

Детектор Плагиата v. 2867 - Отчёт оригинальности: 15.06.2025 16:14:11

Проанализированный документ: дипломна_Люта.docx Лицензия: ВОЛОДИМИР МАТІЄВСЬКИЙ

Тип поиска: Поиск переписанного Язык: Uk
Тип проверки: Интернет
ТЭЕ и кодировка: DocX n/a

Детальный анализ тела документа:

Диаграмма соотношения частей:



Граф распределения зон:



Источники плагиата: 47

	6%		743	1. http://imit.luguniv.edu.ua/sites/default/files/docs/OK28.Vykonannya-kvalifikatsiynoyi-roboty-bakalavra.pdf
	6%		705	2. http://imit.luguniv.edu.ua/sites/default/files/docs/Vykonannya-kvalifikatsiynoyi-roboty-bakalavra.pdf
	1%		138	3. https://leadpanda.media/blog/rezervne-kopiyuvannya-danix-yak-kerivati-bekapami/

Детали обработанных ресурсов: 238 - ОК / 6 - Ошибка

Важные замечания:

Википедия:	Google Книги:	Сервисы платных работ:	Античит:
[не обнаружено]	[не обнаружено]	[не обнаружено]	[не обнаружено]

Античит-отчет UACE:

1. Статус: Анализатор Включен Нормализатор Включен сходство символов установлено на 100%
2. Обнаруженный процент загрязнения UniCode: 3,6% с лимитом: 4%
3. Документ не нормализован: процент не достигнут 5%
4. Все подозрительные символы будут отмечены фиолетовым цветом: Abcd...
5. Найдены невидимые символы: 0
Рекомендации по оценке: Никаких особых действий не требуется. Документ в порядке.
Алфавитная статистика и анализ символов:

 Активные ссылки (URL-адреса, извлеченные из документа):

URL не найдены

 Исключённые ресурсы:













URL не найдены

 Включённые ресурсы:

URL не найдены

 Детальний аналіз документа:

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ДЕРЖАВНИЙ ЗАКЛАД

	Цитування: 0,04%	id: 1
„ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА”		
	Обнаружен Плагиат: 0,08% http://imit.luguniv.edu.ua/sites/default... + 2 ресурсів!	id: 2
Навчально-науковий інститут математики та інформаційних технологій Кафедра інформаційних технологій та систем		
Люта Марія Миколаївна АВТОМАТИЗАЦІЯ РЕЗЕРВНОГО КОПІЮВАННЯ ФАЙЛІВ У ЛОКАЛЬНІЙ МЕРЕЖІ З ВИКОРИСТАННЯМ PYTHON ТА FTP кваліфікаційна робота здобувача вищої освіти першого (бакалаврського) рівня освітньої програми		
	Цитування: 0,05%	id: 3
« Комп’ютерні науки та інформаційні технології »		
за спеціальністю 122 Комп’ютерні науки Особистий підпис_____ Марія Люта Науковий керівникМикола СЕМЕНОВ к. пед. н., зав. кафедри		
	Обнаружен Плагиат: 0,13% http://imit.luguniv.edu.ua/sites/default... + 2 ресурсів!	id: 4
інформаційних технологій та систем Завідувач кафедри Микола СЕМЕНОВ Полтава – 2025 Міністерство освіти і науки України Державний заклад		
	Цитування: 0,04%	id: 5
„Луганський національний університет імені Тараса Шевченка”		
	Обнаружен Плагиат: 0,28% http://imit.luguniv.edu.ua/sites/default... + 3 ресурсів!	id: 6
Факультет (інститут) Навчально-науковий інститут математики та інформаційних технологій Кафедра Кафедра математики та інформатики Освітній ступень Бакалавр Напрям підготовки (спеціальність) 122 Комп’ютерні науки (код, назва) Галузь знань 12, Інформаційні технології (код, назва) ЗАТВЕРДЖУЮ Завідувач кафедри МІ (підпис)(ініціали, прізвище)		
	Цитування: 0,01%	id: 7
“ ”		
	Обнаружен Плагиат: 0,39% http://imit.luguniv.edu.ua/sites/default... + 5 ресурсів!	id: 8
2025 р. ЗАВДАННЯ НА БАКАЛАВРСЬКУ РОБОТУ Лютої Марії Миколаївни (прізвище, ім’я, по батькові) Тема проекту (роботи) Автоматизація резервного копіювання файлів у локальній мережі з використанням Python та FTP Керівник кваліфікаційної роботиСеменов М.А., К. пед. н., зав. кафедри інформаційних технологій та систем (прізвище, ініціали, науковий ступінь, вчене звання) затверджена наказом по університетувід		
	Цитування: 0,01%	id: 9
“ ”		
2		
	Обнаружен Плагиат: 0,39% http://imit.luguniv.edu.ua/sites/default... + 5 ресурсів!	id: 10
025 року № Строк подання студентом проекту (роботи) Вихідні дані до роботи програма створена на Python . Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) визначення та особливості мови програмування Python , використання FTP серверу Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) Слайди презентації Консультанти розділів проекту (роботи) Розділ Прізвище, ініціали та посада		
консультанта Підпис, дата завдання видав завдання прийняв Дата видачі завдання		
	Цитування: 0,01%	id: 11
“ ”		
202 р. КАЛЕНДАРНИЙ ПЛАН № з/п Назва етапів дипломного проекту (роботи) Строк виконання етапів проекту (роботи) Примітка 1		
	Обнаружен Плагиат: 1,26% http://imit.luguniv.edu.ua/sites/default... + 2 ресурсів!	id: 12

Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника. До 1 березня 2 Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно-експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень. До 20 березня 3 Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником. До 1 квітня 4 Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання. До 15 квітня 5 Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи. До 30 квітня 6 Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації. До 30 травня 7 Попередній захист роботи на кафедрі травень 8 Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії За 10 днів до державної атестації 9 Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом. За 5 днів до державної атестації Здобувач вищої освіти М.М. Люта підпис (ініціали, прізвище) Керівник

проекту (роботи) М.А. Семенов АНОТАЦІЯ Люта М.М. Тема: автоматизація резервного копіювання файлів у локальній мережі з використанням [Python](#) та [FTP](#). Спеціальність: 122

Цитування: **0,01%**

id: 13

„Комп’ютерні науки”

Обнаружен Плагиат: **0,14%** <http://imit.luguniv.edu.ua/sites/default...>

id: 14

Установа: ДЗ ЛНУ імені Т.Шевченка, 202р. Кваліфікаційна робота містить: 71 с., 1 додат., 17 рис., 23 джерела. Об’єкт дослідження

– застосунок на [Python](#). Предмет дослідження – розробка застосунку мовою програмування [Python](#). Мета роботи – розробка застосунку мовою програмування [Python](#). Результати роботи. Розроблений застосунок мовою програмування [Python](#) та з використанням [FTP](#)-сервера. Висновки. Обґрунтовано вибір інструментів та технологій, практична реалізація програми з використанням мови програмування [Python](#). Ключові слова. [Python](#), [FTP](#), ЗАСТОСУНОК, ЛОКАЛЬНИЙ, СЕРВЕР, РЕЗЕРВ, КОПІЯ, ВІДДАЛЕНИЙ ДОСТУП. [ABSTRACT Lyuta M.M.](#)

Topic: [Automation of File Backup in a Local Network Using Python and FTP](#)

Specialty: 122

Цитування: **0,01%**

id: 15

"Computer Science"

Institution: [Dnipro National University named after T. Shevchenko](#), 2021. [The qualification thesis includes: 71 pages, 1 appendix, 17 figures, 23 sources.](#)

Object of study: [Python application.](#)

Subject of study: [Development of an application using Python programming language.](#)


Goal of the work: [Development of an application using the Python programming language.](#)

Results: [The application was developed using Python programming language.](#)

Conclusions: [The choice of tools and technologies was justified, and the practical implementation of the program was carried out using Python.](#) Keywords: [Python](#), [FTP](#), [APPLICATION](#), [LOCAL](#), [SERVER](#), [BACKUP](#), [COPY](#), [REMOTE ACCESS](#).


ЗМІСТ ВСТУП 3 РОЗДІЛ 1. ОГЛЯД ТЕХНОЛОГІЙ ТА ПРОТОКОЛІВ 5 1.1. Основи резервного копіювання: типи та методи 5 1.2. Огляд протоколу [FTP](#): особливості та можливості 14 1.3. Використання [Python](#) для роботи з [FTP](#) 24 1.4. Інші альтернативи автоматизації резервного копіювання 28 РОЗДІЛ 2. РОЗРОБКА СИСТЕМИ АВТОМАТИЗОВАНОГО РЕЗЕРВНОГО КОПІЮВАННЯ 33 2.1. Архітектура системи резервного копіювання 33 2.2. Вибір програмних засобів та бібліотек [Python](#) 39 2.3. Реалізація клієнтської частини: створення та налаштування скрипта 41 2.4. Налаштування [FTP](#)-сервера для зберігання резервних копій 44 2.5. Автоматизація процесу копіювання та планування завдань 46 РОЗДІЛ 3. ТЕСТУВАННЯ ТА ОПТИМІЗАЦІЯ 48 3.1. Методологія тестування системи 48 3.2. Оцінка продуктивності та аналіз помилок 51 3.3. Оптимізація швидкості передачі даних 52 3.4. Захист та безпека переданих файлів 54 ВИСНОВКИ 57 СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 59 ДОДАТКИ 63 ВСТУП У сучасному світі інформація стала основним активом будь-якої організації, і її збереження та захист є ключовими аспектами

для безперебійної роботи інформаційних систем. Технології обробки даних, зокрема в організаціях, стають все більш складними і об'ємними, що підвищує ризик втрати важливої інформації. Випадкова втрата даних може призвести до серйозних наслідків: від фінансових збитків до втрати репутації компанії. Тому питання резервного копіювання інформації та забезпечення її надійного збереження набуває особливої актуальності. Одним з основних способів захисту даних є створення резервних копій. Однак традиційні методи резервного копіювання, такі як копіювання на фізичні носії або навіть зберігання даних на віддалених серверах, часто вимагають значних ресурсів і постійного втручання з боку користувача. Крім того, багато організацій не мають автоматизованих систем для резервного копіювання, що збільшує ймовірність людських помилок і знижує ефективність процесу. Автоматизація резервного копіювання є одним з найбільш ефективних способів мінімізації цих ризиків. Використання програмування для автоматичного виконання завдань резервного копіювання дозволяє значно підвищити надійність і ефективність цього процесу. Мова програмування **Python**, завдяки своїй простоті та багатому набору бібліотек, є ідеальним інструментом для вирішення цих задач. Використання протоколу **FTP** для зберігання резервних копій дозволяє створити безпечну і доступну систему зберігання даних, яка може бути використана як у локальних мережах, так і в хмарних середовищах. Таким чином, автоматизація процесу резервного копіювання файлів у локальній мережі з використанням **Python** і **FTP** є актуальним завданням, яке дозволяє вирішити проблему надійного зберігання даних, зменшити ризики втрати важливої інформації і підвищити ефективність роботи з резервними копіями. Це має особливу важливість для організацій будь-якого масштабу, де інформація є критично важливим ресурсом. Метою даної дипломної роботи є розробка рішення для автоматизації процесу

 **Обнаружен Плагиат: 0,36%** <https://www.morningdough.com/uk/a...> + 5 ресурсов! id: 16

резервного копіювання файлів у локальній мережі з використанням мови програмування **Python** і протоколу **FTP**. Такий підхід дозволяє створити надійний механізм для збереження даних без постійного втручання користувача, а також забезпечити високий рівень безпеки і ефективності процесу. Основними завданнями роботи є: Розробка програми для автоматичного резервного копіювання файлів у


локальній мережі з використанням **Python**. Налаштування **FTP**-сервера для зберігання резервних копій. Створення графічного інтерфейсу для полегшення використання програми кінцевим користувачем. Тестування та оптимізація процесу для забезпечення його ефективності та надійності. Робота має практичну значимість, оскільки розроблене рішення дозволяє значно знизити ризик втрати даних і автоматизувати процес резервного копіювання в організаціях або на локальних мережах. Вона також може бути використана для подальших досліджень та вдосконалення механізмів резервного копіювання в реальних умовах. РОЗДІЛ 1. ОГЛЯД ТЕХНОЛОГІЙ ТА ПРОТОКОЛІВ Основи резервного копіювання: типи та методи Резервне копіювання (бекап) є важливою частиною стратегії захисту даних, що дозволяє зберігати копії важливої інформації та відновлювати її в разі втрати чи пошкодження. Існують різні типи та методи резервного копіювання, залежно від вимог до безпеки, швидкості та простору для зберігання. Типи резервного копіювання наведено на рис. 1.1. Рисунок 1.1 – Типи резервного копіювання Повне резервне копіювання. Копіюється весь вміст даних або системи. Переваги: швидке відновлення, оскільки вся інформація вже збережена в одному архіві. Недоліки: займає багато місця для зберігання і може зайняти значний час для виконання.

 **Обнаружен Плагиат: 0,64%** <https://www.hostpark.ua/ua/rezervne...> + 5 ресурсов! id: 17

Інкрементне резервне копіювання. Копіюються тільки ті дані, які були змінені або додані після останнього резервного копіювання (повного або інкрементного). Переваги: економить місце для зберігання і час на виконання. Недоліки: для відновлення потрібно спочатку відновити останнє повне копіювання, а потім застосувати всі інкрементні копії. Диференціальне резервне копіювання. Копіюються всі зміни з моменту останнього повного резервного копіювання. Переваги: менш затратне, ніж повне копіювання, але забезпечує швидке відновлення. Недоліки: займає більше місця, ніж інкрементне копіювання, оскільки кожне нове диференціальне копіювання включає всі зміни з моменту останнього повного резервного копіювання.


Таким чином, можна визначити, що: - повне резервне копіювання - це копіювання всього вмісту даних або системи. Це дозволяє швидко відновлювати інформацію, оскільки вся інформація зберігається в одному архіві. Однак цей метод займає багато місця для

зберігання та може зайняти значний час для виконання; - інкрементне резервне копіювання - копіюються лише ті дані, що були змінені або додані після останнього резервного копіювання. Це економить місце для зберігання і зменшує час виконання. Проте для відновлення потрібно спочатку відновити останнє повне копіювання, а потім застосувати всі інкрементні копії; - диференціальне

 **Обнаружен Плагиат: 0,31%** <https://www.hostpark.ua/ua/rezervne...> + 4 ресурсов! id: 18

резервне копіювання - копіюються всі зміни з моменту останнього повного резервного копіювання. Цей метод менш затратний, ніж повне копіювання, і забезпечує швидке відновлення, хоча займає більше місця, ніж інкрементне копіювання, оскільки кожне нове диференціальне копіювання містить усі зміни з моменту останнього повного

копіювання. Методи резервного копіювання наведені на рис. 1.2. Рисунок 1.2 – Методи резервного копіювання Резервне копіювання на локальних носіях. Використовуються жорсткі диски, флешки, зовнішні жорсткі диски або оптичні носії (DVD, Blu-ray). Переваги: контроль над фізичними носіями, швидкий доступ до даних. Недоліки: можливі фізичні пошкодження носіїв, ризик втрати даних при катастрофах (пожежі, повені тощо). Резервне копіювання в хмарі. Зберігання даних на віддалених серверах через інтернет. Переваги: доступ до даних з будь-якої точки світу, зручність у масштабуванні. Недоліки: залежність від інтернет-з'єднання, необхідність регулярних витрат на підписки, питання безпеки даних. Резервне копіювання на мережевих пристроях Використання спеціальних пристроїв, які підключаються до локальної мережі для зберігання резервних копій. Переваги: централізоване зберігання даних, зручність в управлінні резервними копіями. Недоліки: необхідність в налаштуванні мережі і підтримці NAS-системи. Резервне копіювання через автоматизовані програми Використовуються спеціальні програми для автоматичного створення резервних копій з можливістю планування. Переваги: автоматизація процесу, зниження ризику людської помилки. Недоліки: необхідність у налаштуванні програмного забезпечення та можливі обмеження на безкоштовних версіях. Вибір між різними типами та методами резервного копіювання залежить від таких факторів: Якщо дані є критичними, варто застосовувати регулярне повне резервне копіювання. Якщо даних багато, інкрементне або диференціальне резервне копіювання може бути більш ефективним. Якщо потрібно зберігати резервні копії великого обсягу даних на тривалий час, хмарне зберігання може стати вигідним варіантом. Резервне копіювання є важливою частиною стратегії захисту даних, яка дозволяє зберігати копії важливої інформації та відновлювати її в разі втрати чи пошкодження. Існують різні типи та методи резервного копіювання, і вибір між ними залежить від вимог до безпеки, швидкості і простору для зберігання [1]. Основні типи резервного копіювання: Методи резервного копіювання можна поділити на кілька основних груп: Резервне копіювання на локальних носіях - для цього використовуються жорсткі диски, флешки, зовнішні диски або оптичні носії. Це дає змогу контролювати фізичні носії та швидко отримувати доступ до даних. Проте є

 **Обнаружен Плагиат: 0,1%** <https://www.morningdough.com/uk/a...> id: 19

ризик втрати даних через фізичні пошкодження носіїв або при катастрофах. Резервне копіювання в хмарі

- це зберігання даних на віддалених серверах через інтернет. Це дозволяє отримувати доступ до даних з будь-якої точки світу і зручно масштабувати зберігання, але також залежить від інтернет-з'єднання і може вимагати регулярних витрат на підписки. Резервне копіювання на мережевих пристроях - передбачає використання спеціальних пристроїв, підключених до локальної мережі для зберігання резервних копій. Це дозволяє централізовано зберігати дані та зручно управляти резервними копіями, але вимагає налаштування мережі та підтримки пристроїв. Резервне копіювання через автоматизовані програми - за допомогою програм можна автоматизувати процес створення резервних копій, плануючи їх на певний час. Це знижує ймовірність помилок, але потребує налаштування програмного забезпечення. Вибір методу резервного копіювання залежить від кількох факторів. Якщо дані критичні, найкращим варіантом буде регулярне повне резервне копіювання. Якщо ж обсяг даних великий, можна вибрати інкрементне або диференціальне копіювання. Також важливо враховувати бюджет, оскільки зберігання великих обсягів даних на тривалий час може бути дорогим, особливо при використанні хмарних сервісів. План резервного копіювання також є важливою частиною стратегії збереження даних. Слід визначити, як часто створювати резервні копії залежно від

важливості даних. Для критичних даних резервні копії мають створюватися частіше, а для менш важливих – рідше. Крім того, потрібно організувати зберігання резервних копій, забезпечити їхню безпеку за допомогою шифрування та паролів, а також регулярно тестувати їх для перевірки працездатності. Однією з найпоширеніших проблем є невідповідність між резервною копією та поточним станом даних, неправильна частота резервного копіювання або недостатнє зберігання копій в одному місці. Це збільшує ризик втрати даних через фізичні пошкодження носіїв або катастрофи. Резервне копіювання є невід'ємною частиною стратегії збереження даних, що мінімізує ризики їх втрати і забезпечує швидке відновлення в разі непередбачених ситуацій. Правильний вибір типу і методу резервного копіювання, регулярне тестування і моніторинг цього процесу є важливими для забезпечення стабільності і безпеки даних. Для забезпечення ефективності та надійності резервного копіювання слід також враховувати деякі додаткові аспекти: Щоб знизити ризик людських помилок і забезпечити регулярність процесу, важливо налаштувати автоматичне створення резервних копій. Багато програм для резервного копіювання мають функцію планування, що дозволяє налаштувати періодичність копіювання даних. Це дозволяє уникнути пропуску важливих резервних копій і зберегти актуальність даних. Інколи важливо зберігати кілька версій резервних копій, щоб мати можливість відновити дані в разі помилкових змін або випадкового видалення. Наприклад, можна зберігати резервні копії даних за останні кілька днів або тижнів, щоб мати можливість вибору більш старої версії в разі необхідності. Це особливо важливо при роботі з великими базами даних чи важливими документами, де можуть бути зроблені несанкціоновані зміни. Якщо для резервного копіювання використовуються фізичні носії, такі як зовнішні жорсткі диски, флешки або оптичні носії, потрібно врахувати їхню фізичну безпеку. Це означає, що носії повинні зберігатися в безпечних місцях, захищених від пошкоджень, крадіжки або природних катастроф (пожежі, повені тощо). Використання спеціальних сейфів для зберігання носіїв або резервних копій може суттєво підвищити рівень безпеки. Не менш важливим є резервне копіювання для мобільних пристроїв, таких як смартфони або планшети, на яких також зберігаються важливі дані. Сучасні сервіси хмарного резервного копіювання, як правило, пропонують автоматичне створення копій для таких пристроїв. Проте слід також використовувати додаткові методи, наприклад, копіювання важливих файлів на локальні пристрої чи інші хмарні платформи, щоб зберігати всі дані в безпеці [2]. План відновлення даних є критично важливим для швидкого і ефективного відновлення системи в разі її пошкодження чи втрати даних. Відновлення має бути максимально швидким і зручним для користувача. Тому важливо тестувати не тільки самі резервні копії, але й процес відновлення, щоб у разі потреби можна було оперативно відновити роботу. Якщо використовується хмарне резервне копіювання, слід ретельно вибирати постачальника послуг. Важливо оцінити не тільки вартість та обсяг хмарного сховища, а й рівень безпеки, надані інструменти для шифрування даних, а також політику конфіденційності. Перш ніж обрати постачальника, корисно перевірити його репутацію та гарантії щодо збереження даних. У деяких сферах діяльності (наприклад, у медичній, фінансовій або юридичній) можуть бути додаткові вимоги щодо зберігання та захисту даних. У такому випадку резервне копіювання повинно відповідати певним стандартам або нормативним актам, що вимагають шифрування, зберігання копій на певному обладнанні або в конкретних географічних регіонах. Якщо бізнес або проект зростає, потрібно забезпечити масштабованість системи резервного копіювання. З часом обсяг даних може збільшуватися, тому важливо обирати такі методи зберігання та резервного копіювання, які дозволяють розширювати сховище або кількість копій без значних витрат часу і ресурсів. Належно розроблений план резервного копіювання має супроводжуватися чіткою документацією, яка описує кожен етап процесу, а також процес відновлення даних. Це дозволяє знизити ризик помилок і забезпечити правильне виконання процедур у разі необхідності. Крім того, важливо проводити навчання персоналу для забезпечення правильного виконання всіх етапів резервного копіювання. Резервне копіювання є критично важливою складовою частиною інформаційної безпеки. Воно допомагає забезпечити цілісність та доступність даних у разі різних непередбачуваних ситуацій. Планування, правильний вибір методів і інструментів резервного копіювання дозволяє мінімізувати ризики та забезпечити безперебійну роботу як особистих, так і бізнесових систем. Продовжуючи тему резервного копіювання, варто звернути увагу на деякі додаткові аспекти, які можуть допомогти вдосконалити процес і зробити його ще більш надійним та ефективним. Іноді найкращий підхід до резервного копіювання полягає в комбінуванні різних методів. Наприклад, можна використовувати

повне резервне копіювання для основних даних і комбінувати його з інкрементними або диференціальними копіями для менш критичних даних. Це дозволить оптимізувати використання ресурсів та зберігання, зберігаючи при цьому високий рівень захисту даних. Для великих обсягів даних, як-от відеофайли, великі бази даних або інші ресурсоємні файли, важливо використовувати спеціалізовані стратегії резервного копіювання. У таких випадках може знадобитися використання технологій дедуплікації (для зменшення обсягу даних, що зберігаються) або спеціальних алгоритмів, які знижують навантаження на мережу і зберігання. Якщо резервні копії зберігаються в хмарі або на віддалених серверах, дуже важливо переконатися, що дані доступні для відновлення в будь-який час, навіть у разі перебоїв в інтернет-з'єднанні або інших проблем. Для цього можна використовувати можливості для офлайн-доступу або забезпечити синхронізацію з кількома резервними місцями зберігання, щоб не залежати від одного сервера чи локації. Для контролю за процесом резервного копіювання можна використовувати спеціалізовані програми та платформи, які не лише автоматизують процес, а й надають звіти про статус резервних копій, а також попереджають про можливі проблеми, наприклад, недостатній простір для зберігання або помилки при створенні копій. Це дозволяє вчасно вжити необхідних заходів і забезпечити постійну доступність даних. Враховуючи зростання використання віртуалізації, важливо передбачити створення резервних копій для як фізичних, так і віртуальних середовищ. Віртуальні машини, контейнери та інші елементи віртуалізованих середовищ мають свої особливості при резервному копіюванні. Існують спеціалізовані інструменти для резервного копіювання віртуальних середовищ, які дозволяють зберігати стан віртуальних машин, включаючи системні файли, додатки і конфігурації. Резервне копіювання повинно бути інтегровано з іншими заходами безпеки, такими як антивірусні програми, фаєрволи та системи моніторингу безпеки. Це забезпечить додатковий рівень захисту даних від зловмисників, а також дозволить виявити аномалії, що можуть вказувати на спроби викрадення або змінювання даних. Ідеально, коли резервні копії зберігаються на кількох рівнях: локально (для швидкого доступу), на мережевих пристроях або серверах (для забезпечення відновлення у разі поломки основної системи) та в хмарі (для захисту від фізичних катастроф, таких як пожежі чи повені). Цей багаторівневий підхід дає змогу гарантувати збереження даних навіть у найгірших ситуаціях. Завжди потрібно пам'ятати, що резервне копіювання — це лише одна частина стратегії відновлення після катастроф. Потрібно мати чіткий план відновлення, який описує, як саме будуть відновлюватися дані і системи в разі надзвичайної ситуації. План має включати всі етапи, від визначення пріоритетів відновлення до часу, необхідного для повного відновлення роботи організації [14]. Однією з найбільших загроз сьогодення є програми-вимагачі, які можуть зашифрувати дані і вимагати викуп за їх відновлення. Тому необхідно налаштувати резервне копіювання таким чином, щоб копії даних зберігались в захищеному вигляді, і доступ до них був обмежений лише авторизованим особам. Важливо також регулярно перевіряти, чи не зазнали резервні копії атак, а також чи можливе їх відновлення безпосередньо з резервних джерел. Резервне копіювання є основою для забезпечення безпеки даних і їх відновлення в разі непередбачених ситуацій. Воно повинно бути частиною загальної стратегії інформаційної безпеки та регулярно перевірятися для забезпечення ефективності та працездатності. Важливо обирати відповідні методи, стратегії і технології резервного копіювання залежно від обсягу та критичності даних, а також регулярно вдосконалювати ці стратегії, щоб вони відповідали зростаючим вимогам та викликам безпеки.

1.2. Огляд протоколу FTP: особливості та можливості

Протокол [FTP \(File Transfer Protocol\)](#) є одним з найстаріших і найбільш поширених протоколів для передачі файлів через мережу. [FTP](#) використовується для обміну файлами між клієнтами та серверами в інтернеті чи локальних мережах. Хоча на сьогодні з'явилися більш безпечні протоколи, [FTP](#) все ще залишається популярним завдяки своїй простоті та широкому використанню. Основні особливості [FTP](#): [FTP](#) працює за принципом двоетапного з'єднання: один канал для управління (управління командними повідомленнями та відповідями) і один канал для передачі даних (файлів). Канал управління працює на порті 21, а канали передачі даних можуть використовувати інші порти, що робить [FTP](#) гнучким для великих обсягів передаваних файлів. [FTP](#) підтримує аутентифікацію користувачів за допомогою імені користувача та пароля, що дозволяє контролювати доступ до файлів. Існує також можливість використовувати анонімний доступ, де користувач може підключитися без імені користувача та пароля (зазвичай для загальнодоступних файлів). [FTP](#) підтримує два режими з'єднання: активний і пасивний. Активний режим передбачає, що сервер відкриває з'єднання до клієнта для передачі даних. Пасивний режим є безпечнішим і зазвичай

використовується в умовах суворих правил фаєрволу або [NAT \(Network Address Translation\)](#), оскільки клієнт сам ініціює з'єднання для передачі даних. [FTP](#) дозволяє передавати файли практично будь-якого розміру, що робить його зручним для роботи з великими даними. Також [FTP](#) дає змогу передавати файли як окремо, так і групами. [FTP](#) підтримує кілька режимів передачі даних, зокрема [ASCII](#) для текстових файлів та [Binary](#) для двійкових файлів. Це дозволяє зберігати цілісність даних під час передачі і уникати проблем із кодуванням символів. [FTP](#) дозволяє не лише передавати файли, але й виконувати базові операції з файлами на сервері, такі як перегляд вмісту каталогів, копіювання, видалення та перейменування файлів. Можливості [FTP](#) наведені на рис. 1.3. Рисунок 1.3 – Можливості [FTP](#) [FTP](#) дає змогу користувачам завантажувати файли на сервери або завантажувати їх із серверів на свої пристрої, що особливо корисно для веб-розробників, адміністраторів серверів та користувачів, що працюють з великими обсягами даних. За допомогою [FTP](#) можна підключатися до віддалених серверів, що дає можливість працювати з файлами з будь-якої точки світу, за умов наявності доступу до мережі. [FTP](#) дозволяє здійснювати передачу файлів між різними платформами (наприклад, [Windows](#) до [Linux](#), [Linux](#) до [macOS](#) і т.д.), оскільки підтримує стандартизовану обробку даних у двійковому та текстовому форматах. [FTP](#) ідеально підходить для передачі великих файлів, таких як архіви, відео, бази даних, тому широко застосовується в різних професійних сферах, де необхідна передача великих даних. [FTP](#) підтримує одночасні з'єднання до сервера, що дозволяє кільком користувачам одночасно завантажувати або завантажувати файли з одного серверу без значного зниження швидкості. [FTP](#) є досить швидким протоколом завдяки ефективному використанню мережевих ресурсів для передачі даних. Це робить його популярним для обміну великими обсягами інформації в бізнесі та інших сферах [11]. Недоліки [FTP](#): Одна з основних проблем [FTP](#) — це відсутність вбудованого шифрування. Дані, що передаються через [FTP](#), передаються у відкритому вигляді, що робить їх вразливими для атак типу

Цитування: 0,01%

id: 20

"Man-in-the-Middle".

Обнаружен Плагиат: 0,18% <https://realhost.pro/blog/what-is-ftp>

id: 21

Через це [FTP](#) не рекомендується для передачі чутливих або конфіденційних даних. [FTP](#) використовує відкриті порти та стандартні методи аутентифікації, що робить його вразливим до

атак, таких як підбір пароля або перехоплення даних. Без додаткових засобів безпеки, таких як [VPN](#) або шифрування, [FTP](#) може бути небезпечним для використання в умовах високої загрози. [FTP](#) не завжди є найбільш зручним протоколом для мобільних пристроїв через необхідність налаштування з'єднання і специфічні клієнти, які можуть бути не завжди доступні або зручні в використанні. Для забезпечення більш високого рівня безпеки та функціональності, [FTP](#) було доповнено більш сучасними протоколами, такими як [SFTP \(SSH File Transfer Protocol\)](#) і [FTPS \(FTP Secure\)](#), які використовують шифрування для захисту даних і забезпечують більш високий рівень безпеки при передачі файлів [8]. [FTP](#) є зручним і ефективним інструментом для обміну файлами, але його вразливості до атак і відсутність шифрування роблять його непридатним для передачі конфіденційних даних без додаткових заходів безпеки. Для більш безпечної передачі даних рекомендується використовувати більш сучасні протоколи, такі як [SFTP](#) або [FTPS](#). Однак [FTP](#) досі залишається важливим інструментом для організацій, які потребують ефективного обміну великими обсягами даних. [FTP \(File Transfer Protocol\)](#) залишається важливим інструментом для обміну файлами в мережах, незважаючи на його обмеження, зокрема відсутність шифрування та вразливість до атак. Завдяки своїй простоті та ефективності [FTP](#) широко використовується в ситуаціях, де безпека не є першочерговим пріоритетом. Він продовжує бути популярним для публікацій на веб-сайтах, де веб-розробники використовують його для завантаження та оновлення контенту, таких як медіафайли, [HTML](#), [CSS](#) та інші файли на сервери. Це дозволяє швидко здійснювати зміни та підтримувати актуальність сайту [9]. [FTP](#) також інтегрується в автоматизовані системи, де необхідно регулярно передавати дані. Наприклад, у виробничих чи корпоративних системах [FTP](#) може бути налаштований для автоматичної передачі даних або резервних копій у певний час, що значно полегшує управління великими обсягами інформації. Сценарії [FTP](#) можуть бути налаштовані для безперервної роботи без втручання людини, що робить цей процес ефективним і зручним. У великих організаціях і промислових мережах [FTP](#) часто використовується для зберігання

та обміну великими даними, такими як звіти або файли для аналізу. Завдяки здатності передавати файли великих розмірів [FTP](#) є дуже зручним для внутрішнього обміну інформацією між різними підсистемами чи філіями компанії. Однак із розвитком нових технологій з'явилися більш безпечні протоколи, які використовують шифрування для захисту переданих даних. Це важливо для захисту чутливої інформації, особливо коли [FTP](#) застосовується для передавання конфіденційних даних. Протоколи, такі як [SFTP](#) ([SSH File Transfer Protocol](#)) або [FTPS](#) ([FTP Secure](#)), дозволяють вирішити проблему безпеки, забезпечуючи шифрування даних на всіх етапах передачі, і вони з часом витісняють [FTP](#) у тих сферах, де безпека є пріоритетною. [FTP](#) все ще використовується через свою простоту і доступність, але для організацій, які працюють з конфіденційними або критичними даними, рекомендовано використовувати сучасніші варіанти, які забезпечують кращий рівень захисту. Це дозволяє знизити ризики і відповідати вимогам сучасної безпеки даних. Попри те, що [FTP](#) все ще використовується для певних задач, його обмеження в плані безпеки змушують компанії все більше звертатися до сучасніших альтернатив. Протоколи, які підтримують шифрування, такі як [SFTP](#) і [FTPS](#), забезпечують надійніший захист даних під час передачі. [SFTP](#), зокрема, використовує [SSH](#) ([Secure Shell](#)) для шифрування каналу зв'язку, що робить передачу файлів більш безпечною. [FTPS](#), в свою чергу, додає підтримку шифрування до традиційного [FTP](#), забезпечуючи захист даних, як під час аутентифікації, так і під час фактичної передачі файлів. Незважаючи на більшу безпеку, яка надається сучасними протоколами, [FTP](#) залишається популярним у ситуаціях, де вимоги до захисту не є настільки строгими або коли доступ до простого протоколу потрібен для внутрішніх мереж. Багато користувачів [FTP](#) продовжують використовувати його для завантаження публічних файлів, таких як медіафайли або відкриті ресурси, де безпека не є пріоритетом. Однак для таких ситуацій [FTP](#) повинен працювати в обмеженому контексті, коли конфіденційність даних не загрожує. Враховуючи сучасні тенденції до глобалізації та зростання обсягу переданих даних, [FTP](#) все більше втрачає популярність у сфері передачі чутливих даних. Однак у межах внутрішніх мереж та для менш критичних завдань [FTP](#) продовжує використовуватися завдяки своїй простоті та швидкості. Користувачі повинні ретельно оцінювати, де і як застосовувати [FTP](#), і в разі потреби, переходити на більш захищені альтернативи, щоб забезпечити належний рівень захисту своїх даних у мережі. Незважаючи на деякі обмеження, [FTP](#) все ще залишається важливим інструментом у багатьох сферах. Однак з розвитком цифрових технологій і зростанням вимог до безпеки, варто звернути увагу на те, як [FTP](#) може адаптуватися або поступово переходити на більш безпечні протоколи. Важливо пам'ятати, що багато організацій намагаються інтегрувати шифрування та додаткові методи захисту, щоб забезпечити безпечний обмін даними в умовах сучасних кіберзагроз. Завдяки гнучкості [FTP](#) він продовжує бути корисним для специфічних завдань, де інші протоколи можуть бути менш зручними або більш складними для впровадження. Наприклад, у віддаленому адмініструванні серверів, в обміні великими файлами або в ситуаціях, де необхідно швидко перенести не надто чутливі дані, [FTP](#) може залишатися хорошим варіантом. В таких випадках організації можуть приймати рішення про використання [FTP](#) з додатковими заходами безпеки, наприклад, через використання [VPN](#) або обмеження доступу лише для певних [IP](#)-адрес. Одним із важливих аспектів є розвиток зручних [FTP](#)-клієнтів, які дозволяють користувачам без зайвих зусиль працювати з файлами на віддалених серверах. Ці програми, такі як [FileZilla](#), [WinSCP](#) або [Cyberduck](#), інтегрують [FTP](#) з додатковими можливостями, наприклад, підтримкою шифрування через [FTPS](#) або [SFTP](#). Завдяки таким інструментам користувачі можуть комбінувати простоту [FTP](#) з високим рівнем захисту даних, що робить процес передачі файлів більш безпечним. У будь-якому випадку, важливо оцінювати потреби конкретної організації, враховувати вимоги до безпеки та інфраструктури і вибирати відповідний метод передачі даних. Для роботи з конфіденційними або чутливими даними завжди краще використовувати більш захищені протоколи, ніж [FTP](#). Однак для обміну менш важливими файлами або для простих завдань [FTP](#) може бути достатнім, якщо до цього процесу будуть застосовані належні заходи безпеки. Якщо говорити про подальший розвиток технологій і обміну даними, варто зазначити, що все більше організацій переходять на хмарні сервіси для обміну та зберігання файлів, що вимагає додаткових мір безпеки. Сервіси на кшталт [Google Drive](#), [Dropbox](#), [OneDrive](#) та інші значно спрощують обмін великими файлами та забезпечують високий рівень захисту даних завдяки вбудованим механізмам шифрування. Ці платформи стають все популярнішими через їх зручність та безпеку, оскільки вони автоматично обробляють більшість питань щодо шифрування та збереження даних. Однак навіть попри це, [FTP](#) зберігає свою актуальність у специфічних ситуаціях, зокрема в тих випадках, коли

потрібна підтримка більш старих систем або деякі організації не готові відмовитися від класичних підходів. Наприклад, у випадках, коли потрібно передавати дані між старими системами, яким ще не доступні сучасні хмарні технології, [FTP](#) все ще може бути ефективним рішенням. Для тих, хто все ж використовує [FTP](#) у своїй роботі, варто звернути увагу на деякі важливі моменти для підвищення безпеки. Наприклад, обов'язково потрібно використовувати такі протоколи як [FTPS](#) або [SFTP](#), які забезпечують шифрування даних при передачі. Це значно підвищить рівень захисту і дозволить мінімізувати ризики, пов'язані з перехопленням даних. Також важливо вживати

 **Обнаружен Плагиат: 0,28%** <https://www.softkey.ua/ua/useful/artic...>

id: 22

додаткових заходів безпеки, таких як регулярна зміна паролів для доступу до серверів, застосування багатофакторної аутентифікації (куди це можливо) та використання фаєрволів для обмеження доступу до [FTP](#)-серверів тільки з дозволених [IP](#)-адрес. Це допоможе захистити сервери від несанкціонованого доступу і

знижить вірогідність атак, таких як [brute-force](#). Відновлення даних із [FTP](#)-серверів також є важливою частиною стратегії резервного копіювання для багатьох організацій. [FTP](#) дозволяє налаштувати автоматичне створення резервних копій на віддалених серверах, що дозволяє зберігати дані в безпеці і забезпечувати доступ до них навіть у разі збою локальних систем. Зважаючи на всі ці аспекти, можна зробити висновок, що [FTP](#), хоча і не є ідеальним вибором для передачі чутливих даних, залишається корисним інструментом для конкретних завдань, таких як обмін неконфіденційними файлами, обмін великими обсягами даних або для роботи з віддаленими серверами. Однак, при використанні [FTP](#), дуже важливо враховувати його обмеження з точки зору безпеки і застосовувати додаткові методи захисту для мінімізації ризиків. Важливо зазначити, що з часом технології обміну даними продовжують розвиватися, і з'являються нові стандарти та інструменти для ще більш ефективної та безпечної передачі інформації. Так, зокрема, все більше організацій намагаються впроваджувати рішення, що підтримують не тільки файли, але й цілі робочі процеси. Це включає в себе більш складні інтеграції для автоматичного обміну даними, підтримку [API](#) для зручної роботи з файлами через інтерфейси та розширені можливості для інтеграції з іншими бізнес-платформами [3]. [FTP](#), хоча і є надійним протоколом для базових завдань обміну файлами, може бути не так ефективним у разі великої кількості паралельних запитів або коли дані потребують складних маніпуляцій. Наприклад, у випадках, коли потрібно інтегрувати передачу файлів із бізнес-логікою або забезпечити високий рівень автоматизації, хмарні сервіси або [API](#) можуть бути набагато більш зручними і потужними інструментами для обміну даними. Незважаючи на це, [FTP](#) все ще використовують у багатьох галузях, включаючи веб-розробку, наукові дослідження та виробничі процеси, де є потреба в простому і швидкому обміні великими файлами. Завдяки тому, що [FTP](#) є добре відомим і стабільним протоколом, він залишається важливим інструментом для багато організацій. Для підвищення зручності використання [FTP](#), деякі підприємства комбінують його з іншими інструментами. Наприклад, розгортання [FTP](#) через [VPN](#) або використання шифрованих з'єднань може забезпечити додатковий рівень захисту при роботі з чутливими даними, навіть якщо сам протокол не підтримує вбудоване шифрування. Для малих компаній або тих, хто працює з менш важливими даними, [FTP](#) часто може бути цілком достатнім, особливо якщо необхідно швидко передати великі обсяги інформації. В таких випадках можна обмежити доступ до сервера, налаштувати регулярні перевірки і використовувати інші засоби для захисту інформації, що передається. Тим не менш, для більш серйозних вимог безпеки, зокрема у фінансових установах, медичних закладах або при роботі з державною інформацією, організаціям краще відмовитися від використання стандартного [FTP](#) і перейти на більш безпечні технології. Це може включати [SFTP](#) або [FTPS](#), які, хоч і менш популярні, забезпечують набагато вищий рівень захисту при передачі файлів. Більш того, для забезпечення повної безпеки даних, важливо також налаштувати відповідні політики доступу, моніторинг активності та аудити системи для виявлення потенційних загроз. Враховуючи сучасні тенденції та загрози, організації повинні регулярно переглядати свої стратегії обміну даними та відповідно до цього адаптувати інструменти і протоколи, що використовуються. В кінцевому підсумку, вибір між [FTP](#) і більш сучасними протоколами залежить від специфіки завдання, необхідного рівня безпеки та доступних ресурсів для підтримки цих технологій. Важливо, щоб підприємства регулярно перевіряли і оновлювали свої стратегії безпеки для підтримки високого рівня захисту даних і забезпечення безпечного обміну інформацією в умовах постійно змінюваного цифрового середовища. Сьогодні все більше

організацій застосовують комплексні рішення для забезпечення ефективного обміну даними між різними платформами та застосунками. Інтеграція протоколів передачі даних з існуючими корпоративними системами дозволяє автоматизувати багато процесів, включаючи обмін даними, моніторинг і архівацію, що дає змогу знижувати операційні витрати і підвищувати ефективність. Одним із найбільш популярних напрямків у галузі є підтримка [API](#) для інтеграції передачі даних між різними сервісами та додатками. Це дозволяє автоматизувати багато аспектів роботи з [FTP](#)-серверами і полегшити обмін даними між різними системами в межах організації або з зовнішніми партнерами. [API](#) дозволяють створювати спеціалізовані інтерфейси для обміну файлами без потреби вручну керувати з'єднанням через [FTP](#), а також інтегрувати цей процес у більші автоматизовані робочі потоки. Проте, хоча [FTP](#) залишається широко використовуваним протоколом, деякі організації все більше віддають перевагу хмарним сервісам і платформам для обміну та зберігання даних, оскільки вони забезпечують більшу зручність, масштабованість і знижують витрати на підтримку інфраструктури. Хмарні платформи дозволяють працювати з файлами в реальному часі з будь-якої точки світу, що знижує потребу в локальних [FTP](#)-серверах і дає можливість зберігати дані в зручних та безпечних умовах [15]. Проте, [FTP](#) все ще є незамінним для специфічних задач, таких як інтеграція з [legacy](#)-системами (старими або застарілими інформаційними системами), де модернізація або інтеграція з більш новими рішеннями є економічно недоцільною. Крім того, в деяких випадках [FTP](#) може бути ідеальним для використання в організаціях, де необхідний контроль над фізичним доступом до серверів та де дані не потребують високого рівня захисту. Важливо зазначити, що в умовах постійно зростаючих кіберзагроз організації повинні активно інвестувати в інструменти для моніторингу і виявлення аномалій у своїх мережах. Це включає в себе регулярний аудит, застосування антивірусних рішень і фаєрволів, а також впровадження комплексних систем для виявлення і запобігання несанкціонованим спробам доступу до даних. Якщо [FTP](#) використовується, слід також налаштувати систему сповіщень про підозрілі дії, такі як багаторазові спроби підключення або невдалі спроби аутентифікації, які можуть свідчити про спроби злому. Останнім часом також популярні рішення для автоматизації і керування передачею файлів через [FTP](#) за допомогою хмарних або локальних серверів, що дозволяє забезпечити більшу гнучкість і контроль. Вони дозволяють реалізувати масштабовані рішення для обміну даними, автоматичну архівацію і навіть автоматичне шифрування файлів перед їхнім відправленням на [FTP](#)-сервер. У підсумку, [FTP](#) залишається важливим інструментом для багатьох бізнес-процесів, але важливо не забувати про його обмеження і забезпечувати додаткові заходи безпеки для збереження цілісності та конфіденційності даних. Постійний моніторинг і адаптація до нових технологій допоможуть забезпечити безперервний і безпечний обмін даними у вашій організації.

1.3. Використання [Python](#) для роботи з [FTP](#)

Для роботи з [FTP](#) у [Python](#) використовується бібліотека [ftplib](#), яка є частиною стандартної бібліотеки. Вона дозволяє підключатися до [FTP](#)-серверів, передавати файли, завантажувати їх, працювати з каталогами та виконувати інші операції. При використанні [Python](#) для роботи з [FTP](#) основні етапи включають підключення до серверу, аутентифікацію, навігацію по каталогах і виконання операцій з файлами. Першим кроком є створення об'єкта [FTP](#) і підключення до потрібного сервера. Після цього, залежно від необхідних завдань, можна завантажувати файли з сервера або завантажувати їх на сервер, переглядати вміст каталогів, створювати нові каталоги або видаляти файли. Оскільки [ftplib](#) підтримує анонімний доступ, можна підключатися до багатьох публічних [FTP](#)-серверів без необхідності вводити ім'я користувача і пароль. Однак для приватних серверів необхідно використовувати правильні облікові дані для аутентифікації. [Python](#) також дозволяє налаштувати [FTP](#)-з'єднання на більш високому рівні, додаючи підтримку шифрування (наприклад, за допомогою [FTPS](#)), а також автоматизацію процесів обміну файлами за допомогою сценаріїв. Зазвичай для таких завдань використовуються додаткові бібліотеки або налаштування протоколів. Завдяки простоті використання та великим можливостям [Python](#) є відмінним інструментом для автоматизації задач, пов'язаних з роботою з [FTP](#)-серверами, особливо для завантаження або архівації файлів, а також для обробки великих обсягів даних. Крім основних операцій з файлами, [Python](#) також дозволяє налаштувати більш складні сценарії для роботи з [FTP](#)-серверами, що можуть включати в себе такі функції, як: Автоматизація завантаження та завантаження файлів: Завдяки можливості автоматичного виконання завдань, [Python](#) може бути використаний для регулярного завантаження файлів з [FTP](#)-серверів або їх завантаження на сервери в певний час або при певних умовах. Це особливо корисно для резервного копіювання даних, архівації або

синхронізації великих обсягів інформації. Перевірка наявності файлів і каталогів: З допомогою [Python](#) можна легко перевіряти наявність файлів на сервері перед їх завантаженням або видаленням. Це дає змогу ефективно керувати даними та уникати дублювання чи втрат. Побудова інтерфейсів для роботи з [FTP](#): [Python](#) можна використовувати для створення власних інтерфейсів або скриптів для керування [FTP](#)-серверами. Це може бути корисно для компаній або організацій, які регулярно працюють з великою кількістю файлів і хочуть автоматизувати процеси без необхідності вручну виконувати команди. Інтеграція з іншими системами та базами даних: [Python](#) дозволяє інтегрувати [FTP](#)-операції з іншими системами, такими як бази даних, сервіси для обробки великих даних або системи для аналізу інформації. Це дає змогу будувати більш складні і ефективні робочі процеси, де передача файлів через [FTP](#) є лише частиною великої автоматизованої системи. Обробка великих обсягів даних: Використовуючи [Python](#), можна працювати з великими обсягами файлів, зокрема, передавати їх по [FTP](#)-серверах для подальшої обробки чи зберігання. Бібліотека [ftplib](#) підтримує великі файли, тому можна налаштувати ефективну систему для роботи з великими даними в автоматичному режимі. Безпека та шифрування: Хоча стандартний [FTP](#) не забезпечує шифрування, [Python](#) також дозволяє працювати з більш безпечними варіантами [FTP](#), такими як [FTPS \(FTP Secure\)](#) або [SFTP \(SSH File Transfer Protocol\)](#), що гарантує захист даних під час передачі. Це особливо важливо при роботі з конфіденційною або чутливою інформацією. Використовуючи [Python](#) для автоматизації процесів роботи з [FTP](#), можна значно зменшити час, витрачений на ручне виконання завдань, і підвищити ефективність роботи з великими обсягами даних. [Python](#) також дозволяє створювати складні сценарії для роботи з [FTP](#)-серверами, що відкриває широкі можливості для інтеграції з іншими системами та підвищення рівня безпеки при роботі з даними. Також [Python](#) може бути використаний для додаткових вдосконалень у роботі з [FTP](#), таких як паралельне завантаження та завантаження файлів. Це дозволяє значно зменшити час на виконання операцій, особливо при роботі з великою кількістю файлів або великими файлами. Паралельне завантаження дає змогу одночасно передавати кілька файлів або великий файл частинами, що пришвидшує процес. Обробка помилок також є важливим аспектом при роботі з [FTP](#). Важливо обробляти виключення, такі як помилки підключення до серверу, відмови в аутентифікації або недоступність файлів. Це дозволяє створювати більш надійні і стійкі до збоїв скрипти, які можуть автоматично повторювати операції у разі помилок або інформувати користувача про необхідність втручання. Ще однією корисною можливістю є автоматична перевірка цілісності файлів після їх передачі. Наприклад, можна порівнювати контрольні суми файлів, щоб переконатися, що файли були передані без помилок. Це особливо важливо при роботі з великими даними або чутливими файлами. Завдяки можливостям [Python](#), можна автоматизувати ці процеси, що дозволяє значно зменшити навантаження на користувачів та підвищити ефективність обміну даними. Підключення до [FTP](#)-серверів через [Python](#) можна інтегрувати з іншими системами для централізованого управління файлами, архівації або синхронізації даних, що додає додаткову гнучкість у вирішенні завдань обміну даними. Оскільки [Python](#) має потужні бібліотеки для роботи з іншими протоколами, можна інтегрувати [FTP](#)-сценарії з іншими частинами інфраструктури, що дозволяє побудувати комплексні рішення для обміну даними в корпоративному середовищі. Це може включати синхронізацію з базами даних, обробку даних в реальному часі, а також управління великими наборами даних. Крім основних функцій, [Python](#) дозволяє створювати більш складні рішення для роботи з [FTP](#). Наприклад, можна налаштувати сценарії для автоматичного обміну даними між різними серверами або здійснювати моніторинг [FTP](#)-серверів на наявність нових файлів для обробки. Це дозволяє створювати автоматизовані процеси, які можуть працювати без участі людини, що особливо корисно для великих організацій, де потрібно обробляти великий обсяг даних. Також важливим аспектом є логування і створення звітів. За допомогою [Python](#) можна автоматично записувати всі операції з файлами, такі як завантаження, завантаження, помилки або спроби несанкціонованого доступу, у лог-файли. Це дозволяє проводити аудит і моніторинг, а також швидко виявляти проблеми в роботі системи, якщо щось пішло не так. [Python](#) також може бути корисним для налаштування перевірок на доступність [FTP](#)-сервера, що дозволяє автоматично реагувати на неполадки в мережі. У разі відсутності доступу до серверу можна налаштувати автоматичне сповіщення, наприклад, через електронну пошту або [SMS](#), що дозволяє оперативно вирішувати проблеми. Використання [Python](#) для інтеграції з іншими системами забезпечує гнучкість у роботі з [FTP](#)-серверами. Наприклад, можна налаштувати автоматичне передавання даних з [FTP](#)-серверів в бази даних для подальшого

аналізу або обробки. Це дозволяє створювати потужні робочі процеси для збору, обробки та зберігання даних без потреби у вручну виконуваних операціях. Завдяки своїй простоті, гнучкості та потужним можливостям, [Python](#) є відмінним інструментом для роботи з [FTP](#). З його допомогою можна автоматизувати великі обсяги роботи, знижуючи ймовірність помилок і підвищуючи ефективність обміну даними, що робить його ідеальним вибором для проектів, де необхідно регулярно працювати з [FTP](#)-серверами.

1.4. Інші альтернативи автоматизації резервного копіювання

Існує багато альтернатив для автоматизації резервного копіювання, залежно від потреб і вимог до безпеки. Одним з варіантів є хмарне резервне копіювання. Хмарні сервіси, такі як [Google Drive](#), [Dropbox](#), [OneDrive](#) або [Amazon S3](#), дозволяють зберігати файли на віддалених серверах, забезпечуючи доступ з будь-якої точки світу і захист від фізичних пошкоджень. Більшість хмарних сервісів підтримують автоматичне завантаження файлів, що полегшує процес резервного копіювання. Іншим варіантом є використання програм для резервного копіювання, таких як [Acronis True Image](#), [Cobian Backup](#) або [EaseUS Todo Backup](#). Ці програми дозволяють планувати резервне копіювання, підтримують різні типи копій, зокрема повні, диференціальні та інкрементні, а також дозволяють зберігати резервні копії на локальних носіях або в хмарі. Для локальних мереж можна використовувати мережеві пристрої [NAS \(Network Attached Storage\)](#). Вони дають змогу централізовано зберігати резервні копії всіх комп'ютерів в локальній мережі і автоматизувати процес резервного копіювання. Існують також інструменти для резервного копіювання на основі командного рядка, такі як [Rsync](#) або [Robocopy](#). Вони дозволяють налаштувати автоматичне резервне копіювання та синхронізацію даних, а також можуть використовуватися для зберігання резервних копій на локальних дисках, мережевих папках або в хмарних сховищах. Для віртуалізованих середовищ, таких як [VMware](#) або [Hyper-V](#), використовуються спеціальні інструменти для резервного копіювання віртуальних машин. Віртуалізовані середовища вимагають спеціальних підходів до резервного копіювання, щоб зберегти як дані, так і конфігурації віртуальних машин. Системні адміністратори можуть також автоматизувати резервне копіювання за допомогою скриптів. Наприклад, за допомогою [PowerShell](#), [Bash](#) або [Python](#) можна створювати сценарії для планування резервних копій, перевірки їх цілісності або навіть шифрування даних перед їхнім завантаженням на сервер. Для великих компаній з високими вимогами до зберігання даних можливе використання [SAN \(Storage Area Network\)](#) для резервного копіювання. Це дозволяє централізовано зберігати великі обсяги даних з високою швидкістю доступу і налаштувати резервне копіювання на рівні всієї мережі. Резервне копіювання також може бути організовано через архівування даних, наприклад, за допомогою [ZIP](#) або інших форматів. Це зручно для тривалого зберігання великих обсягів даних, оскільки архівування дозволяє заощадити простір для зберігання і зменшити витрати на фізичні носії. Кожен з цих підходів має свої переваги, і вибір залежить від обсягів даних, вимог до безпеки, доступності і вартості. У деяких випадках комбінація різних методів, таких як локальне і хмарне резервне копіювання, може забезпечити максимальний рівень захисту даних. Існує багато альтернатив для автоматизації резервного копіювання, залежно від потреб і вимог до безпеки. Одним з варіантів є хмарне резервне копіювання. Хмарні сервіси, такі як [Google Drive](#), [Dropbox](#), [OneDrive](#) або [Amazon S3](#), дозволяють зберігати файли на віддалених серверах, забезпечуючи доступ з будь-якої точки світу і захист від фізичних пошкоджень. Більшість хмарних сервісів підтримують автоматичне завантаження файлів, що полегшує процес резервного копіювання. Іншим варіантом є використання програм для резервного копіювання, таких як [Acronis True Image](#), [Cobian Backup](#) або [EaseUS Todo Backup](#). Ці програми дозволяють планувати резервне копіювання, підтримують різні типи копій, зокрема повні, диференціальні та інкрементні, а також дозволяють зберігати резервні копії на локальних носіях або в хмарі. Для локальних мереж можна використовувати мережеві пристрої [NAS \(Network Attached Storage\)](#). Вони дають змогу централізовано зберігати резервні копії всіх комп'ютерів в локальній мережі і автоматизувати процес резервного копіювання. Існують також інструменти для резервного копіювання на основі командного рядка, такі як [Rsync](#) або [Robocopy](#). Вони дозволяють налаштувати автоматичне резервне копіювання та синхронізацію даних, а також можуть використовуватися для зберігання резервних копій на локальних дисках, мережевих папках або в хмарних сховищах. Для віртуалізованих середовищ, таких як [VMware](#) або [Hyper-V](#), використовуються спеціальні інструменти для резервного копіювання віртуальних машин. Віртуалізовані середовища вимагають спеціальних підходів до резервного копіювання, щоб зберегти як дані, так і конфігурації віртуальних машин. Системні адміністратори можуть також автоматизувати резервне копіювання за допомогою

скриптів. Наприклад, за допомогою [PowerShell](#), [Bash](#) або [Python](#) можна створювати сценарії для планування резервних копій, перевірки їх цілісності або навіть шифрування даних перед їхнім завантаженням на сервер. Для великих компаній з високими вимогами до зберігання даних можливе використання [SAN \(Storage Area Network\)](#) для резервного копіювання. Це дозволяє централізовано зберігати великі обсяги даних з високою швидкістю доступу і налаштувати резервне копіювання на рівні всієї мережі. Резервне копіювання також може бути організовано через архівування даних, наприклад, за допомогою [ZIP](#) або інших форматів. Це зручно для тривалого зберігання великих обсягів даних, оскільки архівування дозволяє заощадити простір для зберігання і зменшити витрати на фізичні носії. Кожен з цих підходів має свої переваги, і вибір залежить від обсягів даних, вимог до безпеки, доступності і вартості. У деяких випадках комбінація різних методів, таких як локальне і хмарне резервне копіювання, може забезпечити максимальний рівень захисту даних. Зважаючи на різноманіття доступних рішень, важливо також враховувати специфіку вашої організації або особистих потреб, коли обираєте метод резервного копіювання. Наприклад, для домашніх користувачів або малих підприємств хмарні сервіси можуть бути зручним і економічним вибором, оскільки вони не потребують великих початкових вкладень в обладнання та дозволяють швидко налаштувати автоматичне резервне копіювання без складних налаштувань. Проте для великих компаній або підприємств, що працюють з великими обсягами даних або чутливою інформацією, необхідно враховувати більш складні і масштабовані рішення. Використання мережевих пристроїв зберігання ([NAS](#)) дає можливість централізувати резервне копіювання на одному пристрої в локальній мережі, що полегшує управління даними і підвищує безпеку. Застосування [NAS](#) є хорошим варіантом для середнього бізнесу або компаній, які потребують збереження великих обсягів даних в межах локальної мережі з можливістю відновлення даних у разі потреби. У випадку, коли критично важливим є швидке відновлення даних або безперервна робота бізнесу, можна використовувати рішення, орієнтовані на віртуалізовані середовища. Віртуалізація дозволяє ефективно управляти резервним копіюванням віртуальних машин, що важливо для компаній, які використовують сервери на базі віртуальних машин, що зменшує час відновлення і знижує ризики відмови обладнання. Для великих і складних інфраструктур, де важлива висока доступність і швидкість обміну даними, рішення на базі [Storage Area Network \(SAN\)](#) дають змогу організувати резервне копіювання на рівні всієї мережі. Такі рішення можуть бути особливо корисними для великих організацій або підприємств, де необхідно зберігати великі обсяги даних і забезпечити їхню надійність. З іншого боку, використання командних інструментів, таких як [Rsync](#) або [Robocopy](#), може бути особливо корисним для досвідчених користувачів та адміністраторів. Ці інструменти дозволяють налаштувати гнучкі і потужні сценарії для синхронізації файлів між серверами, автоматизувати резервне копіювання, а також здійснювати перевірку цілісності та шифрування файлів під час їх передачі [7]. Незалежно від вибору конкретного методу, важливо регулярно тестувати резервні копії, щоб переконатися, що дані можна відновити у разі їх втрати або пошкодження. Крім того, потрібно враховувати політику збереження резервних копій і визначити, як довго зберігати копії для відповідності до вимог безпеки та нормативних актів. Регулярне оновлення і перевірка стратегій резервного копіювання дозволяє мінімізувати ризики втрати важливої інформації та забезпечити безперебійну роботу організації. Усі ці методи можуть бути інтегровані в більш комплексні стратегії управління даними, забезпечуючи надійний захист, швидке відновлення і безперервність бізнес-процесів.

РОЗДІЛ 2.

РОЗРОБКА СИСТЕМИ АВТОМАТИЗОВАНОГО РЕЗЕРВНОГО КОПІЮВАННЯ 2.1. Архітектура системи резервного копіювання

Архітектура системи резервного копіювання включає кілька ключових компонентів, які разом забезпечують ефективне збереження та відновлення даних. Першим компонентом є джерела даних, які включають сервери, робочі станції або інші пристрої, з яких потрібно створювати резервні копії. Ці джерела можуть бути фізичними або віртуальними машинами, а також можуть бути підключені до системи через локальну мережу чи хмару. Основною частиною архітектури є сервер резервного копіювання, який координує весь процес резервного копіювання. Цей сервер здійснює операції зі зберігання резервних копій, управління процесами копіювання і забезпечує їх збереження у відповідних сховищах. Сховище, у свою чергу, є місцем для зберігання резервних копій. Воно може бути локальним, мережевим або хмарним, і вибір типу сховища залежить від вимог до безпеки, масштабованості та доступності. Процес резервного копіювання включає вибір даних для копіювання, створення копій і збереження їх у сховищі. Важливими аспектами є типи резервних копій (повні, диференціальні,

інкрементні) та графік їх створення. Залежно від потреб організації, можна вибирати різні стратегії для забезпечення регулярності і ефективності резервного копіювання. Крім того, процес відновлення даних є важливою частиною архітектури. Це забезпечує можливість відновлення даних у разі їх втрати чи пошкодження. Система повинна підтримувати різні варіанти відновлення, як повне, так і часткове відновлення, а також можливість відновлення віртуальних машин або окремих файлів. Інтерфейс для адміністрування дозволяє налаштовувати систему резервного копіювання, моніторити її стан та управляти процесами. Цей інтерфейс може бути веб-інтерфейсом або програмою для робочого столу, що дозволяє користувачам налаштовувати графіки резервного копіювання, отримувати звіти та сповіщення про помилки або інші події. Моніторинг і звітність є важливими для підтримки належного функціонування системи. Вони дозволяють вчасно виявляти проблеми, перевіряти статус виконаних завдань, а також здійснювати аудит діяльності системи резервного копіювання. Безпека системи резервного копіювання включає в себе шифрування даних під час їх збереження і передачі, а також контроль доступу для обмеження



Обнаружен Плагиат: 0,15% <https://leadpanda.media/blog/rezervn...>

id: 23

доступу до резервних копій лише авторизованим користувачам. Це також може включати використання багатофакторної аутентифікації для захисту доступу до резервних копій.

Стійкість і масштабованість є важливими аспектами архітектури, оскільки система повинна бути здатною працювати в умовах зростаючих обсягів даних і забезпечувати високу доступність. Для цього можуть використовуватися розподілені рішення, що дозволяють масштабувати систему в разі потреби. Інтеграція з іншими системами також є важливою частиною архітектури резервного копіювання. Це дозволяє автоматизувати багато процесів і знижує ймовірність помилок, а також інтегрує резервне копіювання в більші інформаційні системи. Схему архітектури наведено на рис. 2.1. Рисунок 2.1 – Архітектура системи резервного копіювання

Загалом архітектура системи резервного копіювання повинна бути спроектована таким чином, щоб забезпечити не тільки ефективне збереження даних, але й їхню доступність для відновлення в будь-який момент, а також надавати необхідний рівень захисту даних. Окрім основних компонентів, архітектура системи резервного копіювання повинна враховувати також можливість гнучкого налаштування і адаптації до змін у вимогах бізнесу. Це включає в себе підтримку різних типів даних і специфічних вимог до їх зберігання, таких як швидке відновлення критично важливих даних або збереження резервних копій на різних носіях. Крім того, система резервного копіювання має бути здатною до інтеграції з іншими компонентами інфраструктури компанії, такими як системи для управління ІТ-ресурсами, антивірусні системи та рішення для моніторингу. Це дозволяє забезпечити комплексний підхід до управління даними та гарантує, що резервне копіювання буде виконуватись без помилок, а в разі виникнення проблем з системою — вчасно вжиті необхідні заходи. Не менш важливим аспектом є можливість використання стратегії




Цитирования: 0,03%

id: 24

"дізнаватися про помилки заздалегідь".

Для цього система повинна мати механізми передбачення потенційних проблем і попереджень, наприклад, для виявлення збоїв резервного копіювання до того, як це вплине на працездатність бізнесу. Це може включати в себе перевірку цілісності файлів резервних копій, автоматичну перевірку доступності серверів або створення журналів для аудиту змін у резервних копіях. Важливе значення має також забезпечення автоматизації всіх етапів процесу резервного копіювання, щоб мінімізувати людський фактор і зменшити ймовірність помилок. Зокрема, система повинна мати можливість самостійно планувати і запускати резервне копіювання на основі заздалегідь визначених налаштувань, наприклад, за часом, типом даних або важливістю об'єктів, що копіюються. Для великих організацій, де дані можуть зберігатися на численних пристроях і серверах, резервне копіювання має бути розподіленим. Це означає, що система повинна бути здатною керувати резервними копіями з різних джерел даних, синхронізувати їх і забезпечувати їх захист незалежно від того, де ці дані зберігаються — на локальних серверах, віртуальних машинах або в хмарних середовищах. Ще одним важливим аспектом є відновлення даних. Ключовим моментом є забезпечення різноманітних варіантів відновлення: від швидкого доступу до важливих файлів до відновлення цілих систем або баз даних у випадку серйозних збоїв. Це також має включати в себе можливість відновлення до певного часу

або точки, що може бути критично важливим для запобігання втратам важливих даних після помилок або атак. Для підвищення ефективності резервного копіювання в організаціях з великими обсягами даних часто використовують технології дедуплікації. Це дозволяє значно зменшити обсяг даних, що зберігаються, за рахунок видалення дубльованої інформації. Така технологія знижує витрати на зберігання і прискорює процес резервного копіювання. Крім того, з огляду на постійно зростаючі вимоги до захисту даних, система резервного копіювання повинна враховувати нормативні вимоги щодо зберігання і обробки даних. Врахування стандартів безпеки і конфіденційності, таких як **GDPR**, **HIPAA** або інших місцевих та міжнародних норм, є важливим аспектом, який гарантує відповідність усіх резервних копій вимогам законодавства. Загалом, архітектура системи резервного копіювання має бути гнучкою, масштабованою, безпечною та надійною, щоб ефективно обробляти великі обсяги даних, забезпечувати їх доступність для відновлення в разі непередбачених ситуацій і гарантувати збереження конфіденційності та цілісності даних [4]. Оскільки резервне копіювання має бути надійним і безпечним, тестування відновлення є критично важливим етапом. Окрім регулярних перевірок резервних копій на наявність помилок, необхідно також регулярно тестувати можливість їх відновлення. Це дозволяє виявити потенційні проблеми до того, як система буде використовуватися для відновлення даних у реальній ситуації. Також важливо мати можливість швидкого відновлення даних навіть у разі масштабних збоїв. Для цього може бути застосована стратегія відновлення після катастроф (**disaster recovery**), яка передбачає розподілене зберігання резервних копій на віддалених локаціях. Вибір таких локацій може варіюватися від хмарних сховищ до фізичних резервних копій на віддалених серверах, що дозволяє забезпечити доступність даних навіть у разі серйозних інцидентів на основних майданчиках. Що стосується безпеки, архітектура системи повинна бути розроблена з урахуванням найкращих практик кібербезпеки. Це включає в себе шифрування всіх резервних копій як на етапі передачі, так і на етапі зберігання. Враховуючи зростаючі загрози з боку кіберзлочинців, зокрема атаки з використанням програм-вимагачів, система резервного копіювання повинна мати можливості для

 **Обнаружен Плагиат: 0,15%** <http://imit.luguniv.edu.ua/sites/default...> + 4 ресурсів! id: 25

захисту від несанкціонованого доступу до даних. Одним із таких підходів є використання багатофакторної аутентифікації для доступу до резервних копій і

обмеження доступу лише для авторизованих користувачів. Система повинна бути достатньо гнучкою, щоб дозволяти організації налаштовувати стратегії резервного копіювання відповідно до змін в інфраструктурі або у вимогах до збереження даних. Наприклад, якщо кількість даних збільшується, система повинна мати можливість масштабувати ресурси для забезпечення безперервності процесів резервного копіювання без впливу на швидкість чи якість роботи. Ще одним важливим аспектом є інтеграція з іншими частинами інфраструктури, такими як системи управління даними, антивірусні системи, платформи для управління безпекою та інші корпоративні системи. Це дозволяє не тільки автоматизувати процес резервного копіювання, але й забезпечити його цілісність і безпеку на всіх етапах — від створення резервних копій до їх зберігання і відновлення [5]. І, звісно, важливо враховувати варіанти відновлення після катастроф. Система резервного копіювання повинна бути частиною більш широкої стратегії відновлення, яка включає також плани для відновлення критично важливих даних в разі серйозних збоїв чи інцидентів. Цей план має бути добре документований і перевірений у реальних умовах, щоб організація могла швидко і ефективно відновити роботу у разі непередбачених ситуацій [6]. У результаті, архітектура системи резервного копіювання повинна бути максимально адаптивною, безпечною і надійною, щоб забезпечити не тільки збереження даних, але й їх доступність у будь-який час, а також швидке відновлення у разі необхідності.

2.2. Вибір програмних засобів та бібліотек **Python**

Вибір програмних засобів та бібліотек **Python** для автоматизації резервного копіювання залежить від конкретних вимог до задач, таких як типи даних, розмір даних, рівень безпеки, а також від платформи, на якій працює система. **Python** має велику кількість бібліотек, які дозволяють інтегрувати резервне копіювання в автоматизовані робочі процеси. Нижче наведено кілька популярних програмних засобів та бібліотек для резервного копіювання, які можна використовувати для цих цілей. Однією з основних бібліотек **Python** для резервного копіювання є **shutil**. Це стандартна бібліотека, яка забезпечує функції для роботи з файлами та каталогами. За допомогою **shutil** можна легко копіювати, переміщати, видаляти файли, а також створювати архіви. Наприклад, бібліотека дозволяє створювати резервні копії каталогів

або окремих файлів шляхом копіювання їх в іншу директорию або на зовнішній диск. Для створення архівів можна використовувати бібліотеку [zipfile](#), яка дозволяє створювати та працювати з [ZIP](#)-архівами. Вона дозволяє автоматично архівувати дані перед їхнім копіюванням на резервне місце, що дозволяє економити простір для зберігання даних і знижує ризик втрати важливої інформації. [Rsync](#) є потужним інструментом для синхронізації файлів і директорій, і для [Python](#) є бібліотека [pyrsync](#). Вона дозволяє виконувати швидке резервне копіювання, зберігаючи лише зміни, що відбулися з часу останнього копіювання. Ця бібліотека ідеально підходить для великих обсягів даних, оскільки оптимізує процес передачі та зберігання лише тих даних, що змінюються. Ще однією потужною бібліотекою для роботи з резервними копіями є [paramiko](#), яка забезпечує [SSH](#)-з'єднання для віддаленого доступу до серверів. Якщо резервне копіювання потрібно виконувати на віддалених серверах, [paramiko](#) дозволяє передавати дані через зашифроване з'єднання, що забезпечує високий рівень безпеки. Для роботи з хмарними сервісами, такими як [Amazon S3](#), [Google Cloud Storage](#) або [Dropbox](#), існують спеціалізовані бібліотеки, які дозволяють інтегрувати [Python](#) зі службами хмарного зберігання. Для [Amazon S3](#) можна використовувати бібліотеку [boto3](#), яка надає можливість автоматизувати завантаження резервних копій в хмару. Для [Google Cloud Storage](#) є бібліотека [google-cloud-storage](#), яка дозволяє зберігати резервні копії в хмарі [Google](#). Для [Dropbox](#) можна використовувати [dropbox-sdk-python](#) для зберігання файлів у цьому популярному хмарному сервісі. Ще одним важливим аспектом є [shutil](#) та [os](#), які є основними бібліотеками для роботи з файлами та операціями над ними. Ці бібліотеки можуть бути використані для автоматизації процесу перевірки наявності резервних копій, організації структури каталогів для зберігання резервних копій, а також для виконання регулярних перевірок на наявність змін у файлах. Для шифрування резервних копій [Python](#) має бібліотеки, такі як [cryptography](#) та [pycryptodome](#), які дозволяють забезпечити високий рівень безпеки при передачі та зберіганні даних. Зашифрування резервних копій є важливим кроком для захисту конфіденційних даних і забезпечення їхньої безпеки. Для автоматизації завдань можна використовувати [schedule](#), бібліотеку для планування виконання завдань. Вона дозволяє налаштувати регулярні інтервали для виконання резервного копіювання, наприклад, щодня або щотижня. [Fabric](#) — це інструмент для автоматизації процесів на віддалених серверах. Він дозволяє виконувати резервне копіювання на віддалених машинах за допомогою [SSH](#), автоматизуючи всі етапи створення і зберігання резервних копій. Для перевірки цілісності резервних копій можна використовувати бібліотеки для обчислення контрольних сум, такі як [hashlib](#) або [xxhash](#). Вони дозволяють створювати хеші для файлів і перевіряти їх на цілісність, щоб бути впевненим, що резервні копії не пошкоджені. Залежно від конкретних потреб, можна комбінувати ці бібліотеки та інструменти для створення комплексних рішень для автоматизованого резервного копіювання. Це дозволяє створювати масштабовані та надійні системи для зберігання даних, які легко інтегруються з іншими системами та відповідають вимогам безпеки.

2.3. Реалізація клієнтської частини: створення та налаштування скрипта Користувач може вибрати файл або папку для копіювання через графічний інтерфейс, після чого ці файли або папки завантажуються на [FTP](#)-сервер. Скрипт підключається до сервера через [SSL](#) ([FTPS](#)) для забезпечення захищеного з'єднання. Після підключення перевіряється наявність віртуальної директорії на сервері, і якщо вона відсутня, вона створюється. Потім усі вибрані файли завантажуються на сервер у зазначену директорию. Графічний інтерфейс дозволяє користувачеві вибрати файл або папку для резервного копіювання. Після цього програма копіює ці файли в зазначену директорию на сервері. Якщо обрана директорія, програма обходить всі файли в ній, а якщо вибрано один файл, він буде просто переданий на сервер. Для графічного інтерфейсу використовується бібліотека [Tkinter](#). [Tkinter](#) — це стандартна бібліотека [Python](#) для створення графічних інтерфейсів користувача ([GUI](#)). Вона є об'єктно-орієнтованим шаром над бібліотекою [Tcl/Tk](#) і дозволяє розробникам створювати вікна, кнопки, текстові поля та інші елементи інтерфейсу без необхідності писати код на [Tcl](#). Основними компонентами [Tkinter](#) є різні віджети для створення інтерфейсу. До них належать кореневе вікно програми ([Tk](#)), текстова мітка ([Label](#)), кнопка ([Button](#)), однорядкове текстове поле ([Entry](#)), многострочне текстове поле ([Text](#)), прапорець ([Checkbutton](#)), перемикач ([Radiobutton](#)), повзунок ([Scale](#)), список ([Listbox](#)), полоса прокрутки ([Scrollbar](#)), меню ([Menu](#)) та холст для малювання графіки ([Canvas](#)). Ці віджети дозволяють створювати інтерфейси різної складності, від простих форм до складних графічних додатків. Для створення простого вікна в [Tkinter](#) можна використати код, який створює вікно з міткою та кнопкою. При натисканні кнопки текст мітки

змінюється. Наприклад, код створює вікно з назвою, задає розмір, додає мітку та кнопку з відповідною функцією обробки події натискання. Перевагами [Tkinter](#) є те, що це вбудована бібліотека, яка є частиною стандартної бібліотеки [Python](#), тому не потребує додаткових установок. Додатки, створені з її допомогою, працюють на різних операційних системах, таких як [Windows](#), [macOS](#) та [Linux](#). [Tkinter](#) легкий у вивченні та використанні, особливо для початківців, і підтримує різні менеджери геометрії для розміщення віджетів, як-от [pack\(\)](#), [grid\(\)](#) та [place\(\)](#). Серед недоліків варто зазначити, що інтерфейси, створені за допомогою [Tkinter](#), можуть виглядати застаріло, хоча з виходом [Tcl/Tk](#) 8.5 зовнішній вигляд було значно покращено. Офіційна документація [Tkinter](#) може бути недостатньо детальною, але існує багато ресурсів і прикладів у відкритому доступі. Підсумовуючи, [Tkinter](#) є потужним інструментом для створення графічних інтерфейсів у [Python](#). Його простота та інтеграція з [Python](#) роблять його відмінним вибором для розробки десктопних додатків. Хоча він може не мати найсучаснішого вигляду, його функціональність і кросплатформеність забезпечують популярність серед розробників. На рис. 2.2 наведено графічний інтерфейс розроблюваної програми, який є зручним і інтуїтивно зрозумілим для користувача. Основний екран програми містить всі необхідні елементи для налаштування процесу резервного копіювання. Тут користувач може легко ініціювати процес вибору файлів або папок для подальшого збереження. Рисунок 2.2 – Графічний інтерфейс програми

Наступним етапом є вибір файлів або папок для резервного копіювання. Користувач може вказати один або кілька файлів або навіть цілі директорії, що мають бути збережені. Для цього програма надає можливість вибору за допомогою спеціальної кнопки або вікна для вибору файлів (рис. 2.3). Це дозволяє точно налаштувати, які саме дані будуть резервуватися. Рисунок 2.3 – Вибір файлу або папки

Після вибору необхідних файлів або папок, користувач може запустити процес створення резервної копії. Це можна зробити за допомогою натискання відповідної кнопки в інтерфейсі. Після цього програма згенерує відповідне повідомлення про успіх процесу копіювання (рис. 2.4), що підтверджує правильність виконання операції. Рисунок 2.4 – Повідомлення про успіх

Створена резервна копія буде збережена в зазначеному місці, і користувач отримає підтвердження про успішне виконання копіювання. Це можна побачити на рис. 2.5, де показано, як виглядає збережена резервна копія в графічному інтерфейсі програми. Рисунок 2.5 – Створена резервна копія

Розроблена програма надає користувачу зручний графічний інтерфейс для вибору файлів або папок для резервного копіювання, що забезпечує інтуїтивно зрозумілий процес. Після вибору файлів або директорій, програма автоматично підключається до [FTP](#)-сервера через захищене з'єднання [SSL \(FTPS\)](#) для передачі даних. Якщо на сервері відсутня віртуальна директорія для зберігання копій, вона створюється, після чого файли завантажуються в обрану директорію. Процес резервного копіювання автоматизовано, що дозволяє зменшити потребу в ручному втручанні та мінімізувати ризик помилок. Користувач отримує підтвердження про успішне виконання завдання, що додає зручності в роботі з програмою. Також важливою частиною є можливість вибору як окремих файлів, так і цілих папок для збереження на сервері, що дає користувачеві гнучкість при роботі з великими обсягами даних. Завдяки цьому підходу, програма забезпечує надійне та безпечне резервне копіювання важливих даних, автоматизуючи більшість операцій та забезпечуючи простоту використання.

2.4. Налаштування [FTP](#)-сервера для зберігання резервних копій

Для налаштування [FTP](#)-сервера виконаємо наступні дії. Відкриваємо [FileZilla Server](#). Створимо користувача [Admin](#). Уведемо йому пароль (рис.2.6). Рисунок 2.6 – Створення користувача

Після створення користувача необхідно налаштувати шляхи для доступу до файлів на [FTP](#)-сервері. У графі

Цитування: 0,01%

id: 26

"Mount points"

потрібно створити віртуальний шлях, за яким буде здійснюватися доступ до резервних копій. Це дозволить забезпечити організацію і зручний доступ до даних. Також важливо визначити, які саме шляхи будуть дійсними для резервних копій, що дасть можливість уникнути проблем із доступом (рис. 2.6). Для забезпечення нормального функціонування [FTP](#)-сервера і доступу до резервних копій, необхідно встановити рівень доступу

Цитування: 0,01%

id: 27

"Read+Write",

що дозволить не лише читати дані, а й записувати їх на сервер. Це важливо для процесу збереження резервних копій на [FTP](#)-сервері, оскільки без цього користувач не зможе

вносити зміни в дані або записувати нові файли на сервер [12]. Таким чином, налаштування **FTP**-сервера

 **Обнаружен Плагиат: 0,64%** <https://phoneinfo8.info/rezervne-kopi...> + 5 ресурсов! id: 28

для зберігання резервних копій є важливим кроком для забезпечення надійного зберігання даних. Це дозволяє зберігати резервні копії на віддалених серверах, що підвищує рівень безпеки та дає можливість отримати доступ до резервних копій в будь-який час, звідки завгодно. Налаштування **FTP**-сервера для зберігання резервних копій є важливим кроком для забезпечення надійного та безпечного збереження даних. Створення користувача на сервері з відповідними параметрами доступу, а також налаштування віртуальних шляхів доступу до резервних копій дозволяє організувати ефективне управління даними. Визначення правильних шляхів для зберігання резервних копій і встановлення рівня доступу

 **Цитирования: 0,01%** id: 29

"Read+Write"

 **Обнаружен Плагиат: 0,24%** <https://phoneinfo8.info/rezervne-kopi...> + 5 ресурсов! id: 30

гарантує можливість не лише читати, а й записувати дані, що є важливим для безперебійної роботи системи резервного копіювання. Завдяки таким налаштуванням **FTP**-сервер забезпечує високий рівень безпеки і доступності для зберігання резервних копій.

Це дозволяє користувачам отримати доступ до збережених даних в будь-який час і з будь-якого місця, що робить процес резервного копіювання надійним та зручним. 2.5. Автоматизація процесу копіювання та планування завдань Автоматизація процесу резервного копіювання є важливим елементом для забезпечення регулярного збереження даних без втручання користувача. Це можна досягти за допомогою планування завдань на операційних системах, таких як **Linux** або **Windows**. Використання планувальників завдань, таких як **cron** на **Linux** або **Task Scheduler** на **Windows**, дозволяє створити графік для автоматичного виконання резервного копіювання в потрібний час. **Cron** на **Linux** — це інструмент для планування завдань, що дозволяє запускати певні скрипти або програми на основі заданого графіка. **Cron** використовується для автоматизації задач, таких як резервне копіювання файлів, оновлення системи або виконання скриптів, що потребують регулярного виконання. Щоб налаштувати резервне копіювання з використанням **cron**, потрібно створити **cron**-завдання, яке буде запускати певний **Python**-скрипт для копіювання файлів або створення архівів. Ось основні кроки для налаштування: Спочатку створюється **Python**-скрипт, який виконує резервне копіювання, наприклад, копіює важливі файли з однієї директорії в іншу або створює архіви. Скрипт може використовувати бібліотеки, як **shutil** для копіювання файлів або **zipfile** для архівування. Щоб автоматично запускати цей скрипт на регулярній основі, потрібно додати його в **cron**. Для цього використовується команда **crontab -e**, яка відкриває редактор для налаштування **cron**-завдань. Завдання в **cron** можна налаштувати таким чином, щоб воно запускалося щодня, щотижня або в будь-який інший час, визначений користувачем. Приклад налаштування завдання для запуску скрипта кожного дня о 3:00 ночі: 0 3 * * * /usr/bin/python3 /path/to/your_backup_script.py Важливо забезпечити механізм сповіщень або моніторингу для перевірки, чи успішно виконуються завдання резервного копіювання. Це можна реалізувати через логування в скрипті або налаштування сповіщень по електронній пошті, якщо резервне копіювання не вдалося. **Task Scheduler** на **Windows** має схожу функціональність, дозволяючи автоматизувати виконання завдань у заданий час. Щоб налаштувати автоматичне резервне копіювання за допомогою **Task Scheduler**, потрібно виконати кілька кроків: Як і в разі з **cron**, необхідно створити **Python**-скрипт для резервного копіювання, який виконуватиме потрібні операції, наприклад, копіювання файлів або створення архівів. Для налаштування автоматичного виконання цього скрипта потрібно відкрити **Task Scheduler**, створити нову задачу і вказати, коли саме скрипт має виконуватись. Наприклад, для щоденного запуску скрипта потрібно вибрати **New Task** і налаштувати запуск за розкладом, наприклад, кожен день о 3:00 ранку. У полі для сценарію потрібно вказати шлях до **Python**-інтерпретатора і шлях до вашого скрипта. Наприклад:

 **Цитирования: 0,01%** id: 31

"C:\path\to\python.exe"

 **Цитирования: 0,01%** id: 32

"C:\path\to\your_backup_script.py"

[Task Scheduler](#) надає можливість переглядати історію виконання завдань. Це дозволяє контролювати успішність резервного копіювання і вчасно виявляти проблеми. Планування завдань через [cron](#) або [Task Scheduler](#) забезпечує регулярне виконання резервного копіювання, що допомагає зменшити ризик втрати важливих даних. Обидва інструменти дозволяють налаштувати резервне копіювання на будь-який графік, що відповідає потребам організації або індивідуальних користувачів. Також важливо забезпечити належну безпеку для скриптів, обмежити доступ до конфіденційних даних і регулярно перевіряти стан процесу резервного копіювання [10, 13].

РОЗДІЛ 3. ТЕСТУВАННЯ ТА ОПТИМІЗАЦІЯ

3.1. Методологія тестування системи

Методологія тестування системи є важливою частиною процесу розробки програмного забезпечення, яка сприяє забезпеченню якості, безпеки та ефективності продукту. Тестування дозволяє не тільки перевірити відповідність системи вимогам, але й виявити потенційні дефекти, знизити ризики та покращити кінцевий результат. Першим етапом у методології тестування є планування тестування. На цьому етапі необхідно визначити основні цілі тестування, включаючи перевірку функціональності, безпеки, продуктивності та інших характеристик системи. Крім того, створюється загальний план тестування, в якому визначаються критерії успішності тестів, обсяги тестування, ресурси, необхідні для виконання тестів, а також терміни. План тестування повинно включати наступні аспекти: визначення тестових задач; стратегію виконання тестів; розподіл відповідальності за проведення тестів; визначення часу та ресурсів, необхідних для тестування. Тестові випадки є основою для виконання тестування, і саме їх якість визначає результативність тестування. Розробка тестових випадків починається з аналізу вимог і документації для визначення необхідних тестових сценаріїв. На цьому етапі важливо: визначити всі функціональні та нефункціональні вимоги до системи; Розробити конкретні тестові сценарії для кожної з вимог; прописати очікувані результати для кожного тесту. При розробці тестових випадків використовуються такі техніки: Тестування чорного ящика — тестування функціональності системи без знання її внутрішньої структури. Тестування білого ящика — тестування внутрішніх механізмів та структури системи. Тестування грей-бокса — поєднання тестування чорного та білого ящика. На етапі виконання тестування тестові випадки, розроблені раніше, виконуються в середовищі тестування. Зазвичай тестування включає автоматизовані та ручні тести, залежно від типу та складності операцій. Тести можуть бути розділені на такі категорії: Функціональні тести: перевірка функціональності програмного забезпечення відповідно до вимог. Нефункціональні тести: перевірка продуктивності, безпеки, сумісності, зручності користування тощо. Інтеграційні тести: перевірка взаємодії різних компонентів системи. Системні тести: перевірка всієї системи в цілому. Після виконання тестів важливо здійснити збір результатів і провести їх аналіз. Це дозволяє оцінити ефективність тестування, виявити дефекти та вивести рекомендації щодо подальших дій. Результати тестування документуються у вигляді звітів, де вказується: які тести були виконані; які тести пройшли успішно, а які зазнали невдачі; причини невдач, дефекти, виявлені під час тестування. Збір та аналіз результатів тестування дозволяє здійснити оцінку якості системи. На цьому етапі проводиться: Кореляційний і регресійний аналіз — оцінка зв'язку між різними параметрами системи. Багатомірний аналіз даних — допомагає виявити приховані закономірності в поведінці системи. Інформаційно-аналітичні системи — забезпечують збір та аналіз даних в реальному часі для своєчасного реагування на проблеми. Безпека є важливим аспектом тестування, і тестування безпеки включає в себе перевірку на вразливості, захист від несанкціонованого доступу, тестування на шахрайство та інші аспекти безпеки. Тестування на проникнення — перевірка захищеності системи від несанкціонованого доступу. Аудит безпеки — перевірка на відповідність безпековим стандартам [16]. Вибір методології тестування залежить від типу проекту та умов його реалізації. Основними підходами до тестування є: Тестування водоспадної моделі — тестування на кожному етапі розробки, після завершення кожного етапу. Аджайл-тестування — більш гнучкий підхід, що передбачає виконання тестів під час кожного циклу розробки та постійне взаємодія з замовником. Модульне тестування — перевірка окремих компонентів системи. Для виконання тестування використовуються різні інструменти: Інструменти для автоматизованого тестування: [Selenium](#), [JUnit](#), [TestComplete](#), [Postman](#) для тестування [API](#). Інструменти для тестування безпеки: [OWASP ZAP](#), [Burp Suite](#). Інструменти для аналізу продуктивності: [LoadRunner](#), [Apache JMeter](#) [17]. Методологія тестування системи дозволяє

забезпечити високу якість програмного забезпечення, виявити помилки та дефекти на ранніх етапах, а також запобігти появі критичних проблем під час експлуатації. Застосування правильної методології, вибір інструментів і постійний аналіз результатів тестування допомагають забезпечити відповідність системи вимогам і підвищити її ефективність. Основні змінні для налаштування FTP-з'єднання — це `FTP_HOST` (адреса сервера), `FTP_PORT` (порт, зазвичай 21), `FTP_USER` (логін), `FTP_PASS` (пароль) та `FTP_DIR` (віддалена директорія для збереження резервних копій). Глобальна змінна `LOCAL_PATH` зберігає шлях, який обирає користувач для копіювання — це може бути файл або папка на локальному комп'ютері. Функція `connect_to_ftp()` створює об'єкт `ftp` для підключення через `ftplib.FTP_TLS()`, підключається до сервера за адресою та портом (`FTP_HOST`, `FTP_PORT`), виконує вхід з логіном і паролем (`FTP_USER`, `FTP_PASS`) та встановлює захищений режим передачі даних. Якщо підключення не вдається, користувач отримує повідомлення про помилку. Функція `upload_files(ftp, local_path, remote_dir)` відповідає за завантаження файлів. Вона приймає об'єкт FTP-з'єднання, локальний шлях `local_path` та віддалену директорію `remote_dir`. Якщо `local_path` — папка, функція рекурсивно обходить усі файли в ній, для кожного формує локальний шлях `local_file_path` та віддалений шлях `remote_file_path`, і завантажує файли на сервер. Якщо `local_path` — файл, він одразу завантажується у вказану папку на сервері. В разі помилки виводиться повідомлення. Функція `backup_files()` перевіряє, чи обрано шлях `LOCAL_PATH`, потім викликає `connect_to_ftp()`, формує віддалену папку для резервної копії, комбінуючи `FTP_DIR` та ім'я локального файлу чи папки, і передає управління `upload_files()`. По завершенню з'єднання закривається, а користувач отримує повідомлення про успіх. Графічний інтерфейс створено за допомогою Tkinter: `path_entry` — поле для відображення або введення шляху до локального файлу/папки (`LOCAL_PATH`), кнопка `browse_button` відкриває діалог вибору файлу або папки, шлях зберігається у `LOCAL_PATH`, кнопка `start_button` запускає функцію `start_backup()`, яка передає шлях із `path_entry` до `LOCAL_PATH` і ініціює резервне копіювання. Основний цикл `root.mainloop()` запускає графічний інтерфейс і підтримує роботу програми в інтерактивному режимі.

3.2. Оцінка продуктивності та аналіз помилок

Оцінка продуктивності та аналіз помилок є важливими етапами в процесі тестування програмного забезпечення. Вони дозволяють не лише виявити дефекти, але й оцінити ефективність системи та процесу тестування, що сприяє підвищенню якості кінцевого продукту. Оцінка продуктивності системи тестування включає в себе вимірювання та аналіз таких параметрів:

- Час виконання тестів: визначення часу, необхідного для виконання тестових сценаріїв, що дозволяє оцінити ефективність тестування та виявити можливі затримки.
- Покриття тестами: оцінка того, яку частину коду або функціональності системи охоплено тестами, що допомагає виявити невиконані ділянки.
- Продуктивність тестових інструментів: аналіз ефективності використаних інструментів для автоматизації тестування, їх впливу на загальну продуктивність процесу тестування.
- Витрати ресурсів: оцінка використання апаратних та програмних ресурсів під час тестування, що дозволяє оптимізувати процеси та зменшити витрати.

Аналіз помилок є ключовим етапом у процесі тестування, що дозволяє:

- Виявити причини дефектів: дослідження кореневих причин виникнення помилок допомагає запобігти їх повторенню в майбутньому.
- Класифікувати помилки: розподіл помилок за типами (функціональні, нефункціональні, безпеки тощо) дозволяє визначити пріоритети для їх виправлення.
- Оцінити вплив помилок: визначення серйозності дефектів допомагає в прийнятті рішень щодо їх виправлення та впливу на реліз продукту.
- Визначити ефективність тестування: аналіз кількості виявлених помилок на одиницю часу або за допомогою конкретних тестових сценаріїв дозволяє оцінити ефективність процесу тестування.

Метод статистичної моделі Миллса - це статистичний підхід, що дозволяє оцінити кількість невиявлених помилок у програмному коді. Він базується на порівнянні кількості виявлених дефектів з кількістю штучно внесених помилок, що дозволяє зробити висновки про загальну кількість помилок у системі. Аналіз причинно-наслідкових зв'язків: дослідження взаємозв'язків між різними факторами, що впливають на якість продукту, дозволяє виявити основні причини дефектів та оптимізувати процеси розробки та тестування. Використання метрик якості: вимірювання таких показників, як кількість дефектів на одиницю коду, час на виправлення помилки, кількість повторних дефектів, дозволяє оцінити ефективність процесу тестування та виявити області для покращення [16].

Оцінка продуктивності та аналіз помилок є невід'ємною частиною процесу тестування програмного забезпечення. Вони дозволяють не лише виявити дефекти, але й оцінити ефективність тестування, виявити слабкі місця в процесах та прийняти обґрунтовані рішення щодо поліпшення якості продукту.

Використання статистичних методів, таких як модель Миллса, а також аналіз причинно-наслідкових зв'язків і метрик якості, сприяє підвищенню ефективності тестування та забезпеченню високої якості кінцевого продукту.

3.3. Оптимізація швидкості передачі даних

Оптимізація швидкості передачі даних є критичним аспектом у забезпеченні ефективної роботи банкоматної мережі. Вона включає в себе вдосконалення інфраструктури, налаштування мережевих протоколів та використання сучасних технологій для зменшення затримок і підвищення пропускної здатності. Для забезпечення високої швидкості передачі даних необхідно обрати відповідне мережеве обладнання, таке як маршрутизатори, комутатори та точки доступу, що підтримують сучасні стандарти, наприклад, [Ethernet 10 Gbps](#) або [Wi-Fi 6](#). Також важливо правильно налаштувати параметри обладнання, включаючи [MTU \(Maximum Transmission Unit\)](#), [QoS \(Quality of Service\)](#) та [VLAN \(Virtual Local Area Network\)](#), щоб оптимізувати трафік і зменшити затримки. Віртуалізація мереж, зокрема програмно визначені мережі ([SDN](#)) та віртуалізація функцій мереж ([NFV](#)), дозволяють централізовано керувати трафіком, автоматично налаштовувати маршрути та балансувати навантаження між серверами. Це сприяє зменшенню затримок і підвищенню ефективності використання мережевих ресурсів. Правильне проектування топології мережі є важливим для забезпечення ефективної передачі даних. Використання ієрархічної структури з виділенням основних і резервних каналів дозволяє зменшити навантаження на окремі сегменти мережі та забезпечити резервування для критичних з'єднань [18]. Балансування навантаження між серверами та мережевими каналами дозволяє рівномірно розподіляти трафік, запобігаючи перевантаженню окремих вузлів і зменшуючи час відгуку системи. Це особливо важливо для банкоматної мережі, де кожен затриманий запит може вплинути на обслуговування клієнтів. Регулярний моніторинг продуктивності мережі за допомогою інструментів, таких як [PRTG](#), [Nagios](#) або [Zabbix](#), дозволяє вчасно виявляти проблеми, такі як перевантаження каналів передачі або зниження швидкості роботи. Аналіз отриманих даних допомагає виявити вузькі місця та оптимізувати мережеву інфраструктуру [19].

Оптимізація швидкості передачі даних у банкоматній мережі

є багатогранним процесом, що включає в себе вдосконалення мережевого обладнання, використання сучасних технологій віртуалізації, оптимізацію топології мережі, балансування навантаження та постійний моніторинг продуктивності. Комплексний підхід до цих аспектів дозволяє забезпечити високу швидкість передачі даних, зменшити затримки та підвищити ефективність роботи банкоматної мережі.

3.4. Захист та безпека переданих файлів

Забезпечення захисту та безпеки переданих файлів є критичним аспектом у функціонуванні банкоматної мережі, оскільки передача конфіденційної інформації, такої як дані карток, особисті дані користувачів та фінансові транзакції, вимагає високого рівня захисту від несанкціонованого доступу, перехоплення та модифікації. Для захисту переданих файлів необхідно використовувати надійні протоколи шифрування, такі як [TLS \(Transport Layer Security\)](#), [SSL \(Secure Sockets Layer\)](#), [IPsec \(Internet Protocol Security\)](#) та [SFTP \(Secure File Transfer Protocol\)](#). Ці протоколи забезпечують шифрування даних та захищають їх від несанкціонованого доступу під час передачі. Важливим елементом безпеки є також аутентифікація та авторизація. Для обмеження доступу до переданих файлів необхідно впровадити надійні механізми аутентифікації, такі як багатофакторна аутентифікація ([MFA](#)), рольова авторизація та сертифікати X.509. Це дозволяє забезпечити контроль за доступом та знизити ризик несанкціонованого доступу до важливих файлів. Для виявлення та запобігання несанкціонованому доступу до файлів необхідно використовувати системи виявлення та запобігання вторгненням ([IDS/IPS](#)), а також вести журнали аудиту для фіксації всіх операцій з файлами. Це дозволяє оперативно реагувати на потенційні загрози та підвищити рівень безпеки. Захист від атак також є важливим аспектом. Атаки

Цитування: 0,01%

id: 33

"людина посередині"

([MITM](#)) запобігаються за допомогою шифрування з'єднань та автентифікації учасників. Для забезпечення цілісності даних використовуються хеш-функції та цифрові підписи, що дозволяють виявити будь-які зміни в переданих файлах. Для запобігання атакам на доступність впроваджуються механізми резервування та відмовостійкості. Регулярне оновлення та патчі програмного забезпечення також є важливими для забезпечення безпеки переданих файлів. Автоматичне застосування оновлень для операційних систем, мережевих пристроїв та програмного забезпечення дозволяє усувати відомі вразливості і підтримувати систему в актуальному стані. Забезпечення захисту та безпеки переданих

файлів у банкоматній мережі вимагає комплексного підходу, що включає використання надійних протоколів шифрування, впровадження ефективних механізмів аутентифікації та авторизації, контролю доступу та моніторингу, захисту від атак та регулярного оновлення систем [20]. Лише за умови реалізації цих заходів можна забезпечити конфіденційність, цілісність та доступність переданих файлів, що є критичними для безпеки фінансових операцій та довіри користувачів до банкоматної мережі.

3.5. Перспективи розвитку та покращення системи

Перспективи розвитку та покращення системи автоматизованого резервного копіювання файлів у локальній мережі з використанням [Python](#) та [FTP](#). Автоматизація процесу резервного копіювання є критично важливою для забезпечення безпеки даних у локальних мережах. Використання [Python](#) та [FTP](#)-протоколу для цієї мети вже довело свою ефективність. Однак, з огляду на швидкий розвиток технологій, існують численні можливості для вдосконалення та розширення функціональності існуючих систем. Для підвищення надійності зберігання резервних копій доцільно інтегрувати систему з хмарними сервісами, такими як [Google Drive](#), [Dropbox](#) або [Amazon S3](#). Це дозволить забезпечити додатковий рівень захисту даних та забезпечити доступ до резервних копій з будь-якої точки світу. [FTP](#) є застарілим і не забезпечує належного рівня безпеки. Рекомендується перейти на більш сучасні та захищені протоколи, такі як [SFTP](#) або [FTPS](#), які забезпечують шифрування даних під час передачі, що значно знижує ризик їх перехоплення або модифікації. Для збереження історії змін у файлах та можливості відновлення попередніх версій доцільно впровадити систему версій. Це дозволить зберігати кілька копій файлів з різними часовими мітками та забезпечить гнучкість при відновленні даних. Використання технологій, таких як моніторинг файлової системи або хешування, дозволить автоматично виявляти зміни у файлах та запускати процес резервного копіювання лише для змінених даних. Це зменшить навантаження на мережу та зекономить ресурси. Розробка зручного графічного інтерфейсу користувача ([GUI](#)) для налаштування та моніторингу процесу резервного копіювання зробить систему більш доступною для користувачів без технічної підготовки. Інтеграція з існуючими системами моніторингу дозволить оперативно отримувати сповіщення про успішне або неуспішне виконання резервного копіювання, а також про можливі помилки або збої в процесі. Це забезпечить своєчасне реагування на проблеми та підвищить надійність системи.

Перспективи розвитку системи автоматизованого резервного копіювання файлів у локальній мережі з використанням [Python](#) та [FTP](#) полягають у впровадженні сучасних технологій та протоколів, що забезпечать підвищення безпеки, ефективності та зручності використання системи. Інтеграція з хмарними сервісами, використання захищених протоколів, впровадження системи версій та автоматичне виявлення змін у файлах дозволять створити надійне та ефективне рішення для резервного копіювання даних. □

ВИСНОВКИ

У процесі виконання дипломної роботи було досліджено актуальність та можливості застосування автоматизованих систем резервного копіювання для захисту важливих даних у середовищі розподілених обчислень. Однією з основних цілей роботи було створення програмної системи автоматизованого резервного копіювання з використанням протоколу [FTP](#) і мови програмування [Python](#). Завдяки цьому було забезпечено ефективне рішення для регулярного збереження даних у реальному часі, що є важливим для підприємств, організацій і користувачів, які працюють із великими обсягами інформації. У першому розділі роботи було проведено аналіз існуючих технологій і протоколів для резервного копіювання даних. Вивчено переваги та недоліки різних методів, що дозволило вибрати найбільш підходящий для реалізації системи [FTP](#). Розгляд альтернативних способів резервного копіювання допоміг зрозуміти важливість оптимізації та забезпечення надійності даних при передаванні через мережу. У другому розділі було представлено процес проєктування та реалізації програмного забезпечення для автоматизації резервного копіювання. Особлива увага була приділена створенню клієнтської частини системи, яка дозволяє автоматично здійснювати копіювання файлів на віддалений сервер. Застосування мови [Python](#) забезпечило простоту інтеграції з [FTP](#)-сервером і дозволило реалізувати зручний і зрозумілий інтерфейс для користувачів, що значно спрощує процес резервного копіювання. Третій розділ дипломної роботи був присвячений тестуванню розробленої системи, оцінці її продуктивності та аналізу можливих помилок. У ході тестування було перевірено різні аспекти системи, зокрема швидкість передачі даних, надійність з'єднання з сервером, а також безпеку переданих даних. Згідно з результатами тестування, система показала високу ефективність і стабільність роботи навіть при великих обсягах даних. Оптимізація швидкості передачі даних була досягнута через використання методів стиснення файлів перед їх передачею,

що значно зменшило час копіювання великих обсягів інформації. Захист переданих файлів реалізовано за допомогою шифрування, що дозволяє забезпечити високу безпеку даних під час їх передачі по мережі. Завдяки автоматизації процесу резервного копіювання вдалося знизити кількість помилок, що можуть виникати через людський фактор, а також забезпечити безперервність і своєчасність створення резервних копій важливих даних. Крім того, система демонструє високу гнучкість і зручність в налаштуванні, що дозволяє використовувати її в різних сферах, де необхідна регулярна передача та збереження інформації. У цілому, розроблена система автоматизованого резервного копіювання є надійним і ефективним інструментом для забезпечення безпеки даних у реальному часі. Це рішення може бути використане як в малих організаціях, так і в великих корпораціях, де важливо мати автоматизований процес збереження даних і їх швидке відновлення в разі непередбачених ситуацій. Дослідження показали, що запропонована система не лише задовольняє вимоги до надійності та швидкості резервного копіювання, але й може бути покращена шляхом впровадження додаткових функцій, таких як автоматичне відновлення даних, а також підтримка нових методів шифрування для ще більшого підвищення рівня безпеки.

Заявление об ограничении ответственности:

Этот отчет должен быть правильно истолкован и проанализирован квалифицированным специалистом, который несет ответственность за оценку!

Любая информация, представленная в этом отчете, не является окончательной и подлежит ручному просмотру и анализу. Пожалуйста, следуйте инструкциям: [Рекомендации по оценке](#)

Детектор Плагиата - Ваше право на оригинальность! ☐ SkyLine LLC