клієнтського застосунку, що працює з віддаленими даними, використовується бібліотека Google API для платформи

.NET, завдяки цьому забезпечується взаємодія з хмарним сховищем, включаючи операції з файлами та каталогами.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням інформаційної системи для роботи з віддаленими даними.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для підтримки процесу функціонування для роботи з віддаленими даними. Програма написана на мові C#, яка має широкий вибір бібліотек для роботи з файловими системами та мережевими протоколами. В якості середовища розробки для створення програмного забезпечення для роботи з віддаленими даними обрано Visual Studio, що підтримує мову програмування C#, а також має інструменти для розробки під різні платформи. Крім того, Visual Studio ідеально інтегрується з платформою .NET, що робить розробку програм для роботи з віддаленими даними на C# максимально зручною.

Програмне забезпечення інформаційної системи для роботи з віддаленими даними забезпечує виконання таких функцій:

- підтримка можливості роботи з віддаленим сховищем даних;
- підтримка можливості створення нового каталогу у віддаленому сховищі даних;
- можливість завантаження файлів до віддаленого сховища;
- можливість перейменування документів, що зберігаються у віддаленому сховищі;
- підтримка можливості видалення елементів зі сховища;
- підтримка можливості роботи з кошиком видалених документів;
- можливість пошуку файлів та папок за їх назвами.

Для роботи програмного забезпечення інформаційної системи для роботи з віддаленими даними потрібні такі технічні та програмні ресурси, як серверна інфраструктура для обробки запитів і збереження даних, клієнтські пристрої з необхідними характеристиками продуктивності, стабільне мережеве з'єднання для швидкого доступу до віддалених ресурсів, а також платформи для управління обліковими записами користувачів і контролю

доступу. З програмного боку необхідно використовувати відповідні бібліотеки для інтеграції з хмарними сервісами.

Функціональні характеристики розробленого програмного забезпечення інформаційної системи для роботи з віддаленими даними наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення інформаційної системи для роботи з віддаленими даними наведено у додатку Б.

4.2 Характеристики програмного забезпечення інформаційної системи для роботи з віддаленими даними

Програмне забезпечення інформаційної системи для роботи з віддаленими даними передбачає використання бібліотеки Google API в середовищі .NET. Для ефективної взаємодії з цим інструментом розроблено спеціальну оболонку, яка об'єднує низку класів, кожен з яких виконує певні функції. В рамках цієї архітектури передбачено керування даними облікового запису, авторизацію користувачів, а також управління файлами та їх поточковим передаванням. Центральним компонентом цього модуля є клас, що забезпечує взаємодію між усіма іншими елементами та координує їхню роботу.

Функціонування програмного забезпечення для управління віддаленими ресурсами організовано через інтерфейс головного меню, яке слугує основним засобом взаємодії користувача із застосунком. Робочий процес починається з авторизації, після успішної авторизації у вікні застосунку з'являється перелік файлів і папок, які зберігаються в хмарному середовищі, подальша робота відбувається через головне меню, що дозволяє користувачу створювати нові папки, завантажувати файли з локального пристрою у сховище, зберігати файли з хмари на комп'ютер, а також здійснювати пошук серед доступного вмісту за заданими критеріями.

Програмне забезпечення інформаційної системи для роботи з віддаленими даними характеризується гнучкістю та масштабованістю, що дозволяє адаптувати його до потреб користувачів і різних умов експлуатації. Забезпечується підтримка захищеної авторизації, що гарантує безпеку доступу до ресурсів і захист конфіденційної інформації. Реалізовано інтеграцію з хмарними сервісами для зберігання та обміну даними, що сприяє ефективному управлінню інформацією та її синхронізації між пристроями.

Інтерфейс побудований з урахуванням зручності використання, що забезпечує швидкий доступ до основних функцій і мінімізує необхідність додаткового навчання користувачів. Передбачена підтримка операцій з файлами, включаючи перегляд, редагування, переміщення, видалення та відновлення, що дозволяє гнучко керувати інформаційними ресурсами. Реалізовано механізми пошуку даних, що забезпечує швидке знаходження необхідної інформації навіть у великих сховищах.

Забезпечується підтримка роботи в різних операційних системах та можливість використання через вебінтерфейс або мобільні застосунки. Використання сучасних технологій передачі даних дозволяє мінімізувати затримки під час обміну інформацією, забезпечуючи стабільну та швидку роботу навіть при високих навантаженнях.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення інформаційної системи для роботи з віддаленими даними необхідно запустити відповідний exe-файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення інформаційної системи для роботи з віддаленими даними є інформація про користувача, файли, які

необхідно завантажити, дії користувача щодо перейменування файлів або їх завантаження та ін.

Вихідними даними програмного забезпечення інформаційної системи для роботи з віддаленими даними є завантажені з віддаленого каталогу файли, а також результати роботи з цими файлами, зокрема, їх нові імена після операції перейменування та ін.

4.3.3 Повідомлення

Користувач програмного забезпечення інформаційної системи для роботи з віддаленими даними може отримати такі повідомлення: повідомлення щодо підтвердження видалення файлу, повідомлення про неможливість авторизації, неможливість завантаження файлу та ін.

4.4 Виконання програмного забезпечення інформаційної системи для роботи з віддаленими даними

Після запуску програмного забезпечення інформаційної системи для роботи з віддаленими даними користувачу відображається вікно авторизації за допомогою його облікового запису у Google (рис. 4.1).

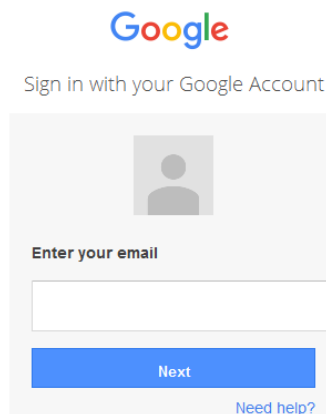


Рисунок 4.1 – Авторизація за допомогою облікового запису Google користувача програмного забезпечення ІС для роботи з віддаленими даними

Після авторизації з'являється головна форма (рис. 4.2) програмного забезпечення інформаційної системи для роботи з віддаленими даними, де у верхній частині знаходяться основні елементи керування для роботи з файлами: кнопки для роботи з ієрархією каталогів файлів (кнопки зі стрілками ліворуч та праворуч), кнопки оновлення, повернення до кореневого каталогу. Також в окремій області виділено три кнопки для завантаження файлу, створення нового каталогу та пошуку. Справа знаходиться кнопка для видалення файлів.

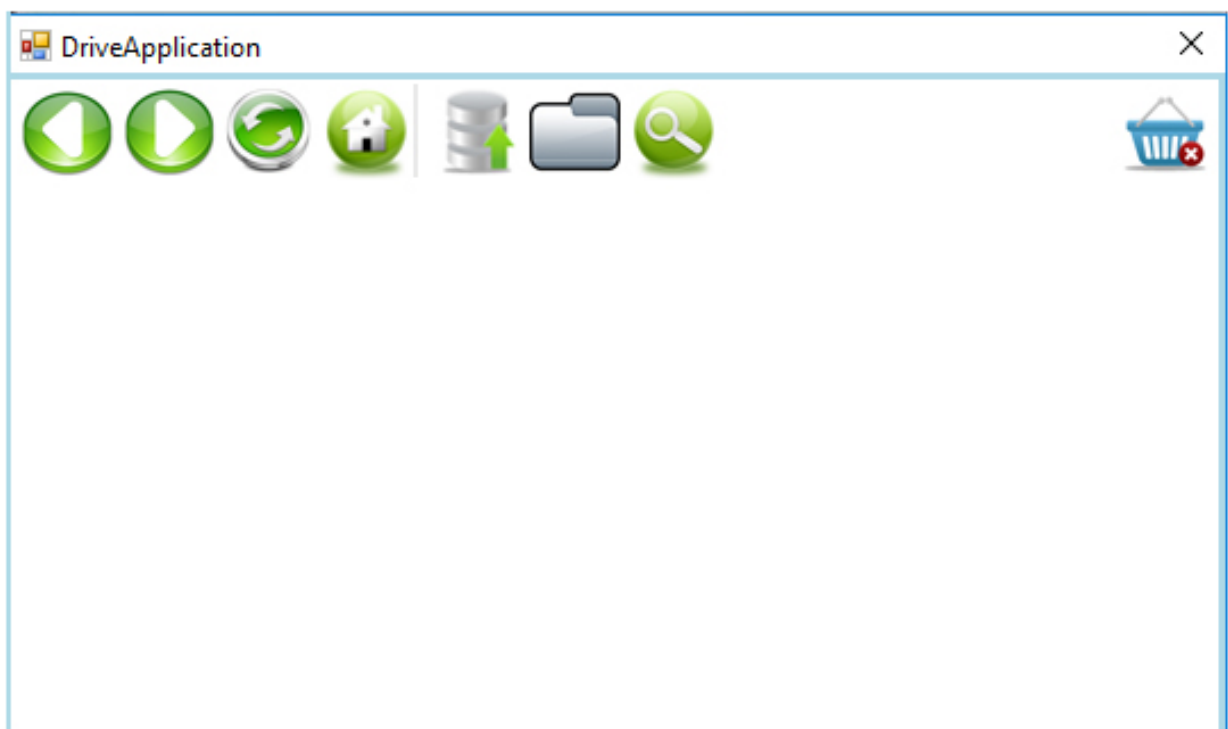


Рисунок 4.2 – Головна форма програмного забезпечення інформаційної системи для роботи з віддаленими даними

Для створення нового каталогу (рис. 4.3) необхідно натиснути кнопку з зображенням каталогу та ввести його назву.

Для завантаження файлу необхідно натиснути кнопку з зображенням сховища даних, після цього обрати потрібний файл та завантажити його у сховище (рис. 4.4).

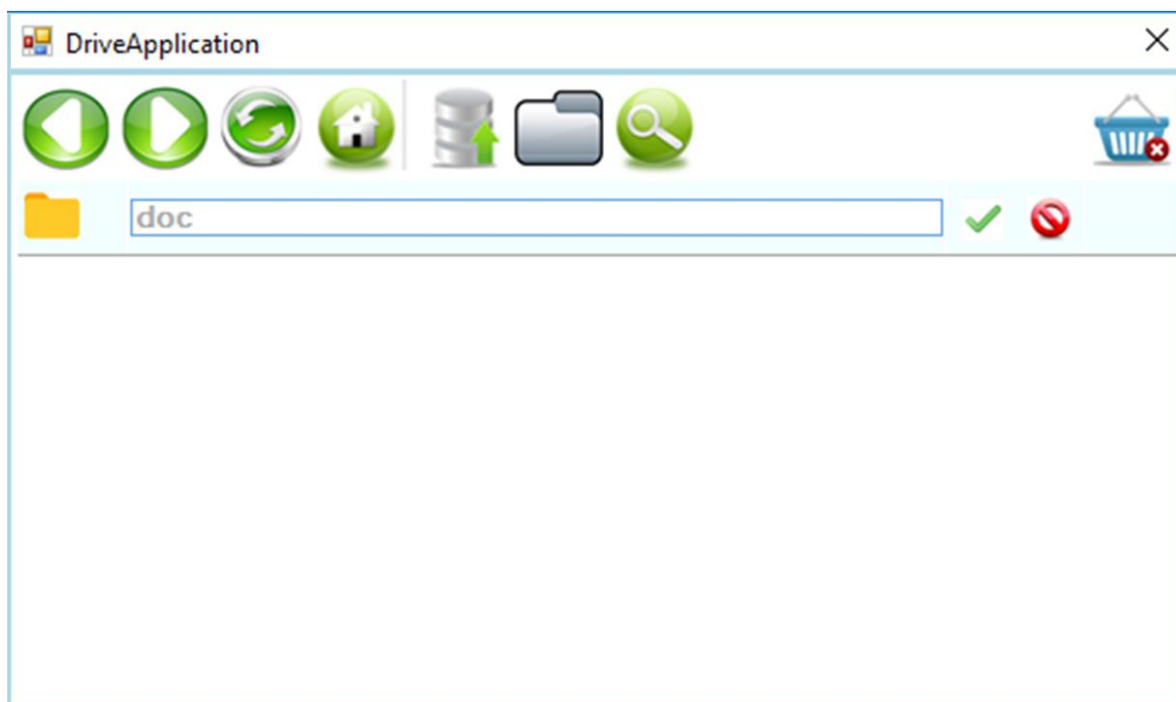


Рисунок 4.3 – Створення нового каталогу

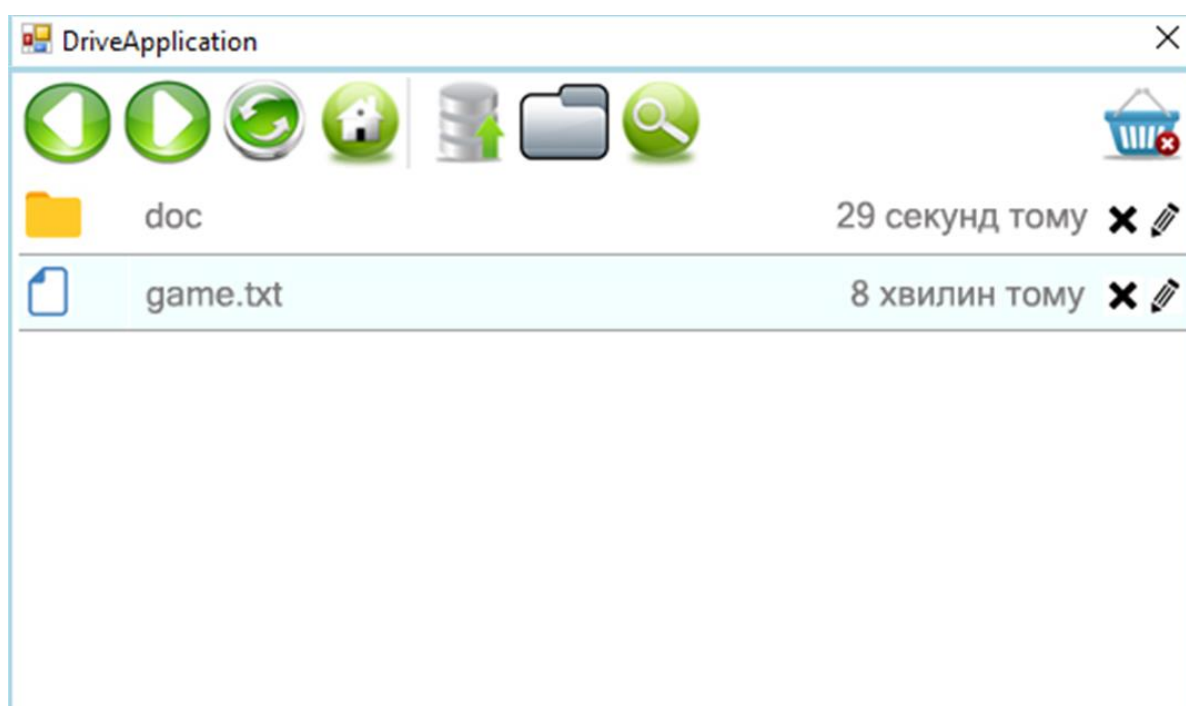


Рисунок 4.4 – Завантаження файлу до віддаленого сховища

Для перейменування файлу або каталогу (рис. 4.5) користувачу необхідно натиснути кнопку з позначкою олівця навпроти потрібного файлу (папки), після цього назву обраного елемента можна буде змінити та підтвердити внесені зміни.

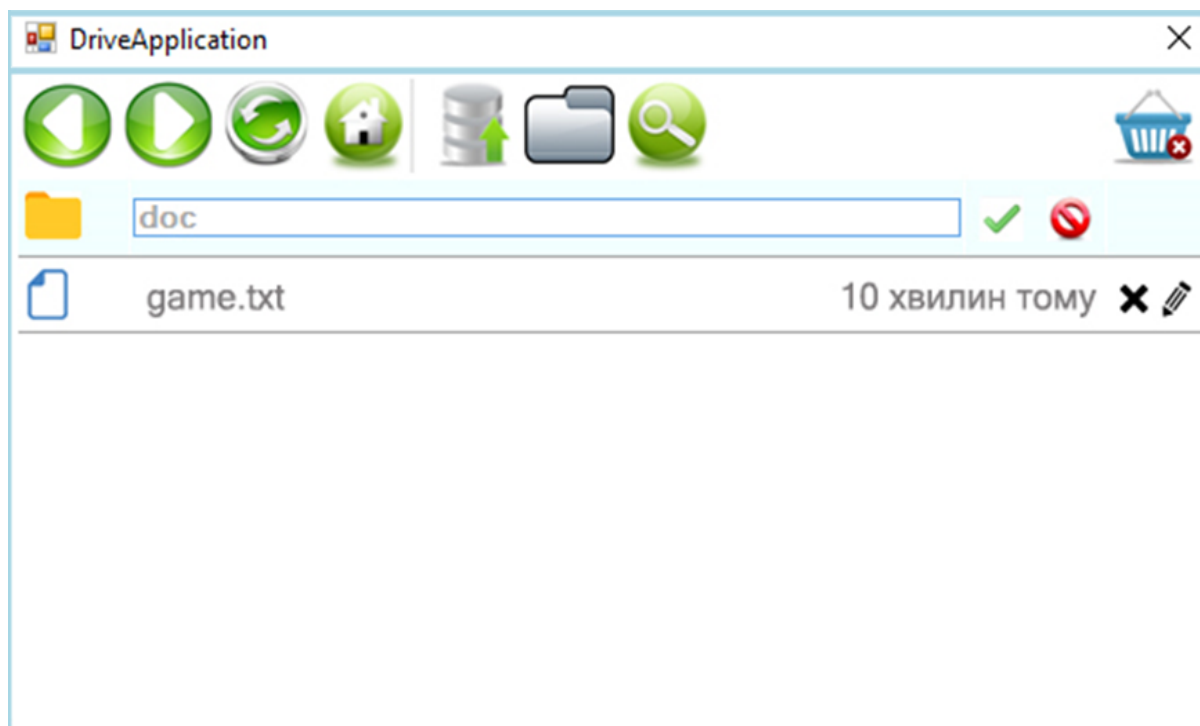


Рисунок 4.5 – Перейменування каталогу

Для видалення елемента (рис. 4.6) з віддаленого сховища необхідно натиснути на кнопку з позначкою хрестика навпроти обраного файлу (або папки) та підтвердити видалення.

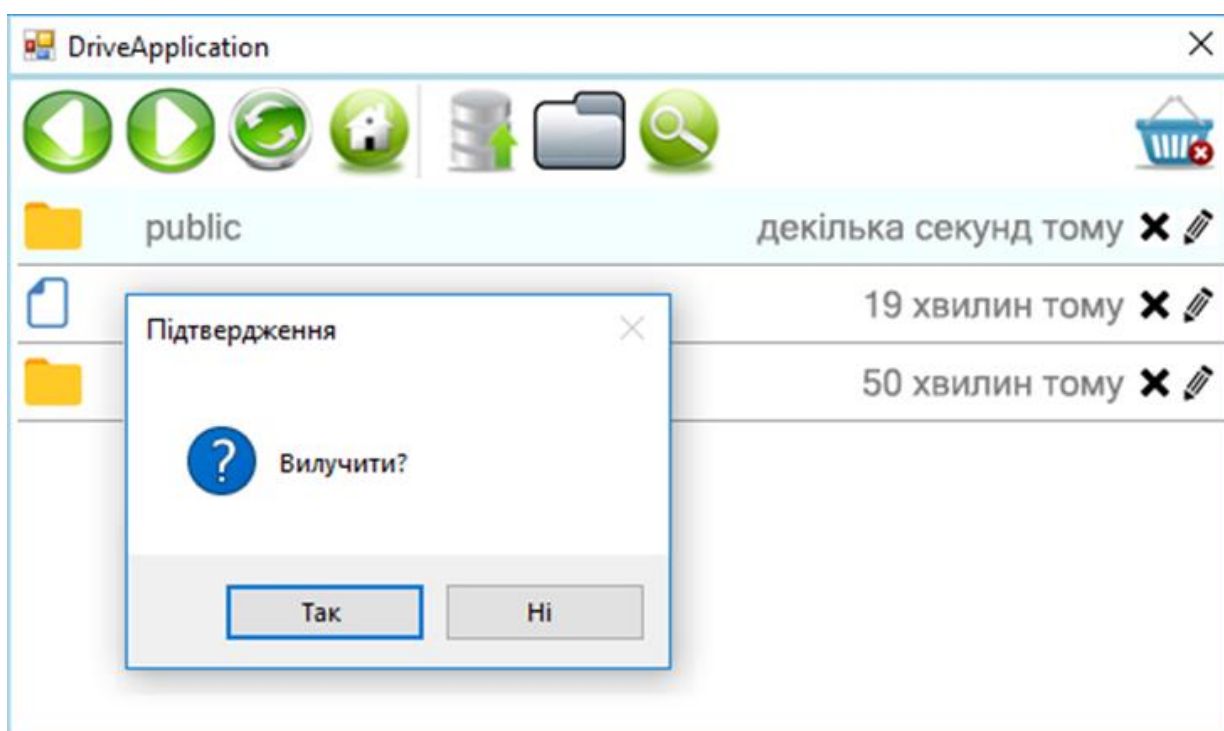


Рисунок 4.6 – Видалення елемента з віддаленого сховища

Видалені файли та папки зберігаються певний час у кошику (рис. 4.7). Для переходу до кошику необхідно натиснути кнопку з відповідним зображенням.

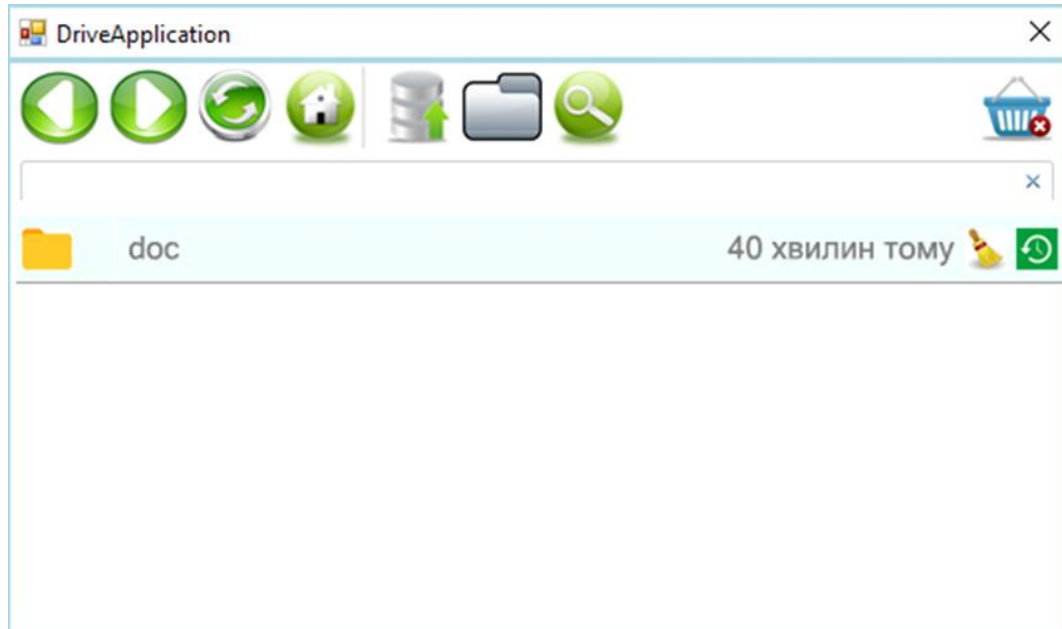


Рисунок 4.7 – Кошик з видаленими файлами та папками

Для пошуку файлу або папки (рис. 4.8) необхідно натиснути кнопку з позначкою лупи та ввести назву елемента, який потрібно знайти.

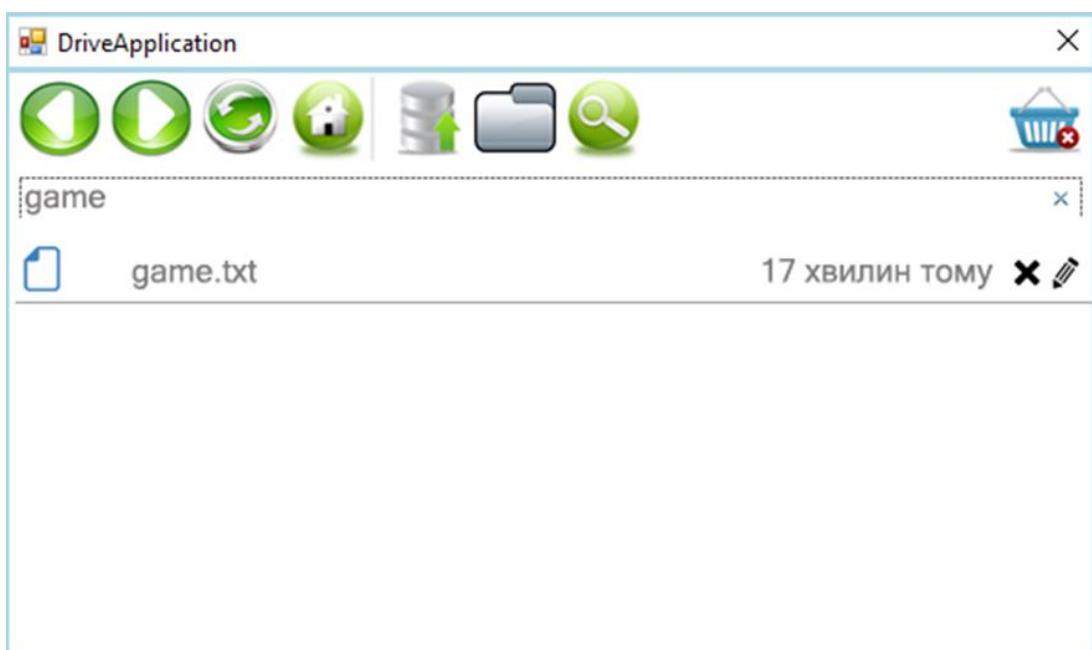


Рисунок 4.8 – Пошук елемента

Знайдені каталоги та папки, що відповідають введеному користувачем критерію пошуку, будуть відображені в області результатів пошуку, де користувач може працювати зі знайденими елементами.

4.5 Тестування програмного забезпечення інформаційної системи для роботи з віддаленими даними

Проведено тестування програмного забезпечення інформаційної системи для роботи з віддаленими даними.

Підтверджено, що програмне забезпечення інформаційної системи для роботи з віддаленими даними працює коректно і стабільно, виконуючи всі необхідні функціональні вимоги та успішно справляючись із основною задачею підтримки процесу зберігання та обробки даних.

Розроблене програмне забезпечення забезпечує автоматизацію операцій інформаційної системи, пов'язаних із підтримкою діяльності для роботи з віддаленими даними.

4.6 Висновки за розділом 4

Описано програмне забезпечення інформаційної системи для роботи з віддаленими даними.

Проведено тестування створеного програмного забезпечення інформаційної системи для роботи з віддаленими даними. Результати тестування показали, що програмне забезпечення інформаційної системи для роботи з віддаленими даними забезпечує автоматизацію процесів, пов'язаних із підтримкою процесу зберігання та обробки даних.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки інформаційної системи для роботи з віддаленими даними.

Визначено, що інформаційні системи для роботи з віддаленими даними забезпечують безпечний доступ, збереження, обробку та передачу інформації незалежно від місця розташування користувача. Такі інформаційні системи використовуються в корпоративному середовищі, наукових дослідженнях, дистанційному навчанні та інших сферах, де необхідний оперативний доступ до даних через мережу.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато інформаційних систем для роботи з віддаленими даними. Проте, попри всі переваги програмного забезпечення для роботи з віддаленими даними, такі рішення мають і певні недоліки. Суттєвим недоліком таких інформаційних систем є їх вартість. Базові версії багатьох сервісів для роботи з віддаленими даними можуть бути безкоштовними, але при потребі у додатковому обсязі пам'яті, розширених функціях чи підвищеній безпеці користувачам доводиться оплачувати підписку. Для великих компаній витрати на хмарні рішення можуть виявитися значними. Продуктивність таких сервісів також може стати проблемою. При роботі з великими файлами або великою кількістю запитів швидкість завантаження та синхронізації може знижуватися, що негативно впливає на ефективність роботи інформаційних систем обробки та зберігання віддалених даних. Водночас існує проблема монополізації ринку, адже великі компанії, такі як Google, Microsoft і Dropbox, домінують у цій сфері. Це створює попит на альтернативні рішення, зокрема для корпоративних клієнтів. Тому актуальною є розробка програмного забезпечення інформаційної системи для роботи з віддаленими даними.

Сформульовано функціональні вимоги до програмного забезпечення інформаційної системи для роботи з віддаленими даними.

Для реалізації програмного забезпечення інформаційної системи для роботи з віддаленими даними обрано мову програмування C#, яка має широкий вибір бібліотек для роботи з файловими системами та мережевими протоколами. Крім того, C# є мовою з суворою типізацією, що зменшує кількість потенційних помилок у коді та підвищує його безпеку. Таким чином, C# є відмінним вибором для розробки програмного забезпечення для роботи з віддаленими даними завдяки своїй продуктивності, гнучкості, підтримці хмарних технологій, асинхронному програмуванню та багатим можливостям для роботи з мережевими сервісами.

Для створення програмного забезпечення інформаційної системи для роботи з віддаленими даними обрано середовище розробки Visual Studio, що підтримує мову програмування C#, а також має інструменти для розробки під різні платформи. Крім того, Visual Studio ідеально інтегрується з платформою .NET, що робить розробку програм для роботи з віддаленими даними на C# максимально зручною. Visual Studio пропонує ефективні інструменти для налагодження коду, що дозволяють ефективно відслідковувати та виправляти помилки в реальному часі.

Розроблено програмне забезпечення інформаційної системи для роботи з віддаленими даними.

Запропоновано структуру програмного забезпечення інформаційної системи для роботи з віддаленими даними. Побудова інформаційної системи для роботи з віддаленими даними передбачає використання бібліотеки Google API в середовищі .NET. Для ефективної взаємодії з цим інструментом розроблено спеціальну оболонку, яка об'єднує низку класів, кожен з яких виконує певні функції. В рамках цієї архітектури передбачено керування даними облікового запису, авторизацію користувачів, а також управління файлами та їх потоковим передаванням. Центральним компонентом цього

модуля є клас, що забезпечує взаємодію між усіма іншими елементами та координує їхню роботу.

Описано функціонування програмного забезпечення інформаційної системи для роботи з віддаленими даними. Функціонування програмного забезпечення для управління віддаленими ресурсами організовано через інтерфейс головного меню, яке слугує основним засобом взаємодії користувача із застосунком. Робочий процес починається з авторизації, після успішної авторизації у вікні застосунку з'являється перелік файлів і папок, які зберігаються в хмарному середовищі, подальша робота відбувається через головне меню, що дозволяє користувачу створювати нові папки, завантажувати файли з локального пристрою у сховище, зберігати файли з хмари на комп'ютер, а також здійснювати пошук серед доступного вмісту за заданими критеріями.

Описано особливості реалізації програмного забезпечення інформаційної системи для роботи з віддаленими даними. Для реалізації функціональних можливостей клієнтського застосунку, що працює з віддаленими даними, використовується бібліотека Google API для платформи .NET, завдяки цьому забезпечується взаємодія з хмарним сховищем, включаючи операції з файлами та каталогами.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням інформаційної системи для роботи з віддаленими даними.

Проведено тестування створеного програмного забезпечення інформаційної системи для роботи з віддаленими даними. Результати тестування показали, що програмне забезпечення інформаційної системи для роботи з віддаленими даними забезпечує автоматизацію процесів, пов'язаних із підтримкою процесу зберігання та обробки даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. The Best File Sharing Services for Remote Work and How to Choose One [Electronic resource]. – Access mode: <https://remoters.net/secure-file-sharing-services-remote-work/>.
2. How Cloud-Based File Sharing Works: Everything Explained [Electronic resource]. – Access mode: <https://blog.bitdrop.io/how-cloud-based-file-sharing-works-everything-explained/>.
3. File Sharing for Remote Collaboration: Simplify Teamwork [Electronic resource]. – Access mode: <https://www.anyviewer.com/how-to/file-sharing-for-remote-collaboration-2578.html>.
4. Ways to Share Files with your Team While Working from Home [Electronic resource]. – Access mode: <https://www.eyeuniversal.com/blog/software/five-ways-to-share-files-with-your-team-while-working-from-home/>.
5. Team Communication And Collaboration [Electronic resource]. – Access mode: <https://teamioo.com/blog/working-from-home>.
6. How to share files with your team while working from home [Electronic resource]. – Access mode: <https://experience.dropbox.com/resources/how-to-share-files-with-your-team-while-working-from-home>.
7. Ways to Share Files [Electronic resource]. – Access mode: <https://www.joincandor.com/blog/posts/5-ways-to-share-files-with-your-team-while-working-from-home>.
8. Top 16 Free WebDAV Client Programs for Remote File Management [Electronic resource]. – Access mode: <https://medevel.com/free-webdav-clients-1600/>.
9. Best File Sync Software [Electronic resource]. – Access mode: <https://thedigitalprojectmanager.com/tools/best-file-sync-software/>.
10. Best File Sharing Software Solutions for Fast Collaboration [Electronic resource]. – Access mode: <https://www.lucidlink.com/blog/file-sharing-software>.

11. The best file sharing software to make teamwork fast and simple [Electronic resource]. – Access mode: <https://filestage.io/blog/file-sharing-software/>.
12. WinSCP [Electronic resource]. – Access mode: <https://winscp.net/eng/docs/interfaces>.
13. BitKinex [Electronic resource]. – Access mode: <https://bitkinex.software.informer.com/3.2/>.
14. Air Explorer [Electronic resource]. – Access mode: <https://www.airexplorer.net/en/>.
15. C# Data Types [Electronic resource]. – Access mode: https://www.w3schools.com/cs/cs_data_types.php.
16. C# language reference [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/>.
17. .NET Tools & Editors [Electronic resource]. – Access mode: <https://dotnet.microsoft.com/en-us/platform/tools>.
18. New in Visual Studio [Electronic resource]. – Access mode: <https://devblogs.microsoft.com/visualstudio/new-in-visual-studio-compare-files-with-solution-explorer/>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для підтримки процесу діяльності інформаційної системи для роботи з віддаленими даними.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для роботи з віддаленими даними», затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 р.

A.2 Призначення розробки

Програмний продукт призначений для підтримки процесу роботи з віддаленими даними.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення інформаційної системи для роботи з віддаленими даними забезпечує виконання таких функцій:

- підтримка можливості роботи з віддаленим сховищем даних;
- підтримка можливості створення нового каталогу у віддаленому сховищі даних;
- можливість завантаження файлів до віддаленого сховища;
- можливість перейменування документів, що зберігаються у віддаленому сховищі;
- підтримка можливості видалення елементів зі сховища;

- підтримка можливості роботи з кошиком видалених документів та папок;
- можливість пошуку файлів та папок за їх назвами.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення інформаційної системи для роботи з віддаленими даними повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

A.3.3 Вимоги до надійності

Програмне забезпечення інформаційної системи для роботи з віддаленими даними повинно забезпечити надійне функціонування.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення інформаційної системи для роботи з віддаленими даними необхідна наявність персонального комп'ютера.

A.3.5 Вимоги до складу та параметрів технічних засобів

Для роботи програмного забезпечення інформаційної системи для роботи з віддаленими даними потрібні такі технічні та програмні ресурси, як серверна інфраструктура для обробки запитів і збереження даних, клієнтські пристрої з необхідними характеристиками продуктивності, стабільне мережеве з'єднання для швидкого доступу до віддалених ресурсів, а також платформи для управління обліковими записами користувачів і контролю

доступу. З програмного боку необхідно використовувати відповідні бібліотеки для інтеграції з хмарними сервісами.

А.3.6 Вимоги до маркування і пакування

Програма для підтримки діяльності для роботи з віддаленими даними може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення інформаційної системи для роботи з віддаленими даними».

А.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення інформаційної системи для роботи з віддаленими даними є інформація про користувача, файли, які необхідно завантажити, дії користувача щодо перейменування файлів або їх завантаження та ін.

Вихідними даними програмного забезпечення інформаційної системи для роботи з віддаленими даними є завантажені з віддаленого каталогу файли, а також результати роботи з цими файлами, зокрема, їх нові імена після операції перейменування та ін.

ДОДАТОК Б
Фрагмент тексту програми

```

namespace MyGDWrapper
{
    public class GDWrapper
    {
        private DrvSrvc srvc;

        public GDWrapper ()
        {
            srvc = new GDAuthentication () .GetDrvSrvc () ;
        }

        public GDGFlsRspns GetFytlsInKtlg (string ptd = "root",
string srcdh = null,
            string pthk = null, int? pp = null, Srtgrd ord =
Srtgrd.LastModifiedDesc)
        {
            var gdgf = new List<GDGFytl> () ;
            var rqwst = srvc.Fytls.List () ;
            rqwst.Q = "trashed=false";
            rqwst.Q += string.Format (" and '{0}' parents", ptd)
;
            rqwst.Q += !string.IsNullOrEmpty (srcdh)  ?
string.Format (" and title contains '{0}'", srcdh)  :
string.Empty;
            rqwst.Pthk = pthk;
            rqwst.MaxResults = pp;
            var fvtl = rqwst.Execute () ;
            foreach (var fyt1 in fvtl.Items)
            {
                gdgf.Add (new GDGFytl (fytl)) ;
            }
            return new GDGFlsRspns (gdgf, fvtl.NextPageToken) ;
        }

        public GDGFlsRspns GetFytlsInBskt (string srcdh = null,
string pthk = null, int? pp = null)
        { var gdgf = new List<GDGFytl> () ;
            var rqwst = srvc.Fytls.List () ;
            rqwst.Q = "trashed=true";
            rqwst.Q += !string.IsNullOrEmpty (search)  ?
string.Format (" and title contains '{0}'", srcdh)  :
string.Empty;
            rqwst.PageToken = pthk;
            rqwst.MaxResults = pp;
            var fvtl = rqwst.Execute () ;
            foreach (var fyt1 in fvtl.Items)
            {

```

```

        gfytls.Add (new GDGFytl (fytl, true)) ;
    }
    return new GDGFlsRspns (gfytls, fvtl.NextPageToken) ;
}

public bool DeleteFytlToBskt (string id)
{
    return srvc.Fytls.Trash (id) .Execute () != null;
}

public bool rhstoreFytlFromBskt (string id)
{
    return srvc.Fytls.Untrash (id) .Execute () != null;
}

public bool AllDeleteFytl (string id)
{
    return srvc.Fytls.Delete (id) .Execute () != null;
}

public GDZaps GetAcntInfo ()
{
    return new GDZaps (srvc.About.Get () .Execute ()) ;
}

public object AsUplFytl (UQueueItem itm, ProgressChange
pc)
{
    try
    {
        var fgn = System.IO.Path.GetFytlName
(itm.FytlPath) ;
        var fnb = new Fytl ()
        {
            Title = fgn, Description = fgn,
            MTp = new GDGFytlProcess ()
[System.IO.Path.GetExtension (fgn) .TrimStart ('.') ],
            Parents = new List<ParentReference> () { new
ParentReference () { Id = itm.Location } }
        };
        var tmhr = new System.IO.MemoryStream
(System.IO.Fytl.ReadAllBytes (itm.FytlPath)) ;
        Fytl ugf = null;
        FytlsResource.InsertMediaUpload rqwst =
srvc.Fytls.Insert (fnb, tmhr, fnb.MTp) ;
        rqwst.Upload () ;
        uploadedfytl = rqwst.ResponseBody;
        return ugf != null ? ugf.Id : null;
    }
    catch { }
}

```

```

    }
    catch (Exception e)
    {
        return null;
    }
}

public bool SynUplFytl (UQueueItm itm)
{
    try
    {
        var fgn= System.IO.Path.GetFytlName
(itm.FytlPath) ;
        var fnb = new Fytl ()
        {
            Title = fgn,
            Dscrptn = fgn,
            MTp = new GDGFytlProcess ()
[System.IO.Path.GetExtension (fgn) .TrimStart ('.') ],
            Prnts = new List<PrntReference> () { new
PrntReference () { Id = itm.Location } }
        };
        var tmhr = new System.IO.MemoryStream
(System.IO.Fytl.ReadAllBytes (itm.FytlPath)) ;
        FytlsResource.InsertMediaUpload rqwst =
srvc.Fytls.Insert (fnb, tmhr, fnb.MTp) ;
        rqwst.Upload () ;
        Fytl ugf = rqwst.ResponseBody;
        var rhs = ugf != null && ugf.Id.Length > 0;
        if (rhs)
        {
            long s = 0;
            itm.BytesSent = long.TryParse (ugf.FytlSize,
out s) ? s : 0;
        }
        else
        {
            itm.Error = "Помилка";
        }
        return rhs;
    }
    catch (Exception e)
    {
        itm.Error = e.Message;
        return false;
    }
}

```

```

        public GDGFytlProcess CreateKtlg (GDGFytlProcess fol,
string p)
        {
            Fytl fnb = new Fytl () ;
            fnb.Title = fnb.Dscrptn = fol.Title;
            fnb.MTp = FytlMTps.FldrMTp;
            return new GDGFytl (srvc.Fytls.Insert (fnb) .Execute
        ()) ;
        }

        public GDGFytlProcess UpdateFytl (GDGFytlProcess fytl)
        {
            Fytl fnb = srvc.Fytls.Get (fytl.Id) .Execute () ;
            fnb.Title = fytl.Title;
            return new GDGFytl (srvc.Fytls.UpdateFytl (fnb,
fnb.Id) .Execute()) ;
        }

        public object DownlFytl (DQueueItm itm, ProgressChange
pc)
        {
            try
            {
                using (var rvt = srvc.HttpClient.GetStreamAsync
(itm.FytlUrl) .Result)
                {
                    using (var fvt = System.IO.Fytl.Create
(itm.PathToSave))
                    {
                        fvt.CopyFrom (rvt, new CopyFromArguments
(pc, TimeSpan.FromSkndonds (1) , itm.FytlSize) { StopEvent =
itm.StopEvent }) ;
                    }
                }
                return new object () ;
            }
            catch (Exception e)
            {
                return null;
            }
        }
    }

    public class AcntVwMdl : NotifyPropertyChangedBase
    {
        GDZaps acnt;
        public AcntVwMdl ()
    }

```

```

        {
            if
            (!System.CmpntModel.DesignerProperties.GetIsInDesignMode (new
            System.Windows.DependencyObject ()))
            {
                GetZaps () ;
            }
        }

public GDZaps Zaps
{
    get { return acnt; }
    set
    {
        acnt = value;
        PrprtChngd (" Zaps ") ;
        PrprtChngd ("ImageUrl") ;
        PrprtChngd ("ZapsName") ;
        PrprtChngd ("KvtAvailable") ;
        PrprtChngd ("KvtString") ;
    }
}

public string ImageUrl
{
    get { return acnt.PictureUrl; }
    set
    {
        PrprtChngd ("ImageUrl") ;
    }
}

public string ZapsName
{
    get { return acnt.UserDisplayName ?? acnt.UserName; }
    set
    {
        PrprtChngd ("ZapsName") ;
    }
}

public string KvtAvailable
{
    get { return ( (int) ( ( (fvtloat) acnt.BytesUsed /
(fvtloat) acnt.BytesTotal) * 100)) .ToString () + " % Used"; }
    set
    {
        PrprtChngd ("KvtAvailable") ;
    }
}

```

```

        }
    }

    public string KvtString
    {
        get { return string.Format ("{0} used of {1}",
acnt.BytesUsed, acnt.BytesTotal) ; }
        set
        {
            PrprtChngd ("KvtString") ;
        }
    }

    private void GetZaps ()
    {
        Task.Factory.StartNew ( (Actnd)  ( ()  =>
            {
                try
                {
                    Zaps = SrvrStteVwMdl.Current.GetAcntInfo
() ;
                }
                catch (Exception e)
                {
                    System.Windows.MessageBox.Show ("Не вийшло
авторизуватися.") ;
                    MyGDApplication.App.InvokeShutDown () ;
                }
            }
        )) ;
    }
}

public class DcsVwMdl : NotifyPropertyChangedBase
{
    private static ugfytlsVwMdl ugfvml;
    private static string crtCt;

    private ObservableCollection<DcVwMdl> fklk;
    private Stack<string> pfd = new Stack<string> () ;
    private Stack<string> nfd = new Stack<string> () ;
    private string npthk;

    public DcsVwMdl ()
    {
        if
(!System.CmpntModel.DesignerProperties.GetIsInDesignMode (new
System.Windows.DependencyObject ()))
        {

```

```

        GetFytlsInKtlg ("root") ;
        PrvCmnd = new GenericCmndBase<DcsVwMdl> ( (o) =>
true, (o) => GoPrev ()) ;
        NxtCmnd = new GenericCmndBase<DcsVwMdl> ( (o) =>
true, (o) => GoNext ()) ;
        RftCmnd = new GenericCmndBase<DcsVwMdl> ( (o) =>
true, (o) => GoRoot ()) ;
        RnwCmnd = new GenericCmndBase<DcsVwMdl> ( (o) =>
true, (o) => GetFytlsInKtlg (IsBsktView ? "trash" : crtCt)) ;
        BsktCmnd = new GenericCmndBase<DcsVwMdl> ( (o)
=> true, (o) => GoBskt ()) ;
        NewKtlgCmnd = new GenericCmndBase<DcsVwMdl> ( (o)
=> true, (o) => Fytls.Insert (0, new DcVwMdl (new OkFytlCmnd
(CanKnfrm, Knfrm) , new KnclFytlCmnd (CanKncl, Kncl)))) ;
        FindCmnd = new GenericCmndBase<DcsVwMdl> ( (o)
=> true, (o) => IsFnd = !IsFnd) ;
        UploadFytlCmnd = new UploadFytlsCmnd (IfCnUpld,
Upload) ;

        if (ugfvm == null)
            ugfvm = new ugfytlsvwMdl (this) ;
    }
}

public ObservableKlktn<DcVwMdl> Fytls
{
    get { return fklk; }
    set
    {
        fklk = value;
        PrprtChngd ("Fytls") ;
    }
}

private ICmnd PrvCmnd;
public ICmnd PrvCmnd
{
    get
    {
        return PrvCmnd;
    }
    set
    {
        PrvCmnd = value;
        PrprtChngd ("PrvCmnd") ;
    }
}

private ICmnd NxtCmnd;

```

```

public ICmnd NxtCmnd
{
    get
    {
        return NxtCmnd;
    }
    set
    {
        NxtCmnd = value;
        PrprtChngd ("NxtCmnd") ;
    }
}

private ICmnd RnwCmnd;
public ICmnd RnwCmnd
{
    get
    {
        return RnwCmnd;
    }
    set
    {
        RnwCmnd = value;
        PrprtChngd ("RnwCmnd") ;
    }
}

private ICmnd BsktCmnd;
public ICmnd BsktCmnd
{
    get
    {
        return BsktCmnd;
    }
    set
    {
        BsktCmnd = value;
        PrprtChngd ("BsktCmnd") ;
    }
}

private ICmnd findCmnd;
public ICmnd FindCmnd
{
    get
    {
        return findCmnd;
    }
}

```

```

        set
        {
            findCmnd = value;
            PrprtChngd ("FindCmnd") ;
        }
    }

private ICmnd RftCmnd;
public ICmnd RftCmnd
{
    get
    {
        return RftCmnd;
    }
    set
    {
        RftCmnd = value;
        PrprtChngd ("RftCmnd") ;
    }
}

private ICmnd uploadFytlCmnd;
public ICmnd UploadFytlCmnd
{
    get
    {
        return uploadFytlCmnd;
    }
    set
    {
        uploadFytlCmnd = value;
        PrprtChngd ("UploadFytlCmnd") ;
    }
}

private ICmnd newKtlgCmnd;
public ICmnd NewKtlgCmnd
{
    get
    {
        return newKtlgCmnd;
    }
    set
    {
        newKtlgCmnd = value;
        PrprtChngd ("NewKtlgCmnd") ;
    }
}

```

```

public bool IsPprevtEnabled
{
    get
    {
        return pfd.Any () ;
    }
}

public bool IsNextEnabled
{
    get
    {
        return nfd.Any () ;
    }
}

private bool CanOpen (DcVwMdl fyt1)
{
    return fyt1 == null || !string.IsNullOrEmpty
(fyt1.Id) ;
}

private void Open (DcVwMdl fyt1)
{
    if (fytl != null)
    {
        if (fytl.IsFldr)
        {
            nfd.Clear () ;
            pfd.Push (crTct) ;
            GetFytlsInKtlg (fytl.Id) ;
        }
        else if (!string.IsNullOrEmpty (fytl.Fyt1Name)
&& !string.IsNullOrEmpty (fytl.DownloadUrl))
        {
            var od = new Microsoft.Win32.SaveFyt1Dialog
() { Title = "Зберегти у ", OvrwitPrmpt = true, Fyt1Name =
fytl.Fyt1Name };
            if (od.ShowDialog () .Value)
            {
                DQueue.AddToDQueue (fytl, od.Fyt1Name) ;
            }
        }
    }
}

private bool IfCnUpld (IEnumerable<Fyt1Info> fklk)

```

```

{
    return fklk == null || fklk.Any () ;
}

private void Upload (IEnumerable<FytlInfo> fklk)
{
    if (fklk != null && fklk.Any ())
    {
        fklk.ToList () .ForEach (fytl => AddFytl
(fytl.FullName)) ;
    }
}

private bool IfCanUnBskt (DcVwMdl f)
{
    return f == null || !string.IsNullOrEmpty (f.Id) ;
}

private void rhstoreFromBskt (DcVwMdl f)
{
    if (MessageBox.Show ("Відновити з кошика?",
"Підтвердження", MessageBoxButton.YesNo, MessageBoxImage.Question) ==
MessageBoxResult.Yes)
    {
        if (!f.IsFldr && f.RestoreFromBskt ())
        {
            GetFytlsInKtlg ("trash") ;
        }
    }
}

private void Knfrm (DcVwMdl f)
{
    if (f.IsFldr && string.IsNullOrEmpty (f.Id))
    {
        if (f.Create (crTct))
        {
            GetFytlsInKtlg (crTct) ;
        }
    }
    else
    {
        if (f.UpdateFytl ())
        {
            GetFytlsInKtlg (crTct) ;
        }
    }
}

```

```

private bool CanKncl (DcVwMdl f)
{
    return f == null || f.IsEditing ||
!string.IsNullOrEmpty (f.Id) ;
}

private void Kncl (DcVwMdl f)
{
    if (f != null)
    {
        f.IsEditing = false;
        if (string.IsNullOrEmpty (f.Id))
        {
            Fytl.Remove (fyt1) ;
        }
    }
}

private void GetFytlInKtlg (string fid)
{
    IsCn = true;
    IsBsktView = fid == "trash";
    if (!IsBsktView)
        crtCt = fldrId;

    Task.Factory.StartNew ( (Actnd) ( () =>
    {
        var fklk = new List<DcVwMdl> () ;

        var ss = IsFnd && !string.IsNullOrEmpty (FndTxt)
? FndTxt : null;

        GDGFlsRspns fvtlsRspns = null;
        try
        {
            fvtlsRspns = IsBsktView ?
SrvrStteVwMdl.Current.GetFytlInBskt (ss) :
SrvrStteVwMdl.Current.GetFytlInKtlg (fid, ss) ;
        }
        catch (Exception e)
        {
            System.Windows.MessageBox.Show ("Не вдалося
авторизуватися ") ;
            MyGDApplication.App.InvokeShutDown () ;
        }

        if (fvtlsRspns != null)

```

```

        {
            npthk = fvtlsRspns.NextPageToken;
            fvtlsRspns.Fytls.ToList () .ForEach (f =>
            {
                var ff = new DcVwMdl (f) ;

                ff.OpenCmnd = new DownloadFytlCmnd
(CanOpen, Open) ;
                ff.EditCmnd = new EditFytlCmnd (CanEdit,
Edit) ;
                ff.DeleteCmnd = new DeleteFytlCmnd
(IfCanDelete, Delete) ;
                ff.OkCmnd = new OkFytlCmnd (CanKnfrm,
Knfrm) ;
                ff.KnclCmnd = new KnclFytlCmnd (CanKncl,
Kncl) ;
                ff.RestoreFromBskt Cmnd = new
rhstoreFytlFromBsktCmnd (IfCanUnBskt, rhstoreFromBskt) ;
                fklk.Add (ff) ;
            }) ;

            Fytls = new ObservableKlktn<DcVwMdl> (fklk) ;

            PrprtChngd ("IsPevtEnabled") ;
            PrprtChngd ("IsNextEnabled") ;

            IsCn = false;
        }
    ))) ;
}

private void GoPrev ()
{
    if (pfd.Any ())
    {
        nfd.Push (crTct) ;
        GetFytlsInKtlg (pfd.Pop ()) ;
    }
}

private void GoNext ()
{
    if (nfd.Any ())
    {
        pfd.Push (crTct) ;
        GetFytlsInKtlg (nfd.Pop ()) ;
    }
}

```

```

private void GoRoot ()
{
    pfd.Clear () ;
    nfd.Clear () ;
    GetFytlInKtlg ("root") ;
}

private void GoBskt ()
{
    pfd.Clear () ;
    nfd.Clear () ;
    GetFytlInKtlg ("trash") ;
}

private bool isUpld;
public bool IsUpld
{
    get
    {
        return isUpld;
    }
    set
    {
        isUpld = value;
        PrprtChngd ("IsUpld") ;
    }
}

private bool IsCn;
public bool IsCn
{
    get
    {
        return IsCn;
    }
    set
    {
        IsCn = value;
        PrprtChngd ("IsCn") ;
    }
}

private bool isBsktView;
public bool IsBsktView
{
    get
    {

```

```

        return isBsktView;
    }
    set
    {
        isBsktView = value;
        PrprtChngd ("IsBsktView") ;
    }
}

private bool isFnd;
public bool IsFnd
{
    get
    {
        return isFnd;
    }
    set
    {
        isFnd = value;
        PrprtChngd ("IsFnd") ;

        if (!isFnd)
        {
            FndTxt = null;
            GetFytlInKtlg (IsBsktView ? "trash" : crtCt)
;
        }
    }
}

private string fndTxt;
public string FndTxt
{
    get
    {
        return fndTxt;
    }
    set
    {
        fndTxt = value;
        PrprtChngd ("FndTxt") ;

        if (IsFnd)
        {
            GetFytlInKtlg (IsBsktView ? "trash" : crtCt)
;
        }
    }
}

```

```

    }

    public int uplKnt;
    public int UplKnt
    {
        get
        {
            return uplKnt;
        }

        set
        {
            uplKnt = value;
            PrprtChngd ("UplKnt") ;
        }
    }

    public void RefrthshFldr (string id)
    {
        if (CurrentFldrId.Equals (id))
        {
            GetFytlsInKtlg (CurrentFldrId) ;
        }
    }

    internal static void AddFytl (string sp)
    {
        UploadVwMdl.AddFytl (new ugfytlVwMdl (sp,
CurrentFldrId)) ;
    }
}

public ugfytlsVwMdl (DcsVwMdl mhvm)
{
    InitilizeFytlKlktn () ;
    this.mhvm = mhvm;
}

public ObservableKlktn<UFytlVwMdl> Fytls
{
    get { return fytlcol; }
    set
    {
        fytlcol = value;
        PrprtChngd ("Fytls") ;
    }
}

```

```

public void AddFytl (IEnhmerable<UFytlVwMdl> fhf)
{
    if (fhf!= null && fhf.Any ())
    {
        fytlcol = new ObservableKlktn<UFytlVwMdl>
(fytlcol.Union (fhf)) ;
        RememberToArchive (fhf) ;
        PrprtChngd ("Fytl") ;
        PrprtChngd ("IsUpld") ;
        PrprtChngd ("UplKnt") ;
    }
}

public void AddFytl (ugfytlVwMdl fytl)
{
    fytl.PropertyChanged += ProperChangedHandler;
    tfgs += fytl.Size;
    AddFytl (new[] { fytl }) ;
    UQueue.AddToUQueue (fytl) ;
}

public void RemoveFytl (ugfytlVwMdl fytl)
{
    fytl.PropertyChanged -= ProperChangedHandler;
    if (fytlcol.Remove (fytl))
    {
        ForgetToArchive (fytl) ;
        PrprtChngd ("Fytl") ;
        PrprtChngd ("IsUpld") ;
        PrprtChngd ("UplKnt") ;
    }
}

public void RemoveAll ()
{
    if (fytlcol.Any ())
    {
        fytlcol.Clear () ;
        ForgetAllFytl () ;
        PrprtChngd ("Fytl") ;
        PrprtChngd ("IsUpld") ;
        PrprtChngd ("UplKnt") ;
    }
}

private void ProperChangedHandler (object o,
PropertyChangedEventArgs e)

```

```

        {
            if (e.PropertyName.Equals ("IsUpld") ||
e.PropertyName.Equals ("PercentageUploaded"))
            {
                var frths = o as ugfytlVwMdl;
                if (frths != null)
                {
                    if (fytlcol.Any (f => f.IsUpld))
                    {
                        var st = fytlcol.Sum (f => f.BytesSent)
+ cfhs;
                        mhvm.Up1Knt = (int) ( ( (fvtfloat) st /
tfgs) * 100) ;
                    }

                    if (frths.PercentageUploaded >= 100)
                    {
                        cfhs += frths.BytesSent;
                        RemoveFytl (frths) ;
                        mhvm.RefrthshFldr (frths.Location) ;
                    }
                    else if (!frths.IsUpld)
                    {
                    }
                    mhvm.IsUpld = fytlcol.Any (f => f.IsUpld) ;
                }
            }
        }

        public object TmCnvrt (object v, Type tt, object p,
CultureInfo c)
        {
            DateTime dat;
            var rhs = string.Empty;
            var dayflag = false;

            if (DateTime.TryParse (v.ToString () , out dat))
            {
                var nhm = (DateTime.Now - dt) .TotalDays;
                if (nhm > 365)
                    rhs = (int) (nhm / 365) + " років";
                else if (nhm > 30)
                    rhs = (int) (nhm / 30) + " місяців";
                else if (nhm > 7)
                    rhs = (int) (nhm / 7) + " тижнів";
                else if (nhm > 1)
                {
                    dayflag = true;
                }
            }
        }
    }

```

```

        rhs = dt.DayOfWeek.ToString () ;
    }
    else
    {
        var sknd = (DateTime.Now - dt)
.TotalSkndonds;

        if (sknd > 3600)
            rhs = (int) (sknd / 3600) + " годин";
        else if (skndonds > 60)
            rhs = (int) (sknd / 60) + " хвилин";
        else
            rhs = skndonds < 0 ? "декілька секунд" :
(int) sknd + " секунд";
    }
}

if (res.StartsWith ("1 "))
    rhs = rhs.TrimEnd ('s') ;

return dayflag? rhs : rhs + " тому";
}

private void ProcessCmnd (IAddInClient cl)
{
    if (cl != null && !string.IsNullOrEmpty (Com))
    {
        var fp = client.ActiveDocument;
        if (Fytl.Exists (p))
            cf.ExecuteVerb (Com, p) ;
        else
            MsgBox.Show ("Збережіть спочатку документ ")
;
    }
}

private void InitilizeFytlKlktn ()
{
    if (!Drktr.Exists (up))
    {
        fytlcol = new ObservableKlktn<UFytlVwMdl> () ;
        Drktr.CreateDrktr (up) .Attributes =
FytlAttributes.Drktr | FytlAttributes.Hidden;
    }
    else
    {
        var fs = new List<UFytlVwMdl> () ;

```

```

        Drktr.GetFytlInKtlg (up, "*" + AC) .ToList ()
.ForEach (f =>
    {
        var rCt = Fytl.ReadAllText (f) ;
        f = System.IO.Path.GetFytlName (f) .TrimEnd
(AC.ToCharArray ()) .Replace (BC, "\\") .Replace (CC, ":")
.Replace (up + "\\ ", string.Empty) ;
        var fgi = new ugfytlVwMdl (f, rCt) ;
        fs.Add (fi) ;
    }) ;

        fytlcol = new ObservableKlkt<UFytlVwMdl>
(fs.Distinct ()) ;
    }

    PrprtChngd ("Fytl") ;
}
}
}

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Програмне забезпечення для роботи з віддаленими даними

Виконав: Денис СКИБА

ст. групи КНТ-212сп

Керівник: Михайло КОЦУР

к.т.н., доцент

Рисунок В.1 – Слайд 1

ОБ'ЄКТ, ПРЕДМЕТ ТА МЕТА РОБОТИ

- Об'єкт дослідження – процес розробки інформаційної системи для роботи з віддаленими даними.
- Предмет дослідження – інформаційні системи для роботи з віддаленими даними.
- Метою роботи є розробка програмного забезпечення інформаційної системи для роботи з віддаленими даними.

2

Рисунок В.2 – Слайд 2

ЗАВДАННЯ РОБОТИ

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та інформаційних систем для роботи з віддаленими даними;
- здійснити проєктування програмного забезпечення інформаційної системи для роботи з віддаленими даними;
- створити програмне забезпечення інформаційної системи для роботи з віддаленими даними;
- виконати тестування розробленого програмного забезпечення інформаційної системи для роботи з віддаленими даними.

3

Рисунок В.3 – Слайд 3

ПОРІВНЯННЯ ІСНУЮЧИХ АНАЛОГІВ

Критерій порівняння	WinSCP	BitKinex	Air Explorer
Інтерфейс користувача	+-	+	+
Підтримка хмарних облікових записів	-	-	+
Простота налаштувань	+	+-	-
Підтримка FTP та FTPS	+	+	+-
Підтримка WebDAV	-	-	+
Синхронізація файлів	+-	+-	+

4

Рисунок В.4 – Слайд 4

ПОСТАНОВКА ЗАВДАННЯ

При розробці програмного забезпечення інформаційної системи для роботи з віддаленими даними необхідно забезпечити такі функціональні вимоги:

- підтримка можливості роботи з віддаленим сховищем даних;
- підтримка можливості створення нового каталогу у віддаленому сховищі даних;
- можливість завантаження файлів до віддаленого сховища;
- можливість перейменування документів, що зберігаються у віддаленому сховищі;
- підтримка можливості видалення елементів зі сховища;
- підтримка можливості роботи з кошиком видалених документів;
- можливість пошуку файлів та папок за їх назвами.

5

Рисунок В.5 – Слайд 5

ПОРІВНЯННЯ МОВ ПРОГРАМУВАННЯ

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Продуктивність	+	+—	+
Асинхронне програмування	+	+—	+—
Підтримка хмарних сервісів	+	+	+—
Зручність для створення ПЗ інформаційної системи для роботи з віддаленими даними	+	—	—
Масштабованість та ефективність при великій кількості запитів	+	+—	+

6

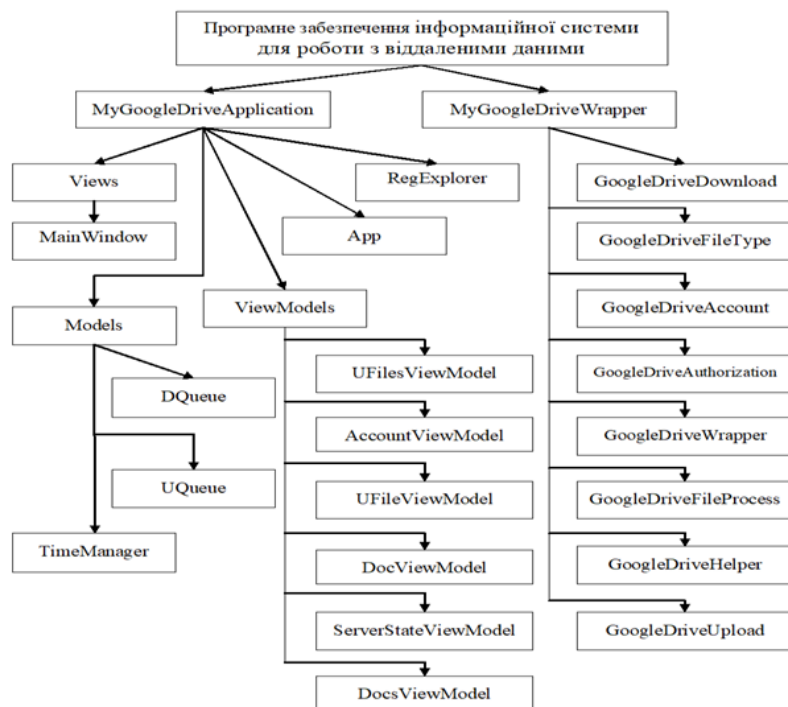
Рисунок В.6 – Слайд 6

ПОРІВНЯННЯ СЕРЕДОВИЩ РОЗРОБКИ

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	PyCharm	Eclipse
Зручність інструментів для тестування та налагодження	+	+-	+
Інтеграція з базами даних	+	+-	+-
Простота використання і навчання	+	+-	-
Продуктивність і швидкість роботи	+	+-	+-
Підтримка контейнеризації	+	+-	+-

7

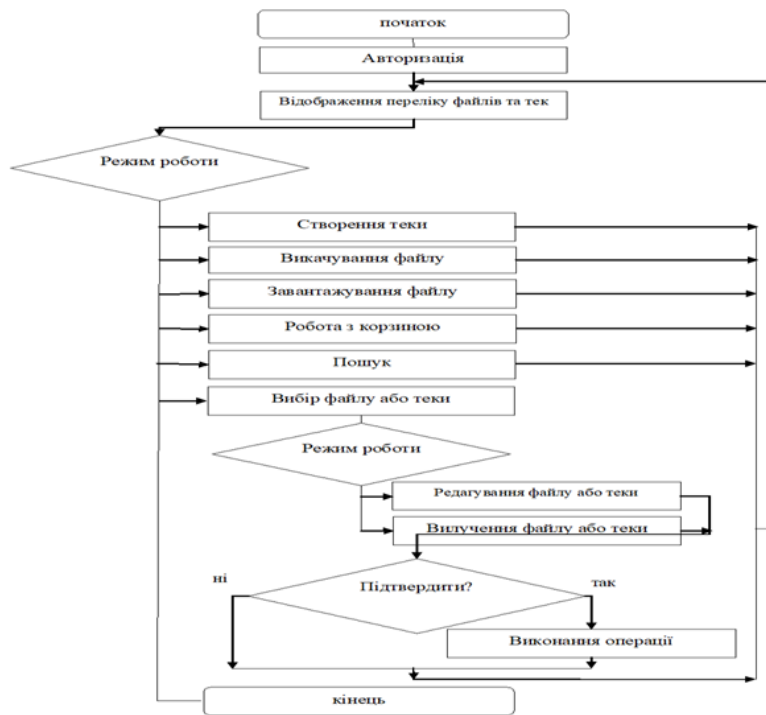
Рисунок В.7 – Слайд 7



СТРУКТУРА ПРОГРАМИ

8

Рисунок В.8 – Слайд 8

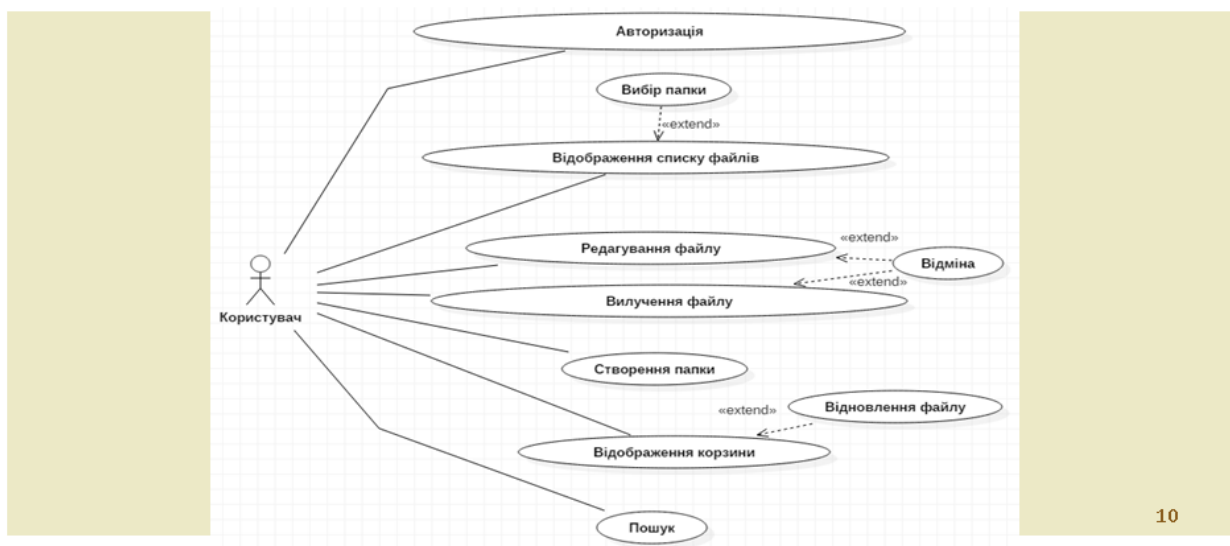


ФУНКЦІОНУВАННЯ ПРОГРАМИ

9

Рисунок В.9 – Слайд 9

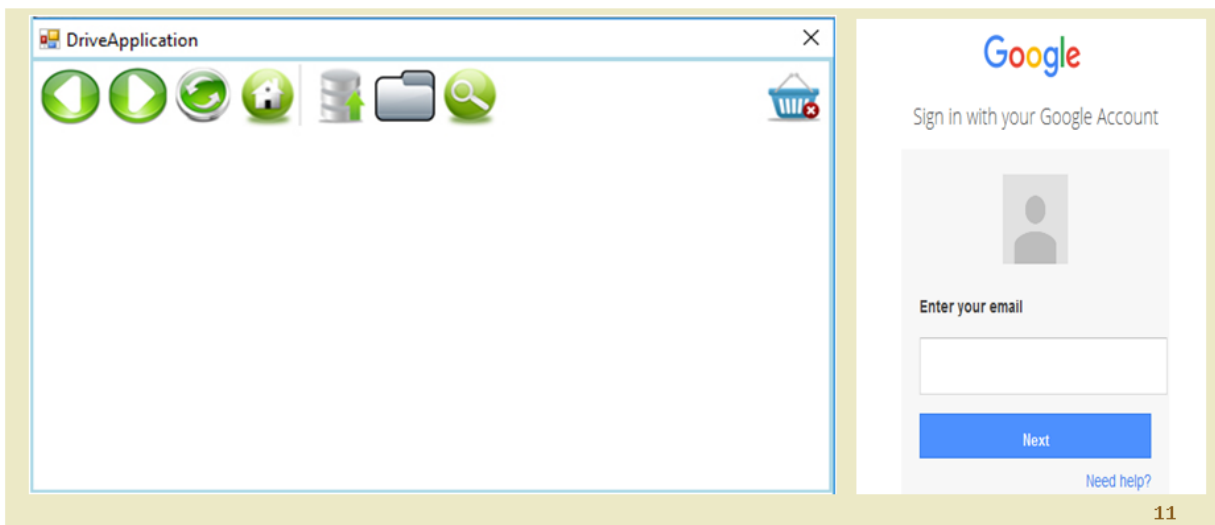
ДІАГРАМА ПРЕЦЕДЕНТІВ



10

Рисунок В.10 – Слайд 10

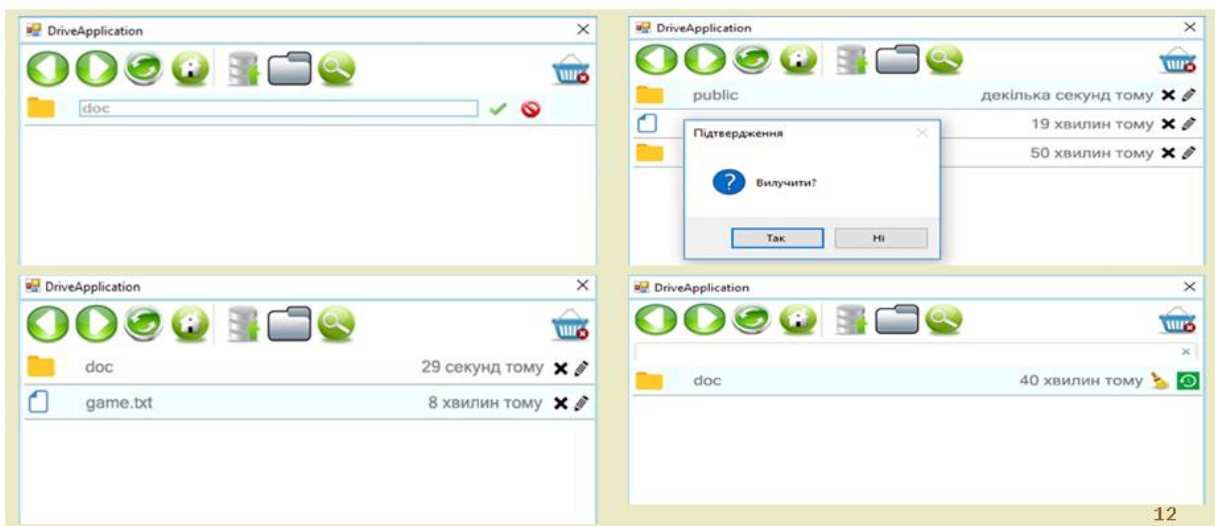
ІНТЕРФЕЙС ЗАСТОСУНКУ



11

Рисунок В.11 – Слайд 11

ІНТЕРФЕЙС ЗАСТОСУНКУ



12

Рисунок В.12 – Слайд 12

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки інформаційної системи для роботи з віддаленими даними.

Сформульовано функціональні вимоги до програмного забезпечення інформаційної системи для роботи з віддаленими даними. Для реалізації програмного забезпечення інформаційної системи для роботи з віддаленими даними обрано мову програмування C# та середовище розробки Visual Studio . Запропоновано структуру програмного забезпечення інформаційної системи для роботи з віддаленими даними. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням інформаційної системи для роботи з віддаленими даними. Виконано тестування розробленого програмного забезпечення інформаційної системи для роботи з віддаленими даними.

13

Рисунок В.13 – Слайд 13