Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

**Бакалаврська дипломна робота на тему:**
«Програмний засіб для резервного копіювання даних»

Виконав: студент 4 курсу групи 1БС–20б
спеціальності 125 Кібербезпека
_____ Олексій ЛЕСЬКО

Керівник: к. т. н. доцент каф. ЗІ
_____ Леонід КУПЕРШТЕЙН
«14» червня 2024 р.

Рецензент: к. т. н. доцент каф. ПЗ
_____ Олександр ХОШАБА
«14» червня 2024 р.

**Допущено до захисту**
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«14» червня 2024 р.

Вінниця ВНТУ – 2024 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти І (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

**ЗАТВЕРДЖУЮ**
**зав. кафедри ЗІ, д. т. н., проф.**
**Володимир ЛУЖЕЦЬКИЙ**
**12 березня 2024 року**

**ЗАВДАННЯ**
**НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ**

Леську Олексію Віталійовичу

1. Тема роботи: «Програмний засіб для резервного копіювання даних»
керівник роботи: Куперштейн Леонід Михайлович, к.т.н., доц. каф. ЗІ.
затверджені наказом ректора ВНТУ №80 від 11 березня 2024 року.

2. Строк подання студентом роботи 14 червня 2024 року.

3. Вихідні дані до роботи:
   - методи резервного копіювання – повне, диференціальне, інкрементальне;
   - тип сховища – локальне, хмарне;
   - операційна система – Windows.

4. Зміст розрахунково-пояснювальної записки: Вступ. Аналіз предметної області. Технічне проектування системи. Робоче проектування системи. Висновки. Перелік використаних джерел. Додатки.

5. Перелік ілюстративного матеріалу: Вигляд архітектури системи (плакат, А4), Структура алгоритму повного резервного копіювання (плакат, А4), Структура алгоритму диференційованого та інкрементного резервного копіювання (плакат, А4), Структура алгоритму відновлення копії (плакат А4), Фрагменти інтерфейсу (плакат А4), Фрагменти тестування (плакат, А4). Акт впровадження (плакат А4).

6. Консультанти розділів роботи:

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1 | Куперштейн Л.М., к.т.н., доц. каф. ЗІ. | 11.03.24 | 31.03.24 |
| 2 | Куперштейн Л.М., к.т.н., доц. каф. ЗІ. | 11.03.24 | 16.04.24 |
| 3 | Куперштейн Л.М., к.т.н., доц. каф. ЗІ. | 11.03.24 | 04.05.24 |

7. Дата видачі завдання 12 березня 2024 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів бакалаврської дипломної роботи | Строк виконання етапів роботи | Пр |
|---|---|---|---|
| 1 | Аналіз завдання. Вступ | 12.03.24 – 14.03.24 | |
| 2 | Аналіз інформаційних джерел за напрямком бакалаврської дипломної роботи | 15.03.24 – 31.03.24 | |
| 3 | Розробка рішень, моделей, алгоритмів | 01.04.24 – 15.04.24 | |
| 4 | Практична реалізація, моделювання, експериментування, результати | 16.04.24 – 01.05.24 | |
| 5 | Оформлення пояснювальної записки | 02.05.24 – 10.05.24 | |
| 6 | Попередній захист БДР | 30.05.24 – 07.06.24 | |
| 7 | Виправлення зауважень, підготовка ілюстративного матеріалу | 10.06.24 – 12.06.24 | |
| 8 | Перевірка на наявність тестових запозичень | 12.06.24 – 13.06.24 | |
| 9 | Представлення БДР до захисту, рецензування | 13.06.24 – 17.06.24 | |
| 10 | Захист БДР | 18.06.24 – 21.06.24 | |

Студент _____ Олексій ЛЕ
( підпис )

Керівник роботи _____ Леонід КУПЕРШ
( підпис )

Бакалавр
є 20 рисунків,
Бакалавр
сучасного зас
проведений
копіювання, ш
збереження
розглянутих
використанні
У рам
реалізований
повного, диф
ефективність
свої особлив
підхід в залеж
Розроб
надійне збер
Його архіте
швидкодії і
ідеальним рі
Ключо
обсяги зберіг

# АНОТАЦІЯ

Бакалаврська дипломна робота складається з 73 сторінок формату А4, на яких є 20 рисунків, 2 таблиці, список використаних джерел містить 35 найменувань.

Бакалаврська робота присвячена глибокому дослідженню та розробці сучасного застосунку для резервного копіювання даних. В ході дослідження був проведений детальний аналіз існуючих технологій та засобів резервного копіювання, що використовуються у різних галузях для забезпечення надійності і збереження даних. Дослідження охопило переваги та недоліки кожної з розглянутих технологій, що дозволило зробити обґрунтований вибір у використанні найбільш підходящих інструментів для реалізації проекту.

У рамках роботи було розроблено засіб для резервного копіювання, реалізований на мові програмування Java. Цей засіб використовує технології повного, диференційного та інкрементного копіювання, що забезпечує високу ефективність та гнучкість у процесі збереження даних. Кожна з цих стратегій має свої особливості і призначення, що дозволяє користувачам вибирати оптимальний підхід в залежності від їхніх потреб і вимог.

Розроблений застосунок для резервного копіювання дозволяє забезпечити надійне збереження даних і відновлення інформації в разі втрати або пошкодження. Його архітектура та алгоритми оптимізовані для забезпечення максимальної швидкодії і ефективності процесів резервного копіювання, що робить його ідеальним рішенням для сучасних систем збереження даних.

Ключові слова: стратегії резервного копіювання, відновлення системи, обсяги зберігання, ресурси.

## ABSTRACT

The bachelor's thesis consists of 73 A4 pages, which have 73 figures, 2 tables, and a list of references containing 35 titles.

The bachelor's thesis is dedicated to in-depth research and development of a modern application for data backup. The research included a detailed analysis of existing backup technologies and tools used across various industries to ensure data reliability and preservation. The study covered the advantages and disadvantages of each technology, allowing for a well-founded selection of the most suitable tools for project implementation.

As part of the thesis, a backup tool was developed using the Java programming language. This tool employs full, differential, and incremental backup technologies, ensuring high efficiency and flexibility in data storage processes. Each of these strategies has its own characteristics and purposes, enabling users to choose the optimal approach based on their needs and requirements.

The developed backup application enables reliable data storage and information recovery in case of loss or damage. Its architecture and algorithms are optimized for maximum speed and efficiency in backup processes, making it an ideal solution for modern data storage systems.



Keywords: backup strategies, system recovery, storage volumes, resources.

**ЗМІСТ**

# ВСТУП

У сучасному світі обсяги цифрової інформації стрімко зростають, що робить її збереження і захист важливими завданнями для будь-якої організації та індивідуального користувача. Втрата даних через збої апаратного забезпечення, помилки програмного забезпечення або зловмисні дії може мати серйозні наслідки. Тому забезпечення надійного резервного копіювання даних є критично важливим аспектом інформаційної безпеки.

Однією з ключових проблем є швидкість та ефективність процесу резервного копіювання. Великі обсяги даних потребують значних ресурсів для їх обробки та збереження, що може вплинути на продуктивність системи та зайняти багато часу. Сучасні технології резервного копіювання постійно вдосконалюються, однак залишається проблема забезпечення швидкого і надійного відновлення даних у разі втрати.

Безпека даних під час резервного копіювання також є важливим питанням. Дані повинні бути захищені від несанкціонованого доступу та зловмисних дій як під час передачі, так і під час зберігання. Відповідно, програмне забезпечення для резервного копіювання повинно включати надійні механізми шифрування та автентифікації, щоб гарантувати безпеку даних.

Простота використання програмного забезпечення для резервного копіювання є ще одним важливим аспектом. Користувачі повинні мати можливість легко налаштовувати та керувати процесом резервного копіювання без необхідності глибоких технічних знань. Автоматизація процесів резервного копіювання допомагає мінімізувати ризики людських помилок і забезпечує безперервність процесу.

Розробка програмних засобів, які забезпечують автоматизацію процесу резервного копіювання, мінімізацію ризиків втрати даних і зручність для користувача, є актуальною темою дослідження. Вдосконалення таких засобів сприятиме підвищенню надійності та ефективності збереження інформації, що є надзвичайно важливим у сучасному інформаційному середовищі.

Загалом, забезпечення цілісності та доступності даних через їх резервне копіювання є невід'ємною частиною стратегії інформаційної безпеки будь-якої організації. Наукові дослідження та практичні розробки в цій сфері мають великий потенціал для покращення існуючих рішень та розробки нових методів і засобів захисту даних.

Об’єктом бакалаврської дипломної роботи є процеси резервного копіювання.

Предметом бакалаврської дипломної роботи є методи та засоби резервного копіювання.

Метою бакалаврської роботи є забезпечення цілісності та доступності даних за рахунок їх резервного копіювання.

Для досягнення мети були поставлені наступні задачі:

- проаналізувати сучасні методи та програмні засоби для резервного копіювання даних;
- здійснити розробку архітектуру програмного засобу;
- реалізувати програмний засіб;
- провести тестування програмного засобу на різних сценаріях використання.

Результати з поведеної роботи стосовно аналізу стратегій резервного копіювання даних доповідалися на LII міжнародній науково-практичній інтернет-конференції студентів аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи» Вінницького національного технічного університету (Вінниця, 2024 р.) [1].

Таким чином, робота спрямована на розробку сучасного програмного засобу для резервного копіювання даних, який би забезпечував високу надійність, безпеку та зручність у використанні, що є важливим аспектом інформаційної безпеки у сучасному цифровому світі.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Аналіз сучасних загроз втрати даних

У сучасному світі інформація є одним із найцінніших ресурсів, тому її збереження та захист стають пріоритетними завданнями для будь-якої організації чи приватної особи. Втрата даних може призвести до значних фінансових втрат, порушення бізнес-процесів, а інколи навіть до репутаційних ризиків. Тому важливо розглянути основні загрози, які можуть призвести до втрати даних, та методи їх подолання (рис. 1.1) [2].


Рисунок 1.1 – Загрози втрати даних та їх причини

Даний рисунок наочно демонструє основні загрози втрати даних та їх причини. Тепер розглянемо кожну з цих загроз детальніше, щоб зрозуміти їхній вплив на безпеку даних та ефективні способи захисту.

Апаратні збої є однією з найпоширеніших причин втрати даних [3]. Це можуть бути збої жорстких дисків, проблеми з оперативною пам'яттю, збої живлення, а також вихід з ладу інших апаратних компонентів комп'ютера чи серверного обладнання. Незважаючи на розвиток технологій і підвищення надійності сучасного обладнання, апаратні збої залишаються значною загрозою

для збереження даних. Регулярне технічне обслуговування та моніторинг стану обладнання можуть знизити ризик втрати даних через апаратні збої.

Людські помилки можуть призвести до випадкового видалення файлів, форматування жорстких дисків або неправильного налаштування систем. Навіть висококваліфіковані спеціалісти можуть допускати помилки, особливо у стресових ситуаціях або при роботі з великими обсягами даних [3].

Людський фактор залишається одним із основних ризиків, що вимагає додаткових заходів для мінімізації. Використання чітких інструкцій, регулярне навчання персоналу та запровадження систем автоматизованого резервного копіювання можуть значно зменшити ризик втрат через людські помилки.

Віруси, трояни, програми-вимагачі та інше шкідливе програмне забезпечення можуть спричиняти серйозні втрати даних. Програми-вимагачі, наприклад, шифрують дані користувача і вимагають викуп за розшифрування. Вірусні атаки можуть призводити до пошкодження файлів, крадіжки конфіденційної інформації або повного знищення даних [3].

Використання антивірусного програмного забезпечення, регулярні оновлення системи та обізнаність користувачів щодо безпечного використання інтернету є важливими заходами для захисту від шкідливих програм.

Природні катастрофи, такі як пожежі, повені, землетруси, можуть знищити фізичні носії даних та обладнання. Такі події є менш ймовірними, але їх наслідки можуть бути катастрофічними для організацій, які не мають належних планів резервного копіювання та відновлення даних [3].

Для захисту від природних катастроф важливо зберігати резервні копії в географічно віддалених місцях, використовувати хмарні сервіси для збереження даних, а також розробити та впровадити плани відновлення після катастроф.

Кібератаки стають дедалі більш поширеними і складними. Хакери можуть здійснювати атаки з метою крадіжки, знищення або зміни даних. Використовуючи соціальну інженерію, фішингові атаки, експлойти та інші методи, зловмисники можуть отримати доступ до систем і даних, завдаючи значної шкоди [3].

Для захисту від кібератак важливо впроваджувати багатофакторну автентифікацію, використовувати сучасні засоби захисту мережі та систем, а також проводити регулярні перевірки безпеки.

Внутрішні загрози, включаючи дії недобросовісних працівників або ненавмисні помилки співробітників, також можуть призвести до втрати або компрометації даних. Працівники можуть випадково видалити важливі файли або намагатися вкрасти конфіденційну інформацію. Впровадження політик безпеки, обмеження доступу до критичних даних, а також моніторинг активності користувачів можуть допомогти знизити ризики, пов'язані з внутрішніми загрозами.

Історія знає чимало випадків, коли втрати даних мали значний вплив на організації та людей. Ось декілька відомих прикладів:

- Влітку 2017 року вірус-шифрувальник Petya (відомий також як NotPetya) завдав серйозного удару по українським компаніям та державним установам. Цей вірус поширювався через мережі, використовуючи вразливості в програмному забезпеченні, і шифрував дані на комп'ютерах, роблячи їх недоступними. Серед постраждалих були банки, енергетичні компанії, державні установи, медіа, а також аеропорти та навіть метро [4];
- У 1998 році студія Pixar ледь не втратила всі дані анімаційного фільму Toy Story 2. Через помилку один із співробітників випадково видалив більшість файлів фільму. Врятувати проект вдалося завдяки тому, що технічний директор зберігав резервну копію вдома [5];
- У 2013 році Yahoo зазнала однієї з найбільших втрат даних в історії. Було зламано близько 3 мільярдів облікових записів користувачів, що призвело до компрометації особистих даних, включаючи імена, електронні адреси та паролі [6];
- У 2017 році Equifax, одна з найбільших кредитних агенцій США, зазнала масштабного злому, внаслідок якого було викрадено особисті дані 147

мільйонів людей, включаючи номери соціального страхування, дати народження та іншу конфіденційну інформацію [7];

- У 2019 році MySpace оголосила про втрату всіх медіафайлів, завантажених користувачами до 2016 року, через помилку під час перенесення серверів. Це призвело до зникнення мільйонів пісень, фотографій та відео [8];
- У 2021 році у французькому дата-центрі OVH сталася пожежа, що знищила сервери і призвело до втрати даних багатьох клієнтів. Багато компаній зазнали значних втрат через відсутність резервних копій [9].

Ці випадки підкреслюють, що необхідність надійного програмного забезпечення для резервного копіювання даних є критично важливою. Розробка такого програмного засобу дозволить зменшити ризики втрати даних та забезпечити безперервність бізнес-процесів. Програмний засіб для резервного копіювання повинен забезпечувати автоматичне створення резервних копій, зберігання копій у безпечних місцях, а також швидке і зручне відновлення даних у разі втрати.

### 1.2 Технології та методи резервного копіюванням

Резервне копіювання є одним з ключових компонентів забезпечення безпеки інформації, адже воно дозволяє відновлювати дані у випадку їх втрати або пошкодження. Існують різні технології та методи резервного копіювання, кожен з яких має свої особливості, переваги та недоліки.

Ручне копіювання даних є одним із найпростіших і найбільш базових методів збереження інформації [10]. Воно полягає у ручному перенесенні файлів і папок з одного носія на інший, наприклад, з комп'ютера на зовнішній жорсткий диск, флешку або інший пристрій зберігання даних. Цей метод не потребує спеціалізованого програмного забезпечення і може бути виконаний будь-яким користувачем з базовими знаннями комп'ютера.

Основною перевагою ручного копіювання є його простота і доступність [10]. Користувачі можуть самостійно вибирати, які файли і папки копіювати, і коли це робити. Це дозволяє швидко створювати резервні копії важливих документів без необхідності в складних налаштуваннях або витратах на програмне забезпечення. Крім того, цей метод дозволяє точно контролювати процес резервного копіювання, забезпечуючи копіювання лише тих даних, які дійсно важливі.

Однак ручне копіювання даних має свої недоліки [10]. Головним з них є ризик людської помилки. Користувачі можуть забути зробити копію, помилково пропустити важливі файли або випадково видалити дані під час процесу копіювання. Це може призвести до втрати важливої інформації, особливо якщо копіювання не виконується регулярно. Крім того, процес ручного копіювання може бути трудомістким і вимаг

Повне резервне копіювання є одним з найстаріших і найбільш надійних методів збереження даних [11]. Воно передбачає створення копії всіх даних на носії в певний момент часу. Весь вміст системи, включаючи файли, програми та системні налаштування, зберігається в окремому архіві. Завдяки цьому методу можна легко і швидко відновити систему до стану на момент створення резервної копії, що робить його незамінним для критичних даних і систем.

Основна перевага повного резервного копіювання полягає у високій надійності і простоті відновлення даних [11]. У випадку втрати або пошкодження інформації, користувачеві потрібно лише відновити останню повну резервну копію, що значно зменшує час на відновлення і мінімізує ризик втрати даних. Оскільки вся інформація зберігається в одному архіві, це також спрощує управління резервними копіями, дозволяючи легко перевіряти їх цілісність і консистентність.

Однак повне резервне копіювання має й свої недоліки. Головним з них є значний обсяг даних, який потрібно зберігати. Оскільки кожна резервна копія містить повний набір даних, дисковий простір швидко заповнюється, особливо у випадку великих обсягів інформації. Це може призводити до значних витрат на зберігання, особливо якщо використовуються хмарні сервіси. Крім того, процес

створення повної резервної копії потребує значного часу і ресурсів системи. Під час резервного копіювання система може сповільнюватися, що негативно впливає на продуктивність, особливо у великих організаціях із значними обсягами даних.

Диференціальне резервне копіювання – це метод збереження даних, який зменшує обсяг збереженої інформації та прискорює процес резервування. Цей метод передбачає створення копій лише тих даних, які були змінені з моменту останнього повного резервного копіювання. Таким чином, кожна диференціальна резервна копія містить усі зміни, внесені після створення останньої повної резервної копії. Це дозволяє значно зменшити обсяг даних, що зберігаються, порівняно з повним резервним копіюванням [12].

Головною перевагою диференціального резервного копіювання є скорочення часу, необхідного для створення резервної копії. Оскільки копіюються лише змінені дані, процес резервного копіювання займає менше часу, що знижує навантаження на систему і дозволяє виконувати резервне копіювання частіше. Це особливо важливо для систем, які потребують частих оновлень і збереження даних у реальному часі.

Недоліками диференціального резервного копіювання є те, що обсяг даних, який потрібно зберігати, зростає з кожним новим резервним копіюванням, доки не буде створено нову повну резервну копію [12].

З часом розмір диференціальних копій може значно збільшуватися, наближаючись до обсягу повної резервної копії. В результаті, хоча кожна окрема диференціальна копія зберігає менше даних, сукупний обсяг збережених даних може бути значним, що вимагає додаткових ресурсів для зберігання.

Інкрементальне резервне копіювання є одним з найбільш ефективних методів збереження даних з точки зору обсягу збережених даних та часу, необхідного для створення резервної копії [13]. Основний принцип інкрементального копіювання полягає у створенні копій лише тих даних, які були змінені з моменту останнього резервного копіювання, незалежно від того, було це повне, диференціальне чи інше інкрементальне копіювання. Це дозволяє значно зменшити обсяг збережених даних, оскільки зберігаються тільки нові або змінені файли.

Перевагою інкрементального резервного копіювання є його висока ефективність. Порівняно з повним резервним копіюванням, інкрементальне копіювання потребує набагато менше дискового простору, що робить його особливо привабливим для організацій із великими обсягами даних. Крім того, процес створення інкрементальних копій значно швидший, оскільки копіюються тільки ті файли, які були змінені, що зменшує навантаження на систему та мінімізує час простою.

Іншим важливим аспектом є зменшення часу на створення резервних копій. Оскільки інкрементальне копіювання потребує обробки меншої кількості даних, процес створення копій може виконуватися частіше, що забезпечує більш актуальні резервні копії. Це особливо важливо для критичних систем, де втрата навіть невеликої кількості даних може мати серйозні наслідки [13].

Проте, інкрементальне резервне копіювання має й свої недоліки. Одним з головних є складність і тривалість процесу відновлення даних. Для відновлення повного набору даних необхідно мати останню повну резервну копію, а також всі інкрементальні копії, створені після неї. Це означає, що процес відновлення може бути досить тривалим, оскільки потрібно обробити велику кількість копій. Крім того, будь-яка помилка або відсутність однієї з інкрементальних копій може призвести до втрати даних або неможливості їх відновлення.

Після детального аналізу чотирьох основних технологій резервного копіювання стає очевидним, що кожна з них має свої переваги та недоліки, які потрібно враховувати під час вибору відповідного методу для конкретної організації чи завдання. Для кращого розуміння відмінностей між цими технологіями, наведемо порівняльну таблицю (табл. 1.1), яка допоможе візуально оцінити ключові характеристики кожного методу.

Таблиця 1.1 – Порівняльна таблиця характеристик технологій резервного копіювання

| Характеристика | Повне резервне копіювання | Диференціальне резервне копіювання | Інкрементальне резервне копіювання | Ручне копіювання |
|---|---|---|---|---|
| Обсяг збережених даних | Дуже великий, всі дані зберігаються в кожній копії | Менший, ніж у повному, лише змінені дані з моменту останнього повного копіювання | Найменший, тільки змінені дані з моменту останнього будь-якого копіювання | Залежить від обсягу даних, що копіюються вручну |
| Час створення копії | Дуже тривалий, потребує багато часу | Середній, швидше ніж повне, але залежить від обсягу змінених даних | Швидкий, створює копії лише змінених даних | Залежить від обсягу даних і швидкості користувача |
| Час відновлення даних | Швидкий, всі дані в одній копії | Середній, потребує останню повну копію і всі диференціальні | Тривалий, потребує всі інкрементальні копії | Залежить від організації і швидкості користувача |
| Надійність | Висока, всі дані зберігаються в одній копії | Висока, але залежить від наявності останньої повної копії | Висока, але залежить від наявності всіх інкрементальних копій | Низька, ризик людської помилки |
| Витрати на зберігання | Високі, потребує багато дискового простору | Середні, менший обсяг даних порівняно з повним копіюванням | Низькі, зберігає тільки змінені дані | Низькі, залежить від наявних носіїв даних |
| Вплив на продуктивність | Високий, потребує багато системних ресурсів | Середній, менший вплив порівняно з повним копіюванням | Низький, мінімальний вплив на систему | Низький, але залежить від користувача |
| Масштабованість | Обмежена, потребує додаткового обладнання | Середня, потребує більше місця для збереження даних | Висока, ефективне зберігання змінених даних | Обмежена, залежить від фізичних носіїв даних |
| Безпека даних | Висока, залежить від фізичної безпеки зберігання | Висока, залежить від фізичної безпеки зберігання | Висока, залежить від фізичної безпеки зберігання | Низька, залежить від фізичної безпеки носіїв |
| Автоматизація | Обмежена, потребує ручного налаштування | Середня, можливість автоматизації створення копій | Висока, легко автоматизується | Низька, ризик людської помилки |
| Доступність | Локальна, дані зберігаються на фізичних носіях | Локальна, дані зберігаються на фізичних носіях | Локальна, дані зберігаються на фізичних носіях | Низькі, залежить від наявних носіїв даних |

Вибір конкретного методу резервного копіювання залежить від вимог до безпеки, обсягу даних, доступного дискового простору та інших факторів. У деяких випадках доцільно використовувати комбіновані підходи, що включають кілька методів резервного копіювання для досягнення оптимальної надійності та ефективності.

Таким чином, технології та методи резервного копіювання є важливими компонентами забезпечення безпеки даних. Кожен метод має свої переваги та недоліки, і вибір конкретного методу залежить від конкретних потреб користувача або організації. Важливо враховувати різні фактори, такі як обсяг даних, вимоги до безпеки, доступний дисковий простір та інші, щоб забезпечити оптимальну надійність та ефективність резервного копіювання.

### 1.3 Типи сховищ для резервного копіювання

Резервне копіювання даних є критично важливим для забезпечення їх збереження та відновлення у разі втрат. Існує кілька типів сховищ для резервного копіювання, які можна поділити на локальні, мережеві та хмарні.

Локальні сховища включають жорсткі диски (HDD), твердотільні накопичувачі (SSD), стрічкові накопичувачі (Tape), Blu-ray диски та флеш-носії [14].

Жорсткі диски (HDD) забезпечують велику ємність і низьку вартість за гігабайт. Вони популярні для зберігання великих обсягів даних, але мають рухомі частини, що підвищує ризик фізичних пошкоджень і зношування.

Твердотільні накопичувачі (SSD) пропонують високу швидкість доступу до даних і стійкість до фізичних ударів завдяки відсутності рухомих частин. Вони дорожчі за HDD при однаковій ємності, але мають більшу надійність і триваліший термін служби.

Стрічкові накопичувачі (Tape) використовуються для довгострокового зберігання великих обсягів даних. Вони мають низьку вартість зберігання та високу надійність, але доступ до даних значно повільніший порівняно з HDD або SSD.

Blu-ray диски забезпечують значний обсяг зберігання (до 128 ГБ на диск) та високу надійність для довготривалого зберігання. Вони підходять для архівування великих файлів, таких як відео та фотографії, але процес запису даних тривалий, і ємність обмежена.

Флеш-носії (USB-флешки, SD-карти) є портативними і зручними для переносу та резервного копіювання даних. Вони швидкі, але менш надійні для довготривалого зберігання через обмежену кількість циклів запису та можливі фізичні пошкодження.

Технологія RAID (Redundant Array of Independent Disks) використовується для об'єднання кількох жорстких дисків або SSD з метою підвищення надійності, продуктивності та об'єму зберігання даних [15]. Існує кілька конфігурацій RAID, кожна з яких має свої особливості та переваги.

RAID 0 розподіляє дані між кількома дисками, що підвищує швидкість доступу до інформації, оскільки дані записуються і зчитуються паралельно. Проте цей режим не забезпечує захисту даних: вихід з ладу одного диска призводить до втрати всієї інформації. RAID 1 використовує дзеркальне копіювання, коли дані дублюються на два або більше дисків. Це забезпечує високий рівень надійності, оскільки у разі виходу з ладу одного диска дані залишаються доступними на іншому. Однак такий підхід вимагає вдвічі більше дискового простору.

RAID 5 розподіляє дані та паритетну інформацію між трьома або більше дисками. Це дозволяє відновлювати дані у разі виходу з ладу одного диска, поєднуючи надійність з ефективним використанням дискового простору. RAID 6 схожий на RAID 5, але використовує два паритетні блоки, що дозволяє витримувати одночасний вихід з ладу двох дисків, підвищуючи надійність системи.

RAID 10 комбінує переваги RAID 0 і RAID 1, поєднуючи швидкість доступу та високий рівень надійності. Дані розподіляються між кількома дзеркальними наборами дисків, що забезпечує паралельний доступ та резервування. Водночас ця конфігурація вимагає значних витрат на дисковий простір.

RAID масиви можуть бути апаратними або програмними. Апаратні RAID масиви реалізуються за допомогою спеціальних контролерів, що забезпечує вищу

продуктивність і менше навантаження на центральний процесор. Програмні RAID масиви створюються за допомогою операційної системи, що може бути більш економічним варіантом, але з меншим рівнем продуктивності.

Вибір конфігурації RAID залежить від конкретних потреб у швидкості доступу до даних, надійності та вартості. RAID масиви широко використовуються у бізнесі, дата-центрах та системах, де критично важливим є забезпечення безперебійного доступу до інформації та захист даних від втрат.

NAS (Network Attached Storage) і SAN (Storage Area Network) є двома основними типами мережевих сховищ для резервного копіювання та зберігання даних [16].

NAS – це спеціалізований пристрій, підключений до локальної мережі, що надає централізований доступ до даних для декількох користувачів або пристроїв [16]. NAS функціонує як автономний сервер, керуючи доступом до файлів і забезпечуючи їхнє резервне копіювання. Він використовує стандартні мережеві протоколи, такі як NFS, SMB/CIFS та FTP, що робить його сумісним з різними операційними системами. NAS легко налаштовується та управляється, що робить його популярним рішенням для малого та середнього бізнесу, а також для домашніх користувачів, які потребують спільного доступу до файлів. Основні переваги NAS включають простоту використання, відносно низьку вартість та можливість масштабування шляхом додавання додаткових дисків.

SAN – це високошвидкісна спеціалізована мережа, яка надає доступ до блокового зберігання даних [17]. SAN використовує протоколи, такі як Fibre Channel або iSCSI, що забезпечують швидкий і надійний доступ до даних на рівні блоків. Це рішення переважно використовується в великих підприємствах і дата-центрах, де потрібна висока продуктивність, низька затримка та масштабованість. SAN дозволяє підключати безліч серверів до спільного пулу зберігання, що спрощує управління великими обсягами даних та забезпечує високу доступність і відмовостійкість. Основні переваги SAN включають високу швидкість доступу до даних, можливість обробки великих навантажень та підтримку складних корпоративних додатків.

Хмарні сховища забезпечують віддалене зберігання даних на серверах, що управляються сторонніми провайдерами. Вони включають такі популярні рішення, як Amazon S3, Google Cloud Storage та Microsoft Azure.

Amazon S3 (Simple Storage Service) пропонує масштабоване сховище для будь-якого обсягу даних з можливістю доступу з будь-якого місця. Amazon S3 забезпечує високу доступність, безпеку і можливості резервного копіювання та відновлення. Користувачі можуть автоматично архівувати дані на Glacier для довготривалого зберігання з низькою вартістю [18].

Google Cloud Storage надає високонадійне і масштабоване сховище з інтеграцією з іншими сервісами Google Cloud. Він підтримує автоматичне резервне копіювання, зберігання об'єктів різних типів і доступ до даних з низькою затримкою. Google Cloud Storage також пропонує різні класи зберігання, що дозволяє оптимізувати витрати залежно від потреб [19].

Microsoft Azure Storage пропонує різні рішення для зберігання даних, включаючи Blob Storage, File Storage та Disk Storage. Azure забезпечує високу доступність і надійність, а також інтеграцію з іншими сервісами Microsoft. Azure Storage підтримує функції резервного копіювання, архівування та відновлення даних [20].

Основні переваги хмарних сховищ включають легке масштабування, доступність з будь-якої точки світу, високу безпеку та автоматичне резервне копіювання. Недоліки полягають у необхідності стабільного інтернет-з'єднання, можливих високих витратах на зберігання та операції, а також занепокоєнні щодо конфіденційності даних на серверах сторонніх провайдерів.

Таким чином, вибір типу сховища для резервного копіювання залежить від конкретних вимог до швидкості доступу, надійності, вартості та обсягу даних, які необхідно зберігати. Локальні сховища підходять для більш швидкого доступу та контролю над даними, мережеві сховища забезпечують централізований доступ та додаткову надійність, тоді як хмарні сховища забезпечують віддалене зберігання та високу доступність. Технологія RAID може бути інтегрована як у локальні, так і у мережеві сховища для підвищення їх надійності та продуктивності.

### 1.4 Аналіз засобів резервного копіювання даних

У сучасних умовах резервне копіювання даних є критично важливим для забезпечення безперебійної роботи як підприємств, так і окремих користувачів. Зі зростанням обсягу даних та складності інформаційних систем на ринку з'являється все більше рішень для резервного копіювання, кожне з яких має свої переваги та недоліки.

EaseUS Todo Backup є популярним інструментом для резервного копіювання, відомим своєю зручністю у використанні та різноманітними можливостями [21]. Програма підтримує повне, диференційоване та інкрементне резервне копіювання, що дозволяє користувачам вибирати найбільш підходящий тип резервного копіювання для своїх потреб. Вона також має функції синхронізації файлів, що забезпечує актуальність резервних копій без зайвих зусиль.

EaseUS Todo Backup також пропонує можливість створення завантажувального диска, що є корисним у випадках відновлення системи після серйозних збоїв. Крім того, програма дозволяє здійснювати резервне копіювання на різні носії, включаючи локальні диски, зовнішні жорсткі диски та мережеві сховища. Зручність у налаштуванні та можливість автоматизації процесів резервного копіювання роблять її ідеальним вибором для домашніх користувачів та малого бізнесу.

Acronis True Image відомий своєю комплексністю та багатофункціональністю, пропонуючи рішення для резервного копіювання та відновлення, які підходять як для індивідуальних користувачів, так і для бізнесу [22]. Програма підтримує резервне копіювання як файлів, так і образів дисків, забезпечуючи повну захисту даних. Вона також включає функції активного захисту від програм-вимагачів та криптозахисту, що додає додатковий рівень безпеки для користувачів.

Однією з ключових переваг Acronis True Image є її здатність до клонування дисків та міграції систем, що робить її ідеальною для оновлення або заміни жорстких дисків без втрати даних. Програма також підтримує резервне копіювання

в хмарні сховища, що забезпечує додатковий рівень захисту від фізичних пошкоджень або втрати пристроїв.

Macrium Reflect відомий своєю надійністю та високою швидкістю роботи [23]. Програма пропонує повне, диференційоване та інкрементне резервне копіювання, дозволяючи користувачам вибирати найбільш ефективний метод захисту даних. Одна з головних переваг Macrium Reflect – це її здатність створювати образи дисків, що дозволяє легко відновлювати всю систему у випадку збою або пошкодження диска. Крім того, програма підтримує створення завантажувальних носіїв, що забезпечує можливість відновлення системи навіть у випадку, коли операційна система не завантажується.

Macrium Reflect також відзначається своїм інтуїтивно зрозумілим інтерфейсом та широкими можливостями налаштування. Користувачі можуть налаштувати автоматичні резервні копії на певні дні або часи, що робить процес резервного копіювання менш трудомістким і більш зручним. Програма підтримує збереження резервних копій на різні носії, включаючи локальні диски, зовнішні жорсткі диски та мережеві сховища. Завдяки своїй надійності та продуктивності, Macrium Reflect є одним з найкращих виборів для користувачів, які шукають ефективне рішення для резервного копіювання.

AOMEI Backupper є однією з найбільш популярних програм для резервного копіювання завдяки своїй доступності та широкому спектру функцій [34]. Програма підтримує повне, диференційоване та інкрементне резервне копіювання, що дозволяє користувачам вибирати найбільш підходящий метод захисту даних. Однією з головних переваг AOMEI Backupper є її здатність створювати резервні копії на різні носії, включаючи локальні диски, зовнішні жорсткі диски та мережеві сховища. Програма також підтримує синхронізацію файлів та клонування дисків, що робить її ідеальною для оновлення або заміни жорстких дисків.

AOMEI Backupper також відзначається своєю простотою у використанні та інтуїтивно зрозумілим інтерфейсом. Користувачі можуть налаштувати автоматичні резервні копії на певні дні або часи, що робить процес резервного копіювання менш трудомістким і більш зручним. Крім того, програма включає функції перевірки

резервних копій на наявність помилок, створення завантажувальних носіїв та можливість відновлення системи у випадку серйозних збоїв. Завдяки своїм широким можливостям та надійності, AOMEI Backupper є відмінним вибором для домашніх користувачів та малого бізнесу.

Paragon Backup & Recovery пропонує гнучке рішення для резервного копіювання та відновлення даних, яке підходить як для домашніх користувачів, так і для бізнесу [25]. Програма підтримує повне, диференційоване та інкрементне резервне копіювання, що дозволяє користувачам вибирати найбільш підходящий метод захисту даних. Однією з головних переваг Paragon Backup & Recovery є її здатність створювати завантажувальні носії, що забезпечує можливість відновлення системи навіть у випадку, коли операційна система не завантажується. Крім того, програма підтримує створення складних резервних копій, що включає автоматичне резервне копіювання та планування.

Paragon Backup & Recovery також відзначається своєю простотою у використанні та інтуїтивно зрозумілим інтерфейсом. Користувачі можуть налаштувати резервні копії на певні дні або часи, що робить процес резервного копіювання менш трудомістким і більш зручним. Програма підтримує збереження резервних копій на різні носії, включаючи локальні диски, зовнішні жорсткі диски та мережеві сховища. Завдяки своїй надійності та продуктивності, Paragon Backup & Recovery є одним з найкращих виборів для користувачів, які шукають ефективне рішення для резервного копіювання.

Genie Timeline пропонує просте та ефективне рішення для резервного копіювання, яке відзначається своєю здатністю працювати у фоновому режимі без значного впливу на продуктивність системи [26]. Програма має інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам легко налаштовувати резервні копії та відновлення даних. Однією з головних переваг є режим Turbo, який дозволяє прискорити процес резервного копіювання, а також режим Smart, який адаптує швидкість резервного копіювання відповідно до наявних системних ресурсів. Це робить Genie Timeline ідеальним вибором для користувачів, які цінують простоту та ефективність.

NovaBACKUP є надійним інструментом для резервного копіювання, який пропонує повний спектр функцій для захисту даних [27]. Програма підтримує резервне копіювання як файлів, так і образів дисків, що дозволяє користувачам легко відновлювати свої системи у випадку збою. Однією з головних переваг NovaBACKUP є її здатність до шифрування даних, що забезпечує додатковий рівень безпеки. Програма також пропонує різні варіанти резервного копіювання, включаючи локальні диски, зовнішні жорсткі диски та хмарні сховища, що робить її універсальним рішенням для будь-яких потреб.

Comodo Backup є потужним інструментом для резервного копіювання, який пропонує широкий спектр можливостей для захисту даних [28]. Програма підтримує різні види резервного копіювання, включаючи повне, диференційоване та інкрементне, що дозволяє користувачам вибирати найбільш підходящий метод захисту своїх даних. Однією з ключових переваг Comodo Backup є її здатність до резервного копіювання в хмарні сховища, що забезпечує додатковий рівень захисту та доступність даних з будь-якого пристрою.

Cobian Backup є безкоштовним інструментом для резервного копіювання, який відомий своєю простотою у використанні та широкими можливостями [29]. Програма підтримує різні види резервного копіювання, включаючи повне, диференційоване та інкрементне, що дозволяє користувачам вибирати найбільш підходящий метод захисту своїх даних. Однією з головних переваг Cobian Backup є її здатність до автоматизації процесів резервного копіювання, що дозволяє користувачам налаштовувати резервні копії на певні дні або часи без необхідності ручного управління процесом.

FBackup є безкоштовним інструментом для резервного копіювання, який відомий своєю зручністю у використанні та широкими можливостями [30]. Програма підтримує повне та дзеркальне резервне копіювання, що дозволяє користувачам створювати точні копії своїх файлів. Однією з головних переваг FBackup є її здатність до автоматизації процесів резервного копіювання, що дозволяє користувачам налаштовувати резервні копії на певні дні або часи без необхідності ручного управління процесом. Інтерфейс програми простий та

інтуїтивно зрозумілий, що робить її доступною для користувачів будь-якого рівня технічної підготовки.

Git є потужною системою контролю версій, яка дозволяє користувачам легко створювати резервні копії своїх проектів [31]. Однією з головних переваг Git є можливість локального збереження історії змін, що дозволяє відстежувати всі версії файлів у проекті та відновлювати будь-яку з них у будь-який момент. Це забезпечує високий рівень контролю та гнучкості при роботі з кодом. Крім того, Git підтримує роботу з гілками, що дозволяє створювати окремі лінії розробки для нових функцій або виправлення помилок, не впливаючи на основну версію проекту.

Також зберігає копії файлів як послідовні зміни (інкременти), зберігаючи лише різницю між версіями файлів. Кожен коміт в Git представляє собою зміни відноно попередньої версії, а не повну копію всіх файлів.

Git зберігає повні ревізії кожного файлу на кожен коміт, тобто стан файлів на момент конкретного коміту. Це означає, що ви завжди можете відновити будь-яку попередню версію файлу або навіть весь проект до будь-якого стану в його історії.

Отже, Git не створює повних копій на кожному коміті (як це робить повне резервне копіювання), але зберігає зміни (інкременти) і повні ревізії (з ревізіями), що дозволяє відновлювати дані на різних рівнях точності в залежності від потреб.

Порівняння відомих програмних засобів для резервного копіювання наведено в таблиці 1.2.

Таблиця 1.2 – Порівняльна таблиця засобів резервного копіювання

| Програмне забезпечення | Типи резервного копіювання | Типи сховища | Типи даних | Інші характеристики | Вартість |
|---|---|---|---|---|---|
| EaseUS Todo Backup | Повне, Інкр., Дифер. | Локальне, Зовнішнє, Хмарне, NAS | Файли, Система, Диск, Розділ | Клонування, Ущільнення, Планування | Безкоштовно (Обмежено), $29.95/рік (Home), $39.95/рік (Workstation) |

Продовження таблиці 1.2

| Програмне забезпечення | Типи резервного копіювання | Типи сховища | Типи даних | Інші характеристики | Вартість |
|---|---|---|---|---|---|
| Acronis True Image | Повне, Інкр., Дифер. | Локальне, Зовнішнє, Хмарне | Файли, Система, Диск, Розділ | Клонування, Анти-Вимагач, Хмарне резервне копіювання | $49.99/рік (Essential), $89.99/рік (Advanced), $124.99/рік (Premium) |
| Macrium Reflect | Повне, Інкр., Дифер. | Локальне, Зовнішнє, NAS | Файли, Система, Диск, Розділ | Клонування, Завантажувальний носій, Планування | Безкоштовно (Обмежено), $69.95 (Standard), $139.95 (Home), $299.95 (Server) |
| AOMEI Backupper | Повне, Інкр., Дифер. | Локальне, Зовнішнє, NAS | Файли, Система, Диск, Розділ | Клонування, Ущільнення, Завантажувальний носій | Безкоштовно (Standard), $49.95 (Professional), $199.90 (Technician) |
| Paragon Backup & Recovery | Повне, Інкр., Дифер. | Локальне, Зовнішнє, Хмарне, NAS | Файли, Система, Диск, Розділ | Завантажувальний носій USB, Планування | $79.95 (Home), Індивідуальна (Business) |
| Genie Timeline | Повне, Інкр. | Локальне, Зовнішнє | Файли | Режим Turbo, Розумний режим | $49.95 (Одноразова) |
| NovaBACKUP | Повне, Інкр., Дифер. | Локальне, Зовнішнє, Хмарне, NAS | Файли, Система, Диск, Розділ | Шифрування, Завантажувальний носій | $49.95/рік (PC), $199.95/рік (Server) |
| Comodo Backup | Повне, Інкр., Дифер. | Локальне, Зовнішнє, Хмарне | Файли, Система, Диск, Розділ | Шифрування, Завантажувальний носій | Безкоштовно (Basic), $12/рік (Advanced), $35/рік (Pro) |
| Cobian Backup | Повне, Інкр., Дифер. | Локальне, Зовнішнє, NAS | Файли, Система, Диск, Розділ | Шифрування, Планування | Безкоштовно |
| FBackup | Повне, Дзерк. | Локальне, Зовнішнє, Хмарне | Файли | Ущільнення, Плагіни | Безкоштовно |
| Git | Інкрементне з ревізіями | Локальне, Віддалене | Файли, Код | Контроль версій, Гілки | Безкоштовно |

На основі аналізу існуючих рішень можна зробити висновок, що кожне з них має свої унікальні особливості та сфери застосування. Обрання конкретного програмного засобу для резервного копіювання залежить від конкретних потреб користувача або підприємства, обсягу даних, що зберігаються, та бюджету, який виділяється на забезпечення захисту даних.

Загалом, ринок програмних засобів для резервного копіювання пропонує широкий вибір рішень, що дозволяє кожному користувачеві знайти те, що найкраще відповідає його потребам. Незалежно від обраного методу чи програмного засобу, важливо пам'ятати, що резервне копіювання є ключовим елементом у забезпеченні безпеки даних та безперебійної роботи інформаційних систем.

### 1.5 Формалізація вимог та постановка задачі

У процесі розробки програмного засобу для резервного копіювання даних необхідно чітко визначити вимоги та поставити задачі, які цей програмний засіб повинен виконувати. Вимоги повинні враховувати всі аспекти функціональності, безпеки, продуктивності та зручності використання, щоб забезпечити високий рівень задоволення потреб користувачів та відповідність технічним стандартам.

Основні вимоги до програмного засобу включають наступні пункти:

- Типи копіювання:
  а) Повне копіювання: створення повної копії всіх даних, що підлягають резервному копіюванню;
  б) Інкрементне копіювання: збереження лише тих даних, що змінилися з моменту останнього резервного копіювання;
  в) Диференційоване копіювання: збереження всіх змін з моменту останнього повного копіювання;
- Планування резервного копіювання:
  а) Можливість встановлення резервного копіювання на певний час і дату;
  б) Реалізація циклічного копіювання з можливістю налаштування інтервалів: щоденно, щотижнево або щомісячно;
- Шифрування даних:
  а) Опція шифрування певних невеликих резервних копій для захисту особистих даних користувачів;

б) Збереження резервних копій у шифрованому вигляді з використанням алгоритму AES;

- Архівування даних: можливість архівування резервних копій за допомогою формату Zip для зменшення об'єму збережених даних та підвищення ефективності використання дискового простору;
- Алгоритм відновлення даних з обраного типу сховища;
- Сховища для резервних копій:
  а) Локальне сховище: збереження резервних копій на локальних дисках або інших пристроях зберігання, підключених до комп'ютера користувача;
  б) Спільна директорія на іншому локальному пристрої;
  в) Хмарне сховище: збереження резервних копій у хмарному сервісі Google Cloud для підвищення надійності та доступності даних.

Програмний засіб повинен бути розроблений з урахуванням цих вимог, щоб забезпечити надійне, зручне та безпечне резервне копіювання даних користувачів.

### 1.6 Висновки до першого розділу

Проведений аналіз предметної області показав важливість та необхідність розробки надійного програмного засобу для резервного копіювання даних. У сучасному світі цифрова інформація є критично важливим активом, втрата якого може мати серйозні наслідки для організацій та індивідуальних користувачів. Різноманітні загрози, такі як апаратні збої, людські помилки, шкідливе програмне забезпечення, природні катастрофи та кібератаки, постійно ставлять під загрозу цілісність та доступність даних.

У ході аналізу було розглянуто сучасні технології та методи резервного копіювання, включаючи повне, диференціальне та інкрементальне резервне копіювання, а також ручне копіювання даних. Кожен з методів має свої переваги та недоліки, які слід враховувати при виборі відповідного підходу для конкретних потреб користувачів.

Також було розглянуто різні типи сховищ для резервного копіювання, включаючи локальні, мережеві та хмарні рішення. Локальні сховища зазвичай пропонують швидкий доступ до даних та високу надійність, однак можуть бути вразливими до фізичних пошкоджень. Мережеві сховища забезпечують централізоване зберігання даних та зручність у спільному використанні, але вимагають надійної мережевої інфраструктури. Хмарні рішення відрізняються високою масштабованістю та доступністю з будь-якої точки світу, проте можуть бути дорожчими та залежать від стабільності інтернет-з’єднання.

Аналіз існуючих засобів резервного копіювання даних, таких як EaseUS Todo Backup, Acronis True Image, Macrium Reflect, AOMEI Backupper та інші, показав, що на ринку представлено широкий спектр рішень, які задовольняють різні потреби користувачів. EaseUS Todo Backup відзначається простотою використання та широким функціоналом, що підходить для домашніх користувачів. Acronis True Image пропонує потужні можливості для професійного використання, включаючи клонування дисків та захист від зловмисного ПЗ. Macrium Reflect славиться своєю надійністю та швидкістю роботи, що робить його популярним серед IT-фахівців. AOMEI Backupper пропонує безкоштовну версію з багатим набором функцій, що робить його привабливим для малого бізнесу.

На основі проведеного аналізу було сформульовано основні вимоги до програмного засобу для резервного копіювання даних, який планується розробити. Ці вимоги включають підтримку різних типів резервного копіювання, можливість планування, шифрування та архівування даних, а також підтримку різних типів сховищ.

Таким чином, розробка програмного засобу для резервного копіювання є актуальним завданням, яке дозволить забезпечити надійне збереження та відновлення даних у разі їх втрати, а також підвищить загальний рівень інформаційної безпеки для користувачів.

# 2 ТЕХНІЧНЕ ПРОЕКТУВАННЯ СИСТЕМИ

## 2.1 Розробка класифікації системи резервного копіювання

На основі вище описаного аналізу класифікація систем резервного копіювання може бути здійснена за такими ознаками:

- За типом резервного копіювання:
    1) Повне резервне копіювання. Копіює всі дані з об'єкта або системи.
    2) Диференціальне резервне копіювання: Копіює лише зміни відносно останнього повного резервного копіювання;
    3) Інкрементальне резервне копіювання. Копіює лише зміни відносно попереднього резервного копіювання (повного або інкрементального);
- За місцем збереження:
    1) Локальне. Дані зберігаються на пристрої або в локальній мережі;
    2) Хмарне. Дані зберігаються на віддалених серверах, що належать хмарному провайдеру;
- За методом управління:
    1) Ручне. Резервне копіювання виконується вручну;
    2) Автоматичне. Резервне копіювання заплановано і виконується автоматично згідно з певними установками;
    3) Автоматизоване. Резервне копіювання виконується автоматично, але також може включати додаткові функції, такі як моніторинг стану системи або автоматичне відновлення;
- За рівнем програмної або апаратної реалізації:
    1) Програмне. Резервне копіювання виконується за допомогою спеціального програмного забезпечення;
    2) Апаратне. Резервне копіювання виконується за допомогою апаратного забезпечення, такого як пристрої збереження або спеціальні пристрої резервного копіювання
- За часовим інтервалом копіювання:

1) Регулярне. Резервне копіювання виконується з регулярними інтервалами, наприклад, кожну годину або кожну добу;
2) Періодичне. Резервне копіювання виконується згідно з певними періодичними розкладами, наприклад, щоденно, щотижнево, щомісяця тощо;
3) Подійне. Резервне копіювання спрацьовує відразу після певних подій, таких як зміни в файлах або базі даних;

– 6. За методом збереження копій:
1) Інкрементальне збереження. Зміни зберігаються окремо від оригінальних даних, що дозволяє зберігати більше історії змін за менше простору;
2) Синхронізація. Не лише створює резервні копії, але й забезпечує одночасну синхронізацію змін між джерелом і копією;

– За рівнем доступності копій:
1) Єдиноразове резервне копіювання. Виконується лише один раз, без можливості оновлення або додавання нових даних;
2) Мультиверсійне резервне копіювання. Забезпечує зберігання кількох версій даних з різних моментів часу, що дозволяє відновлювати дані з конкретного моменту;

– За рівнем захисту інформації:
1) Шифрування. Можливість шифрування резервних копій для захисту від несанкціонованого доступу;
2) Без шифрування. відсутність можливості шифрування
3) Автентифікація. Вимога до аутентифікації перед відновленням або доступом до резервних копій;
4) Без автентифікації. відсутність вимоги автентифікації для доступу до резервних копій;

– За обсягом даних та ресурсами:

1) Масштабоване. Здатність системи резервного копіювання працювати ефективно навіть з величезними обсягами даних, забезпечуючи стабільну продуктивність;
2) Немасштабоване. Система працює з відносно невеликими резервними копіями;
3) Обмежене. Резервне копіювання обмежується лише обсягом ресурсу для зберігання копій;
4) З Оптимізацією ресурсів. Мінімізація використання ресурсів, таких як мережевий трафік або обсяг дискового простору, для ефективного виконання процесу резервного копіювання;
5) Без оптимізації ресурсів. Резервне копіювання здійснюється в звичайному режимі без ущільнення даних при передачі та зберіганні;

– За типом джерела даних:
1) Файлове резервне копіювання. Копіювання файлів і папок з файлової системи;
2) Базове резервне копіювання. Копіювання баз даних, таких як SQL бази даних або NoSQL системи;
3) Системне резервне копіювання. Копіювання системних налаштувань та конфігурацій операційної системи, сервера, мережевих пристроїв;

– За можливістю розширення та інтеграції:
1) з API доступом / без API доступом. Наявність/відсутність API для інтеграції з іншими системами та сторонніми програмами;
2) Наявність / відсутність модульності. Здатність/відсутність розширення функціоналу за допомогою додаткових модулів або плагінів, що дозволяє налаштовувати її під конкретні потреби та вимоги користувача;

– За призначенням:
1) Системи резервного копіювання загального призначення – системи призначені для забезпечення резервного копіювання різноманітних типів даних та інформації, включаючи файли, бази даних, системи

операцій та інше. Вони зазвичай мають різні опції та конфігурації, які можна налаштувати залежно від потреб користувача чи організації. Такі системи резервного копіювання надають універсальний підхід до забезпечення безпеки даних та можуть бути використані в різних сферах, від домашнього користувача до великих корпорацій;

2) Спеціалізовані системи резервного копіювання – системи призначені для конкретних типів даних або вузько спеціалізованих потреб. Наприклад, існують спеціалізовані системи резервного копіювання для баз даних, які спеціально розроблені для збереження та відновлення інформації з баз даних великих обсягів. Інші спеціалізовані системи можуть бути призначені для забезпечення безпеки тільки для певних типів файлів чи даних, наприклад, медичних записів, фінансових даних тощо. Спеціалізовані системи резервного копіювання можуть мати унікальні функції та можливості, які спеціально враховують особливості конкретного типу даних чи вимоги відповідних галузей;

– За місцем відновлення:

1) З відновленням тільки на вихідній системі – система забезпечує можливість відновлення даних лише на тій системі, на якій була виконана резервна копія. Це може бути зручно, якщо система пережила локальну аварію, але не підходить для випадків, коли весь обладнаний зазнає збою;

2) З відновленням тільки на іншій системі – система передбачає можливість відновлення даних лише на іншій системі, а не на тій, де була виконана резервна копія. Це корисно в разі повного збою обладнання або системи, коли потрібно відновити дані на іншому пристрої або сервері;

3) З кросплатформеним відновленням – система дозволяє відновлювати дані на будь-якій платформі, незалежно від платформи, на якій була здійснена резервна копія. Вона дозволяє легко переміщувати та відновлювати дані між різними операційними системами, що може

бути важливо в гетерогенних середовищах, де використовуються різні платформи.

Класифікація систем резервного копіювання охоплює різні аспекти, включаючи типи резервного копіювання, методи управління, рівень реалізації, часові інтервали, методи збереження копій, рівень доступності, захист інформації (шифрування, автентифікація), обсяг даних та ресурси, тип джерела даних, можливість розширення та інтеграції, призначення, та місце відновлення. Це дозволяє вибрати оптимальну систему для конкретних потреб, забезпечуючи ефективність, надійність і безпеку процесу резервного копіювання.

## 2.2 Розробка архітектури системи

Архітектура програмного засобу для резервного копіювання даних включає в себе кілька ключових модулів, які забезпечують реалізацію всіх необхідних функцій і забезпечують ефективну взаємодію між компонентами системи.

Архітектура системи зображена на рисунку 2.1.

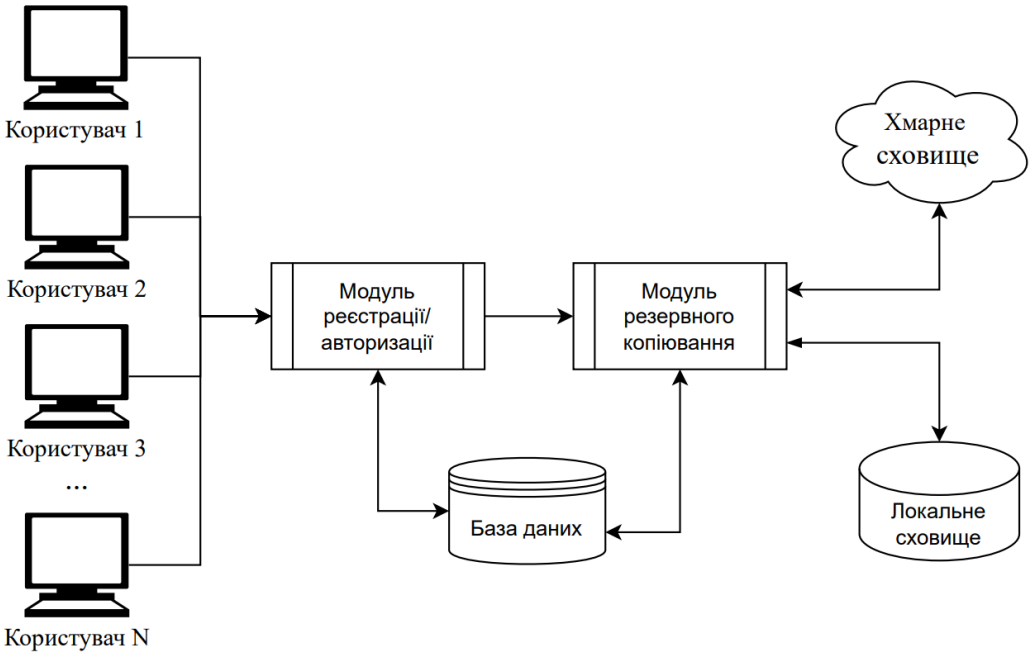



Рисунок 2.1 – Вигляд архітектури системи

Основні елементи архітектури та їх взаємодії описані нижче.

Користувачі (Користувач 1, Користувач 2, Користувач 3, …, Користувач N) є кінцевими користувачами системи, які взаємодіють з нею через інтерфейси користувача. Вони здійснюють операції, що вимагають реєстрації та авторизації, що забезпечується компонентом процесів реєстрації/авторизації. Цей компонент відповідає за реєстрацію нових користувачів та авторизацію існуючих. Він взаємодіє з базою даних для збереження та перевірки даних користувачів.

База даних є центральним елементом системи, який зберігає інформацію про користувачів, включаючи їхні облікові записи та дані реєстрації та авторизації. Вона забезпечує збереження та доступ до необхідної інформації для процесів реєстрації та авторизації. Після авторизації користувачів, дані з бази даних також використовуються процесами резервного копіювання для збереження налаштувань додатку.

Процеси резервного копіювання отримують дані з бази даних про збереженні налаштування додатку. Виконується копіювання у локальному або хмарному сховищах. Локальне сховище призначене для збереження резервних копій даних на локальних серверах або пристроях, забезпечуючи швидкий доступ до резервних даних у разі необхідності їхнього відновлення. Хмарне сховище, у свою чергу, використовується для збереження резервних копій даних в інтернет-середовищі, що забезпечує додатковий рівень безпеки та доступність даних незалежно від місцезнаходження користувачів або фізичних серверів. Також механізм відновлення відповідно для певного користувача.

Ця архітектура забезпечує надійний та безпечний про цес обробки, зберігання та резервного копіювання даних користувачів, що є важливим для функціонування системи в цілому.

### 2.3 Розробка математичної моделі

Для розробки математичної моделі процесу резервного копіювання, необхідно визначити основні параметри та функціональні можливості системи. Основними елементами моделі є обсяг даних, що підлягають резервуванню,

частота резервного копіювання, час, необхідний для резервного копіювання та відновлення даних, а також ресурси, необхідні для виконання цих операцій.

Вхідні параметри:

- $D$ – загальний обсяг даних, що підлягають резервуванню;
- $F$ – частота резервного копіювання;
- $T_{backup}$ – час, необхідний для виконання резервного копіювання;
- $T_{restore}$ – час, необхідний для відновлення даних;
- $R$ – ресурси, необхідні для виконання операцій резервного копіювання та відновлення (наприклад, обсяг пам'яті, пропускна здатність мережі).

Функція резервного копіювання:

$$T_{backup} = f(D, R), \tag{2.1}$$

де $T_{backup}$ – залежить від обсягу даних $D$ та ресурсів $R$.

Функція відновлення даних:

$$T_{restore} = g(D, R), \tag{2.2}$$

де $T_{restore}$ – залежить від обсягу даних $D$ та ресурсів $R$.

Резервне копіювання може виконуватися за допомогою різних методів: повне, інкрементальне та диференціальне резервування. Для кожного з цих методів необхідно враховувати специфічні особливості.

Повне резервне копіювання:

$$T_{full\_backup} = \alpha \cdot D, \tag{2.3}$$

де $\alpha$ – коефіцієнт, що залежить від продуктивності апаратного забезпечення.

Інкрементальне резервне копіювання:

$$T_{incremental\_backup} = \alpha \cdot \Delta D, \tag{2.4}$$

де $\Delta D$ – обсяг змінених даних з моменту останнього резервного копіювання.

Диференціальне резервне копіювання:

$$T_{differential\_backup} = \alpha \cdot \Delta D_{full}, \quad (2.5)$$

де $\Delta D_{full}$ – обсяг змінених даних з моменту останнього повного резервного копіювання.

Час відновлення даних також залежить від методу резервного копіювання.

Відновлення з повного резервного копіювання:

$$T_{full\_restore} = \alpha \cdot D, \quad (2.6)$$

Відновлення з інкрементального резервного копіювання:

$$T_{incremental\_restore} = \alpha \cdot (D + \sum \Delta D_i), \quad (2.7)$$

де $\sum \Delta D_i$ – сума всіх змінених даних з моменту останнього повного резервного копіювання.

Відновлення з диференціального резервного копіювання:

$$T_{differential\_restore} = \alpha \cdot (D + \sum \Delta D_{full}), \quad (2.8)$$

де $\Delta D_{full}$ – загальний обсяг змінених даних з моменту останнього повного резервного копіювання.

Узагальнена формула для відновлення даних:

$$T_{restore} = \alpha \cdot (D + \sum \Delta D_{method}), \quad (2.9)$$

де – $\Delta D_{method}$ залежить від обраного методу резервного копіювання (інкрементального чи диференціального).

У підсумку, розроблена математична модель процесу резервного копіювання даних враховує основні параметри, такі як обсяг даних, частоту резервного

копіювання, час та ресурси, необхідні для виконання операцій. Вона описує функції резервного копіювання та відновлення для різних методів: повного, інкрементального та диференціального резервування. Завдяки цій моделі можна ефективно оцінити та оптимізувати процес резервного копіювання, забезпечуючи мінімальний час простою та оптимальне використання ресурсів. Це сприяє підвищенню надійності та ефективності захисту даних, що є критично важливим для сучасних інформаційних систем.

### 2.4 Розробка алгоритмів функціонування програми

Алгоритми функціонування програмного засобу для резервного копіювання даних є ключовим елементом його ефективної роботи.

Алгоритм повного резервного копіювання починається з отримання списку файлів та папок, які користувач обрав для резервного копіювання. Перш за все, алгоритм перевіряє доступність цільового сховища, переконуючись, що воно готове приймати дані. Після цього створюється нова резервна копія шляхом копіювання всіх вибраних даних до цільового місця зберігання. Під час процесу копіювання алгоритм ретельно перевіряє цілісність даних, щоб уникнути можливих помилок або пошкодження файлів. Цей етап є критично важливим для забезпечення надійності резервного копіювання. Алгоритм використовує методи перевірки, щоб гарантувати, що всі дані копіюються без помилок. Якщо під час копіювання виникає помилка, алгоритм негайно зупиняє процес і повідомляє користувача про проблему. Це дозволяє користувачеві прийняти необхідні заходи для виправлення ситуації та повторити операцію резервного копіювання, щоб забезпечити повноту та точність копійованих даних. Після успішного завершення копіювання алгоритм зберігає метадані про резервну копію у базі даних. Ці метадані включають час створення резервної копії, її розмір та місце зберігання. Зберігання таких метаданих допомагає користувачам легко відслідковувати резервні копії та забезпечувати ефективне управління ними (рис. 2.2).

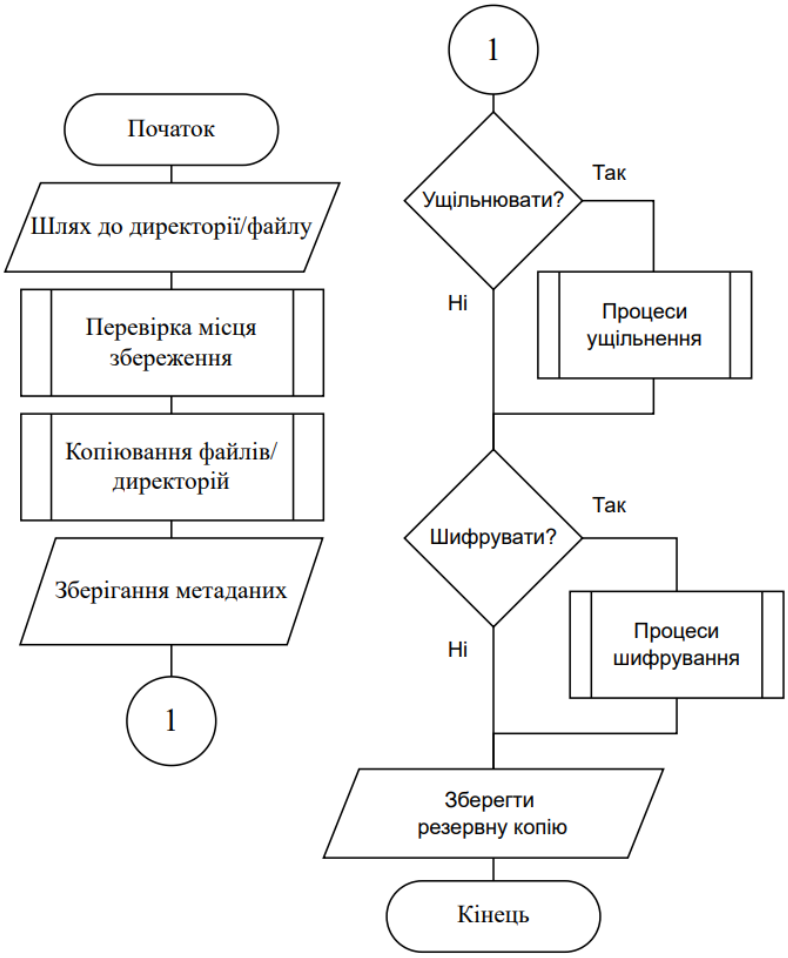



Рисунок 2.2 – Структура алгоритму повного резервного копіювання

Таким чином, алгоритм повного резервного копіювання не лише гарантує збереження важливих даних, але й забезпечує їх цілісність та надійність, надаючи користувачам впевненість у безпеці їх інформації.

Наступним алгоритмом є інкрементне резервне копіювання. Він також починається з отримання списку файлів та папок. Потім алгоритм визначає, які файли були змінені або додані з моменту останнього резервного копіювання, використовуючи метадані, що зберігаються у базі даних. Тільки ці файли копіюються у цільове місце зберігання. Після завершення копіювання метадані оновлюються відповідно до нових змін.

Алгоритм диференційованого резервного копіювання працює подібно до інкрементного, але з тією різницею, що він визначає файли, змінені з моменту останнього повного резервного копіювання. Алгоритм отримує список змінених

файлів і копіює їх у цільове місце зберігання. Після завершення копіювання метадані оновлюються (рис. 2.3).

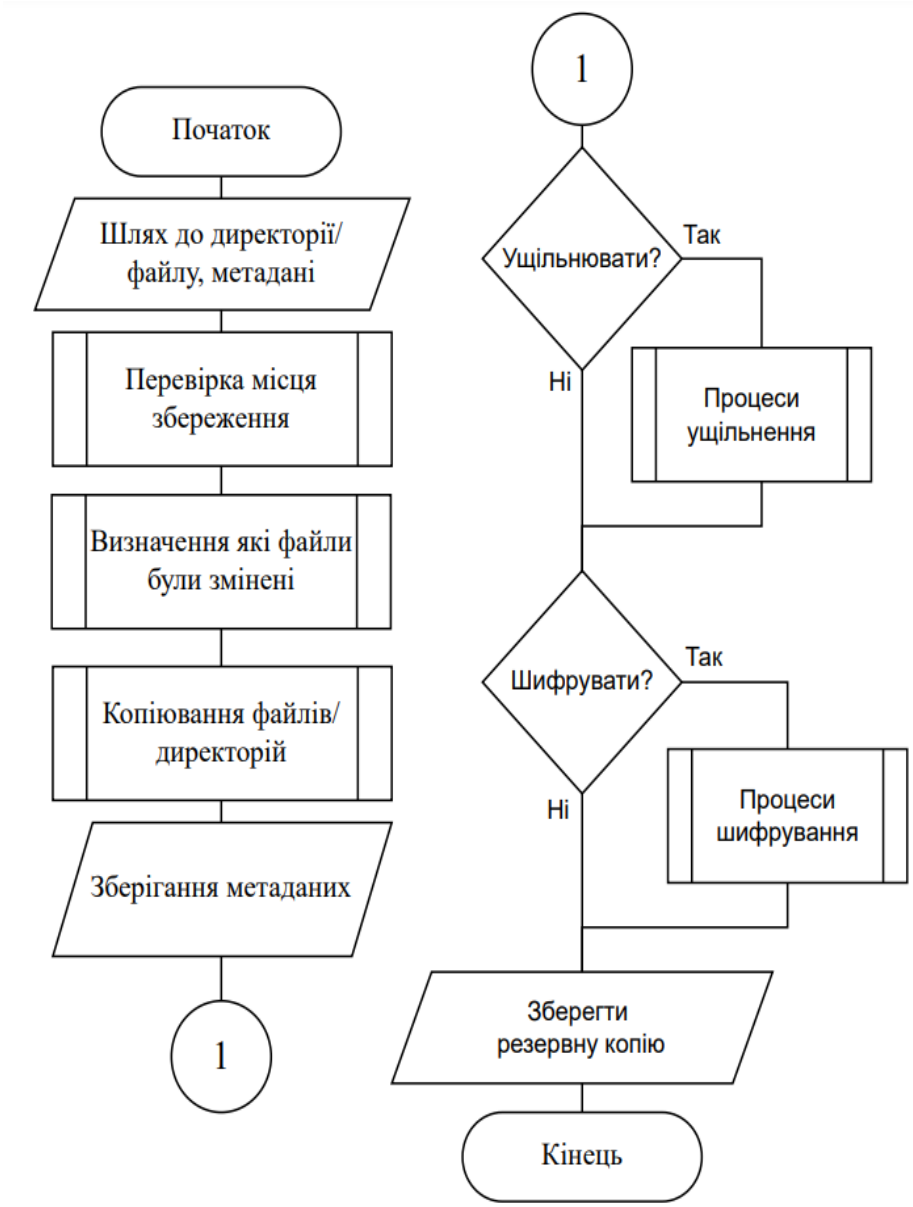



Рисунок 2.3 – Структура алгоритму диференційованого резервного копіювання

Наступним одним з головних алгоритмів застосунку є алгоритм відновлення даниз з резервної копії (рис. 2.4).

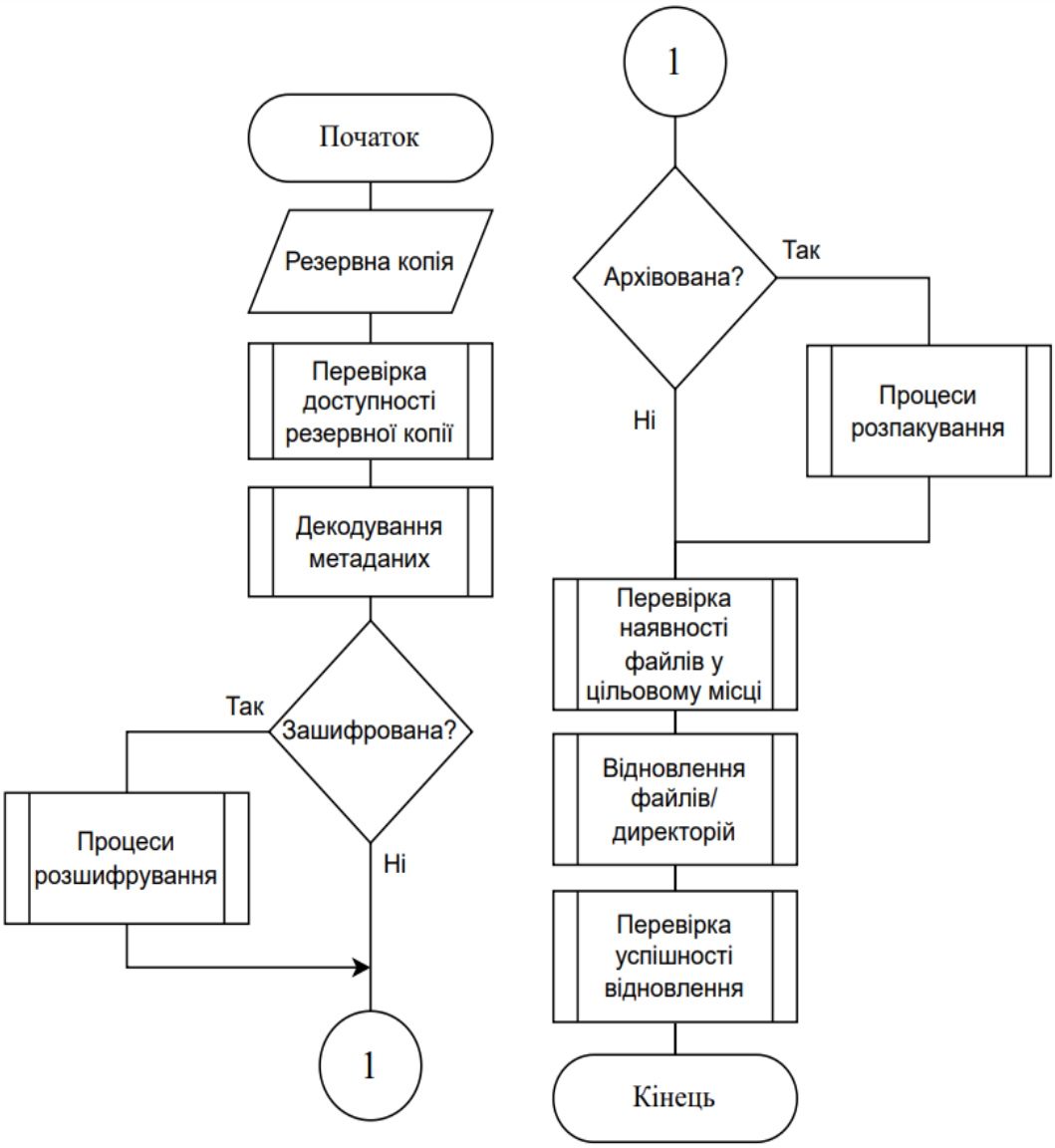



Рисунок 2.4 – Структура алгоритму відновлення копії

Алгоритм відновлення з резервної копії розпочинається з вибору потрібної резервної копії та перевірки її доступності. Далі здійснюється декодування метаданих, що дозволяє визначити структуру файлів та їх властивості. Якщо резервна копія зашифрована, то виконується процес розшифрування, а якщо ущільнена – процес розпакування. Наступним кроком є визначення цільового місця для відновлення файлів, а також перевірка, чи існують файли в цьому місці і чи потрібно їх замінити.

Після визначення місця збереження файли копіюються з резервної копії до цільового місця. Завершальним етапом є перевірка успішності відновлення, яка

включає підтвердження, що всі файли були відновлені правильно та без помилок. Після цього процес відновлення завершується.

Для алгоритму формування ключа було обрано PBKDF2 (Password-Based Key Derivation Function 2) який є ефективним алгоритмом для формування ключа через свою здатність забезпечити високий рівень безпеки за допомогою соління та ітераційного процесу. Соління дозволяє захистити паролі від атак зі заздалегідь підготовленими таблицями (так званими «rainbow tables»), додаючи випадковий елемент до кожного паролю перед його хешуванням. Це робить кожен пароль унікальним навіть якщо два користувачі обирають однаковий пароль. Ітераційний процес, у свою чергу, забезпечує додатковий рівень безпеки, збільшуючи час, необхідний для обчислення хешу, тим самим ускладнюючи виконання атак грубою силою.

Крім того, PBKDF2 є стандартом, рекомендованим Національним інститутом стандартів і технологій США (NIST), що свідчить про його надійність та перевіреність у численних дослідженнях. Алгоритм широко використовується в різних програмних системах та протоколах, таких як WPA2 для захисту бездротових мереж, що свідчить про його практичну ефективність. Завдяки своїй гнучкості, PBKDF2 дозволяє налаштовувати кількість ітерацій та довжину солі відповідно до вимог конкретного застосування, забезпечуючи баланс між безпекою та продуктивністю.

Розробка алгоритмів функціонування програми передбачає створення ефективних, надійних та масштабованих рішень, які забезпечують безперебійну роботу програмного засобу для резервного копіювання даних.

### 2.5 Висновки до другого розділу

У розділі було детально розглянуто основні аспекти технічного проектування програмного засобу для резервного копіювання даних. Спочатку була визначена архітектура програмного засобу, яка включає кілька ключових модулів. Ці модулі реалізують необхідні функції і забезпечують ефективну взаємодію між компонентами системи. Зокрема, були розроблені модулі для реєстрації та

авторизації користувачів, обробки та збереження даних, а також для взаємодії з різними типами сховищ даних, включаючи як локальні, так і хмарні середовища.

Наступним кроком була розробка математичної моделі процесу резервного копіювання. Ця модель враховує основні параметри та функціональні можливості системи, такі як обсяг даних, частота резервного копіювання, а також час, необхідний для виконання процедур резервування та відновлення даних. Це дозволяє точно налаштувати процеси резервного копіювання відповідно до потреб користувача та вимог безпеки.

Крім того, були розроблені ефективні алгоритми для функціонування програмного засобу. Вони включають алгоритми для самого процесу резервного копіювання даних, відновлення інформації у випадку втрати, а також алгоритми для шифрування і стиснення даних, що забезпечують збереження та передачу інформації у безпечному і ефективному форматі.

Ці кроки демонструють систематичний підхід до проектування програмного засобу для резервного копіювання даних, що забезпечує якість, надійність та зручність використання для кінцевих користувачів.

# 3 РОБОЧЕ ПРОЕКТУВАННЯ СИСТЕМИ

## 3.1 Обґрунтування інструментальних засобів

Розробка програмного забезпечення є складним процесом, який вимагає використання різноманітних інструментальних засобів.

Java, завдяки своїй платформонезалежності, стабільності та надійності, стала однією з найпопулярніших мов програмування для серверних додатків [32]. Вона дозволяє розробляти масштабовані та безпечні веб-додатки, що робить її ідеальним вибором для багатьох корпоративних систем. Java використовує віртуальну машину (JVM), яка забезпечує високу продуктивність і оптимізацію коду на різних платформах, що є значною перевагою перед іншими мовами.

Порівнюючи Java з Python [32], варто відзначити, що Python відомий своєю простотою та зручністю у використанні, що робить його популярним серед новачків і для розробки прототипів. Однак, Python поступається Java у продуктивності та масштабованості, що робить його менш придатним для великих корпоративних систем. Крім того, Python інтерпретується, а не компілюється, що може впливати на швидкість виконання програм.

C# є ще однією популярною мовою для розробки серверних додатків, особливо в екосистемі Microsoft. C# має багато спільного з Java у плані синтаксису та можливостей, але його основною перевагою є інтеграція з платформою .NET, що забезпечує тісну взаємодію з іншими технологіями Microsoft. Втім, Java все ще залишається більш універсальною мовою завдяки своїй платформонезалежності та великій кількості бібліотек і фреймворків.

IntelliJ IDEA Ultimate Edition [33] вирізняється своєю інтелектуальною потужністю серед інших середовищ розробки, таких як Eclipse і NetBeans. Завдяки розширеним можливостям автодоповнення коду, рефакторингу та аналізу коду, IntelliJ IDEA значно перевершує своїх конкурентів, надаючи розробникам змогу писати код швидше та з меншою кількістю помилок. Крім того, інтеграція з системами контролю версій, такими як Git, робить процес управління версіями проєкту більш зручним та ефективним.

У порівнянні з Eclipse, який також є потужним і популярним середовищем розробки, IntelliJ IDEA пропонує більш інтуїтивний інтерфейс і кращу підтримку сучасних фреймворків та мов програмування. Хоча Eclipse має велику кількість плагінів, IntelliJ IDEA часто забезпечує кращу інтеграцію з ними і пропонує зручніші інструменти для тестування та налагодження. Це дозволяє розробникам швидше знаходити та виправляти помилки у коді, що знижує час розробки та підвищує якість кінцевого продукту.

NetBeans, ще одне популярне середовище розробки, також підтримує безліч фреймворків і мов програмування, але не може зрівнятися з IntelliJ IDEA у плані інтеграції з базами даних і хмарними сервісами. IntelliJ IDEA надає розробникам можливість швидко та ефективно керувати даними додатку завдяки вбудованим інструментам для роботи з базами даних. Це робить його ідеальним вибором для проєктів, що потребують складного управління даними та інтеграції з різними сервісами.

Ще однією перевагою IntelliJ IDEA є його система плагінів, яка дозволяє розширювати функціональність середовища відповідно до потреб конкретного проекту. Хоча всі три середовища розробки мають свої сильні сторони, IntelliJ IDEA часто виділяється своєю здатністю адаптуватися до сучасних вимог розробки, забезпечуючи зручний інтерфейс, потужні інструменти для тестування та налагодження, а також широкі можливості інтеграції з іншими технологіями та сервісами.

Spring Boot [34] є фреймворком для створення мікросервісів і веб-додатків на базі Java, який спрощує процес розробки завдяки своїй високій автоматизації і конвенційності. Основною перевагою Spring Boot є його здатність швидко розгортати робочі додатки з мінімальними налаштуваннями. Він надає вбудовані засоби для роботи з базами даних, безпекою, веб-службами та іншими критичними аспектами розробки.

Порівняно з іншими фреймворками, такими як Java EE або Dropwizard, Spring Boot виграє завдяки своїй простоті та широкій підтримці. Java EE пропонує повний набір стандартів для корпоративних додатків, але може бути складним у

налаштуванні та використанні. Dropwizard є ще одним популярним вибором для мікросервісів, але Spring Boot пропонує більше інтегрованих рішень і має більшу спільноту.

Vaadin [35] є фреймворком для створення сучасних веб-додатків з використанням Java. Основною перевагою Vaadin є його орієнтація на серверну частину, що дозволяє розробникам створювати складні веб-інтерфейси без необхідності написання великої кількості JavaScript-коду. Vaadin забезпечує повну інтеграцію з екосистемою Java, що дозволяє використовувати знайомі інструменти та бібліотеки.

Порівняно з іншими фреймворками для веб-розробки, такими як Angular або React, Vaadin виділяється тим, що дозволяє зосередитися на серверному коді без потреби в глибокому знанні JavaScript. Хоча Angular і React є потужними інструментами для створення фронтенду, вони вимагають знання JavaScript та пов'язаних технологій, що може ускладнити розробку для Java-розробників. Vaadin забезпечує більш прямий шлях до створення інтерактивних веб-додатків, використовуючи Java.

Вибір інструментальних засобів для розробки веб-додатку є критичним етапом у процесі створення програмного забезпечення. Використання IntelliJ Idea Ultimate Edition, Spring Boot, Vaadin та Java забезпечує ефективність, надійність та гнучкість розробки, що дозволяє створювати високоякісні та масштабовані веб-додатки. Кожен з цих інструментів має свої унікальні переваги та особливості, що робить їх незамінними для сучасних розробників.

### 3.2 Розробка основних алгоритмів програмного засобу

Спочатку створюється метод performBackup, який приймає кілька параметрів, таких як шлях до джерела, шлях до цільового каталогу, тип резервного копіювання, тип сховища, прапори шифрування та стиснення. Цей метод виконує резервне копіювання у окремому потоці, використовуючи ExecutorService. В залежності від типу резервного копіювання (повне, інкрементне,

диференційоване), метод викликає відповідні приватні методи для виконання завдання (рис. 3.1).

```
public void performBackup(Path source, Path target,  1 usage
        String backupType, String storageType,
        boolean isEncrypted, boolean isCompressed) {
    executor.submit(() -> {
        try {
            Instant start = Instant.now();
            switch (backupType) {
                case "Повне":
                    fullBackup(source.toFile().getAbsolutePath(), target.toFile().getAbsolutePath(),
                            isEncrypted, isCompressed);
                    break;
                case "Інкрементне":
                    incrementalBackup(source.toFile().getAbsolutePath(), target.toFile().getAbsolutePath(),
                            isEncrypted, isCompressed);
                    break;
                case "Диференційоване":
                    differentialBackup(source.toFile().getAbsolutePath(), target.toFile().getAbsolutePath(),
                            isEncrypted, isCompressed);
                    break;
            }
            Instant end = Instant.now();
            Duration duration = Duration.between(start, end);
            System.out.println("Backup completed successfully in " + duration.toMinutesPart()
                    + " minutes and " + duration.toSecondsPart() + " seconds.");
        } catch (IOException e) {
            e.printStackTrace();
        }
```

Рисунок 3.1 – Демонстрація реалізації методу performBackup

Далі створюється метод fullBackup, який здійснює повне резервне копіювання. Він приймає шляхи до джерела та цілі, а також прапори шифрування та стиснення. Спочатку створюється каталог призначення, якщо він не існує, а потім всі файли та папки з джерела архівуються у ZIP-файл. Якщо вказано прапор шифрування, архівований файл шифрується методом encryptFile. Після успішного виконання резервного копіювання оновлюється час останнього повного та будь-якого резервного копіювання (рис. 3.2).

```java
public static void fullBackup(String sourceDir, String destDir,  1 usage
                              boolean isEncrypted, boolean isCompressed)
        throws IOException {
    File srcDir = new File(sourceDir);
    File destDirFile = new File(destDir, srcDir.getName());
    if (!destDirFile.exists()) {
        destDirFile.mkdirs();
    }
    String zipFilePath = destDirFile.getAbsolutePath() + ".zip";
    zipDirectory(srcDir, zipFilePath, isCompressed ? ZipOutputStream.DEFLATED : ZipOutputStream.STORED);
    if (isEncrypted) {
        encryptFile(new File(zipFilePath));
    }
    updateLastBackupTime(LAST_FULL_BACKUP_TIME_FILE);
    updateLastBackupTime(LAST_ANY_BACKUP_TIME_FILE);

    System.out.println("Full backup completed successfully.");
}
```

Рисунок 3.2 – Демонстрація реалізації методу fullBackup

Метод differentialBackup здійснює диференційоване резервне копіювання. Якщо було виконано повне резервне копіювання, він копіює тільки ті файли, які були змінені з моменту останнього повного резервного копіювання. Після копіювання файлів вони архівуються у ZIP-файл, а потім, за необхідності, шифруються. Після успішного виконання оновлюється час останнього будь-якого резервного копіювання (рис. 3.3).

```java
public static void differentialBackup(String sourceDir, String destDir,  1 usage
                                      boolean isEncrypted, boolean isCompressed)
        throws IOException {
    if (isFullBackupDone()) {
        File srcDir = new File(sourceDir);
        File destDirFile = new File(destDir, srcDir.getName());
        if (!destDirFile.exists()) {
            destDirFile.mkdirs();
        }
        long lastFullBackupTime = getLastBackupTime(LAST_FULL_BACKUP_TIME_FILE);
        Files.walkFileTree(srcDir.toPath(), new SimpleFileVisitor<Path>() {
            @Override
            public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOException {
                if (attrs.lastModifiedTime().toMillis() > lastFullBackupTime) {
                    Path destFile = destDirFile.toPath().resolve(srcDir.toPath().relativize(file));
                    Files.copy(file, destFile, StandardCopyOption.REPLACE_EXISTING);
                }
                return FileVisitResult.CONTINUE;
            }
        });
        String zipFilePath = destDirFile.getAbsolutePath() + ".zip";
        zipDirectory(destDirFile, zipFilePath, isCompressed ? ZipOutputStream.DEFLATED : ZipOutputStream.STORED);
        if (isEncrypted) {
            encryptFile(new File(zipFilePath));
        }
```

Рисунок 3.3 – Демонстрація реалізації методу differentialBackup

Метод incrementalBackup виконує інкрементне резервне копіювання. Він копіює лише ті файли, які були змінені з моменту останнього будь-якого резервного копіювання. Після копіювання файлів вони також архівуються у ZIP-файл, а потім можуть бути зашифровані. Після успішного виконання оновлюється час останнього будь-якого резервного копіювання.

```
public static void incrementalBackup(String sourceDir, String destDir,   1 usage
                                      boolean isEncrypted, boolean isCompressed)
        throws IOException {
    if (isAnyBackupDone()) {
        File srcDir = new File(sourceDir);
        File destDirFile = new File(destDir, srcDir.getName());
        if (!destDirFile.exists()) {
            destDirFile.mkdirs();
        }
        long lastIncrementalBackupTime = getLastBackupTime(LAST_ANY_BACKUP_TIME_FILE);
        Files.walkFileTree(srcDir.toPath(), new SimpleFileVisitor<Path>() {
            @Override
            public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOException {
                if (attrs.lastModifiedTime().toMillis() > lastIncrementalBackupTime) {
                    Path destFile = destDirFile.toPath().resolve(srcDir.toPath().relativize(file));
                    Files.copy(file, destFile, StandardCopyOption.REPLACE_EXISTING);
                }
                return FileVisitResult.CONTINUE;
            }
        });
        String zipFilePath = destDirFile.getAbsolutePath() + ".zip";
        zipDirectory(destDirFile, zipFilePath, isCompressed ? ZipOutputStream.DEFLATED : ZipOutputStream.STORED);
        if (isEncrypted) {
            encryptFile(new File(zipFilePath));
        }
        updateLastBackupTime(LAST_ANY_BACKUP_TIME_FILE);
```

Рисунок 3.4 – Демонстрація реалізації методу incrementalBackup

Метод zipDirectory архівує всі файли та папки з джерела у ZIP-файл. Він використовує ZipOutputStream для створення архіву і рекурсивно обходить всі файли та папки у каталозі джерела, додаючи їх до архіву (рис. 3.5).

```
private static void zipDirectory(File srcDir, String zipFilePath, int compressionLevel) throws IOException {  3 usages
    try (ZipOutputStream zos = new ZipOutputStream(new FileOutputStream(zipFilePath))) {
        zos.setLevel(compressionLevel);
        Files.walkFileTree(srcDir.toPath(), new SimpleFileVisitor<Path>() {
            @Override
            public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOException {
                try (FileInputStream fis = new FileInputStream(file.toFile())) {
                    ZipEntry zipEntry = new ZipEntry(srcDir.toPath().relativize(file).toString());
                    zos.putNextEntry(zipEntry);

                    byte[] bytes = new byte[1024];
                    int length;
                    while ((length = fis.read(bytes)) >= 0) {
                        zos.write(bytes,  off: 0, length);
                    }
                }
                return FileVisitResult.CONTINUE;
            }
```

Рисунок 3.5 – Демонстрація реалізації методу zipDirectory

Метод encryptFile шифрує файл за допомогою алгоритму AES. Спочатку генерується секретний ключ та вектор ініціалізації (IV). Файл читається блоками та

записується у зашифрованому вигляді до нового файлу з розширенням .enc. Після цього видаляється оригінальний файл, а секретний ключ та IV зберігаються для подальшого розшифрування (рис. 3.6).

```
private static void encryptFile(File file) throws IOException {  3 usages
    SecretKey secretKey = generateAESKey();
    IvParameterSpec iv = generateIV();

    try (FileInputStream fis = new FileInputStream(file);
         FileOutputStream fos = new FileOutputStream( name: file.getAbsolutePath() + ".enc");
         CipherOutputStream cos = new CipherOutputStream(fos, initCipher(Cipher.ENCRYPT_MODE, secretKey, iv))) {

        byte[] buffer = new byte[1024];
        int length;
        while ((length = fis.read(buffer)) > 0) {
            cos.write(buffer,  off: 0, length);
        }
    }

    file.delete();

    // Save the key and IV for decryption (this example uses Base64 encoding for simplicity)
    try (FileWriter keyWriter = new FileWriter(new File(file.getParent(),  child: "secret.key"));
         FileWriter ivWriter = new FileWriter(new File(file.getParent(),  child: "iv.txt"))) {
        keyWriter.write(Base64.getEncoder().encodeToString(secretKey.getEncoded()));
        ivWriter.write(Base64.getEncoder().encodeToString(iv.getIV()));
    }
}
```

Рисунок 3.6 – Демонстрація реалізації методу encryptFile

Метод initCipher ініціалізує об'єкт Cipher для шифрування або розшифрування, використовуючи наданий секретний ключ та IV. Він повертає налаштований об'єкт Cipher.

```
private static Cipher initCipher(int mode, SecretKey key, IvParameterSpec iv) throws IOException {  2 usages
    try {
        Cipher cipher = Cipher.getInstance(AES_ALGORITHM);
        cipher.init(mode, key, iv);
        return cipher;
    } catch (Exception e) {
        throw new IOException("Error initializing cipher", e);
    }
}
```

Рисунок 3.7 – Демонстрація реалізації методу initCipher

Метод generateAESKey генерує секретний ключ AES на основі наданого пароля, використовуючи алгоритм PBKDF2. Він також генерує сіль та зберігає її для подальшого використання при розшифруванні (рис. 3.8).

```
private static SecretKey generateAESKey() throws IOException {  2 usages
    try {
        KeyGenerator keyGen = KeyGenerator.getInstance(AES_KEY_ALGORITHM);
        keyGen.init(AES_KEY_SIZE);
        return keyGen.generateKey();
    } catch (Exception e) {
        throw new IOException("Error generating AES key", e);
    }
}
```

Рисунок 3.8– Демонстрація реалізації методу generateAESKey

Метод generateIV генерує вектор ініціалізації для алгоритму шифрування AES, використовуючи випадкові байти (рис. 3.9).

```
private static IvParameterSpec generateIV() {  1 usage
    byte[] iv = new byte[16];
    new SecureRandom().nextBytes(iv);
    return new IvParameterSpec(iv);
}
```

Рисунок 3.9– Демонстрація реалізації методу generateAESKey

Методи getLastBackupTime, updateLastBackupTime, isFullBackupDone та isAnyBackupDone працюють з файлами, що зберігають час останнього резервного копіювання. Вони читають або записують час у відповідні файли, що дозволяє визначати, коли було виконано останнє резервне копіювання.

На кінець розробляється метод restoreBackup, який приймає кілька параметрів: шлях до резервної копії, шлях до цільового каталогу, прапори шифрування та стиснення, а також пароль для шифрування. Цей метод відновлює дані з резервної копії у зазначений цільовий каталог (рис. 3.10).

```
public void restoreBackup(Path backupPath, Path targetDir,   1 usage
                          boolean isEncrypted, boolean isCompressed,
                          String password) {
    try {
        File backupFile = backupPath.toFile();
        if (!backupFile.exists()) {
            throw new FileNotFoundException("Резервна копія не знайдена за вказаним шляхом: " + backupPath);
        }
        if (isEncrypted) {
            backupFile = decryptFile(backupFile, password);
        }

        if (isCompressed) {
            unzipDirectory(backupFile, targetDir.toFile());
        }

        System.out.println("Відновлення резервної копії виконано успішно.");
    } catch (IOException e) {
        e.printStackTrace();
        System.err.println("Помилка при відновленні резервної копії: " + e.getMessage());
    }
}
```

Рисунок 3.10– Демонстрація реалізації методу restoreBackup

Метод починається з перевірки, чи існує резервна копія за наданим шляхом. Якщо резервна копія зашифрована, спочатку вона розшифровується за допомогою методу decryptFile. Для цього завантажується секретний ключ та вектор ініціалізації (IV) на основі наданого пароля, і виконується розшифрування файлу.

```
private File decryptFile(File file, String password) throws IOException {  1 usage
    SecretKey secretKey = generateAESKey(password);
    IvParameterSpec iv = loadIV();

    File decryptedFile = new File(file.getAbsolutePath().replace( target: ".enc",  replacement: ""));

    try (FileInputStream fis = new FileInputStream(file);
         FileOutputStream fos = new FileOutputStream(decryptedFile);
         CipherInputStream cis = new CipherInputStream(fis, initCipher(Cipher.DECRYPT_MODE, secretKey, iv))) {

        byte[] buffer = new byte[1024];
        int length;
        while ((length = cis.read(buffer)) > 0) {
            fos.write(buffer,  off: 0, length);
        }
    }

    return decryptedFile;
}
```

Рисунок 3.11– Демонстрація реалізації методу decryptFile

Після цього, якщо резервна копія стиснута, вона розархівовується за допомогою методу unzipDirectory. Цей метод приймає шлях до ZIP-файлу та шлях до цільового каталогу, розархівовуючи всі файли та папки у цільовий каталог.


Рисунок 3.12– Демонстрація реалізації методу unzipDirectory

У кінці методу виконується перевірка успішності відновлення. Якщо всі операції пройшли успішно, виводиться повідомлення про успішне відновлення резервної копії.

### 3.3 Тестування програмного засобу

Функціональне тестування програмного забезпечення є критичним етапом у процесі розробки, спрямованим на забезпечення його надійної роботи та відповідності вимогам функціональності. Основні аспекти, які перевіряються під час цього тестування, включають налаштування різних типів копіювання і сховищ для даних, можливість запланувати автоматичне виконання резервних копій, а також забезпечення безпеки через шифрування.

Один з перших кроків у тестуванні є перевірка коректності функціоналу радіо-кнопок. Вони відповідають за вибір типу копіювання (повне, інкрементне, диференціальне) та типу сховища (локальне або хмарне). Важливо переконатися, що користувач може вибрати потрібний варіант і що вибір зберігається і відображається на інтерфейсі програми правильно.

Далі в тестуванні важливо перевірити можливість користувача встановити час і дату для автоматичних резервних копій. Це включає перевірку того, як програма обробляє введені дані щодо часу і дати, і як вона виконує заплановані завдання відповідно до налаштувань користувача.

Ще однією важливою функцією є шифрування даних під час їхнього копіювання. Під час тестування слід перевірити, як програмне забезпечення застосовує шифрування до даних і як це впливає на процес їхнього транспортування і зберігання, забезпечуючи при цьому високий рівень конфіденційності та захисту інформації.

Також важливим елементом є можливість вибору користувачем директорії для зберігання резервних копій. Програмне забезпечення повинне інтегруватися з операційною системою коректно і забезпечувати доступ до необхідних каталогів для збереження копій.

Усі вищезгадані елементи тестування спрямовані на те, щоб забезпечити, що програмне забезпечення працює так, як очікується від нього користувачем, і надає надійність і безпеку під час роботи з даними. (рис. 3.13)

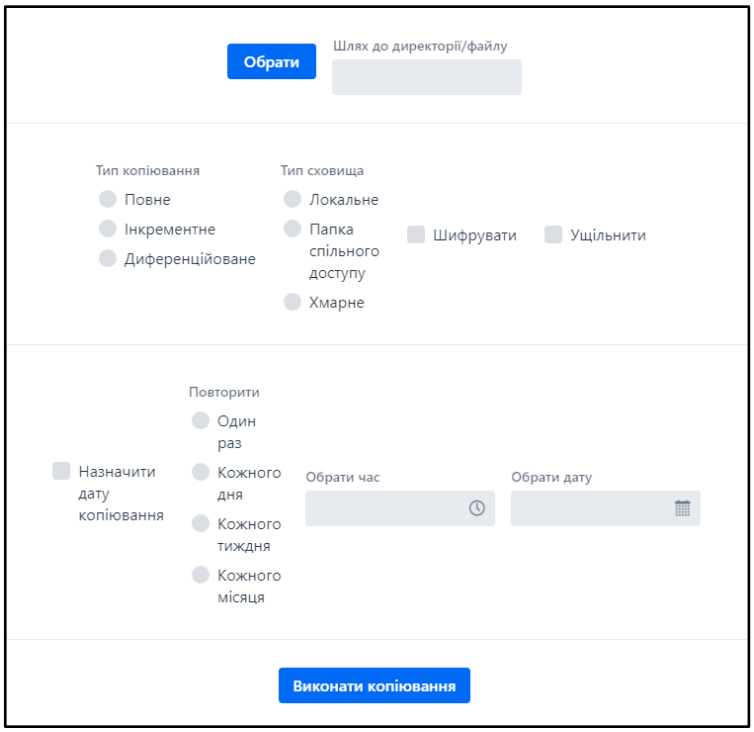



Рисунок 3.13 – Вигляд головного вікна з вибором налаштувань для резервного копіювання

Тестування вибору директорії для копіювання включає перевірку коректності роботи текстового поля та кнопки для вибору директорії. Після вибору директорії шлях до неї повинен відображатись у текстовому полі, і це значення повинно бути використано при виконанні копіювання (рис. 3.14).

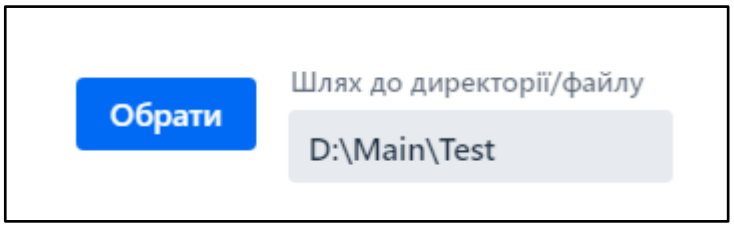



Рисунок 3.14 – Вигляд фрагмента головного вікна з вибором директорії для здійснення копіювання

Проведено тестування повного локального копіювання обраної користувачем директорії, що підлягає копіюванню. Результати тестування зображено на рисунку 3.15.

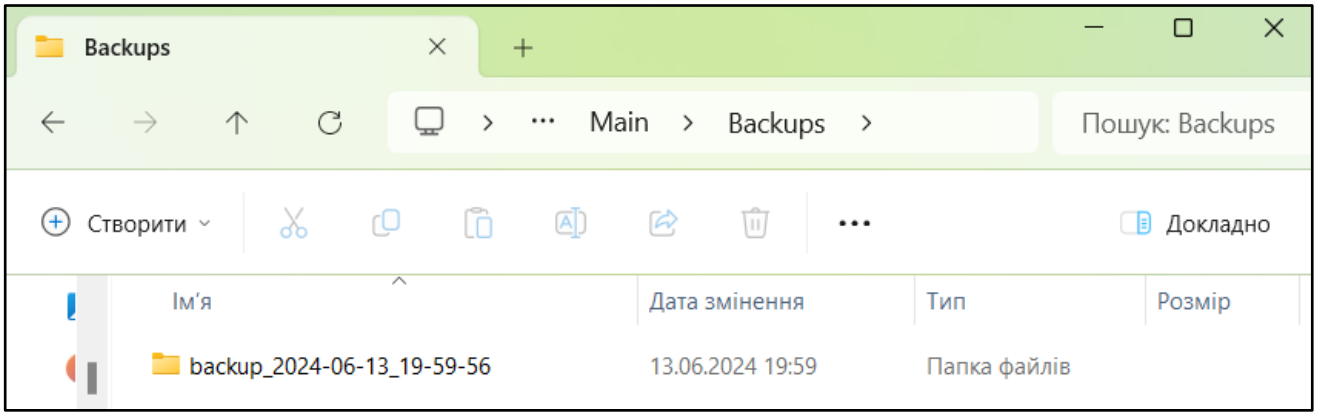



Рисунок 3.15 – Демонстрація наявності копії обраної директорії внаслідок виконання повного локального копіювання

Як видно на рисунку, було здійснено копіювання вказаної раніше користувачем директорії та поміщено дану копію її до директорії «Backups». Перевірено вміст директорії що підлягала копіюванню для визначення коректності роботи алгоритму (рис 3.16).

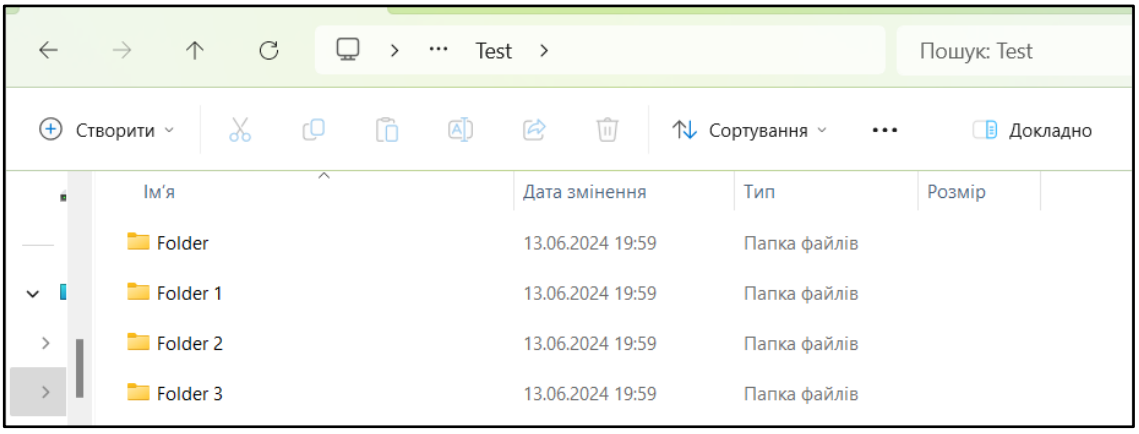



Рисунок 3.16 – Демонстрація цілісності вмісту обраної директорії після виконання повного локального копіювання

Протестовано алгоритм виконання диференціального копіювання. В даному типі повинно виконуватися копіювання лише тих файлів, які було змінено користувачем. Змінено вміст одного з файлів що знаходиться у папці «Folder2» та перевірено коректність роботи алгоритму (рис. 3.17).

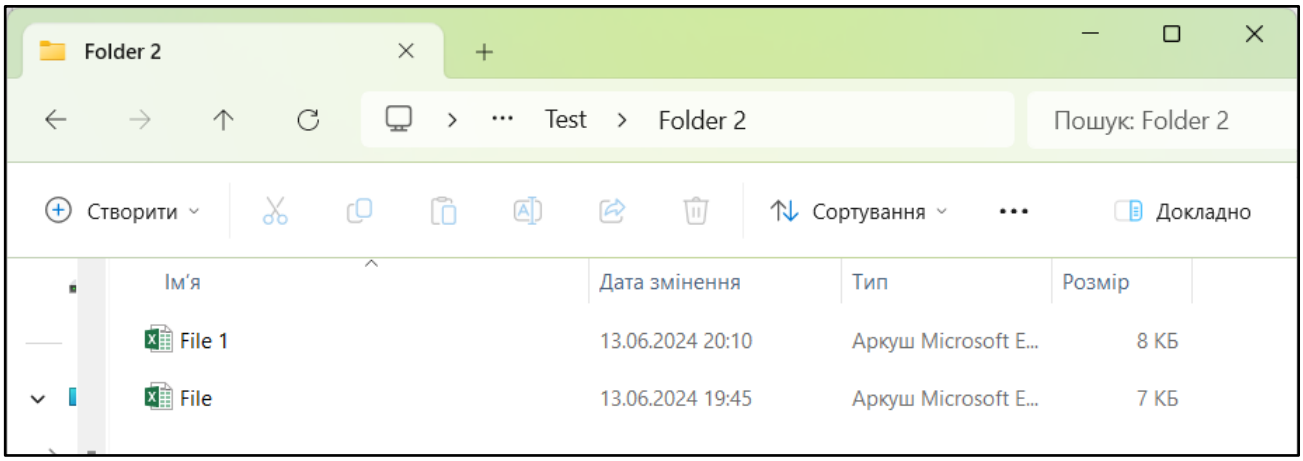



Рисунок 3.17 – Демонстрація відмінності дати змінення у файлі що було відредаговано

Як можна побачити на даному рисунку, дата змінення файлу з назвою «File1» є пізнішою за дати змінення файлу «File». Таким чином При виконанні диференціального копіювання має бути скопійовано лише файл з назвою «File1». Результати тестування зображено на рисунку 3.18.

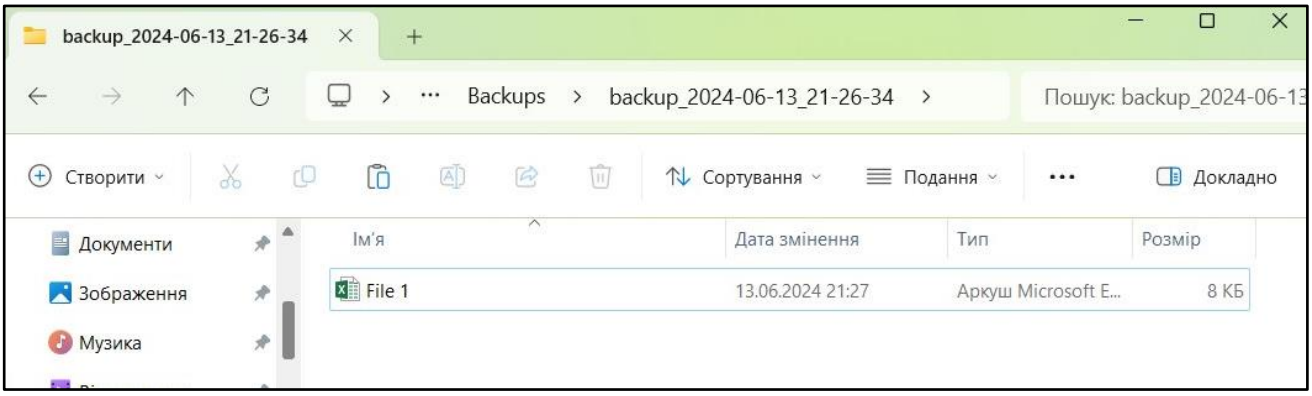



Рисунок 3.18 – Демонстрація створення копії лише відредагованого файлу внаслідок виконання диференціального копіювання

На даному рисунку показано створену копію лише відредагованого файлу, при цьому усі інші файли цільової папки не було скопійовано, а отже, алгоритм виконання даного типу копіювання працює коректно.

Для користувача має бути забезпечено можливість створення ущільненого архіву що містить в собі цільову папку, задля економного зберігання резервних копій на носії. Для цього обрано пункт налаштувань «Ущільнення даних» та повторно виконано копіювання директорії (рис. 3.19).

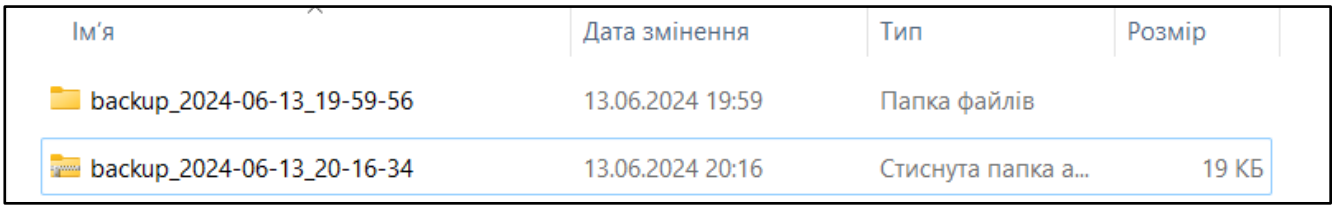



Рисунок 3.19 – Демонстрація наявності створеної резервної копії у форматі ущільненого архіву

Протестовано можливість шифрування директорії за допомогою алгоритму AES для забезпечення безпеки резервних копій що зберігаються на носії. Даний тест включає в себе перевірку як шифрування звичайної директорії, так і ущільненого архіву, при одночасному виборі користувачем пункту налаштувань шифрування і ущільнення даних (рис. 3.20).


Рисунок 3.20 – Демонстрація різних типів резервних копій

Як видно на рисунку, створено резервну копію в зашифрованому вигляді та у вигляді зашифрованого архіву, на що вказують розширення файлів «.dir.aes» та «.zip.aes».

Проведено тестування алгоритму збереження повної копії вказаної директорії у хмарному сховищі Google Cloud. Для цього за допомогою графічного інтерфейсу обрано метод копіювання «повне», а параметр «тип сховища» встановлено на значення «хмарне» Після чого натиснуто кнопку «Виконати копіювання». Результат роботи алгоритму наведено на рисунку 3.21.

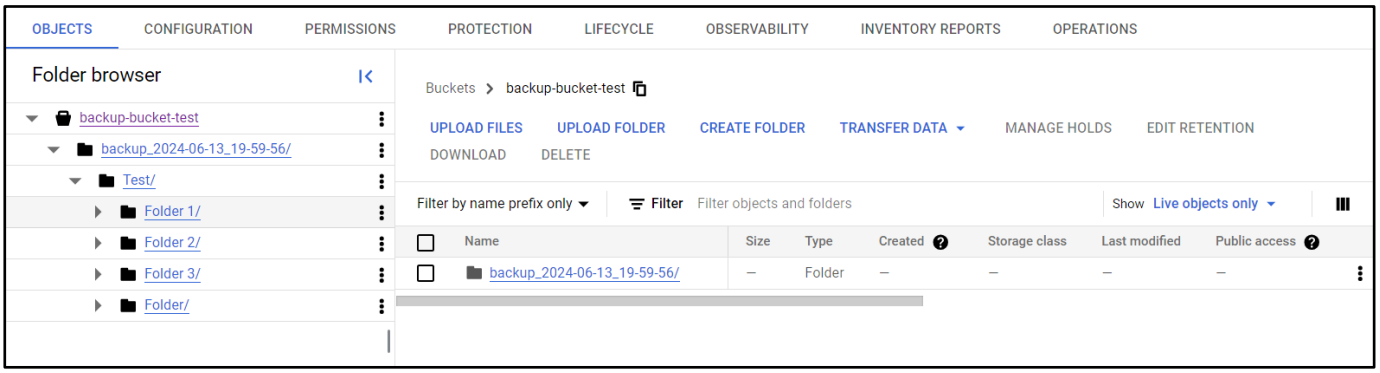



Рисунок 3.21 – Демонстрація наявності копії цільової директорії у хмарному сховищі

Створено зашифровану копію директорії та перевірено коректність її збереження у хмарному сховищі (рис. 3.22).

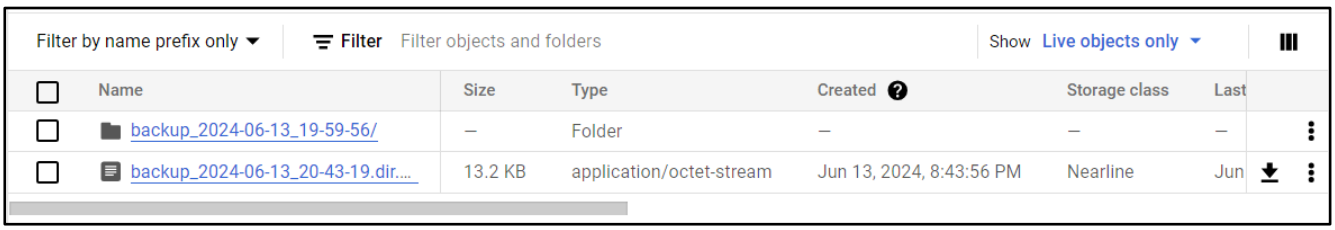



Рисунок 3.22 – Демонстрація наявності звичайної та зашифрованої резервної копії у хмарному сховищі

Отже, алгоритм збереження резервних копій працює коректно, жодних помилок в роботі при тестуванні даної операції не виявлено.

Проведено тестування створення резервної копії до папки спільного доступу (рис. 3.23).

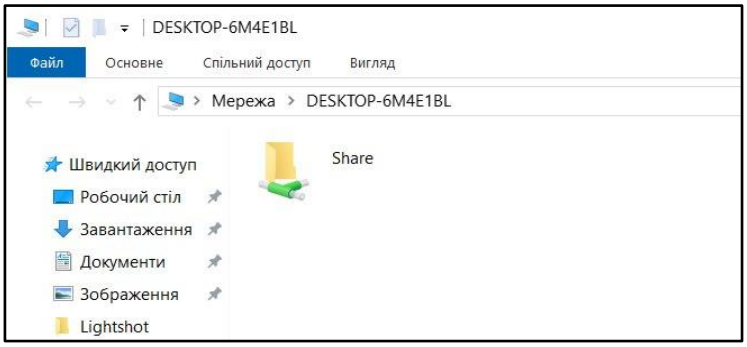



Рисунок 3.23 – Демонстрація папки спільного доступа

Виберемо потрібні налаштування на інтерфейсі та натиснемо кнопку «Виконати копіювання».

Створено резервну копію на іншій локальній машині, з якої був наданий доступ до сховища для збереження резервних копій (рис. 3.24).

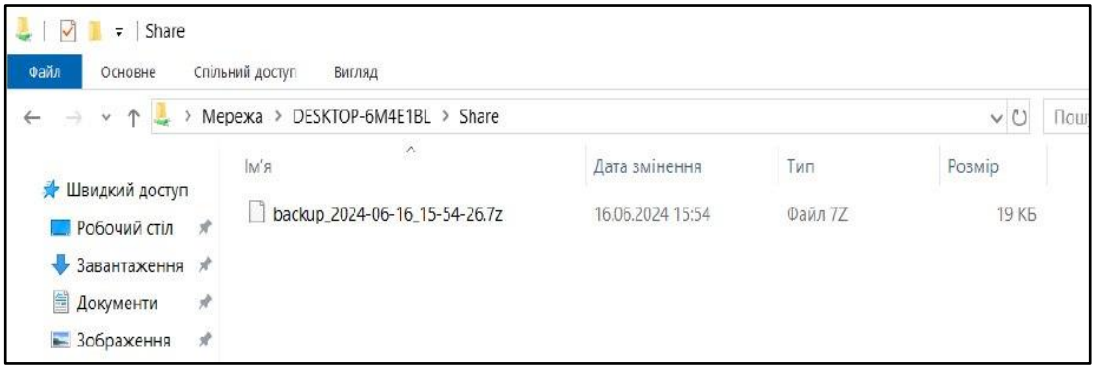



Рисунок 3.24 – Демонстрація успішно створеної копії у папці спільного доступа

Як видно з рисунку  копія створилася у розшареній папці, отже функція працює вірно.

Однією з ключових функцій програмного засобу є виконання відновлення даних з резервних копій при втраті оригінальних даних. Змінено вміст цільової директорії, чию резервну копію було раніше створено, шляхом видалення з диску однієї з папок яка в ній містилася (рис. 3.25).

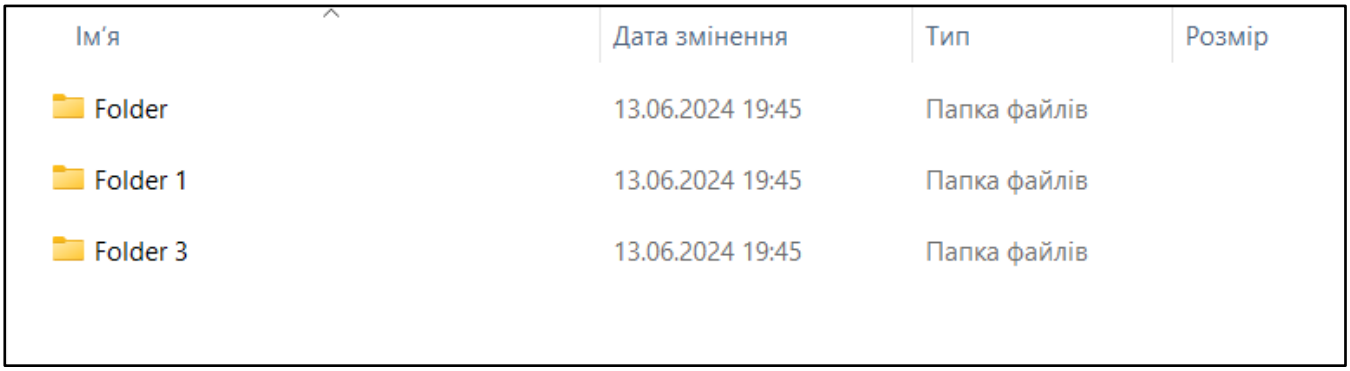



Рисунок 3.25 – Демонстрація видалення однієї з папок цільової директорії

Після цього Відкрито вікно «Recovery» в якому обрано резервну копію з локального, або хмарного сховища, що буде використовуватися в процесі відновлення (рис 3.26).

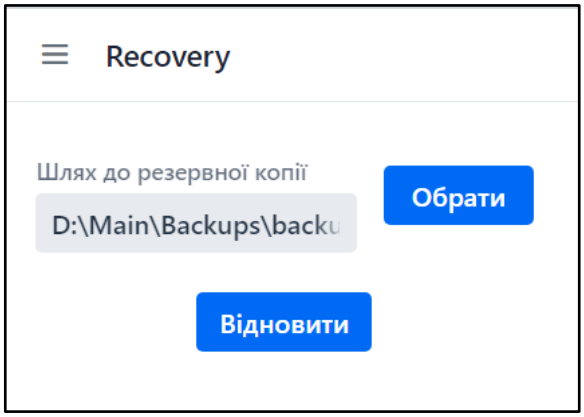



Рисунок 3.26 – Вигляд вікна відновлення даних з наявної резервної копії

Після виконання даних дій, відновлено стан цільової директорії на момент створення резервної копії (рис. 3.27).


Рисунок 3.27 – Вигляд вікна відновлення даних з наявної резервної копії

Як видно на рисунку, папка «Folder 2» з'явилася серед списку, що також видно за датою змінення папок, відповідно, алгоритм працює належним чином.

Проводиться тестування продуктивності, тобто тестування часу створення резервної копії (рис. 3.28).

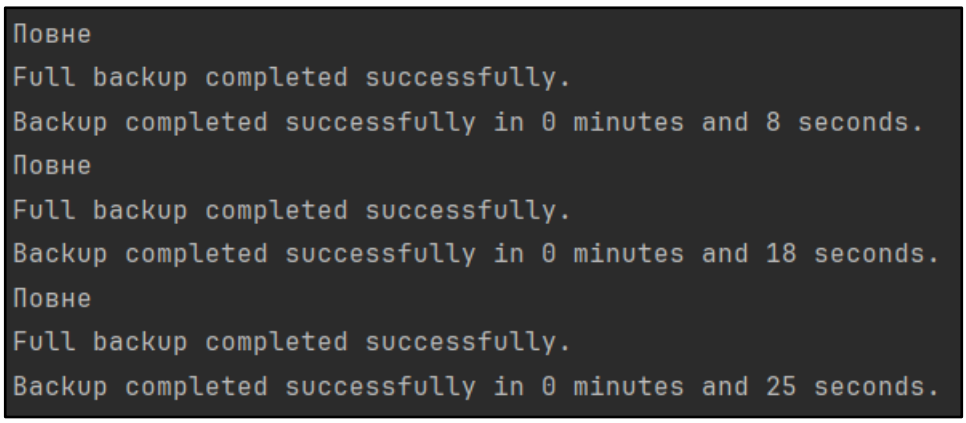



Рисунок 3.28 – Демонстрація часу створення копій

Таким чином, функціональне тестування даного програмного засобу охоплює всі ключові аспекти його роботи, від вибору та збереження налаштувань до безпосереднього виконання процесу резервного копіювання, забезпечуючи його надійну та коректну роботу відповідно до вимог користувача.

### 3.4 Висновки до третього розділу

У розділі 3 проведено детальне обґрунтування вибору інструментальних засобів для розробки програмного забезпечення та описано процес розробки основних алгоритмів програмного засобу для резервного копіювання.

Було розроблено основні алгоритми для виконання повного, інкрементального та диференціального резервного копіювання, а також алгоритми для ущільнення та шифрування даних. Крім того, реалізовано алгоритми для оновлення метаданих файлів. Особлива увага приділена забезпеченню безпеки даних шляхом використання алгоритму шифрування AES.

Проведено функціональне тестування розробленого програмного забезпечення, що включало перевірку коректної роботи всіх основних функцій, таких як вибір типу копіювання, вибір сховища, створення резервних копій, шифрування та ущільнення даних, а також відновлення даних з резервних копій.

# ВИСНОВКИ

У процесі виконання бакалаврської дипломної роботи було досліджено та реалізовано програмний засіб для резервного копіювання даних. Основною метою роботи було забезпечення цілісності та доступності даних за рахунок їх резервного копіювання.

У ході роботи були досягнуті наступні результати:

Проведено огляд існуючих технологій резервного копіювання, включаючи повне, диференційоване та інкрементальне копіювання. Визначено переваги та недоліки кожного з методів. Також досліджено типи сховищ та засоби резервного копіювання. За рахунок даного дослідження розроблено класифікацію систем резервного копіювання.

Сформульовано вимоги до програмного забезпечення, включаючи типи копіювання, планування резервного копіювання, шифрування даних, архівування, а також типи сховищ для резервних копій.

Розроблено архітектуру системи, яка включає ключові модулі для забезпечення всіх необхідних функцій. Описано взаємодію між компонентами системи.

Створено програмний засіб для резервного копіювання з використанням мови програмування Java. Засіб підтримує повне, диференційоване та інкрементальне копіювання, шифрування даних, архівування та збереження копій у локальних та хмарних сховищах.

Проведено тестування розробленого програмного засобу на різних сценаріях використання. Оцінено ефективність роботи засобу та його відповідність визначеним вимогам.

Розроблений програмний засіб забезпечує надійне та зручне резервне копіювання даних, що дозволяє мінімізувати ризики втрати інформації. Впровадження цього засобу може бути корисним для організацій та індивідуальних користувачів, які прагнуть забезпечити безпеку своїх даних.

## ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Куперштейн Л. М., Лесько О. В. Аналіз стратегій резервного копіювання даних // *Матеріали LII міжнародній науково-практичній інтернет-конференції студентів аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи»* ВНТУ, Вінниця, 2024 р. – Електрон. текст. дані. – 2024. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21712 (дата звернення 01.05.24).
2. Caseware Staff. Preventing common causes of data loss. July 20, 2023. URL : https://www.caseware.com/resources/blog/preventing-common-causes-of-data-loss/#:~:text=Top%20causes%20of%20data%20loss%201%20Human%20error, Hardware%20failure%20Machinery%20does%20not%20last%20forever.%20 (date of access: 02.05.2024).
3. CFI Team. Data Loss. URL: https://corporatefinanceinstitute.com/resources/data-science/data-loss/ (date of access: 03.06.2024).
4. Аросєв Г. Кібератака вірусу Petya: що відомо. 28.06.2017. URL: https://www.dw.com/uk/кібератака-вірусу-petya-що-відомо/a-39452258 (дата звернення: 05.06.2024).
5. How toy story 2 got deleted twice, once on accident, again on purpose. May 21, 2012. URL : https://thenextweb.com/news/how-pixars-toy-story-2-was-deleted-twice-once-by-technology-and-again-for-its-own-good (date of access: 05.05.2024).
6. Redfern E. The Yahoo Cyber Attack & What should you learn from it? March 1, 2023. URL : https://www.cashfloat.co.uk/blog/technology-innovation/yahoo-cyber-attack/ (date of access: 07.05.2024).
7. Fruhlinger J. Equifax data breach FAQ: What happened, who was affected, what was the impact? Feb 12, 2020 . URL: https://www.csoonline.com/article/567833/equifax-data-breach-faq-what-happened-who-was-affected-what-was-the-impact.html (date of access: 08.05.2024).

8. Robinson M. Myspace apologizes after losing 12 years' worth of music. March 18, 2019. URL: https://edition.cnn.com/2019/03/18/us/myspace-lost-12-years-music-uploads-apology-intl-scli/index.html#:~:text=Myspace%20apologizes%20after%20losing%2012%20years'%20worth%20of%20music&text="As%20a%20result%20of%20a,a%20statement%20on%20its%20website. (date of access: 09.05.2024).
9. Rud A. Пожежа в дата-центрі OVH SBG2 у Франції. 11.03.2021. URL: https://hyperhost.ua/info/uk/pozhezha-v-data-tsentri-ovh-sbg2-u-frantsii (дата звернення: 11.05.2024).
10. Як вибрати тип і спосіб резервного копіювання. 20.04.2023. URL: http://surl.li/uovqa (дата звернення: 11.06.2024).
11. Beard B. Full Backups. Beginning Backup and Restore for SQL Server. Berkeley, CA, 2018. pp. 3–21. DOI: https://doi.org/10.1007/978-1-4842-3456-3_1 (date of access: 12.05.2024).
12. Beard B. Differential Backups. Beginning Backup and Restore for SQL Server. Berkeley, CA, 2018. pp. 23–37. DOI: https://doi.org/10.1007/978-1-4842-3456-3_2 (date of access: 14.05.2024).
13. Qian C., Nakamura S., Nakagawa T. Optimal Backup Policies for a Database System with Incremental Backup. Electronics and Communications in Japan (Part III: Fundamental Electronic Science). 2002. Vol. 85, no 4. pp. 1–9. DOI: https://doi.org/10.1002/ecjc.1081 (date of access: 15.05.2024).
14. Data storage. / ed. by E. I. Inc. New York, NY, USA : Engineering Information, 2020. 183 p.
15. Kachhala K., Gangarde R. RAID (Redundant Array of independent Disks). International Journal of Engineering Trends and Technology. 2016. Vol. 35, no. 12. pp. 574–577. DOI: https://doi.org/10.14445/22315381/ijett-v35p316 (date of access: 15.05.2024).
16. RAM. Network Attached Storage Fundamentals. John Wiley & Sons Inc, 2003. 408 p.

17. Bigelow S. J. What Is a Storage Area Network? September 2020. URL: https://www.techtarget.com/searchstorage/definition/storage-area-network-SAN#:~:text=A%20storage%20area%20network%20(SAN,critical%20concerns%20for%20enterprise%20computing. (date of access: 16.05.2024).
18. Gulabani S. Getting Started with AWS. Practical Amazon EC2, SQS, Kinesis, and S3. Berkeley, CA, 2017. P. 1–22. DOI: https://doi.org/10.1007/978-1-4842-2841-8_1 (date of access: 16.06.2024).
19. Lichtendahl K. C., Andrasko J., Boatright B. Google Cloud Platform: Cloud Storage. SSRN Electronic Journal. 2022. DOI: https://doi.org/10.2139/ssrn.4133739 (date of access: 16.05.2024).
20. Azure Storage / J. Soh et al. Microsoft Azure. Berkeley, CA, 2020. P. 271–291. DOI: https://doi.org/10.1007/978-1-4842-5958-0_14 (date of access: 16.05.2024).
21. Todo Backup Tips to backup files and data. URL: https://www.easeus.com/todo-backup-guide/ (date of access: 17.05.2024).
22. Acronis Company Information and History. URL: https://www.acronis.com/en-us/company/ (date of access: 20.05.2024).
23. About Macrium Software. URL: https://www.macrium.com/our-story (date of access: 20.06.2024).
24. Aomei Backupper Pro review: All-in-one backup, now with online storage. URL: https://www.pcworld.com/article/394774/aomei-backupper-review.html (date of access: 21.05.2024).
25. Paragon Software. URL: https://www.paragon-software.com/ (date of access: 23.05.2024).
26. Jacobi J. Genie timeline 10 home review: time machine-like backup for windows. Aug 24, 2022. URL: https://www.pcworld.com/article/819876/genie-timeline-10-home-windows-backup-review.html (date of access: 24.05.2024).
27. NovaBACKUP Product Documentation. URL: https://www.novabackup.com/resources/product-documentation (date of access: 24.05.2024).

28. Getting Started with Comodo Backup. URL: https://www.comodo.com/home/backup-online-storage/backup-first-time-setup.php (date of access: 27.05.2024).
29. CobianSoft - The home of Cobian Backup. URL: https://www.cobiansoft.com/cobianbackup.html (date of access: 28.05.2024).
30. Hagos T. Beginning IntelliJ IDEA. Berkeley, CA : Apress, 2022. DOI: https://doi.org/10.1007/978-1-4842-7446-0 (date of access: 28.05.2024).
31. Spring Boot. URL: https://spring.io/projects/spring-boot (date of access: 05.06.2024).
32. Vaadin . The Web App Platform for Java. URL: https://vaadin.com/benefits (date of access: 05.06.2024).

# ДОДАТКИ

**Додаток А**

**Протокол перевірки**
**бакалаврської дипломної роботи**
**на наявність текстових запозичень**

Назва роботи: Програмний засіб для резервного копіювання даних

Автор роботи: Лесько Олексій Віталійович

Тип роботи: бакалаврська дипломна робота .

Підрозділ кафедра захисту інформації ФІТКІ .
(кафедра, факультет)

**Показники звіту подібності Unicheck**

Оригінальність – 98,1%. Схожість – 1,9%.

Аналіз звіту подібності (відмітити потрібне):

√ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Валентина КАПЛУН
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Олексій ЛЕСЬКО
(підпис) (прізвище,ініціали)

Керівник роботи _____ Леонід КУПЕРШТЕЙН
(підпис) (прізвище, ініціали)

# Додаток Б

# Код програмного засобу

## Файл BackupService.java

```
package com.example.application.services;

import com.vaadin.flow.component.notification.Notification;
import org.springframework.stereotype.Service;

import javax.crypto.Cipher;
import javax.crypto.CipherOutputStream;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import java.io.*;
import java.nio.file.*;
import java.nio.file.attribute.BasicFileAttributes;
import java.security.SecureRandom;
import java.time.Duration;
import java.time.Instant;
import java.util.Base64;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;

@Service
public class BackupService {
    private ExecutorService executor = Executors.newSingleThreadExecutor();

    private static final String LAST_FULL_BACKUP_TIME_FILE = "lastFullBackupTime.txt";
    private static final String LAST_ANY_BACKUP_TIME_FILE = "lastAnyBackupTime.txt";

    private static final String AES_ALGORITHM = "AES/CBC/PKCS5Padding";
    private static final String AES_KEY_ALGORITHM = "AES";
    private static final int AES_KEY_SIZE = 256;

    public void performBackup(
            Path source,
            Path target,
            String backupType,
            String storageType,
            boolean isEncrypted,
            boolean isCompressed
    ) {
        executor.submit(() -> {
            try {
                Instant start = Instant.now();
                switch (backupType) {
                    case "Повне":
                        fullBackup(source.toFile().getAbsolutePath(),
target.toFile().getAbsolutePath(), isEncrypted, isCompressed);
                        break;
                    case "Інкрементне":
                        incrementalBackup(source.toFile().getAbsolutePath(),
target.toFile().getAbsolutePath(), isEncrypted, isCompressed);
                        break;
                    case "Диференційоване":
                        differentialBackup(source.toFile().getAbsolutePath(),
target.toFile().getAbsolutePath(), isEncrypted, isCompressed);
                        break;
                    default:
```

```
                        throw new IllegalArgumentException("Невідомий тип копіювання: " +
backupType);
                }
                Instant end = Instant.now();
                Duration duration = Duration.between(start, end);
                System.out.println("Backup completed successfully in " +
duration.toMinutesPart() + " minutes and " + duration.toSecondsPart() + " seconds.");
            } catch (IOException e) {
                e.printStackTrace();
            }
        });
    }

    public static void fullBackup(String sourceDir, String destDir, boolean isEncrypted,
boolean isCompressed) throws IOException {
        File srcDir = new File(sourceDir);
        File destDirFile = new File(destDir, srcDir.getName());

        if (!destDirFile.exists()) {
            destDirFile.mkdirs();
        }

        String zipFilePath = destDirFile.getAbsolutePath() + ".zip";
        zipDirectory(srcDir, zipFilePath, isCompressed ? ZipOutputStream.DEFLATED :
ZipOutputStream.STORED);

        if (isEncrypted) {
            encryptFile(new File(zipFilePath));
        }

        updateLastBackupTime(LAST_FULL_BACKUP_TIME_FILE);
        updateLastBackupTime(LAST_ANY_BACKUP_TIME_FILE);

        System.out.println("Full backup completed successfully.");
    }

    public static void differentialBackup(String sourceDir, String destDir, boolean
isEncrypted, boolean isCompressed) throws IOException {
        if (isFullBackupDone()) {
            File srcDir = new File(sourceDir);
            File destDirFile = new File(destDir, srcDir.getName());

            if (!destDirFile.exists()) {
                destDirFile.mkdirs();
            }

            long lastFullBackupTime = getLastBackupTime(LAST_FULL_BACKUP_TIME_FILE);

            Files.walkFileTree(srcDir.toPath(), new SimpleFileVisitor<Path>() {
                @Override
                public FileVisitResult visitFile(Path file, BasicFileAttributes attrs)
throws IOException {
                    if (attrs.lastModifiedTime().toMillis() > lastFullBackupTime) {
                        Path destFile =
destDirFile.toPath().resolve(srcDir.toPath().relativize(file));
                        Files.copy(file, destFile, StandardCopyOption.REPLACE_EXISTING);
                    }
                    return FileVisitResult.CONTINUE;
                }
            });

            String zipFilePath = destDirFile.getAbsolutePath() + ".zip";
            zipDirectory(destDirFile, zipFilePath, isCompressed ? ZipOutputStream.DEFLATED
: ZipOutputStream.STORED);

            if (isEncrypted) {
                encryptFile(new File(zipFilePath));
            }

            updateLastBackupTime(LAST_ANY_BACKUP_TIME_FILE);
            System.out.println("Differential backup completed successfully.");
```

```
        } else {
            Notification.show("Будь ласка, введіть логін та пароль.");
        }
    }

    public static void incrementalBackup(String sourceDir, String destDir, boolean
isEncrypted, boolean isCompressed) throws IOException {
        if (isAnyBackupDone()) {
            File srcDir = new File(sourceDir);
            File destDirFile = new File(destDir, srcDir.getName());

            if (!destDirFile.exists()) {
                destDirFile.mkdirs();
            }

            long lastIncrementalBackupTime = getLastBackupTime(LAST_ANY_BACKUP_TIME_FILE);

            Files.walkFileTree(srcDir.toPath(), new SimpleFileVisitor<Path>() {
                @Override
                public FileVisitResult visitFile(Path file, BasicFileAttributes attrs)
throws IOException {
                    if (attrs.lastModifiedTime().toMillis() > lastIncrementalBackupTime) {
                        Path destFile =
destDirFile.toPath().resolve(srcDir.toPath().relativize(file));
                        Files.copy(file, destFile, StandardCopyOption.REPLACE_EXISTING);
                    }
                    return FileVisitResult.CONTINUE;
                }
            });

            String zipFilePath = destDirFile.getAbsolutePath() + ".zip";
            zipDirectory(destDirFile, zipFilePath, isCompressed ? ZipOutputStream.DEFLATED
: ZipOutputStream.STORED);

            if (isEncrypted) {
                encryptFile(new File(zipFilePath));
            }

            updateLastBackupTime(LAST_ANY_BACKUP_TIME_FILE);
            System.out.println("Incremental backup completed successfully.");
        }
    }

    private static void zipDirectory(File srcDir, String zipFilePath, int compressionLevel)
throws IOException {
        try (ZipOutputStream zos = new ZipOutputStream(new FileOutputStream(zipFilePath)))
{
            zos.setLevel(compressionLevel);
            Files.walkFileTree(srcDir.toPath(), new SimpleFileVisitor<Path>() {
                @Override
                public FileVisitResult visitFile(Path file, BasicFileAttributes attrs)
throws IOException {
                    try (FileInputStream fis = new FileInputStream(file.toFile())) {
                        ZipEntry zipEntry = new
ZipEntry(srcDir.toPath().relativize(file).toString());
                        zos.putNextEntry(zipEntry);

                        byte[] bytes = new byte[1024];
                        int length;
                        while ((length = fis.read(bytes)) >= 0) {
                            zos.write(bytes, 0, length);
                        }
                    }
                    return FileVisitResult.CONTINUE;
                }

                @Override
                public FileVisitResult preVisitDirectory(Path dir, BasicFileAttributes
attrs) throws IOException {
                    if (!srcDir.equals(dir)) {
```

```
                        zos.putNextEntry(new
ZipEntry(srcDir.toPath().relativize(dir).toString() + "/"));
                        zos.closeEntry();
                    }
                    return FileVisitResult.CONTINUE;
                }
            });
        }
    }

    private static void encryptFile(File file) throws IOException {
        SecretKey secretKey = generateAESKey();
        IvParameterSpec iv = generateIV();

        try (FileInputStream fis = new FileInputStream(file);
             FileOutputStream fos = new FileOutputStream(file.getAbsolutePath() + ".enc");
             CipherOutputStream cos = new CipherOutputStream(fos,
initCipher(Cipher.ENCRYPT_MODE, secretKey, iv))) {

            byte[] buffer = new byte[1024];
            int length;
            while ((length = fis.read(buffer)) > 0) {
                cos.write(buffer, 0, length);
            }
        }

        file.delete();

        // Save the key and IV for decryption (this example uses Base64 encoding for
simplicity)
        try (FileWriter keyWriter = new FileWriter(new File(file.getParent(),
"secret.key"));
             FileWriter ivWriter = new FileWriter(new File(file.getParent(), "iv.txt"))) {
            keyWriter.write(Base64.getEncoder().encodeToString(secretKey.getEncoded()));
            ivWriter.write(Base64.getEncoder().encodeToString(iv.getIV()));
        }
    }

    private static Cipher initCipher(int mode, SecretKey key, IvParameterSpec iv) throws
IOException {
        try {
            Cipher cipher = Cipher.getInstance(AES_ALGORITHM);
            cipher.init(mode, key, iv);
            return cipher;
        } catch (Exception e) {
            throw new IOException("Error initializing cipher", e);
        }
    }

    private static SecretKey generateAESKey() throws IOException {
        try {
            KeyGenerator keyGen = KeyGenerator.getInstance(AES_KEY_ALGORITHM);
            keyGen.init(AES_KEY_SIZE);
            return keyGen.generateKey();
        } catch (Exception e) {
            throw new IOException("Error generating AES key", e);
        }
    }

    private static IvParameterSpec generateIV() {
        byte[] iv = new byte[16];
        new SecureRandom().nextBytes(iv);
        return new IvParameterSpec(iv);
    }

    private static long getLastBackupTime(String fileName) {
        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
            return Long.parseLong(reader.readLine());
        } catch (IOException | NumberFormatException e) {
            return 0;
        }
```

```
    }

    private static void updateLastBackupTime(String fileName) {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName))) {
            writer.write(String.valueOf(System.currentTimeMillis()));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static boolean isFullBackupDone() {
        File file = new File(LAST_FULL_BACKUP_TIME_FILE);
        return file.exists() && file.length() > 0;
    }

    private static boolean isAnyBackupDone() {
        File file = new File(LAST_ANY_BACKUP_TIME_FILE);
        return file.exists() && file.length() > 0;
    }
}
```

## Файл MainView.java

```
package com.example.application.services;

import com.vaadin.flow.component.notification.Notification;
import org.springframework.stereotype.Service;

import javax.crypto.*;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;
import java.io.*;
import java.nio.file.*;
import java.nio.file.attribute.BasicFileAttributes;
import java.security.*;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.KeySpec;
import java.time.Duration;
import java.time.Instant;
import java.util.Base64;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;

@Service
public class BackupService {
    private ExecutorService executor = Executors.newSingleThreadExecutor();

    private static final String LAST_FULL_BACKUP_TIME_FILE = "lastFullBackupTime.txt";
    private static final String LAST_ANY_BACKUP_TIME_FILE = "lastAnyBackupTime.txt";

    private static final String AES_ALGORITHM = "AES/CBC/PKCS5Padding";
    private static final String AES_KEY_ALGORITHM = "AES";
    private static final int AES_KEY_SIZE = 256;
    private static final String PBKDF_ALGORITHM = "PBKDF2WithHmacSHA256";
    private static final int PBKDF_ITERATIONS = 65536;
    private static final int SALT_SIZE = 16;

    public void performBackup(
            Path source,
            Path target,
            String backupType,
            String storageType,
            boolean isEncrypted,
            boolean isCompressed
    ) {
        executor.submit(() -> {
            try {
                Instant start = Instant.now();
                switch (backupType) {
```

```
                    case "Повне":
                        fullBackup(source.toFile().getAbsolutePath(),
target.toFile().getAbsolutePath(), isEncrypted, isCompressed);
                        break;
                    case "Інкрементне":
                        incrementalBackup(source.toFile().getAbsolutePath(),
target.toFile().getAbsolutePath(), isEncrypted, isCompressed);
                        break;
                    case "Диференційоване":
                        differentialBackup(source.toFile().getAbsolutePath(),
target.toFile().getAbsolutePath(), isEncrypted, isCompressed);
                        break;
                    default:
                        throw new IllegalArgumentException("Невідомий тип копіювання: " +
backupType);
                }
                Instant end = Instant.now();
                Duration duration = Duration.between(start, end);
                System.out.println("Backup completed successfully in " +
duration.toMinutesPart() + " minutes and " + duration.toSecondsPart() + " seconds.");
            } catch (IOException e) {
                e.printStackTrace();
            }
        });
    }

    public static void fullBackup(String sourceDir, String destDir, boolean isEncrypted,
boolean isCompressed) throws IOException {
        File srcDir = new File(sourceDir);
        File destDirFile = new File(destDir, srcDir.getName());

        if (!destDirFile.exists()) {
            destDirFile.mkdirs();
        }

        String zipFilePath = destDirFile.getAbsolutePath() + ".zip";
        zipDirectory(srcDir, zipFilePath, isCompressed ? ZipOutputStream.DEFLATED :
ZipOutputStream.STORED);

        if (isEncrypted) {
            encryptFile(new File(zipFilePath), "password"); // Replace "password" with your
password
        }

        updateLastBackupTime(LAST_FULL_BACKUP_TIME_FILE);
        updateLastBackupTime(LAST_ANY_BACKUP_TIME_FILE);

        System.out.println("Full backup completed successfully.");
    }

    public static void differentialBackup(String sourceDir, String destDir, boolean
isEncrypted, boolean isCompressed) throws IOException {
        if (isFullBackupDone()) {
            File srcDir = new File(sourceDir);
            File destDirFile = new File(destDir, srcDir.getName());

            if (!destDirFile.exists()) {
                destDirFile.mkdirs();
            }

            long lastFullBackupTime = getLastBackupTime(LAST_FULL_BACKUP_TIME_FILE);

            Files.walkFileTree(srcDir.toPath(), new SimpleFileVisitor<Path>() {
                @Override
                public FileVisitResult visitFile(Path file, BasicFileAttributes attrs)
throws IOException {
                    if (attrs.lastModifiedTime().toMillis() > lastFullBackupTime) {
                        Path destFile =
destDirFile.toPath().resolve(srcDir.toPath().relativize(file));
                        Files.copy(file, destFile, StandardCopyOption.REPLACE_EXISTING);
                    }
```

```
                    return FileVisitResult.CONTINUE;
                }
            });

            String zipFilePath = destDirFile.getAbsolutePath() + ".zip";
            zipDirectory(destDirFile, zipFilePath, isCompressed ? ZipOutputStream.DEFLATED
: ZipOutputStream.STORED);

            if (isEncrypted) {
                encryptFile(new File(zipFilePath), "password"); // Replace "password" with
your password
            }

            updateLastBackupTime(LAST_ANY_BACKUP_TIME_FILE);
            System.out.println("Differential backup completed successfully.");
        } else {
            Notification.show("Будь ласка, введіть логін та пароль.");
        }
    }

    public static void incrementalBackup(String sourceDir, String destDir, boolean
isEncrypted, boolean isCompressed) throws IOException {
        if (isAnyBackupDone()) {
            File srcDir = new File(sourceDir);
            File destDirFile = new File(destDir, srcDir.getName());

            if (!destDirFile.exists()) {
                destDirFile.mkdirs();
            }

            long lastIncrementalBackupTime = getLastBackupTime(LAST_ANY_BACKUP_TIME_FILE);

            Files.walkFileTree(srcDir.toPath(), new SimpleFileVisitor<Path>() {
                @Override
                public FileVisitResult visitFile(Path file, BasicFileAttributes attrs)
throws IOException {
                    if (attrs.lastModifiedTime().toMillis() > lastIncrementalBackupTime) {
                        Path destFile =
destDirFile.toPath().resolve(srcDir.toPath().relativize(file));
                        Files.copy(file, destFile, StandardCopyOption.REPLACE_EXISTING);
                    }
                    return FileVisitResult.CONTINUE;
                }
            });

            String zipFilePath = destDirFile.getAbsolutePath() + ".zip";
            zipDirectory(destDirFile, zipFilePath, isCompressed ? ZipOutputStream.DEFLATED
: ZipOutputStream.STORED);

            if (isEncrypted) {
                encryptFile(new File(zipFilePath), "password"); // Replace "password" with
your password
            }

            updateLastBackupTime(LAST_ANY_BACKUP_TIME_FILE);
            System.out.println("Incremental backup completed successfully.");
        }
    }

    private static void zipDirectory(File srcDir, String zipFilePath, int compressionLevel)
throws IOException {
        try (ZipOutputStream zos = new ZipOutputStream(new FileOutputStream(zipFilePath)))
{
            zos.setLevel(compressionLevel);
            Files.walkFileTree(srcDir.toPath(), new SimpleFileVisitor<Path>() {
                @Override
                public FileVisitResult visitFile(Path file, BasicFileAttributes attrs)
throws IOException {
                    try (FileInputStream fis = new FileInputStream(file.toFile())) {
                        ZipEntry zipEntry = new
ZipEntry(srcDir.toPath().relativize(file).toString());
```

```
                        zos.putNextEntry(zipEntry);

                        byte[] bytes = new byte[1024];
                        int length;
                        while ((length = fis.read(bytes)) >= 0) {
                            zos.write(bytes, 0, length);
                        }
                    }
                    return FileVisitResult.CONTINUE;
                }

                @Override
                public FileVisitResult preVisitDirectory(Path dir, BasicFileAttributes
attrs) throws IOException {
                    if (!srcDir.equals(dir)) {
                        zos.putNextEntry(new
ZipEntry(srcDir.toPath().relativize(dir).toString() + "/"));
                        zos.closeEntry();
                    }
                    return FileVisitResult.CONTINUE;
                }
            });
        }
    }

    private static void encryptFile(File file, String password) throws IOException {
        SecretKey secretKey = generateAESKey(password);
        IvParameterSpec iv = generateIV();

        try (FileInputStream fis = new FileInputStream(file);
             FileOutputStream fos = new FileOutputStream(file.getAbsolutePath() + ".enc");
             CipherOutputStream cos = new CipherOutputStream(fos,
initCipher(Cipher.ENCRYPT_MODE, secretKey, iv))) {

            byte[] buffer = new byte[1024];
            int length;
            while ((length = fis.read(buffer)) > 0) {
                cos.write(buffer, 0, length);
            }
        }

        file.delete();

        // Save the key and IV for decryption (this example uses Base64 encoding for
simplicity)
        try (FileWriter keyWriter = new FileWriter(new File(file.getParent(),
"secret.key"));
             FileWriter ivWriter = new FileWriter(new File(file.getParent(), "iv.txt"))) {
            keyWriter.write(Base64.getEncoder().encodeToString(secretKey.getEncoded()));
            ivWriter.write(Base64.getEncoder().encodeToString(iv.getIV()));
        }
    }

    private static Cipher initCipher(int mode, SecretKey key, IvParameterSpec iv) throws
IOException {
        try {
            Cipher cipher = Cipher.getInstance(AES_ALGORITHM);
            cipher.init(mode, key, iv);
            return cipher;
        } catch (Exception e) {
            throw new IOException("Error initializing cipher", e);
        }
    }

    private static SecretKey generateAESKey(String password) throws IOException {
        try {
            byte[] salt = new byte[SALT_SIZE];
            SecureRandom random = new SecureRandom();
            random.nextBytes(salt);
```

```
            KeySpec spec = new PBEKeySpec(password.toCharArray(), salt, PBKDF_ITERATIONS,
AES_KEY_SIZE);
            SecretKeyFactory factory = SecretKeyFactory.getInstance(PBKDF_ALGORITHM);
            byte[] keyBytes = factory.generateSecret(spec).getEncoded();

            // Save the salt for decryption (this example uses Base64 encoding for
simplicity)
            try (FileWriter saltWriter = new FileWriter(new File("salt.txt"))) {
                saltWriter.write(Base64.getEncoder().encodeToString(salt));
            }

            return new SecretKeySpec(keyBytes, AES_KEY_ALGORITHM);
        } catch (NoSuchAlgorithmException | InvalidKeySpecException e) {
            throw new IOException("Error generating AES key", e);
        }
    }

    private static IvParameterSpec generateIV() {
        byte[] iv = new byte[16];
        new SecureRandom().nextBytes(iv);
        return new IvParameterSpec(iv);
    }

    private static long getLastBackupTime(String fileName) {
        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
            return Long.parseLong(reader.readLine());
        } catch (IOException | NumberFormatException e) {
            return 0;
        }
    }

    private static void updateLastBackupTime(String fileName) {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName))) {
            writer.write(String.valueOf(System.currentTimeMillis()));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static boolean isFullBackupDone() {
        File file = new File(LAST_FULL_BACKUP_TIME_FILE);
        return file.exists() && file.length() > 0;
    }

    private static boolean isAnyBackupDone() {
        File file = new File(LAST_ANY_BACKUP_TIME_FILE);
        return file.exists() && file.length() > 0;
    }
}
```

**Додаток В**

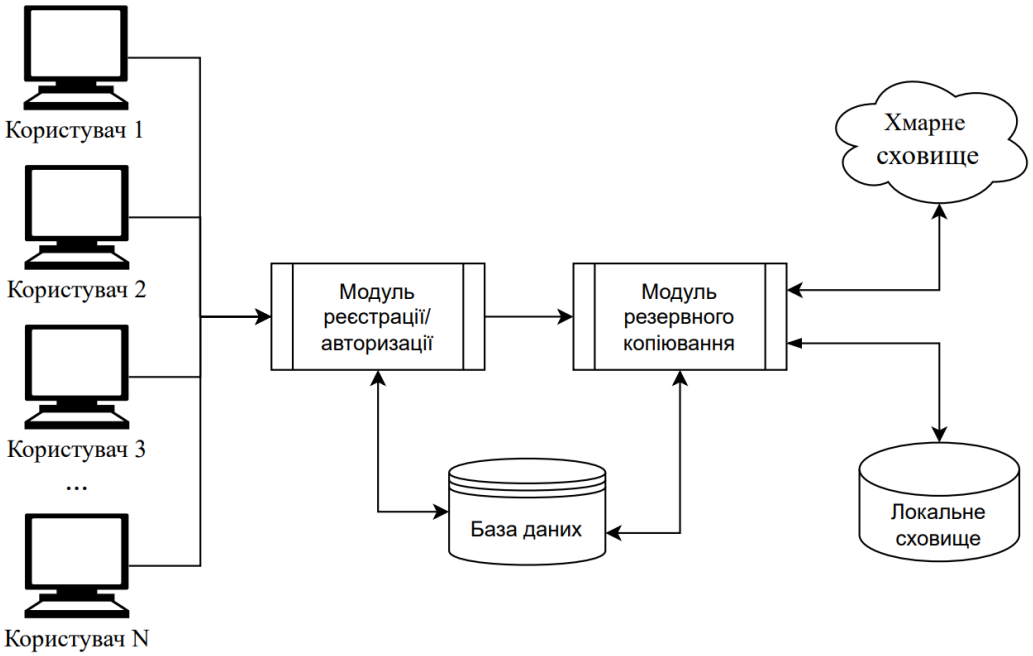
**ІЛЮСТРАТИВНА ЧАСТИНА**

ПРОГРАМНИЙ ЗАСІБ ДЛЯ РЕЗЕРВНОГО КОПІЮВАННЯ ДАНИХ

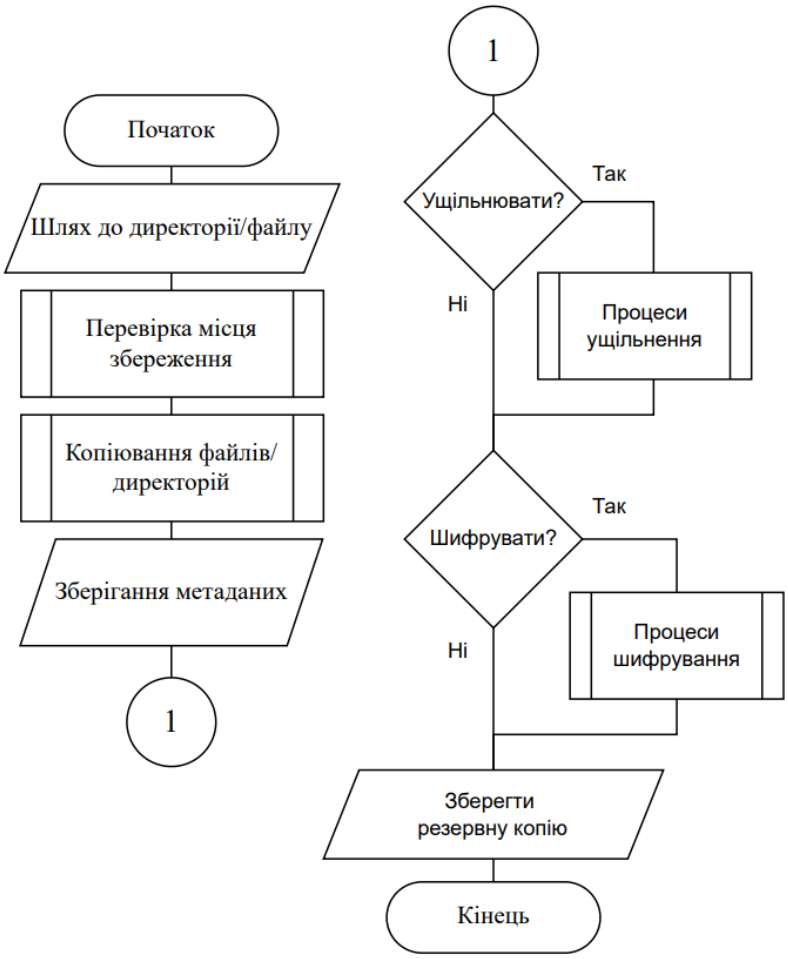
Виконав: студент 4 курсу групи 1БС-20б
спеціальності 125 Кібербезпека
_____Олексій ЛЕСЬКО
14 червня 2024 р.

Керівник: к. т. н., доцент каф. ЗІ
_____Леонід КУПЕРШТЕЙН
14 червня 2024 р.

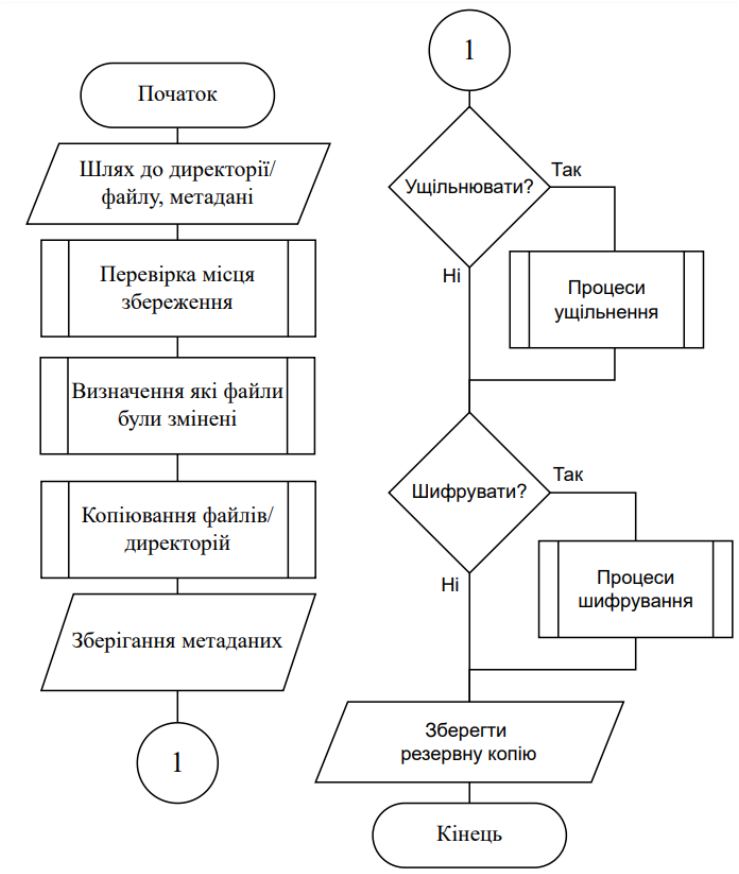
# ВИГЛЯД АРХІТЕКТУРИ СИСТЕМИ



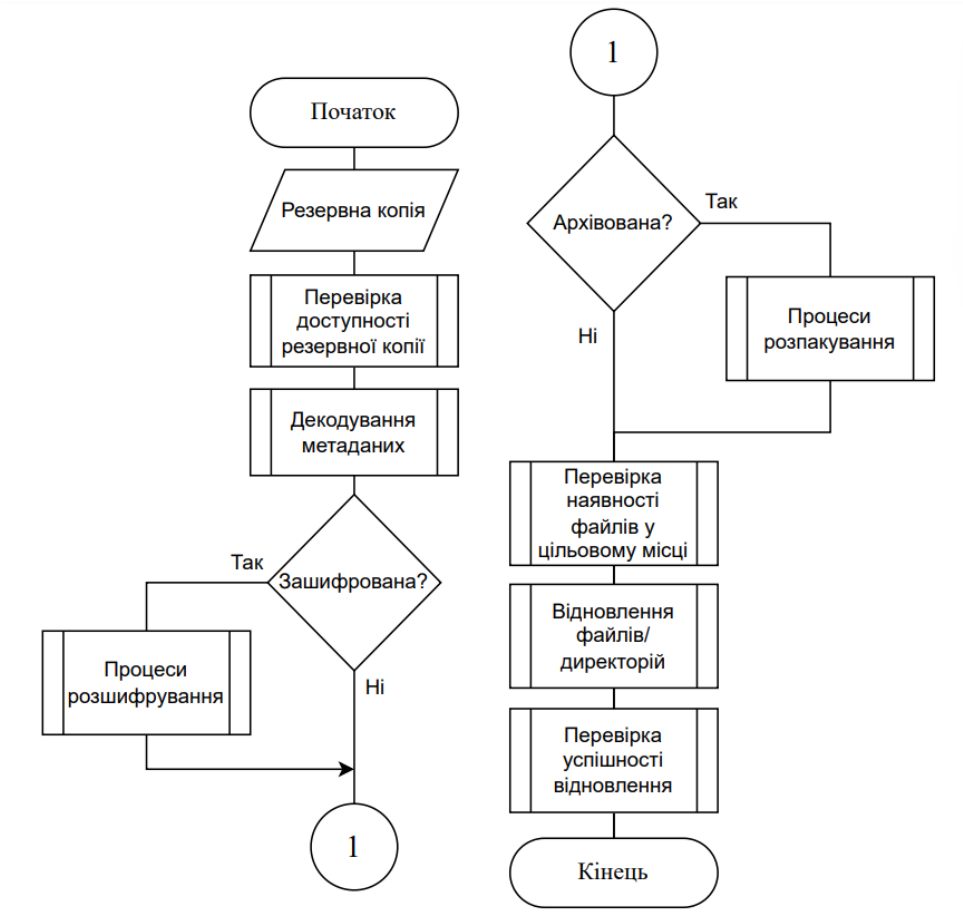
СТРУКТУРА АЛГОРИТМУ ПОВНОГО РЕЗЕРВНОГО КОПІЮВАННЯ



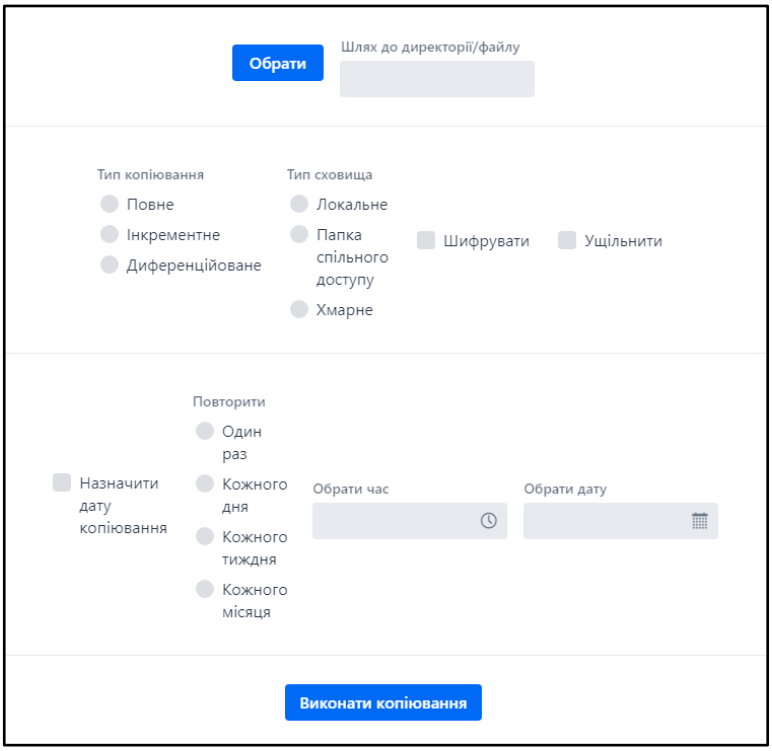
# СТРУКТУРА АЛГОРИТМУ ДИФЕРЕНЦІЙОВАНОГО РЕЗЕРВНОГО КОПІЮВАННЯ



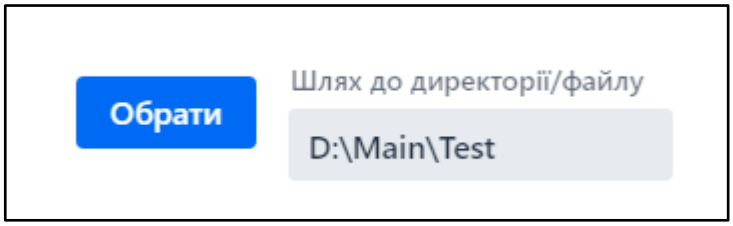
СТРУКТУРА АЛГОРИТМУ ВІДНОВЛЕННЯ КОПІЇ



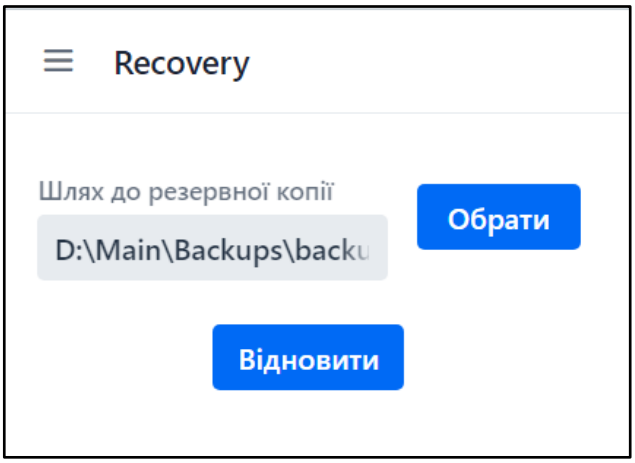
# ФРАГМЕНТИ ІНТЕРФЕЙСУ




Вигляд головного вікна з вибором налаштувань для резервного копіювання

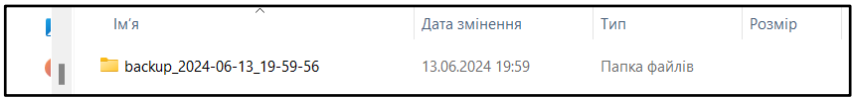



Вигляд фрагмента головного вікна з вибором директорії для здійснення копіювання

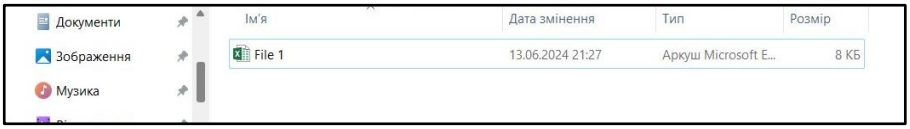



Вигляд вікна відновлення даних з наявної резервної копії

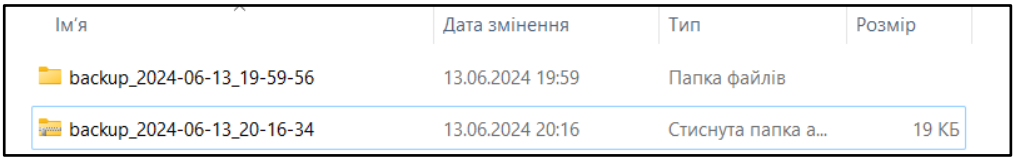
# ФРАГМЕНТ ТЕСТУВАННЯ




Демонстрація наявності копії обраної директорії внаслідок виконання повного локального копіювання




Демонстрація створення копії лише відредагованого файлу внаслідок виконання диференціального копіювання




Демонстрація наявності створеної резервної копії у форматі ущільненого архіву


Демонстрація різних типів резервних копій

```
Повне
Full backup completed successfully.
Backup completed successfully in 0 minutes and 8 seconds.
Повне
Full backup completed successfully.
Backup completed successfully in 0 minutes and 18 seconds.
Повне
Full backup completed successfully.
Backup completed successfully in 0 minutes and 25 seconds.
```

Демонстрація часу копіювання

АКТ ВПРОВАДЖЕННЯ

ТОВ «ДСофтвер»
Україна, м. Вінниця,
вул. 600-річчя, 17

**АКТ**
**ВПРОВАДЖЕННЯ**

Цим актом підтверджується, що окремі результати бакалаврської дипломної роботи на тему «Програмний засіб для резервного копіювання», яку виконав студент групи 1БС-20б Лесько Олексій за спеціальністю 125 «Кібербезпека», впроваджені в діяльність ТОВ «ДСофтвер». Цей документ не є підставою для фінансових розрахунків.

Дата 17.06.24

Директор _____ Бондар Н.П.

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ «ДСОФТВЕР» «DSOFTWARE» Limited Liability Company 41591566 м. Вінниця Україна