МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМ. С.З. ҐЖИЦЬКОГО
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему: **«Створення системи резервного копіювання даних у хмарних середовищах»**

Виконав: студент 5 курсу групи Іт-51з

Спеціальності 126 «Інформаційні системи та технології»
(шифр і назва)

Слічний Назар Андрійович
(Прізвище та ініціали)

Керівник: к.т.н., доцент Падюка Р.І.
(Прізвище та ініціали)

ДУБЛЯНИ-2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА
ІНФОРМАЦІЙНИХ ТЕХНЕОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Перший (бакалаврський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри_____

д.т.н., проф. А. М. Тригуба
«____» _____2026 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Слічному Назару Андрійовичу

1. Тема роботи: «Створення системи резервного копіювання даних у хмарних середовищах»

Керівник роботи Падюка Роман Іванович, к.т.н., доцент
затверджені наказом по університету від 14.01.2026 року № 32-4

2. Строк подання студентом роботи 15.03.2026 р.
3. Вихідні дані до роботи: *1) Вимоги до систем резервного копіювання даних у хмарних середовищах; 2) Характеристики сервісів Google Cloud для зберігання та відновлення даних; 3) Методи проєктування інформаційних систем і програмних модулів; 4) Методика оцінювання надійності, безпеки та ефективності системи резервного копіювання.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)
Вступ.
1. Аналіз предметної області.
2.Постановка задачі створення системи резервного копіювання даних у хмарних середовищах
3.Проектування інформаційної системи резервного копіювання даних у хмарних середовищах
4. Охорона праці та безпека в надзвичайних ситуаціях
Висновки та пропозиції.
Список використаних джерел.

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових креслень): *Актуальність дослідження; Мета та завдання роботи; Об'єкт і предмет дослідження; Теоретичні основи резервного копіювання; Сучасні підходи у хмарних середовищах; Вимоги до розроблюваної системи; Вибір Google Cloud як основи системи; Архітектура інформаційної системи; Головне вікно та логіка взаємодії з користувачем; Вікно налаштування резервного копіювання; Реалізовані модулі та загальні результати.*

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3	*Падюка Р.І., доцент кафедри ІТ*		
4	*Городецький І.М., доцент кафедри інженерної механіки*		

7. Дата видачі завдання 16 січня 2026 р.

Календарний план

№ з/п	Назва етапів дипломного проекту	Терміни виконання етапів роботи	Примітка
1	*Написання першого розділу*	*16.01.26-30.01.26*	
2	*Виконання другого розділу та аркушів ілюстраційного матеріалу до нього*	*31.01-15.02.26*	
3.	*Виконання третього розділу та аркушів ілюстраційного матеріалу до нього*	*16.02-28.02.26*	
4.	*Написання розділу «Охорона праці»*	*01-05.03.26*	
5.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів ілюстраційного матеріалу*	*06-10.03.26*	
6.	*Завершення роботи вцілому*	*11.03 - 15.03.26*	

Студент _____ Слічний Н.А.
(підпис)

Керівник роботи _____Падюка Р.І.
(підпис)

УДК 004.75:004.056.53

Створення системи резервного копіювання даних у хмарних середовищах. Слічний Н.А. Кафедра ІТ – Дубляни, Львівський НУВМБ ім. С.З. Гжицького, 2026.

Кваліфікаційна робота: 66 с. текст. част., 11 рис., 12 арк. ілюстраційного матеріалу, 34 джерел.

У кваліфікаційній роботі розглянуто питання створення системи резервного копіювання даних у хмарних середовищах. Проведено аналіз предметної області, узагальнено сучасні підходи до резервного копіювання, визначено вимоги до надійності, безпеки, масштабованості та відновлення даних. Обґрунтовано вибір платформи Google Cloud як основи для реалізації системи. Запропоновано архітектуру інформаційної системи, що включає модулі керування інтерфейсом користувача, формування резервних копій, налаштування параметрів, планування запуску та передавання даних у хмарне сховище. Описано програмну реалізацію основних модулів і принципи їх взаємодії. Практичне значення роботи полягає у можливості використання розробленого рішення для підвищення надійності зберігання даних і автоматизації процесів резервного копіювання.

Ключові слова: резервне копіювання, хмарне середовище, Google Cloud, інформаційна система, захист даних, відновлення даних, автоматизація.

Key words: backup, cloud environment, Google Cloud, information system, data protection, data recovery, automation.

ЗМІСТ

ВСТУП

Цифровізація суспільства, активний розвиток інформаційних технологій та широке впровадження хмарних сервісів потребує вирішення питання забезпечення надійного зберігання даних. Інформація є одним із найважливіших ресурсів діяльності підприємств, установ, організацій і приватних користувачів, а її втрата, пошкодження або несанкціоноване знищення можуть призвести до істотних матеріальних збитків, порушення безперервності бізнес-процесів, втрати довіри користувачів та зниження ефективності функціонування інформаційних систем.

Одним із ключових засобів забезпечення цілісності та доступності інформації є резервне копіювання даних. Традиційні підходи до резервного копіювання, які базуються на використанні локальних носіїв та внутрішньої інфраструктури організації, у багатьох випадках уже не повною мірою відповідають сучасним вимогам щодо масштабованості, гнучкості, доступності та відмовостійкості. У зв’язку з цим усе більшого поширення набувають хмарні середовища, які надають можливість зберігати резервні копії на віддалених серверах, автоматизувати процес резервування, оптимізувати витрати на інфраструктуру та підвищувати надійність збереження інформаційних ресурсів.

Разом із тим створення системи резервного копіювання даних у хмарних середовищах супроводжується низкою проблем, що потребують ґрунтовного дослідження. До таких проблем належать забезпечення конфіденційності даних під час передавання та зберігання, вибір ефективних методів створення резервних копій, організація механізмів швидкого відновлення інформації, підтримка цілісності даних, а також забезпечення сумісності між локальними інформаційними системами та хмарними платформами. У сучасних умовах підвищення кіберзагроз, зростання обсягів інформації та необхідності забезпечення безперервного доступу до цифрових ресурсів зазначені питання мають не лише теоретичне, а й важливе практичне значення.

Актуальність теми бакалаврської роботи зумовлена необхідністю розроблення ефективних засобів резервного копіювання даних, які б відповідали сучасним вимогам щодо надійності, безпеки, автоматизації та доступності. Використання хмарних технологій у цій сфері відкриває нові можливості для створення гнучких і масштабованих рішень, однак потребує обґрунтованого підходу до вибору архітектури системи, методів захисту інформації та програмних засобів реалізації. Саме тому дослідження особливостей побудови систем резервного копіювання у хмарних середовищах є своєчасним і доцільним.

Необхідність дослідження цієї теми пояснюється тим, що резервне копіювання є невід'ємним елементом забезпечення надійного функціонування інформаційних систем. За відсутності ефективної системи резервування навіть незначні технічні несправності, помилки користувачів або дії шкідливого програмного забезпечення можуть спричинити втрату важливої інформації. Хмарні середовища, у свою чергу, надають додаткові можливості для побудови сучасних систем резервного копіювання, однак вимагають детального аналізу принципів їх функціонування, способів організації зберігання резервних копій та механізмів захисту даних.

Метою бакалаврської роботи є розроблення системи резервного копіювання даних у хмарних середовищах, яка забезпечує надійне збереження інформації, автоматизацію процесів створення резервних копій, захист даних та можливість їх оперативного відновлення у разі втрати або пошкодження.

Для досягнення поставленої мети в роботі необхідно розв'язати такі завдання:

- дослідити теоретичні засади резервного копіювання даних та особливості використання хмарних середовищ;
- проаналізувати сучасні підходи, методи та програмні засоби резервного копіювання;
- визначити основні вимоги до системи резервного копіювання даних у хмарному середовищі;

- спроєктувати архітектуру системи резервного копіювання з урахуванням вимог безпеки, надійності та масштабованості;
- обґрунтувати вибір технологій і програмних засобів для реалізації системи;
- реалізувати програмне рішення для створення, зберігання та відновлення резервних копій;

Об’єктом дослідження є процес резервного копіювання та відновлення даних в інформаційних системах.

Предметом дослідження є методи, моделі, архітектурні рішення та програмні засоби створення системи резервного копіювання даних у хмарних середовищах.

Інформаційну базу дослідження становлять наукові праці вітчизняних і зарубіжних учених, присвячені питанням резервного копіювання даних, хмарних обчислень, інформаційної безпеки та проєктування інформаційних систем, технічна документація сучасних хмарних платформ і сервісів, нормативно-довідкові матеріали, а також результати власних досліджень, проєктування, програмної реалізації та тестування розроблюваної системи.

Практичне значення одержаних результатів полягає в можливості використання розробленої системи резервного копіювання даних у діяльності підприємств, установ, організацій та інших користувачів, які потребують надійного, безпечного та доступного механізму збереження інформації. Запропоноване рішення може бути застосоване для автоматизації процесів резервного копіювання, підвищення рівня захисту даних, зменшення ризиків їх втрати, а також скорочення часу відновлення інформації після збоїв. Окремі положення та результати роботи можуть бути використані у навчальному процесі під час вивчення дисциплін, пов’язаних із хмарними технологіями, інформаційними системами та захистом даних.

РОЗДІЛ 1.
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Теоретичні засади резервного копіювання даних у хмарних середовищах

У практично всіх інформаційних системах дані виступають одним із основних ресурсів, від якого залежить стабільність функціонування цифрових сервісів, прикладного програмного забезпечення, корпоративних інформаційних систем та мережевої інфраструктури. Зростання обсягів електронної інформації, активне використання віддалених сервісів, віртуалізація обчислювальних ресурсів і перехід до хмарних моделей надання IT-послуг суттєво підвищують вимоги до збереження даних. За таких умов резервне копіювання стає базовою складовою забезпечення доступності та цілісності інформації, оскільки саме наявність актуальних резервних копій дає змогу відновити роботу системи після технічного збою, втрати файлів, пошкодження сховища, помилкових дій користувача або кібератаки [1, 2].

Під резервним копіюванням даних доцільно розуміти процес створення копій інформаційних ресурсів з метою їх подальшого відновлення у разі часткової або повної втрати первинних даних. На відміну від звичайного дублювання файлів, система резервного копіювання передбачає визначену політику зберігання копій, періодичність виконання операцій резервування, контроль цілісності копій, засоби керування версіями, а також процедури відновлення. Ефективність такої системи визначається не лише фактом створення копії, а й можливістю гарантованого і своєчасного повернення даних у працездатний стан [2, 3].

Хмарне середовище, у межах якого реалізується резервне копіювання, являє собою модель надання обчислювальних ресурсів у вигляді сервісів через мережу. Відповідно до визначення NIST, хмарні обчислення характеризуються самообслуговуванням на вимогу, широким мережевим доступом, об’єднанням

ресурсів, швидкою еластичністю та вимірюваністю сервісів [1]. Застосування таких принципів до задач резервного копіювання дозволяє відмовитися від жорсткої прив’язки до локального обладнання, спростити масштабування обсягів сховища, автоматизувати виконання резервних операцій і забезпечити географічно віддалене зберігання копій. Саме віддалене розміщення резервних копій є важливою перевагою хмарного підходу, оскільки дозволяє зменшити ризик одночасної втрати робочих і резервних даних унаслідок фізичного пошкодження локальної інфраструктури [1, 3].

У загальному вигляді система резервного копіювання у хмарному середовищі складається з джерела даних, програмного агента або сервісу резервування, каналу передачі, механізмів шифрування і стиснення, хмарного сховища та підсистеми відновлення. Джерелом можуть бути окремі файли, бази даних, віртуальні машини, контейнери, сервери застосунків або цілі інформаційні системи. Програмний компонент резервування здійснює аналіз об’єктів копіювання, формує резервний набір, шифрує його, передає через мережу до сховища та підтримує журнал виконаних операцій. Підсистема відновлення, своєю чергою, повинна забезпечувати вибір необхідної версії даних, перевірку її цілісності та повернення інформації у визначене середовище виконання [2, 3].

Традиційно виділяють три основні типи резервного копіювання: повне, диференціальне та інкрементне (рис. 1.1). Повне резервне копіювання передбачає створення повної копії всього обраного набору даних. Його перевагою є простота відновлення, оскільки для повернення інформації достатньо однієї актуальної копії. Водночас цей підхід потребує найбільших витрат дискового простору та часу на створення резервного набору. Диференціальне копіювання зберігає всі зміни, що відбулися після останнього повного копіювання, завдяки чому скорочуються витрати часу порівняно з повним резервуванням, а відновлення виконується на основі двох наборів, а саме останньої повної та останньої диференціальної копії. Інкрементне копіювання фіксує лише зміни з моменту попередньої резервної операції, що дозволяє

мінімізувати обсяг передаваних даних і знизити навантаження на канал зв’язку, однак ускладнює процедуру відновлення, оскільки вимагає послідовного використання кількох резервних наборів [2, 4].

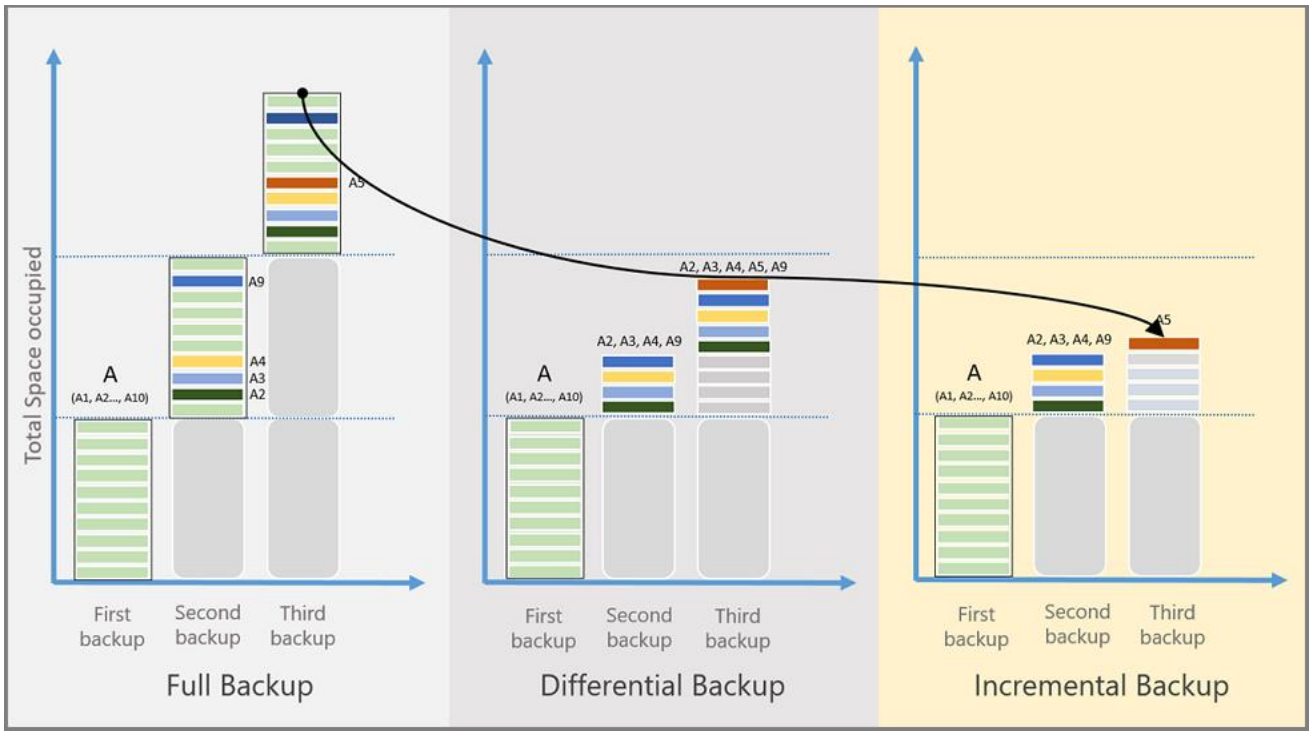

Рисунок 1.1 – Основні типи резервного копіювання: повне, диференціальне та інкрементне

Для хмарних середовищ особливої ваги набуває вибір такого механізму резервування, який забезпечує баланс між швидкістю створення копій, економією мережевих ресурсів і швидкістю відновлення. У практиці побудови сучасних систем часто застосовується комбінований підхід, за якого повні копії створюються через певні проміжки часу, а між ними виконуються інкрементні або диференціальні резервні операції. Такий підхід дозволяє оптимізувати використання хмарного сховища і водночас зберігати прийнятний рівень готовності до відновлення [2, 3].

Однією з ключових характеристик систем резервного копіювання є політика зберігання копій. Вона визначає, як довго резервні набори зберігаються у сховищі, скільки версій даних потрібно підтримувати та за якими правилами

видаляються застарілі копії. У хмарних середовищах політика зберігання безпосередньо впливає не лише на рівень захищеності інформації, а й на економічні параметри системи, оскільки спожитий обсяг сховища, операції читання та запису, а іноді й вихідний трафік оплачуються відповідно до тарифної моделі провайдера [3, 5]. Тому проектування системи резервного копіювання має враховувати не тільки технічні, а й організаційно-економічні аспекти її функціонування.

Окремої уваги потребує питання забезпечення інформаційної безпеки під час резервного копіювання у хмарі. Передавання даних через мережу та їх зберігання у віддаленій інфраструктурі формують додаткові ризики, пов’язані з несанкціонованим доступом, компрометацією облікових даних, витоком конфіденційної інформації та підміною резервних копій. З огляду на це сучасна система резервування повинна підтримувати шифрування даних під час передавання і зберігання, багаторівневу автентифікацію адміністративного доступу, журналювання операцій, контроль версій, політики розмежування прав доступу, а також механізми захисту від шкідливого програмного забезпечення, зокрема від програм-вимагачів [2, 3, 5]. У багатьох платформах хмарного резервування додатково реалізуються функції незмінюваного зберігання окремих копій, що знижує ризик їх навмисного видалення або модифікації.

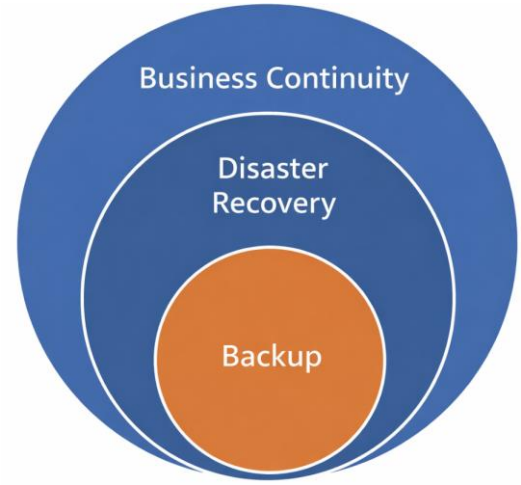

Рисунок 1.2 – Співвідношення резервного копіювання, аварійного відновлення та безперервності бізнесу

Для оцінювання якості системи резервного копіювання важливими є показники RPO та RTO. Показник RPO, тобто допустима точка відновлення, характеризує максимальний обсяг даних, який може бути втрачено між моментом останнього успішного копіювання та виникненням збою. Показник RTO, тобто допустимий час відновлення, визначає граничний інтервал, протягом якого працездатність системи має бути відновлена. Чим менші значення RPO і RTO, тим вищі вимоги ставляться до періодичності резервування, пропускної здатності мережі, продуктивності хмарного сховища та організації самого процесу відновлення [2, 5]. Таким чином, ці показники виступають одними з базових критеріїв проектування системи резервного копіювання в хмарному середовищі.

У предметній області резервного копіювання у хмарних середовищах важливим є також вибір моделі розгортання. Найпоширенішими є публічна, приватна та гібридна хмара. Публічна хмара передбачає використання інфраструктури зовнішнього провайдера, що забезпечує високу масштабованість і порівняно невисокий поріг входу. Приватна хмара будується на ресурсах окремої організації та забезпечує вищий рівень контролю над даними, але вимагає більших витрат на підтримку інфраструктури. Гібридна модель поєднує переваги двох попередніх підходів і часто є найбільш доцільною для резервного копіювання [1, 3].

Важливим аспектом аналізу предметної області є дотримання принципу 3-2-1, відповідно до якого рекомендується зберігати не менше трьох копій даних, використовувати щонайменше два різні типи носіїв та розміщувати хоча б одну копію поза межами основного середовища експлуатації. У хмарних системах ця концепція отримує подальший розвиток завдяки можливості географічного резервування, створення багатозонних копій та автоматичного переміщення резервних наборів між класами зберігання [2, 3].

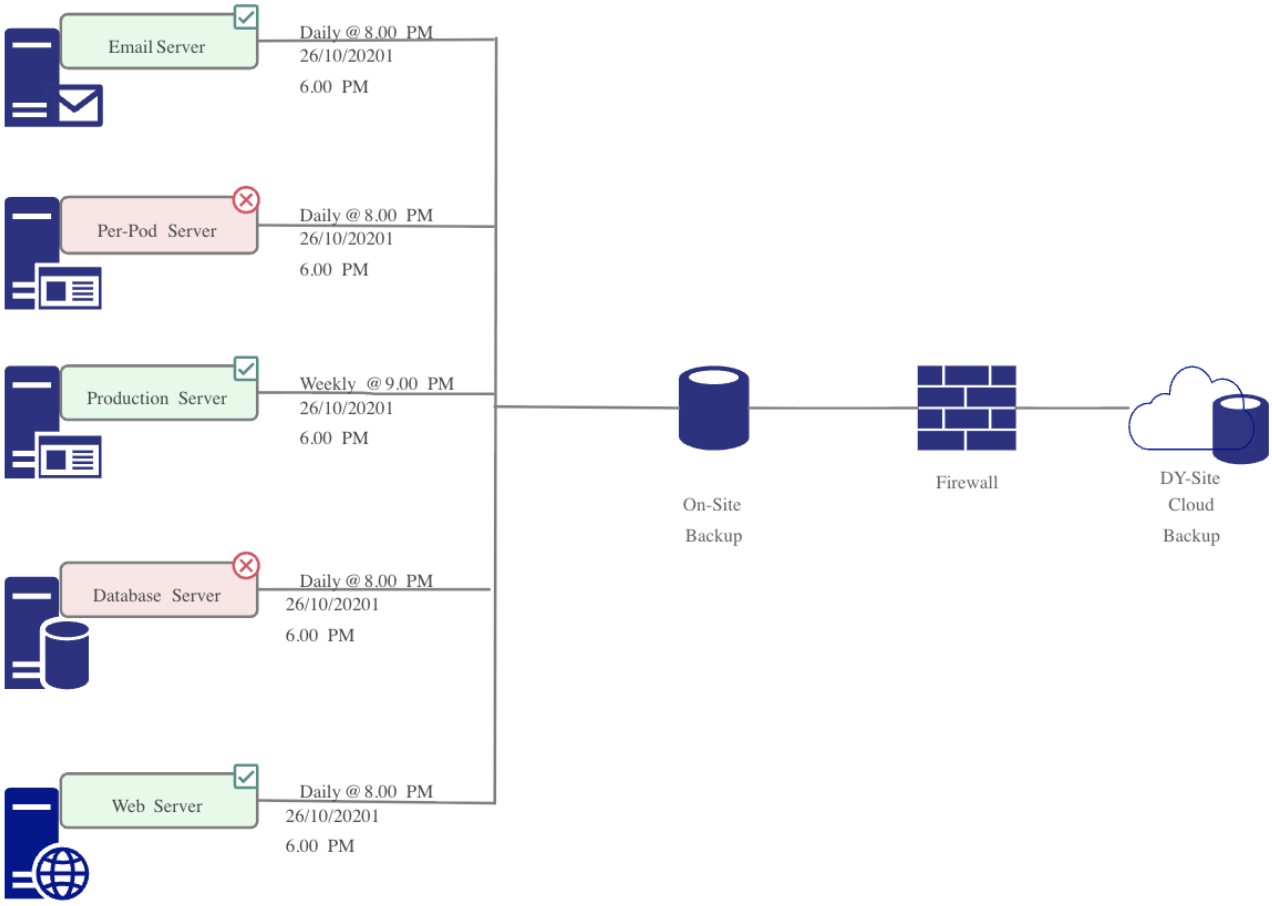

Рисунок 1.3 – Приклад реалізації політики резервного копіювання у хмарному середовищі

Отже, теоретичні засади резервного копіювання даних у хмарних середовищах базуються на поєднанні класичних принципів захисту інформації з можливостями хмарних обчислень. Основу предметної області становлять поняття резервної копії, політики зберігання, методів резервування, процедур відновлення, показників RPO та RTO, а також засобів забезпечення конфіденційності, цілісності й доступності даних. Хмарний підхід до резервного копіювання забезпечує масштабованість, гнучкість, географічну віддаленість копій та високий рівень автоматизації, однак водночас висуває підвищені вимоги до захисту інформації, проектування архітектури та вибору моделі зберігання. Саме тому подальший аналіз предметної області доцільно спрямувати на

вивчення сучасних підходів, сервісів і технологій побудови систем резервного копіювання у хмарних середовищах.

1.2 Сучасні підходи та технології побудови систем резервного копіювання у хмарних середовищах

Сучасний етап розвитку систем резервного копіювання у хмарних середовищах характеризується переходом від ізольованих локальних рішень до централізованих сервісно-орієнтованих платформ, які забезпечують автоматизацію політик захисту даних, масштабованість сховищ, підтримку різнорідних джерел інформації та інтеграцію із засобами моніторингу й інформаційної безпеки. Якщо традиційні підходи були зорієнтовані переважно на копіювання файлів або окремих серверів, то сучасні хмарні рішення охоплюють віртуальні машини, контейнери, бази даних, об’єктні сховища, прикладні сервіси та гібридні інфраструктури [3, 6, 7].

Одним із ключових сучасних підходів є Backup as a Service (BaaS), тобто резервне копіювання як сервіс. Суть цього підходу полягає в тому, що функції створення, зберігання, контролю життєвого циклу та відновлення резервних копій надаються користувачеві як готовий хмарний сервіс. Така модель дозволяє зменшити потребу у власній резервній інфраструктурі, спростити адміністрування, централізувати політики захисту даних і скоротити час розгортання рішення. AWS визначає AWS Backup як повністю керований сервіс, що централізує та автоматизує захист даних для сервісів AWS і гібридних навантажень, а Microsoft характеризує Azure Backup як просте, безпечне та економічно ефективне рішення для створення і відновлення копій з хмари Microsoft Azure [6, 8]. Отже, модель BaaS є одним з основних напрямів розвитку сучасних систем резервного копіювання.

Другим важливим напрямом є побудова політико-орієнтованих систем резервування, де центральним елементом виступає не окрема операція копіювання, а набір правил, за якими система автоматично виконує

резервування, зберігає версії, видаляє застарілі копії та формує точки відновлення. Такий підхід дозволяє уніфікувати захист даних для великої кількості ресурсів і мінімізує залежність від ручного адміністрування. У документації AWS Backup прямо зазначено, що сервіс дає змогу налаштовувати політики резервування та контролювати активність для ресурсів з єдиного місця, а Azure Backup реалізує централізоване керування резервними копіями через відповідні сховища та служби відновлення [7, 8]. Перевагою цього підходу є його придатність до використання в корпоративному середовищі, де захист даних має бути стандартизованим і масштабованим.

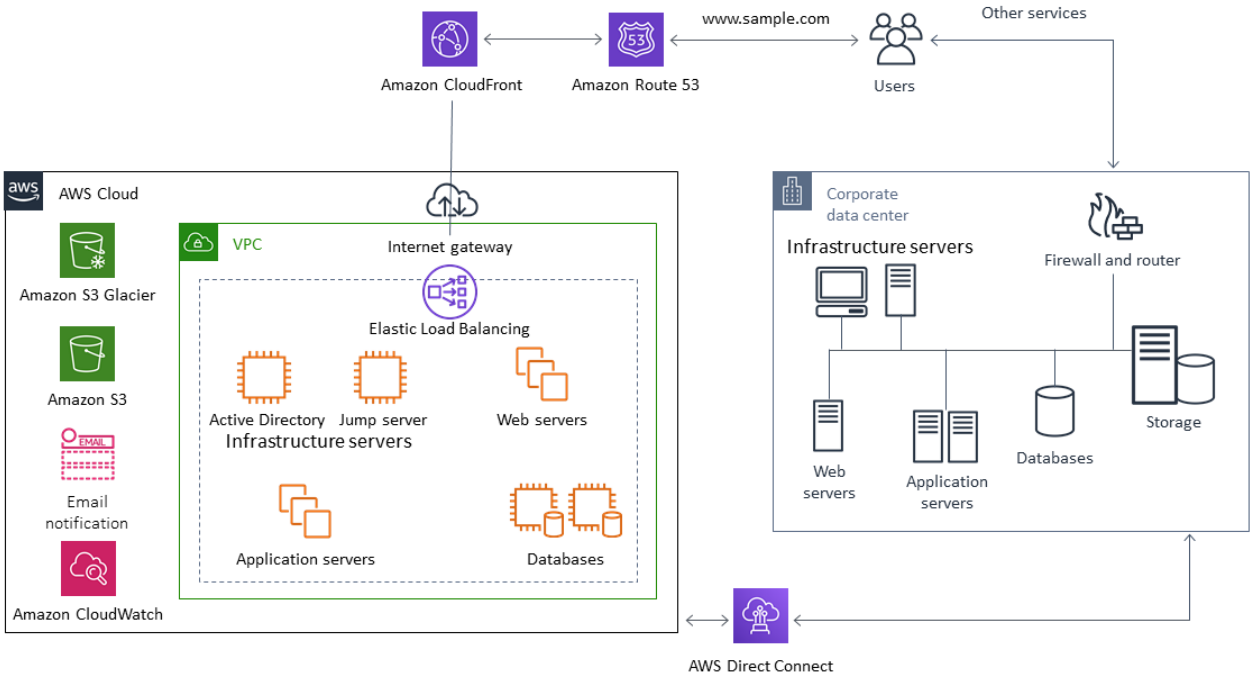

Рисунок 1.4 – Приклад організації резервного копіювання та відновлення для гібридної архітектури.

Поширення гібридних та мультихмарних інфраструктур зумовило формування ще одного важливого підходу, а саме гібридного резервного копіювання. У такій моделі частина даних або резервних копій зберігається в локальній інфраструктурі, а інша частина у публічній чи приватній хмарі. Гібридний підхід доцільний у випадках, коли організація прагне поєднати

швидке локальне відновлення критичних даних із перевагами довгострокового віддаленого зберігання. AWS у керівництві з резервного копіювання для гібридних архітектур розглядає сценарії, у яких застосунки, сервери, каталоги, бази даних і допоміжні сервіси розміщуються як локально, так і в хмарі, а система захисту даних охоплює обидва сегменти інфраструктури [9]. Таким чином, гібридне резервне копіювання можна вважати одним із найпрактичніших сучасних підходів для організацій, що здійснюють поступову міграцію до хмарних сервісів.

Важливе місце серед сучасних технологій посідає централізоване керування резервними копіями для різних типів ресурсів. На практиці це означає, що одна платформа резервування повинна підтримувати копіювання не лише файлових систем, а й віртуальних машин, баз даних, хмарних сервісів, контейнеризованих застосунків і середовищ віртуалізації. Зокрема, Google Cloud Backup and DR позиціонується як централізовано керований сервіс резервного копіювання та відновлення, який забезпечує захист віртуальних машин Compute Engine, середовищ VMware, баз даних і файлових систем, а також підтримує централізоване адміністрування життєвого циклу копій [10, 11]. Такий підхід є особливо цінним у великих організаціях, де використовується неоднорідна IT-інфраструктура і потрібен єдиний механізм політик резервування.

Ще однією суттєвою тенденцією є зміщення акценту з простого копіювання даних на забезпечення кіберстійкості резервної інфраструктури. У сучасних умовах резервні копії розглядаються не лише як технічний інструмент відновлення після збоїв, а як критичний компонент захисту від програм-вимагачів, навмисного видалення інформації та компрометації адміністраторських облікових записів. Саме тому провайдери хмарних платформ дедалі більше інтегрують у сервіси резервного копіювання механізми незмінюваності копій, логічної ізоляції резервних сховищ, захисту від випадкового або зловмисного видалення та додаткового підтвердження критичних адміністративних операцій. Azure Backup серед захисних механізмів зазначає immutable vaults, soft delete та multi-user authorization, а Google Cloud

Backup and DR акцентує увагу на захисті резервних даних від випадкового або шкідливого видалення [10, 12]. Це свідчить про те, що сучасна система резервного копіювання має проектуватися вже не лише як засіб архівації, а як елемент комплексної системи безпеки.

Суттєвим технологічним напрямом є також використання незмінюваних та інденілібельних резервних сховищ, тобто таких, у яких резервні копії неможливо змінити або видалити до завершення встановленого терміну зберігання. Такий підхід значно підвищує стійкість резервної інфраструктури до атак на адміністративний рівень. У документації Google Cloud Backup and DR окремо описано backup vault як компонент для immutable and indelible backups, що призначений для захисту резервних копій від модифікації чи знищення [13]. Використання подібних механізмів особливо важливе для систем, у яких резервні копії мають виконувати не лише функцію відновлення, а й функцію підтвердження цілісності історичних версій даних.

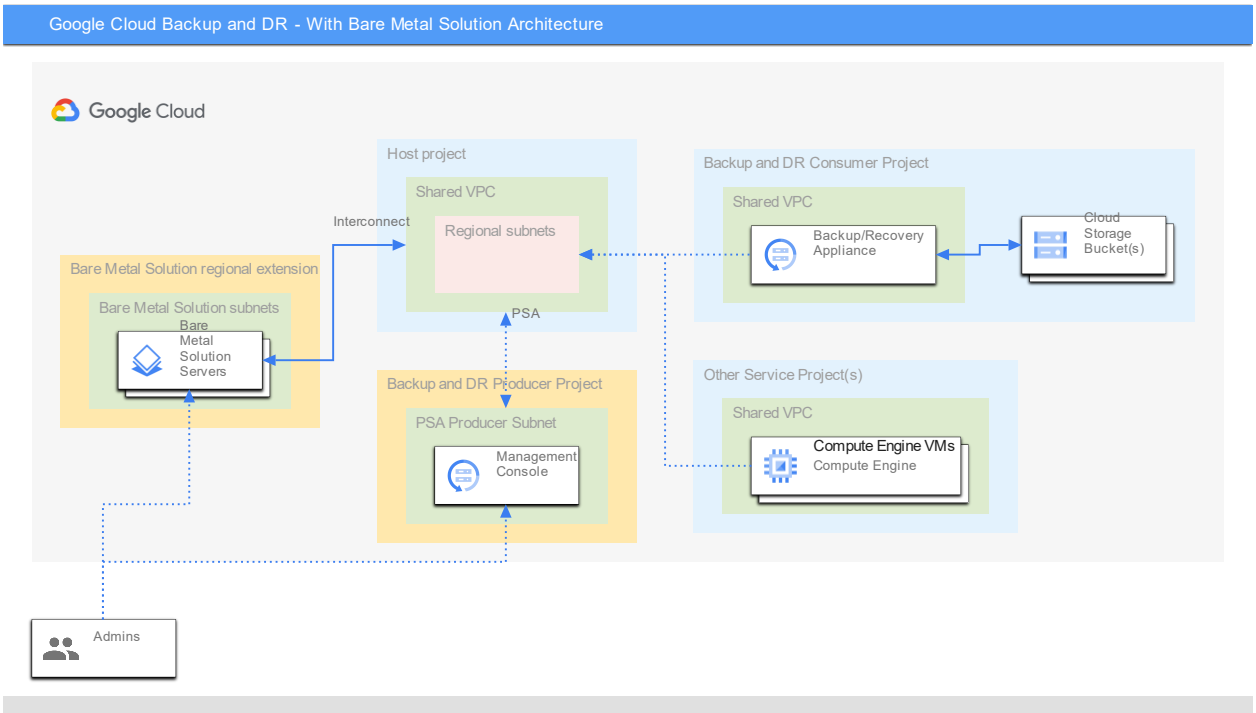

Рисунок 1.5 – Приклад централізованого сервісу Backup and DR у Google Cloud

Окремим напрямом розвитку є орієнтація на життєвий цикл резервних даних. Сучасні хмарні сервіси резервного копіювання не обмежуються створенням копій, а дозволяють керувати переходом даних між різними класами зберігання, визначати строки утримання копій, автоматизувати архівацію і зменшувати витрати на довгострокове зберігання. У матеріалах AWS Prescriptive Guidance наголошується на необхідності врахування не тільки процесу створення копій, а й повного циклу backup-and-restore для виконання вимог RTO, RPO та compliance [2, 7].

Не менш важливою характеристикою сучасних систем є підтримка оркестрації та інтеграції з іншими хмарними сервісами. AWS Backup функціонує як orchestration layer, який інтегрується з Amazon CloudWatch, AWS CloudTrail, AWS Identity and Access Management, AWS Organizations та іншими сервісами [7]. Така інтеграція дозволяє поєднати резервне копіювання з журналюванням подій, централізованим контролем доступу, моніторингом, сповіщеннями та політиками організаційного рівня. У результаті резервне копіювання перестає бути ізольованою функцією і стає складовою системи управління хмарною інфраструктурою.

З позиції архітектури сучасні рішення можна поділити на хмарно-нативні, гібридні та платформно-незалежні. Хмарно-нативні сервіси створюються провайдером конкретної хмари і максимально інтегровані з її екосистемою, що забезпечує високу продуктивність, простоту адміністрування та підтримку вбудованих засобів безпеки. Гібридні рішення орієнтовані на одночасний захист локальних і хмарних ресурсів. Платформно-незалежні підходи застосовуються тоді, коли організація прагне уніфікувати політики резервування для різних середовищ або уникнути критичної залежності від одного провайдера [6, 8, 10].

Отже, сучасні підходи до побудови систем резервного копіювання у хмарних середовищах базуються на сервісній моделі надання функцій резервування, централізованому та політико-орієнтованому керуванні, підтримці гібридних інфраструктур, інтеграції з механізмами інформаційної безпеки та орієнтації на повний життєвий цикл резервних даних [6, 7, 8, 10].

1.3 Вимоги, проблеми та критерії проєктування систем резервного копіювання даних у хмарних середовищах

Проєктування системи резервного копіювання даних у хмарному середовищі є складним багатокомпонентним завданням, оскільки така система повинна одночасно забезпечувати надійне збереження інформації, її безпечне передавання, контрольований доступ до резервних копій, масштабованість, прийнятні витрати на експлуатацію та можливість швидкого відновлення після збою. На відміну від локальних рішень, хмарна система резервування функціонує в умовах розподіленої інфраструктури, залежності від мережевих каналів зв’язку, тарифних моделей провайдера та постійної зміни обсягів даних [3, 7, 14].

Однією з базових вимог до систем резервного копіювання є надійність збереження даних. Вона передбачає, що резервна копія повинна бути створена коректно, збережена без пошкоджень, доступна для вибіркового або повного відновлення та захищена від випадкового чи навмисного знищення. У рекомендаціях NIST щодо безпеки інфраструктури зберігання підкреслюється, що зі зростанням складності сучасних систем зберігання підвищується імовірність конфігураційних помилок, які створюють додаткові ризики для безпеки та доступності інформації [14].

Не менш важливою вимогою є забезпечення повного циклу відновлення, а не лише створення копій. Google Cloud у керівництві з планування disaster recovery наголошує на необхідності проєктувати повний recovery process. Це положення має принципове значення, оскільки на практиці система резервного копіювання може успішно створювати резервні копії, однак виявитися непридатною для оперативного повернення даних у робоче середовище [15]. Отже, під час проєктування необхідно передбачити сценарії часткового й повного відновлення, відновлення окремих файлів, відновлення після логічних помилок, збоїв сховища, компрометації облікових даних або атаки програм-вимагачів [7, 10, 15].

Однією з головних проблем хмарного резервного копіювання є захист резервних даних від кіберзагроз. У сучасних умовах зловмисники часто намагаються не лише зашифрувати або знищити робочі дані, а й вивести з ладу резервні копії, щоб унеможливити відновлення системи. Microsoft у документації Azure Backup описує механізми захисту, зокрема soft delete і secure-by-default підхід, а також засоби додаткового контролю доступу до критичних операцій [12, 16]. Відповідно, однією з ключових вимог до проєктованої системи є наявність механізмів захисту резервних даних на рівні зберігання, політик доступу та адміністративного контролю [12, 16].

Ще однією важливою вимогою є автоматизація резервних операцій. У великих або динамічних хмарних інфраструктурах ручне керування процесами резервування є неефективним, оскільки збільшує ризик помилок і ускладнює дотримання єдиних політик захисту даних. AWS Prescriptive Guidance серед базових кроків визначає необхідність автоматизації резервних операцій, впровадження моніторингу, аудиту конфігурацій та регулярного тестування можливостей відновлення [17, 18].

Для систем резервного копіювання у хмарі суттєвою вимогою є також масштабованість. Обсяги даних, що підлягають резервуванню, можуть швидко змінюватися через розширення інформаційних систем, збільшення кількості користувачів, накопичення журналів, мультимедійного контенту або резервування нових сервісів. Google Cloud Backup and DR акцентує, що сервіс дозволяє керувати копіями протягом їх життєвого циклу та використовувати їх для disaster recovery, business continuity і тестово-розробницьких завдань [10]. Отже, масштабованість повинна розглядатися не лише як можливість додавати новий простір, а як здатність системи підтримувати керованість і продуктивність зі зростанням обсягу резервних даних [10, 15].

Важливою групою вимог є вимоги до безпеки доступу та шифрування. Оскільки резервні копії нерідко містять конфіденційну або критично важливу інформацію, їх захист повинен включати автентифікацію, авторизацію, журналювання доступу та криптографічний захист. AWS серед рекомендацій із

безпеки вказує на необхідність реалізації контролю доступу, шифрування резервних даних і сховищ, захисту копій за допомогою immutable storage, моніторингу, аудиту та перевірки сценаріїв відновлення [17].

Окремою проблемою під час побудови систем резервного копіювання є обмеження мережевої інфраструктури та часові витрати на передавання даних. При великому обсязі інформації резервне копіювання у хмару може створювати суттєве навантаження на мережу, особливо якщо йдеться про повні копії, часті резервні операції або обмежену пропускну здатність каналу. Саме тому вибір методу копіювання, частоти резервування та політики дедуплікації має безпосередній вплив на експлуатаційні характеристики системи. У цьому контексті особливо важливо враховувати показники RPO та RTO [2, 6, 15].

Помітною проблемою є також економічна ефективність системи резервного копіювання. Хоча хмарні сервіси дозволяють уникнути значних початкових витрат на придбання власної інфраструктури, їх використання пов’язане з постійними експлуатаційними витратами, які залежать від обсягу збережених копій, частоти резервування, кількості операцій запису та читання, строків утримання, мережевого трафіку і додаткових сервісів безпеки. Тому під час проєктування системи необхідно визначати компроміс між глибиною історії резервних копій, швидкістю відновлення та вартістю зберігання [3, 7, 10].

Ще однією вимогою є регулярне тестування відновлення. Система резервного копіювання не може вважатися ефективною лише на основі успішного завершення резервних завдань. Її працездатність повинна підтверджуватися тестами відновлення, які перевіряють цілісність копій, повноту даних, коректність процедур повернення сервісу в робочий стан та відповідність очікуваним параметрам RTO. AWS окремо зазначає необхідність тестування recovery capabilities, а Google Cloud під час планування disaster recovery також наголошує на проєктуванні end-to-end recovery process [15, 17].

Узагальнюючи вимоги до системи резервного копіювання у хмарному середовищі, доцільно виділити такі ключові критерії її проєктування: надійність зберігання, безпека доступу, захист від видалення або модифікації копій,

автоматизація резервних операцій, масштабованість, економічна ефективність, керованість життєвим циклом резервних даних, підтримка різних сценаріїв відновлення та відповідність цільовим показникам RPO і RTO [7, 10, 12, 17].

Отже, аналіз предметної області показує, що проєктування систем резервного копіювання даних у хмарних середовищах повинно спиратися на комплексний підхід, який охоплює технічні, безпекові, організаційні та економічні аспекти. Основними проблемами залишаються захист резервних копій від сучасних кіберзагроз, забезпечення повноцінного й швидкого відновлення, оптимізація витрат на зберігання та збереження керованості системи в умовах масштабування. Усе це формує підґрунтя для подальшого обґрунтування функціональних і нефункціональних вимог до розроблюваної системи резервного копіювання [10, 12, 15, 17].

РОЗДІЛ 2.
ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ СИСТЕМИ РЕЗЕРВНОГО КОПІЮВАННЯ ДАНИХ У ХМАРНИХ СЕРЕДОВИЩАХ.

2.1 Обґрунтування вибору напряму дослідження та стислий аналіз об’єкта дослідження

У результаті аналізу предметної області, виконаного в попередньому розділі, встановлено, що сучасні системи резервного копіювання мають забезпечувати не лише створення копій даних, а й їх безпечне зберігання, керування життєвим циклом резервних наборів, масштабованість, зручність адміністрування та можливість оперативного відновлення після збоїв. З урахуванням цих вимог як основний напрям дослідження в роботі обрано розроблення системи резервного копіювання даних із використанням хмарного середовища Google Cloud, оскільки ця платформа надає керовані сервіси зберігання, інструменти автоматизації, логування, безпечного зберігання облікових даних і можливості реалізації резервування для різних типів навантажень. Google Cloud Storage є керованим сервісом об’єктного зберігання, у якому дані зберігаються у вигляді об’єктів у бакетах, а Backup and DR Service орієнтований на захист копій даних протягом їх життєвого циклу та використання для disaster recovery і business continuity.

Об’єктом дослідження в межах другого розділу виступає процес організації резервного копіювання даних у хмарному середовищі Google Cloud. Практично це означає дослідження взаємодії між локальним або прикладним джерелом даних, програмним модулем резервування, сервісами хмарного зберігання, плануванням запуску резервних процедур, журналюванням результатів та механізмами відновлення. На відміну від загального теоретичного аналізу, у цьому розділі об’єкт дослідження розглядається вже як основа для створення конкретної програмної системи, яка повинна реалізувати збереження файлів у Google Cloud, підтримку керування версіями копій, контроль помилок і

можливість автоматизованого запуску резервних операцій. Вибір саме Google Cloud обґрунтовується тим, що ця платформа поєднує об’єктне сховище Cloud Storage, serverless-середовище Cloud Run, сервіс планування Cloud Scheduler, засоби зберігання секретів Secret Manager і централізоване логування через Cloud Logging.

Cloud Storage доцільно розглядати як базовий компонент майбутньої системи, тому що цей сервіс призначений для зберігання неструктурованих даних будь-якого обсягу, а об’єкти в ньому є незмінними одиницями даних, розміщеними в бакетах. Така модель добре узгоджується із задачею резервного копіювання, де кожна резервна копія або архівний файл може розглядатися як окремий об’єкт, для якого задаються атрибути зберігання, клас сховища, правила життєвого циклу та політики доступу. Додатково Cloud Storage підтримує різні класи зберігання, а Object Lifecycle Management дозволяє автоматично керувати строками зберігання, переходом об’єктів до дешевших класів або видаленням застарілих копій. Це є важливим для систем резервного копіювання, де потрібно поєднати технічну надійність і економічну доцільність.

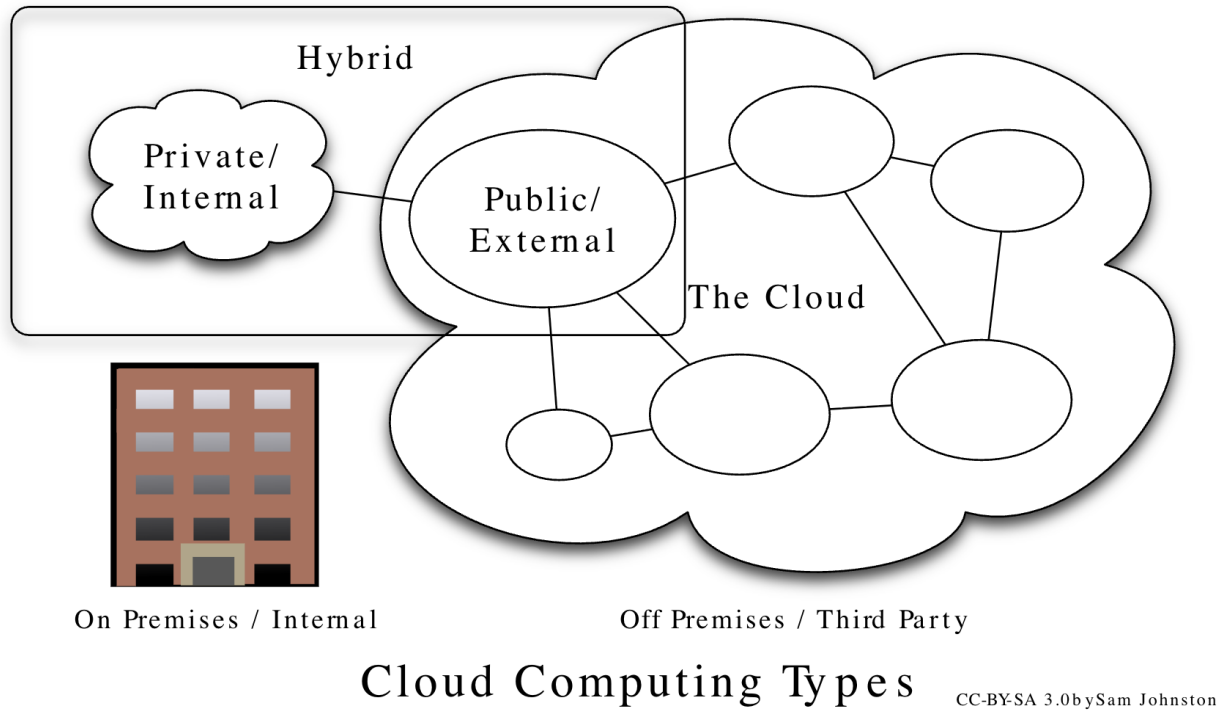

Рисунок 2.1 – Схема типів хмарних обчислень: public, hybrid, private

Для цієї роботи найбільш доцільною є модель використання публічної хмари Google Cloud із можливістю інтеграції з локальним джерелом даних, тобто фактично гібридний сценарій взаємодії. Такий підхід дозволяє зберегти початкові дані в локальному середовищі або на користувацькому пристрої, а резервні копії розміщувати в Google Cloud Storage. Практична цінність цього підходу полягає в тому, що навіть при обмеженій локальній інфраструктурі можна отримати віддалене резервне сховище, механізми керування доступом, журналювання дій та автоматизацію запуску резервних завдань. Водночас гібридна модель є наближеною до реальних умов експлуатації, коли копіювання відбувається з локальної системи в хмару, а відновлення може виконуватися як у вихідне середовище, так і в іншу точку розгортання.

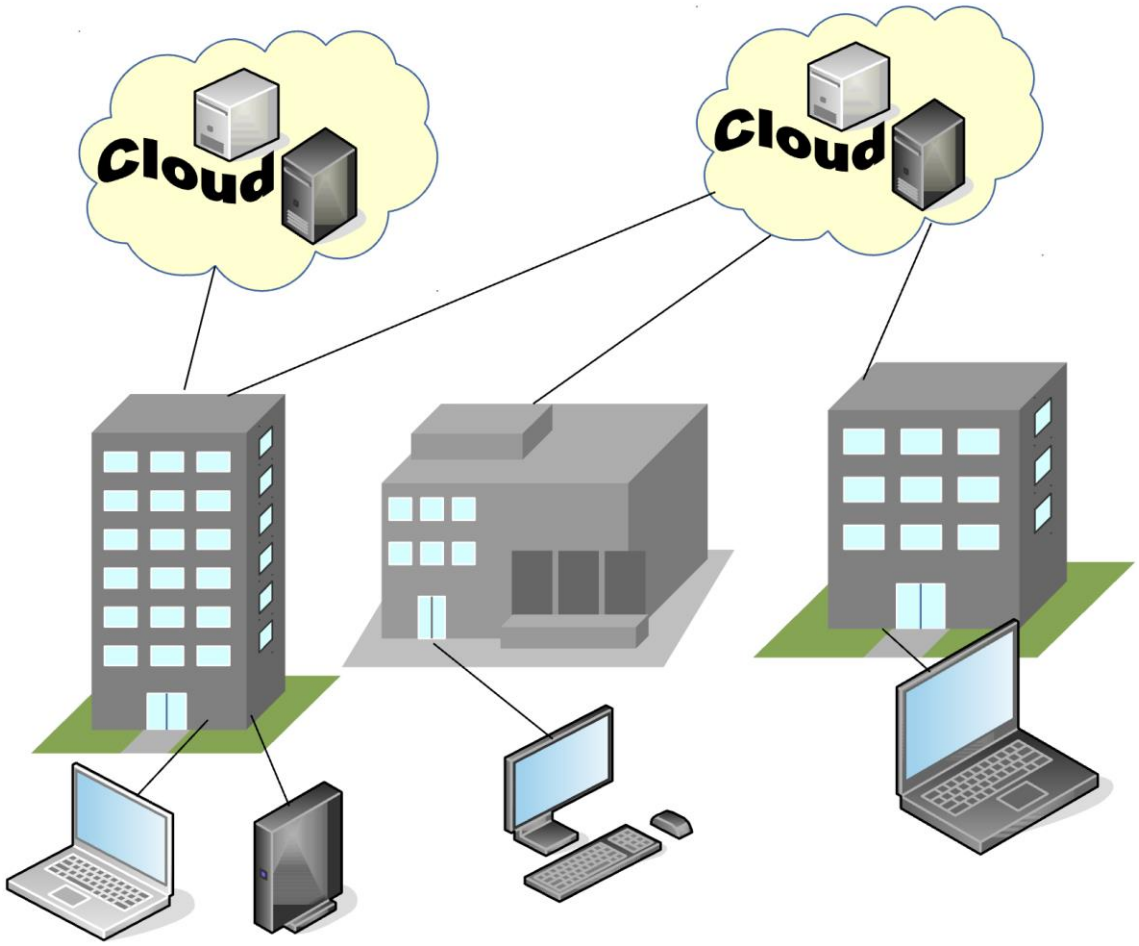

Рисунок 2.2 – Схема гібридної хмари

У межах роботи пропонується орієнтуватися не на побудову повноцінної корпоративної Backup and DR платформи, а на розроблення прикладної системи резервного копіювання, яка використовує можливості Google Cloud як інфраструктурної основи. Це означає, що об’єкт дослідження має прикладний характер: він охоплює механізми підготовки файлів до копіювання, передавання даних у хмарне сховище, збереження метаданих, логування операцій, керування періодичністю запуску та реалізацію процедури відновлення. Саме такий підхід є доцільним для бакалаврської роботи, оскільки дозволяє поєднати теоретичні положення з практичним програмним рішенням, а також чітко визначити межі системи, її функції та критерії ефективності.

З урахуванням зазначеного постановка задачі полягає у створенні програмної системи резервного копіювання даних, яка забезпечує вибір файлів або каталогів для копіювання, передавання резервних копій у хмарне середовище Google Cloud, зберігання резервних об’єктів у Cloud Storage, автоматичний запуск копіювання за розкладом, безпечне зберігання параметрів доступу, ведення журналу виконаних операцій та реалізацію процедури відновлення даних із хмарного сховища. При цьому система повинна бути придатною до розгортання в сучасному хмарному середовищі, масштабованою за обсягом даних і зрозумілою з точки зору адміністрування.

2.2 Методи вирішення поставленої задачі та загальна методика проведення власних досліджень

Розв’язання поставленої задачі доцільно виконувати на основі поєднання методів системного аналізу, порівняльного аналізу хмарних сервісів, структурного проєктування програмної системи та експериментальної перевірки її працездатності. Метод системного аналізу використовується для визначення складу компонентів майбутньої системи, їх взаємодії та меж відповідальності. Порівняльний аналіз застосовується для вибору сервісів Google Cloud, які найкраще відповідають функціям резервного копіювання, планування,

безпечного керування параметрами доступу та журналювання. Структурне проєктування дає змогу побудувати логічну і фізичну архітектуру рішення, а експериментальна перевірка потрібна для оцінювання правильності завантаження резервних копій, їх відновлення та відповідності системи поставленим функціональним вимогам. Вибір саме таких методів зумовлений тим, що Google Cloud надає окремі сервіси для зберігання, запуску контейнеризованих застосунків, планування періодичних завдань, керування секретами та централізованого логування.

Загальна методика проведення власних досліджень у роботі передбачає кілька послідовних етапів. На першому етапі виконується аналіз функціональних вимог до системи резервного копіювання, серед яких визначаються сценарії створення копій, збереження метаданих, відновлення файлів, облік версій резервних об’єктів, вимоги до журналювання та захисту даних. На другому етапі обираються хмарні сервіси Google Cloud, які доцільно використати в розробці. На третьому етапі виконується логічне проєктування системи, під час якого визначаються програмні модулі, точки інтеграції з хмарними API та формати взаємодії між компонентами. На четвертому етапі реалізується програмний прототип системи. На п’ятому етапі проводиться тестування сценаріїв завантаження резервної копії, повторного запуску, логування, помилкового доступу та відновлення даних. Така методика узгоджується з рекомендаціями Google Cloud щодо проєктування DR-процесів, де підкреслюється потреба визначення recovery objectives, побудови recovery process і перевірки відновлення від втрати даних.

У межах розроблюваної системи доцільно використати таку загальну архітектурну модель. Джерелом резервування є локальні файли або каталоги користувача. Програмний модуль резервування виконує сканування джерела, формує архів або набір файлів для відправлення, додає службові метадані, після чого передає об’єкти до бакета Cloud Storage. Плановий запуск резервування організовується через Cloud Scheduler, який у Google Cloud є fully managed cron service для запуску робіт за розкладом. Логіка виконання резервної операції може

бути розміщена в Cloud Run, який є fully managed application platform для запуску коду або контейнера на масштабованій інфраструктурі Google. Секрети доступу та конфігураційні параметри безпеки доцільно розміщувати в Secret Manager, а журнали подій записувати в Cloud Logging. Така архітектура дозволяє чітко розподілити функції між сервісами та побудувати систему з високим рівнем автоматизації.

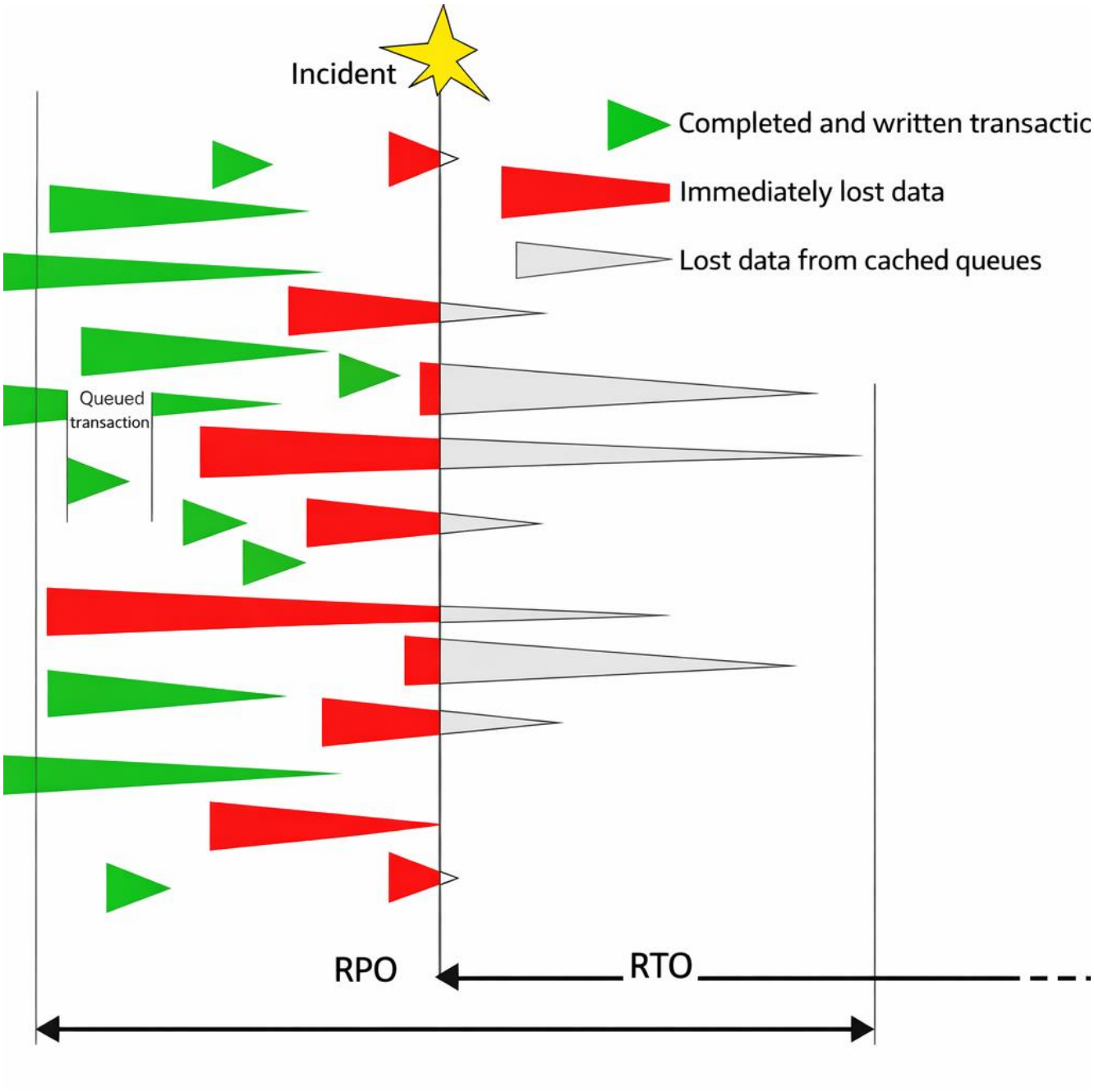

Рисунок 2.3 – Схематичне представлення показників RPO та RTO

Під час формування методики дослідження важливе значення мають показники RPO та RTO, оскільки саме вони визначають цільові характеристики системи резервного копіювання. Для бакалаврської розробки доцільно прийняти прикладний підхід, за якого RPO визначається інтервалом між плановими запусками резервування, а RTO — часом, необхідним для вибору резервної копії, її завантаження з Cloud Storage та відновлення файлів у локальному середовищі. Google Cloud у рекомендаціях щодо тестування відновлення від втрати даних радить чітко визначати RPO, відстежувати стан резервних копій і перевіряти recovery process. Це означає, що в роботі потрібно оцінювати не лише факт створення копії, а й можливість її відновлення в прийнятний проміжок часу.

Вибір Cloud Storage як основного резервного сховища обґрунтовується також підтримкою політик життєвого циклу об’єктів. Object Lifecycle Management у Cloud Storage дає змогу автоматично виконувати дії над об’єктами за встановленими правилами, зокрема реалізовувати Time to Live, зберігати неактуальні версії або переводити об’єкти до дешевших класів зберігання. Для системи резервного копіювання це дозволяє автоматизувати видалення застарілих копій і керувати витратами на сховище без ручного втручання адміністратора. Додатково класи зберігання Cloud Storage можуть бути використані для розмежування “гарячих” резервних копій, що часто відновлюються, та довготривалого архівного зберігання.

Методика досліджень в межах цієї роботи повинна також включати перевірку безпеки доступу до резервних даних. Для цього в системі доцільно передбачити використання Secret Manager, який призначений для зберігання API keys, passwords, certificates та інших sensitive data, а також Cloud Audit Logs і Cloud Logging для фіксації адміністративних та прикладних подій. Поєднання цих сервісів дозволяє побудувати базовий контур контролю доступу, аудит дій і діагностику помилок. У контексті бакалаврської роботи це важливо, оскільки резервне копіювання розглядається не лише як технічна операція перенесення файлів у хмару, а як процес, що має забезпечувати цілісність, контрольованість і простежуваність усіх значущих операцій.

Отже, загальна методика проведення досліджень полягає у поетапному проєктуванні й реалізації системи резервного копіювання на базі Google Cloud із подальшою експериментальною перевіркою її працездатності. Методично це включає визначення вимог, вибір хмарних сервісів, побудову архітектури, реалізацію програмного рішення, налаштування планового запуску, організацію безпечного зберігання конфігураційних даних і проведення тестів на завантаження, зберігання та відновлення резервних копій.

2.3 Обґрунтування вибору мови програмування та засобів реалізації

Важливою складовою постановки задачі є вибір мови програмування, яка буде використана для реалізації системи резервного копіювання. З огляду на те, що основою розробки обрано Google Cloud, доцільно використовувати таку мову, для якої існує повноцінна підтримка клієнтських бібліотек Google Cloud, зручні засоби роботи з файлами, мережевими операціями, архівацією, обробкою винятків і логуванням. Найбільш обґрунтованим вибором для цієї роботи є Python, оскільки Google прямо зазначає, що Cloud Client Libraries є recommended way to access Google Cloud APIs programmatically, а бібліотеки для Python надають високорівневі абстракції, зменшують обсяг шаблонного коду і добре інтегруються зі стандартною бібліотекою мови. Для Cloud Storage, Secret Manager, Logging та інших сервісів також доступні окремі Python client libraries.

На користь Python свідчить і те, що ця мова добре підходить для реалізації утиліт і серверних сервісів, пов’язаних з автоматизацією резервного копіювання. Стандартна бібліотека Python містить засоби для роботи з файловою системою, архівами, датою і часом, винятками, хешуванням, журналюванням та мережевими запитами. У поєднанні з офіційними Google Cloud client libraries це дозволяє реалізувати завантаження об’єктів у Cloud Storage, взаємодію з Secret Manager, передавання журналів до Cloud Logging і побудову прикладного резервного сервісу без потреби в низькорівневому опрацюванні REST-запитів.

Крім того, Google Cloud Storage client library для Python підтримується для актуальних версій Python, що додатково підтверджує практичність цього вибору.

Окремо слід врахувати, що Cloud Run підтримує запуск контейнеризованих застосунків, написаних різними мовами, проте для бакалаврської роботи доцільно обрати таку мову, яка дає змогу швидко створити працездатний прототип, зберігши при цьому достатню гнучкість для подальшого розширення. Python відповідає цій вимозі, тому що поєднує порівняно простий синтаксис, швидкість розроблення, наявність великої кількості бібліотек і офіційну підтримку інструментів Google Cloud. Це робить Python доцільним не лише для локального скрипта резервування, а й для контейнеризованого сервісу, що розгортається в Cloud Run і запускається за розкладом через Cloud Scheduler.

У межах цієї роботи результати вибору мови програмування доцільно сформулювати так: для реалізації системи резервного копіювання використовується Python, оскільки ця мова забезпечує зручну інтеграцію з Google Cloud APIs через офіційні client libraries, дозволяє швидко реалізувати прикладну логіку резервування, підтримує роботу з файлами, архівацією та журналюванням, а також є придатною до розгортання в Cloud Run. Як основне середовище хмарного розгортання обирається Google Cloud, де Cloud Storage використовується як резервне сховище, а Cloud Run як середовище виконання сервісу, Cloud Scheduler як інструмент планування запусків, Secret Manager, як засіб захисту конфігураційних параметрів і ключів доступу, а Cloud Logging, як засіб централізованого журналювання.

Таким чином, у другому розділі обґрунтовано вибір напряму досліджень, що полягає у створенні системи резервного копіювання даних на базі сервісів Google Cloud, визначено об’єкт дослідження, описано методи вирішення поставленої задачі, розроблено загальну методику проведення власних досліджень і наведено результати вибору мови програмування. Це формує методичну й технологічну основу для переходу до подальшого проєктування структури системи, опису її функціональних компонентів і безпосередньої програмної реалізації.

РОЗДІЛ 3
ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ РЕЗЕРВНОГО КОПІЮВАННЯ ДАНИХ У ХМАРНИХ СЕРЕДОВИЩАХ

3.1 Загальна структура та логіка роботи системи

Проєктування інформаційної системи резервного копіювання даних виконувалося з урахуванням вимог, сформульованих у попередніх розділах, а саме необхідності забезпечення вибору файлів для копіювання, захищеного передавання даних у хмарне середовище, керування параметрами резервування, автоматичного запуску завдань та надання користувачеві простого графічного інтерфейсу. Основною ідеєю реалізації стало створення настільного програмного застосунку, який об’єднує модулі взаємодії з користувачем, модулі підготовки резервних даних, механізми роботи з токеном доступу, підсистему планування копіювання та компонент передавання резервних копій до хмарного середовища Google.

Система побудована за модульним принципом. Такий підхід дозволяє розділити відповідальність між окремими частинами програми, спростити супровід коду, ізолювати обробку помилок і забезпечити можливість подальшого розширення функціональності. У загальному вигляді програмна система містить модуль головного вікна, модуль резервного копіювання, модуль налаштувань, модуль додаткових параметрів токена, модуль планування запуску, модуль хмарної взаємодії та сервісний модуль журналювання подій. Центральною ланкою між інтерфейсом і серверною логікою є контролер застосунку, який приймає дії користувача, ініціює відповідні операції та оновлює стан вікон.

Функціональна логіка системи передбачає такий сценарій роботи. Після запуску застосунок відкриває головне вікно, у якому відображається стан токена доступу та доступні розділи системи. Користувач переходить до модуля налаштування резервного копіювання, вибирає каталоги або окремі файли, за

потреби задає пароль, указує дату або розклад виконання резервної операції, після чого запускає копіювання. На рівні бізнес-логіки система формує набір вибраних об’єктів, виконує їх архівацію, за потреби шифрує вміст, створює службовий опис резервної копії та передає підготовлений файл у хмарне сховище. Якщо увімкнено автоматичний запуск, система додатково зберігає параметри завдання та ініціює повторне виконання копіювання у вказаний час.

З урахуванням архітектури, прийнятої в роботі, інформаційна система реалізується як клієнтський застосунок на Python із графічним інтерфейсом на базі Qt. Такий вибір дозволяє поєднати зручні засоби створення настільних вікон із доступом до файлової системи, підтримкою багатопотокової обробки та простою інтеграцією з хмарними API. Для взаємодії з інтерфейсом користувача доцільно використати підхід подієво-орієнтованого програмування, де кожна дія користувача, наприклад натискання кнопки або зміна прапорця, викликає відповідний обробник події.

У структурі системи доцільно виділити три основні рівні. Перший рівень становить інтерфейс користувача, який відповідає за відображення вікон, кнопок, полів введення, календаря, списків вибраних файлів і повідомлень про стан виконання операцій. Другий рівень утворює прикладна логіка, що реалізує перевірку введених параметрів, підготовку резервної копії, формування розкладу та виклики сервісів доступу до хмари. Третій рівень становить шар роботи з даними, який охоплює файлову систему, локальні конфігураційні файли, тимчасові архіви, журнал подій і мережеві операції передавання даних у Google Cloud Storage. Така організація забезпечує логічну цілісність системи й дозволяє окремо вдосконалювати її інтерфейсну та функціональну частини.

Нижче наведено фрагмент коду, який демонструє базову структуру головного класу застосунку:

```
class BackupApplication:
    def __init__(self, ui_manager, backup_service, token_service, scheduler_service):
```

```
        self.ui_manager = ui_manager
        self.backup_service = backup_service
        self.token_service = token_service
        self.scheduler_service = scheduler_service

    def start(self):
        token_state = self.token_service.get_status()
        self.ui_manager.show_main_window(token_state)

    def run_backup(self, config):
        if not config.selected_paths:
            raise ValueError("Не вибрано файли для резервного копіювання")
        archive_path = self.backup_service.create_backup_archive(config)
        self.backup_service.upload_to_cloud(archive_path, config)
```

У наведеному фрагменті коду видно, що головний клас не виконує всі операції самостійно, а лише координує роботу інших сервісів. Саме така побудова є доцільною для даного проєкту, оскільки дозволяє підтримувати чисту структуру коду та уникати надмірної залежності між інтерфейсом і хмарною логікою.

3.2 Проєктування та реалізація головного вікна системи

Головне вікно системи є центральною точкою входу до застосунку та забезпечує переходи до основних функціональних розділів. Його зовнішній вигляд подано на рисунку 3.1. У лівій частині вікна розміщено навігаційну панель із кнопками «Інфо-панель», «ML деталі» та «Налаштування», а в правій частині відображається інформаційна область із повідомленням про стан токена та кнопкою переходу до резервного копіювання. Верхня горизонтальна зона

містить індикатор стану токена, який візуально сигналізує, чи може система виконувати хмарні операції.

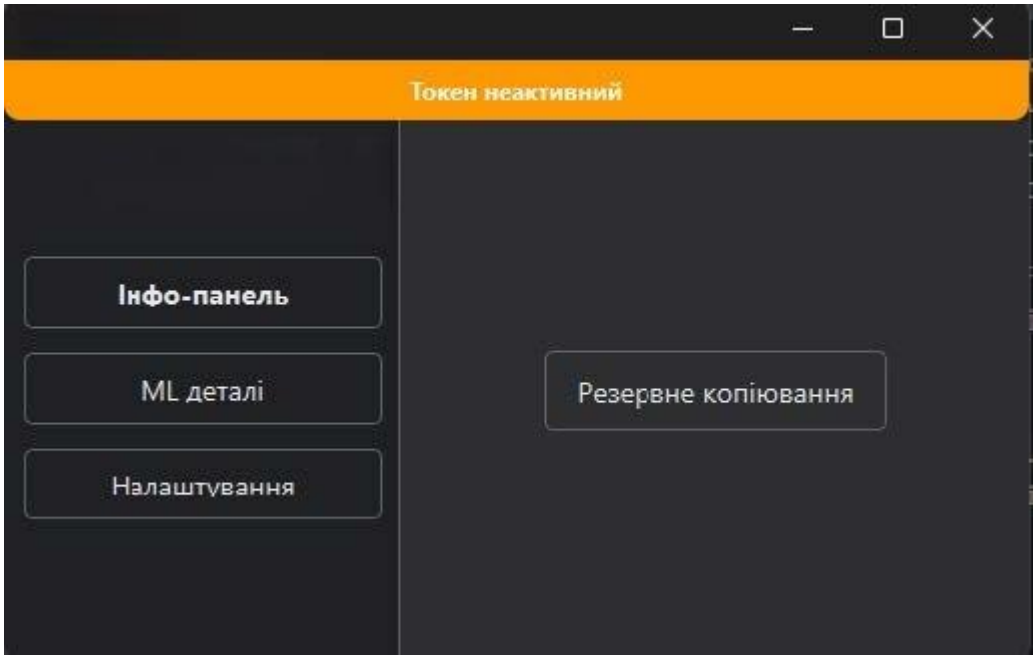

Рисунок 3.1 – Головне вікно системи резервного копіювання

Програмна реалізація головного вікна передбачає створення базового класу, який успадковується від QMainWindow. Усередині цього класу формується контейнер з двох областей: навігаційної панелі та робочої області. Для побудови навігації доцільно використати вертикальний менеджер компонування, у якому послідовно розміщуються кнопки переходу між вкладками. Робоча область формується як окремий контейнер, у якому можуть динамічно відображатися вкладені віджети залежно від вибраного розділу. Код створеного головного вікна додатку подано в додатку.

Функціонально головне вікно виконує кілька завдань. По-перше, воно інформує користувача про поточний стан системи. Якщо токен доступу неактивний, відповідне повідомлення виводиться у верхній частині правої області, що узгоджується з інтерфейсом на рисунку 3.1. По-друге, воно дає можливість швидкого переходу до налаштування резервного копіювання. По-третє, головне вікно виступає контейнером для завантаження інших

функціональних форм без потреби відкривати велику кількість незалежних вікон.

Для забезпечення наочності стану токена доцільно використати окремий метод оновлення індикатора. Його реалізацію наведено нижче.

```
def update_token_status(self, is_active: bool):
    if is_active:
        self.token_label.setText("Токен активний")
        self.token_label.setStyleSheet(
            "background-color: #2f9e44; color: white; padding: 8px;"
        )
    else:
        self.token_label.setText("Токен неактивний")
        self.token_label.setStyleSheet(
            "background-color: #f08c00; color: white; padding: 8px;"
        )
```

У логіці застосунку така функція викликається після перевірки дійсності токена, під час запуску системи, після оновлення токена або після його скидання. Це забезпечує синхронізацію інтерфейсу з поточним станом механізму авторизації.

3.3 Реалізація вікна резервного копіювання

Ключовим елементом системи є вікно налаштування резервного копіювання, подане на рисунку 3.2. Саме через це вікно користувач задає параметри резервної операції: обирає файли, задає пароль, налаштовує дату запуску та ініціює початок копіювання. Інтерфейс має чітку прикладну спрямованість і побудований так, щоб основні етапи формування резервної копії були розміщені послідовно зверху вниз.

У верхній частині цього вікна розміщено компонент «Провідник файлів», який використовується для перегляду доступних дисків і каталогів. Для його реалізації в середовищі Qt доцільно застосувати QTreeView разом із QFileSystemModel, що дозволяє відображати структуру файлової системи у формі дерева. Нижче розміщено область «Вибрані файли», яка містить список шляхів, що будуть включені до резервної копії. Для додавання елементів у цей список використовується кнопка «Додати вибране».

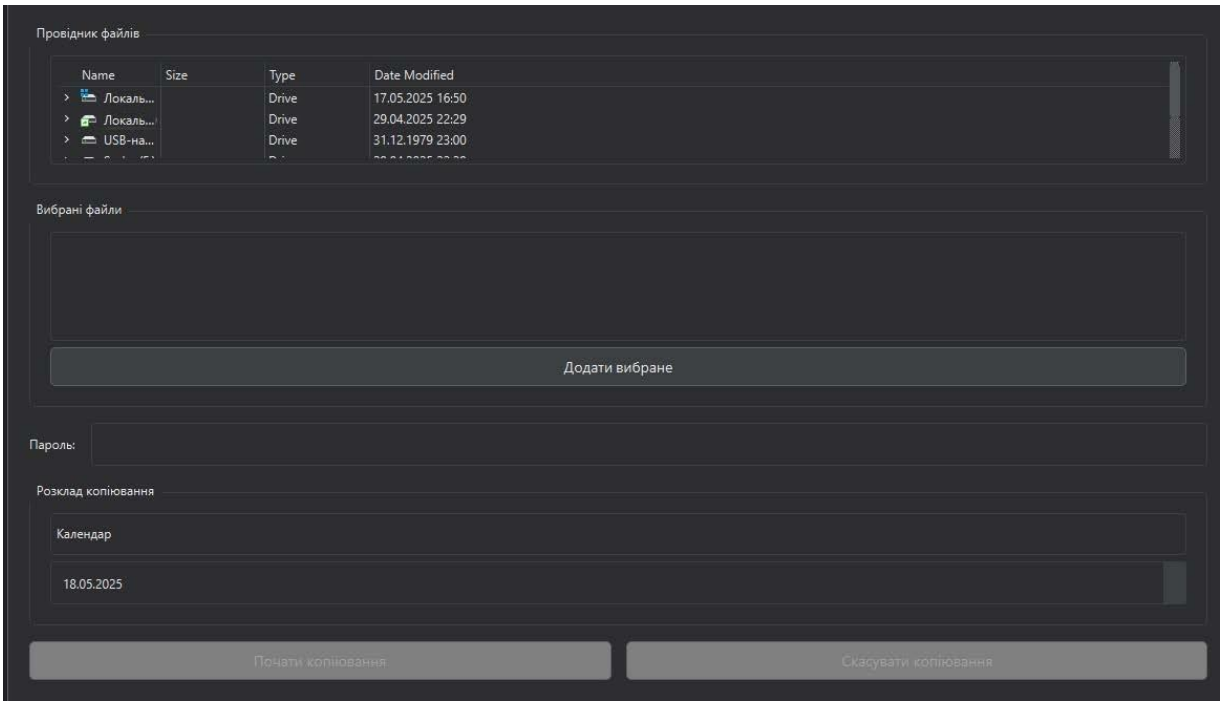

Рисунок 3.2 – Вікно налаштування резервного копіювання

Базова реалізація файлового вибору реалізована таким фрагментом коду:

```
from PyQt6.QtWidgets import QWidget, QVBoxLayout, QPushButton, QListWidget, QTreeView
from PyQt6.QtGui import QFileSystemModel
from PyQt6.QtCore import QDir

class BackupConfigWidget(QWidget):
```

```
def __init__(self):
    super().__init__()

    self.layout = QVBoxLayout(self)

    self.file_model = QFileSystemModel()
    self.file_model.setRootPath(QDir.rootPath())

    self.file_tree = QTreeView()
    self.file_tree.setModel(self.file_model)

    self.selected_files = QListWidget()
    self.add_button = QPushButton("Додати вибране")

    self.layout.addWidget(self.file_tree)
    self.layout.addWidget(self.selected_files)
    self.layout.addWidget(self.add_button)

    self.add_button.clicked.connect(self.add_selected_item)

def add_selected_item(self):
    index = self.file_tree.currentIndex()
    path = self.file_model.filePath(index)
    if path:
        self.selected_files.addItem(path)
```

Наведений фрагмент демонструє базову механіку вибору файлів. Після виділення елемента в дереві каталогів користувач натискає кнопку додавання, а шлях до вибраного файлу або каталогу переноситься до списку резервування. У повній реалізації доцільно додатково перевіряти наявність дублікатів,

забезпечити можливість видалення елементів зі списку та зберігати обрані шляхи у внутрішній конфігурації завдання.

Наступним важливим компонентом вікна є поле «Пароль». Воно призначене для введення символьного ключа, який використовується під час шифрування резервного архіву. З погляду безпеки пароль не повинен зберігатися у відкритому вигляді довше, ніж це потрібно для формування копії. Тому в програмній реалізації доцільно зчитувати пароль із поля, передавати його до сервісу шифрування, а після завершення операції очищати значення в полі або зберігати лише похідний криптографічний ключ.

Далі у вікні розміщено блок «Розклад копіювання», який містить календарний компонент та поле дати. Функціонально цей блок дає змогу користувачеві визначити момент виконання резервної операції. Для реалізації цього фрагмента інтерфейсу доцільно використовувати QCalendarWidget, а також поле введення або мітку для відображення вибраної дати. У випадку розширення системи до повноцінного планувальника можна також додати вибір години, хвилин і режиму повторення.

Реалізація підключення календаря до конфігурації завдання наведена нижче.

```
from PyQt6.QtWidgets import QCalendarWidget, QLineEdit

self.calendar = QCalendarWidget()
self.date_field = QLineEdit()
self.date_field.setReadOnly(True)

self.calendar.selectionChanged.connect(self.update_selected_date)

def update_selected_date(self):
    selected_date = self.calendar.selectedDate()
    self.date_field.setText(selected_date.toString("dd.MM.yyyy"))
```

У нижній частині вікна розміщено дві великі кнопки: «Почати копіювання» та «Скасувати копіювання». Перша кнопка запускає формування резервної копії, а друга перериває активне завдання або очищає тимчасові параметри, якщо операція ще не почалася. Щоб графічний інтерфейс не блокувався під час тривалого копіювання, запуск резервної операції доцільно реалізувати в окремому потоці або асинхронному завданні.

Фрагмент коду запуску операції має такий вигляд:

```
def start_backup(self):
    config = {
        "paths": [self.selected_files.item(i).text() for i in range(self.selected_files.count())],
        "password": self.password_input.text(),
        "date": self.date_field.text()
    }
    self.backup_controller.run_backup(config)
```

На рівні функціоналу це вікно забезпечує повний цикл початкового налаштування резервування. Користувач самостійно формує перелік об’єктів резервування, задає параметри захисту й визначає час виконання. Таким чином, саме вікно на рисунку 3.2 є основним інструментом практичного використання системи.

3.4 Реалізація вікна додаткових налаштувань токена та параметрів безпеки

Для роботи із хмарним середовищем необхідний механізм керування параметрами доступу. У розробленій системі відповідні функції реалізовано у вікні додаткових налаштувань, зовнішній вигляд якого подано на рисунку 3.3.

Це вікно містить поля для теми, параметри автозапуску, вимкнення ML-модуля, вимкнення сповіщень, вибір мови інтерфейсу, а також кнопки оновлення та скидання токена.

У верхній частині вікна відображається поточний стан токена, зокрема повідомлення «Токен активний». Це означає, що програма вже має дійсні параметри доступу до хмарного середовища та може виконувати завантаження резервних копій. Якщо токен втрачає чинність або видаляється, відповідний статус має змінюватися автоматично, а хмарні операції тимчасово блокуються до повторної авторизації.

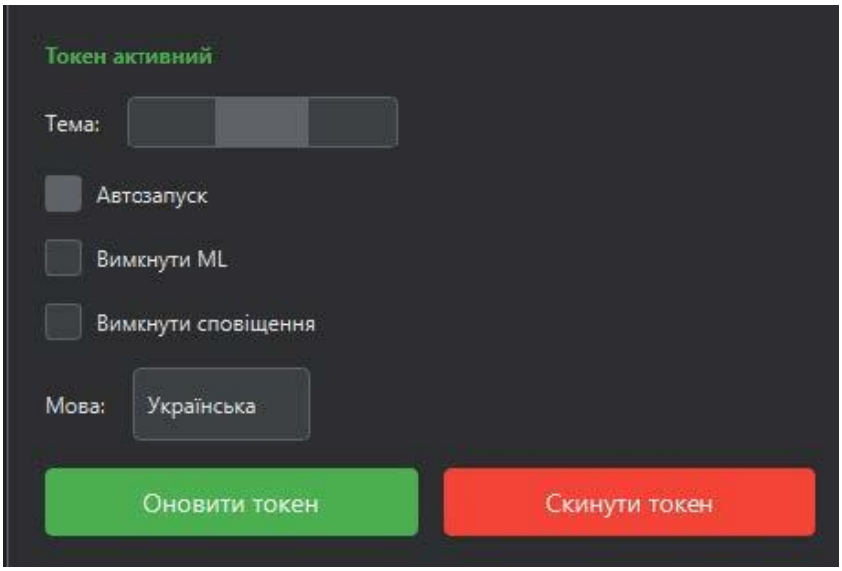

Рисунок 3.3 – Вікно додаткових налаштувань і керування токеном

З погляду програмної реалізації це вікно містить набір прапорців, випадаючий список мов, елементи керування темою оформлення й кнопки сервісних операцій. Для зберігання параметрів доцільно використати окремий конфігураційний файл у форматі JSON. Такий файл містить значення прапорців, обрану мову, ознаку автозапуску й допоміжні параметри інтерфейсу. Під час старту програми налаштування зчитуються, а у разі змін вони записуються назад на диск.

Реалізація класу для роботи з налаштуваннями виглядає так:

```
import json
from pathlib import Path

class SettingsService:
    def __init__(self, config_path="config/settings.json"):
        self.config_path = Path(config_path)
        self.config_path.parent.mkdir(parents=True, exist_ok=True)

    def load(self):
        if not self.config_path.exists():
            return {
                "autostart": False,
                "disable_ml": False,
                "disable_notifications": False,
                "language": "Українська"
            }
        with open(self.config_path, "r", encoding="utf-8") as file:
            return json.load(file)

    def save(self, data):
        with open(self.config_path, "w", encoding="utf-8") as file:
            json.dump(data, file, ensure_ascii=False, indent=4)
```

Кнопка «Оновити токен» повинна викликати процедуру оновлення параметрів доступу до Google Cloud. У простому варіанті реалізації це може бути перевірка наявного ключа сервісного облікового запису, тестовий запит до хмарного сховища або повторне завантаження конфігурації середовища. Кнопка «Скинути токен» видаляє або деактивує локально збережені параметри доступу, після чого стан токена в головному вікні змінюється на неактивний.

Приклад реалізації відповідної логіки в рамках бакалаврської роботи:

```
class TokenService:
    def __init__(self):
        self.active = False

    def refresh_token(self):
        # тут виконується перевірка облікових даних
        self.active = True
        return self.active

    def reset_token(self):
        self.active = False
        return self.active

    def get_status(self):
        return self.active
```

Наявність у цьому вікні параметра «Автозапуск» дозволяє інтегрувати систему з механізмами автоматичного запуску програми після старту операційної системи. Практично це важливо для сценаріїв регулярного резервування, коли користувачеві не потрібно щоразу вручну відкривати застосунок. Прапорець «Вимкнути сповіщення» впливає на відображення повідомлень про завершення резервування або виникнення помилок. Параметр «Вимкнути ML» у структурі системи може використовуватися як службовий перемикач для відключення додаткового аналітичного модуля, якщо він не потрібен у поточній конфігурації.

3.5 Реалізація модуля формування резервної копії

Основною функціональною частиною системи є модуль формування резервної копії. Він отримує від інтерфейсу перелік файлів або каталогів,

перевіряє їх існування, формує тимчасову структуру резервування, стискає її в архів і за потреби виконує шифрування. На рівні внутрішньої логіки цей модуль повинен бути незалежним від графічного інтерфейсу, щоб його можна було перевикористовувати в автоматичному режимі або під час запуску за розкладом.

Для створення архіву в середовищі Python зручно використати стандартний модуль zipfile. Якщо потрібно зберігати також службові метадані, до архіву можна додавати окремий JSON-файл із датою створення, списком шляхів, міткою користувача та параметрами версії резервної копії.

Фрагмент коду, що реалізує цей модуль у нашій системі:

```
import zipfile
from pathlib import Path

class BackupService:
    def create_backup_archive(self, config):
        archive_name = f"backup_{config['timestamp']}.zip"
        archive_path = Path("temp") / archive_name
        archive_path.parent.mkdir(exist_ok=True)

        with zipfile.ZipFile(archive_path, "w", zipfile.ZIP_DEFLATED) as archive:
            for item in config["paths"]:
                path = Path(item)
                if path.is_file():
                    archive.write(path, arcname=path.name)
                elif path.is_dir():
                    for file in path.rglob("*"):
                        if file.is_file():
                            archive.write(file, arcname=file.relative_to(path.parent))
        return archive_path
```

У наведеному фрагменті коду використовується проходження по списку шляхів, де для окремих файлів і каталогів застосовуються різні правила додавання до архіву. Такий підхід дозволяє зберігати структуру каталогів і водночас об’єднувати всі об’єкти резервування в один цілісний архівний файл.

3.6 Реалізація модуля передавання даних у Google Cloud

Для збереження резервних копій у хмарному середовищі система використовує взаємодію з Google Cloud Storage. На рівні програмної реалізації це означає створення окремого сервісу, відповідального за ініціалізацію клієнта, підключення до бакета, формування імені об’єкта та завантаження архіву в хмару. Важливо, щоб цей модуль не залежав від графічного представлення, а приймав готовий шлях до архіву та конфігурацію резервної операції.

Приклад фрагмента коду:

```
from google.cloud import storage

class CloudUploader:
    def __init__(self, bucket_name: str):
        self.client = storage.Client()
        self.bucket = self.client.bucket(bucket_name)

    def upload_file(self, local_path: str, remote_name: str):
        blob = self.bucket.blob(remote_name)
        blob.upload_from_filename(local_path)
        return blob.name
```

У повній реалізації для іменування об’єктів використано структуру, що включає дату, час і логічну назву завдання. Наприклад, файл може зберігатися у вигляді backups/2026-02-12/backup_120000.zip. Такий підхід полегшує відновлення, сортування і подальше очищення застарілих копій.

Окремо реалізована перевірка результату завантаження та обробка винятків. Якщо під час передавання до хмари виникає помилка, система повинна зафіксувати її в журналі та повідомити користувача через інтерфейс.

```
def upload_to_cloud(self, archive_path, config):
    try:
        remote_name = f"backups/{config['timestamp']}.zip"
        return self.cloud_uploader.upload_file(str(archive_path), remote_name)
    except Exception as error:
        self.logger.write(f"Помилка завантаження в хмару: {error}")
        raise
```

Такий підхід забезпечує передбачувану поведінку системи навіть у випадках мережевих проблем, недійсного токена або помилок конфігурації.

3.7 Реалізація модуля планування та журналювання

Оскільки система повинна підтримувати запуск резервних операцій не лише вручну, а й за встановленим розкладом, у її складі передбачено модуль планування. У настільному варіанті розробки базовий розклад можна реалізувати локально через таймер або фоновий сервіс. У подальшому розширенні цей механізм може бути інтегрований із сервісами Google Cloud Scheduler і Cloud Run, однак на етапі бакалаврської розробки, на нашу думку, було достатньо реалізувати локальне збереження дати й часу запланованого запуску та перевірку настання цього моменту.

Журналювання є ще одним обов’язковим компонентом системи. Воно потрібне для фіксації початку резервування, переліку вибраних об’єктів, успішного завершення архівації, результату завантаження до хмари та можливих помилок. Для цього доцільно використати стандартний модуль logging.

```
import logging
logging.basicConfig(
    filename="logs/backup.log",
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s"
)
logging.info("Розпочато формування резервної копії")
```

Наявність журналу суттєво підвищує зручність адміністрування системи, оскільки дозволяє аналізувати причини збоїв, перевіряти факт виконання запланованих операцій і контролювати історію резервування.

У третьому розділі виконано проєктування інформаційної системи резервного копіювання даних та описано реалізацію її основних модулів. Побудована архітектура передбачає поділ на інтерфейсний рівень, прикладну логіку та модулі роботи з даними. Детально розглянуто програмну реалізацію головного вікна системи, вікна налаштування резервного копіювання та вікна додаткових параметрів і керування токеном. Також описано модуль формування резервного архіву, модуль передавання даних до Google Cloud Storage, а також підсистеми планування та журналювання.

Запропонована структура системи забезпечує можливість подальшого розширення функціональності, зокрема додавання повноцінного автоматичного запуску за розкладом, розширеного механізму шифрування, відновлення окремих версій резервних копій і інтеграції з додатковими сервісами Google Cloud.

РОЗДІЛ 4.
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Аналіз стану виробничої санітарії і гігієни праці

Виконання робіт за темою кваліфікаційної роботи пов’язане з розробленням, налагодженням і тестуванням інформаційної системи резервного копіювання даних у хмарному середовищі. Основними видами робіт є робота за персональним комп’ютером, аналіз програмного коду, конфігурування хмарних сервісів, тестування графічного інтерфейсу, перевірка сценаріїв передавання та відновлення даних, а також підготовка технічної документації. Такі роботи належать до категорії робіт із переважним нервово-емоційним та зоровим навантаженням, виконуються в приміщенні з постійним перебуванням працівника на робочому місці та повинні організовуватися з дотриманням вимог до безпечної експлуатації обладнання, організації робочого місця, мікроклімату, освітлення і режиму праці та відпочинку. Загальні вимоги щодо створення безпечних і нешкідливих умов праці покладаються на роботодавця, який має забезпечити належне облаштування робочих місць, навчання працівників, інструктажі та контроль за додержанням вимог охорони праці [29, 31, 33].

Основним обладнанням під час виконання робіт є персональний комп’ютер або ноутбук, монітор, клавіатура, миша, мережеве обладнання, пристрої резервного живлення, а також допоміжні офісні меблі. Потенційно шкідливими та небезпечними чинниками для працівника в таких умовах є тривале статичне навантаження, перенапруження органів зору, психоемоційне напруження, недостатнє або нерівномірне освітлення, підвищене зорове навантаження під час роботи з екраном, електробезпекові ризики, пов’язані з використанням електрообладнання, а також ризик пожежі внаслідок короткого замикання, перевантаження мережі або несправності блоків живлення. Для профілактики цих ризиків на підприємстві або в установі мають функціонувати

служба охорони праці або відповідальна особа, проводитися вступний, первинний, повторний та позаплановий інструктажі, вестися журнали реєстрації інструктажів і забезпечуватися контроль технічного стану обладнання. Такі організаційні заходи прямо випливають із загальних вимог щодо забезпечення роботодавцями охорони праці працівників [34].

Розміщення приміщення, у якому виконується розробка та тестування системи, повинно відповідати санітарно-гігієнічним вимогам до адміністративних та офісних робочих місць. Робоче місце користувача ПК доцільно розміщувати так, щоб природне світло надходило збоку, переважно зліва, а на екрані монітора не виникали відблиски від вікон чи світильників. Відстань від очей працівника до екрана, положення монітора, висота столу і крісла мають забезпечувати зручну робочу позу та мінімізувати навантаження на зір і опорно-руховий апарат. Санітарні правила для роботи з візуальними дисплейними терміналами встановлюють гігієнічні вимоги до організації робочих місць, умов освітлення, параметрів середовища та режиму праці.

Санітарно-гігієнічні умови виконання робіт у межах даної кваліфікаційної роботи повинні враховувати особливості інтелектуальної праці за комп’ютером. Найбільший вплив на працівника мають параметри мікроклімату, рівень освітлення, тривалість безперервної роботи з екраном, ергономічність робочого місця та шумове навантаження від технічних засобів. Для приміщення, у якому виконується програмна розробка, особливе значення має підтримання комфортної температури, достатньої вологості повітря, ефективного повітрообміну та відсутності надмірного шуму. Недотримання цих параметрів може призводити до перевтоми, зниження концентрації уваги, головного болю, подразнення слизових оболонок, погіршення зору та зростання імовірності помилок у роботі. Вимоги до таких умов визначаються санітарними правилами для роботи з ВДТ, а також загальними вимогами до облаштування робочих місць.

Важливим фактором безпечної роботи є освітлення. Для офісних приміщень і приміщень із роботою за ПК параметри природного та штучного

освітлення мають відповідати будівельним нормам щодо освітленості робочих поверхонь, рівномірності освітлення та обмеження сліпучої дії світильників. Недостатня освітленість змушує працівника напружувати зір, а надлишкова або неправильно спрямована створює відблиски на екрані. Для робіт із документами та екранною інформацією зазвичай орієнтуються на нормовану освітленість робочої поверхні, яка забезпечує чітке сприйняття тексту та графічних елементів, а конкретні параметри штучного освітлення визначаються ДБН В.2.5-28:2018 [32].

Окрему увагу слід приділити режиму праці та відпочинку. Робота програміста або оператора ПК характеризується тривалим зосередженням уваги, статичним положенням тіла та повторюваними рухами кистей рук. Тому при виконанні робіт необхідно передбачати регламентовані перерви, зміну виду діяльності, короткочасні вправи для очей і кистей рук, а також недопущення надмірної тривалості безперервної роботи за екраном. Організація раціонального режиму праці й відпочинку є одним із основних профілактичних заходів щодо запобігання перевтомі, професійним захворюванням і помилкам, що можуть спричинити аварійні ситуації або втрату даних. Відповідні вимоги щодо організації праці при роботі з візуальними дисплейними терміналами встановлені санітарними правилами [31, 33].

Загалом стан виробничої санітарії та гігієни праці при виконанні робіт за темою кваліфікаційної роботи можна вважати задовільним за умови дотримання нормативних вимог до організації робочого місця, мікроклімату, освітлення, електробезпеки та режиму праці. Для подальшого підвищення безпеки доцільно рекомендувати: використання ергономічних крісел і столів, застосування моніторів із якісним антибліковим покриттям, періодичний контроль освітленості, провітрювання приміщення, перевірку електрообладнання, проведення інструктажів і ведення відповідної документації. Такі заходи зменшують імовірність нещасних випадків, сприяють зниженню захворюваності та підвищують загальну ефективність роботи персоналу.

4.2 Обґрунтування організаційно-технічних рекомендацій з охорони праці

Для поліпшення умов праці при розробленні системи резервного копіювання доцільно застосувати комплекс організаційних і технічних заходів. До організаційних заходів належать: призначення відповідального за охорону праці, проведення навчання та інструктажів, організація періодичного контролю стану робочих місць, упровадження регламентованих перерв і контроль технічного стану обладнання. До технічних заходів належать: облаштування робочих місць із дотриманням ергономічних вимог, забезпечення нормативного рівня освітленості, правильне розміщення моніторів щодо джерел світла, використання стабільного електроживлення, застосування справних подовжувачів і мережевих фільтрів, а також забезпечення достатнього повітрообміну в приміщенні. Такі заходи відповідають загальним вимогам щодо створення безпечних і нешкідливих умов праці.

У межах цього пункту доцільно виконати розрахунок штучного освітлення для приміщення, де здійснюється розробка та тестування програмної системи. Припустимо, що робота виконується в кімнаті площею 24 м² із трьома робочими місцями, тобто розміри приміщення становлять 6 × 4 м. Для робіт за комп’ютером приймаємо нормовану освітленість робочої поверхні E = 300 лк, що узгоджується з вимогами ДБН В.2.5-28:2018 для офісних та подібних приміщень. Для розрахунку кількості світильників використаємо спрощену формулу методу світлового потоку [32]:

$$N = \frac{E \cdot S \cdot k \cdot z}{F \cdot \eta}$$

де N - кількість світильників; E - нормована освітленість, лк; S - площа приміщення, м²; k - коефіцієнт запасу; z - коефіцієнт нерівномірності; F- світловий потік одного світильника, лм; η - коефіцієнт використання світлового потоку.

Для розрахунку приймаємо такі вихідні дані:

- $S = 24$ м
- $E = 300$ лк;
- $k = 1{,}4$;
- $z = 1{,}1$;
- використовується LED-світильник зі світловим потоком $F = 3200$ лм;
- коефіцієнт використання $\eta = 0{,}6$.

Підставляємо значення у формулу:

$$N = \frac{300 \cdot 24 \cdot 1{,}4 \cdot 1{,}1}{3200 \cdot 0{,}6}$$

$$N = \frac{11088}{1920} \approx 5{,}78$$

Отже, для забезпечення нормативного рівня штучного освітлення потрібно прийняти 6 LED-світильників. Таке рішення забезпечує необхідний запас освітленості, підвищує рівномірність світлорозподілу та зменшує втому очей під час роботи з екраном і паперовими документами. Вибір нормованої освітленості ґрунтується на вимогах ДБН В.2.5-28:2018 [32].

На підставі виконаного розрахунку можна рекомендувати рівномірне розташування шести стельових LED-світильників у два ряди по три світильники. Монітори слід розміщувати так, щоб світловий потік не був спрямований безпосередньо на екрани. Додатково доцільно використовувати жалюзі або рулонні штори для регулювання природного освітлення в денний час. Це дасть змогу зменшити контрастність між яскравістю вікна та екрана й уникнути відблисків. Також варто передбачити регулярне очищення світильників і вікон, оскільки забруднення поверхонь зменшує фактичний рівень освітленості.

До організаційно-технічних рекомендацій з охорони праці для даної роботи доцільно включити такі заходи: забезпечення кожного працівника

справним робочим місцем із сертифікованими меблями; використання якісних LED-світильників; періодичне вимірювання освітленості; регламентування тривалості безперервної роботи за ПК; упорядкування кабельного господарства; використання джерел безперебійного живлення для критичного обладнання; своєчасне технічне обслуговування комп'ютерів; збереження паролів і ключів доступу до хмарного середовища із застосуванням безпечних методів; а також проведення повторних інструктажів з електробезпеки та пожежної безпеки. У результаті реалізації таких заходів підвищується кількість робочих місць, що відповідають нормативним вимогам, знижується ризик перевтоми, зменшується імовірність аварійних ситуацій та підвищується загальна надійність виконання робіт.

4.3 Пожежна безпека

Пожежна безпека під час виконання робіт за темою кваліфікаційної роботи має особливе значення, оскільки експлуатація комп'ютерної техніки, мережевих пристроїв, блоків живлення, зарядних пристроїв та іншого електрообладнання пов'язана з ризиком перегрівання, короткого замикання, іскріння або займання ізоляції. Небезпечними чинниками пожежі є підвищена температура, токсичні продукти горіння, дим, зниження вмісту кисню в повітрі, можливість ураження полум'ям або електричним струмом, а також матеріальні збитки внаслідок знищення обладнання й даних. Система протипожежного захисту в офісному або лабораторному приміщенні повинна поєднувати організаційні заходи та технічні засоби, спрямовані на запобігання пожежі, своєчасне виявлення загоряння, евакуацію людей і успішне гасіння пожежі на початковій стадії. Чинні Правила пожежної безпеки в Україні встановлюють обов'язкові вимоги щодо утримання приміщень, шляхів евакуації, використання електрообладнання та забезпечення первинними засобами пожежогасіння [30].

Для приміщення, у якому виконуються роботи з програмної розробки та тестування інформаційної системи, характерним є наявність горючих матеріалів

у вигляді пластикових елементів корпусів техніки, ізоляції проводів, паперових носіїв і меблів, але відсутні технологічні процеси з використанням легкозаймистих рідин чи вибухонебезпечних сумішей. Тому таке приміщення зазвичай відносять до приміщень з помірною пожежною небезпекою, де основну увагу слід приділяти електробезпеці, справності проводки та недопущенню перевантаження електромережі. Усі електроприлади мають бути справними, підключеними через штатні розетки, а використання пошкоджених шнурів, саморобних подовжувачів або перевантажених трійників є неприпустимим. Ці підходи відповідають загальним правилам пожежної безпеки для будівель і приміщень [31].

У приміщенні мають бути передбачені первинні засоби пожежогасіння. Для офісних і комп’ютерних приміщень доцільно застосовувати вуглекислотні або порошкові вогнегасники, оскільки вони придатні для гасіння загорянь електрообладнання під напругою в установлених межах. Вогнегасники повинні розміщуватися у доступних місцях, позначатися відповідними знаками, періодично перевірятися і проходити технічне обслуговування. Додатково необхідно забезпечити вільний доступ до евакуаційних виходів, наявність плану евакуації, справний стан систем оповіщення, а за потреби - автоматичну пожежну сигналізацію. Організаційні заходи мають включати інструктаж працівників щодо дій у разі пожежі, правил користування вогнегасником і порядку знеструмлення обладнання.

Пожежна профілактика в межах даної кваліфікаційної роботи полягає у впровадженні комплексу запобіжних заходів. До них належать: систематичний контроль стану електропроводки та розеткових груп; заборона залишати ввімкнене обладнання без нагляду на тривалий час; використання джерел безперебійного живлення та мережевих фільтрів заводського виготовлення; розміщення кабелів без механічного пошкодження та перегинів; недопущення захаращення приміщення папером та упаковкою; регулярне прибирання пилу з вентиляційних отворів комп’ютерів і блоків живлення; а також обмеження доступу сторонніх осіб до електротехнічних вузлів. Якщо ці рекомендації

виконуються, імовірність пожежі суттєво знижується, а у разі виникнення загоряння створюються умови для швидкої локалізації небезпечної ситуації без тяжких наслідків для людей і обладнання.

Отже, для робіт із розроблення та експлуатації системи резервного копіювання даних у хмарних середовищах охорона праці повинна розглядатися як комплекс організаційних, санітарно-гігієнічних і технічних заходів. Найважливішими напрямами є забезпечення належної організації робочого місця, нормативного освітлення, раціонального режиму праці та відпочинку, електробезпеки та пожежної профілактики. Реалізація запропонованих рекомендацій дозволяє створити безпечні умови праці для працівника, який виконує проектування, програмну реалізацію та тестування інформаційної системи.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

У кваліфікаційній роботі розв’язано актуальне завдання створення системи резервного копіювання даних у хмарних середовищах. Актуальність обраної теми зумовлена постійним зростанням обсягів цифрової інформації, необхідністю забезпечення її цілісності, доступності та захисту від втрати внаслідок технічних збоїв, помилок користувачів, відмов обладнання або дії шкідливого програмного забезпечення. Проведене дослідження підтвердило, що використання хмарних технологій є доцільним напрямом побудови сучасних систем резервного копіювання, оскільки дозволяє поєднати масштабованість, віддалене зберігання даних, автоматизацію резервних операцій та зручність адміністрування.

У першому розділі виконано аналіз предметної області резервного копіювання даних у хмарних середовищах. Розглянуто теоретичні засади побудови систем резервування, основні типи резервного копіювання, принципи організації зберігання копій, підходи до відновлення інформації, а також показники RPO та RTO. Проаналізовано сучасні підходи й технології побудови хмарних систем резервного копіювання та визначено основні вимоги до їх проєктування. У результаті встановлено, що ефективна система резервування повинна забезпечувати надійність збереження даних, інформаційну безпеку, масштабованість, автоматизацію резервних процесів і підтримку різних сценаріїв відновлення.

У другому розділі обґрунтовано вибір напряму дослідження та визначено основні підходи до розв’язання поставленої задачі. Встановлено, що як основа розроблюваної системи доцільно використовувати хмарне середовище Google Cloud, яке надає необхідний набір сервісів для зберігання резервних копій, автоматизації запуску, журналювання подій та захисту конфігураційних параметрів. Описано загальну методику проведення власних досліджень, що включає аналіз вимог, вибір хмарних сервісів, проєктування архітектури системи, програмну реалізацію та експериментальну перевірку її працездатності.

Також обґрунтовано вибір мови програмування Python як інструменту реалізації системи завдяки її придатності до швидкої розробки, зручній роботі з файлами, підтримці архівації, логування та інтеграції з сервісами Google Cloud.

У третьому розділі виконано проєктування інформаційної системи резервного копіювання та описано реалізацію її основних модулів. Розроблено загальну структуру застосунку, що включає модуль головного вікна, модуль налаштування резервного копіювання, модуль додаткових параметрів і керування токеном, модуль формування резервної копії, модуль передавання даних у Google Cloud Storage, а також підсистеми планування і журналювання. Визначено логіку взаємодії між інтерфейсом користувача, прикладною логікою та компонентами роботи з даними. Описано програмну реалізацію графічного інтерфейсу користувача з орієнтацією на зручність використання, послідовність виконання резервної операції та контроль стану хмарної авторизації.

У результаті виконаної роботи досягнуто поставленої мети, а саме спроєктовано систему резервного копіювання даних у хмарному середовищі, яка забезпечує вибір файлів і каталогів для копіювання, формування резервного архіву, можливість використання пароля, зберігання резервних копій у Google Cloud, налаштування параметрів запуску та ведення журналу виконаних операцій. Таким чином, поставлені в роботі завдання виконано: досліджено теоретичні засади резервного копіювання, проаналізовано сучасні підходи й сервіси, визначено вимоги до системи, обґрунтовано вибір засобів реалізації, спроєктовано структуру програмного рішення та описано реалізацію його основних модулів.

Практичне значення одержаних результатів полягає в тому, що запропоноване рішення може бути використане як основа для створення прикладної системи резервного копіювання даних для невеликих організацій, навчальних закладів або окремих користувачів, які потребують надійного механізму збереження інформації в хмарному середовищі. Розроблена структура системи придатна до подальшого розширення, зокрема в напрямі автоматичного запуску резервних операцій за розкладом, реалізації повноцінного відновлення

окремих версій даних, удосконалення механізмів шифрування, розширення журналювання подій і поглибленої інтеграції з додатковими сервісами Google Cloud.

Отже, результати кваліфікаційної роботи підтверджують доцільність використання хмарних технологій для побудови систем резервного копіювання даних та демонструють можливість створення працездатного програмного рішення, яке поєднує функціональність, зручність використання й перспективи подальшого розвитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mell P., Grance T. The NIST Definition of Cloud Computing : NIST Special Publication 800-145. Gaithersburg : National Institute of Standards and Technology, 2011. URL: https://csrc.nist.gov/pubs/sp/800/145/final (дата звернення: 12.02.2026).

2. Backup and recovery approaches on AWS [Електронний ресурс] // AWS Prescriptive Guidance. URL: https://docs.aws.amazon.com/prescriptive-guidance/latest/backup-recovery/welcome.html (дата звернення: 12.02.2026).

3. Architecture Overview – Azure Backup [Електронний ресурс] // Microsoft Learn. URL: https://learn.microsoft.com/en-us/azure/backup/backup-architecture (дата звернення: 12.02.2026).

4. Incremental Backup vs. Differential Backup – The Differences [Електронний ресурс] // Contabo. URL: https://contabo.com/blog/incremental-vs-differential-backups/ (дата звернення: 12.02.2026).

5. Backup and Disaster Recovery (DR) Service [Електронний ресурс] // Google Cloud. URL: https://cloud.google.com/backup-disaster-recovery (дата звернення: 12.02.2026).

6. Backup and recovery using AWS Backup [Електронний ресурс] // AWS Prescriptive Guidance. URL: https://docs.aws.amazon.com/prescriptive-guidance/latest/backup-recovery/aws-backup.html (дата звернення: 12.02.2026).

7. Backup and recovery from on-premises infrastructure to AWS [Електронний ресурс] // AWS Prescriptive Guidance. URL: https://docs.aws.amazon.com/prescriptive-guidance/latest/backup-recovery/on-premises-to-aws.html (дата звернення: 12.02.2026).

8. What is the Azure Backup service? [Електронний ресурс] // Microsoft Learn. URL: https://learn.microsoft.com/en-us/azure/backup/backup-overview (дата звернення: 12.02.2026).

9. Product overview | Backup and DR [Електронний ресурс] // Google Cloud Documentation. URL: https://docs.cloud.google.com/backup-disaster-recovery/docs/concepts/backup-dr (дата звернення: 12.02.2026).

10. Chandramouli R., Butrico M., Eydt B. Security Guidelines for Storage Infrastructure : NIST Special Publication 800-209. Gaithersburg : National Institute of Standards and Technology, 2020. URL: https://csrc.nist.gov/pubs/sp/800/209/final (дата звернення: 12.02.2026).

11. Secure by default with soft delete for Azure Backup [Електронний ресурс] // Microsoft Learn. URL: https://learn.microsoft.com/en-us/azure/backup/secure-by-default (дата звернення: 12.02.2026).

12. Best practices for backup and recovery of data [Електронний ресурс] // AWS Prescriptive Guidance. URL: https://docs.aws.amazon.com/prescriptive-guidance/latest/security-best-practices/best-practices.html (дата звернення: 12.02.2026).

13. Step 1. Implement a backup strategy [Електронний ресурс] // AWS Prescriptive Guidance. URL: https://docs.aws.amazon.com/prescriptive-guidance/latest/security-best-practices/strategy.html (дата звернення: 12.02.2026).

14. Disaster recovery planning guide [Електронний ресурс] // Google Cloud Architecture Center. URL: https://docs.cloud.google.com/architecture/dr-scenarios-planning-guide (дата звернення: 12.02.2026).

15. Cloud Storage [Електронний ресурс] // Google Cloud. URL: https://cloud.google.com/storage (дата звернення: 12.02.2026).

16. Product overview of Cloud Storage [Електронний ресурс] // Google Cloud Documentation. URL: https://docs.cloud.google.com/storage/docs/introduction (дата звернення: 12.02.2026).

17. Storage classes [Електронний ресурс] // Google Cloud Documentation. URL: https://docs.cloud.google.com/storage/docs/storage-classes (дата звернення: 12.02.2026).

18. Object Lifecycle Management [Електронний ресурс] // Google Cloud Documentation. URL: https://docs.cloud.google.com/storage/docs/lifecycle (дата звернення: 12.02.2026).

19. Backup and DR Service documentation [Електронний ресурс] // Google Cloud Documentation. URL: https://docs.cloud.google.com/backup-disaster-recovery/docs (дата звернення: 12.02.2026).

20. Perform testing for recovery from data loss [Електронний ресурс] // Google Cloud Architecture Center. URL: https://docs.cloud.google.com/architecture/framework/reliability/perform-testing-for-recovery-from-data-loss (дата звернення: 12.02.2026).

21. What is Cloud Run [Електронний ресурс] // Google Cloud Documentation. URL: https://docs.cloud.google.com/run/docs/overview/what-is-cloud-run (дата звернення: 12.02.2026).

22. Cloud Scheduler documentation [Електронний ресурс] // Google Cloud Documentation. URL: https://docs.cloud.google.com/scheduler/docs (дата звернення: 12.02.2026).

23. Running services on a schedule [Електронний ресурс] // Google Cloud Documentation. URL: https://docs.cloud.google.com/run/docs/triggering/using-scheduler (дата звернення: 12.02.2026).

24. Secret Manager overview [Електронний ресурс] // Google Cloud Documentation. URL: https://docs.cloud.google.com/secret-manager/docs/overview (дата звернення: 12.02.2026).

25. Cloud Logging overview [Електронний ресурс] // Google Cloud Documentation. URL: https://docs.cloud.google.com/logging/docs/overview (дата звернення: 12.02.2026).

26. Cloud Audit Logs overview [Електронний ресурс] // Google Cloud Documentation. URL: https://docs.cloud.google.com/logging/docs/audit (дата звернення: 12.02.2026).

27. Python Cloud Client Libraries [Електронний ресурс] // Google Cloud Documentation. URL: https://docs.cloud.google.com/python/docs/reference (дата звернення: 12.02.2026).

28. Cloud Storage client libraries [Електронний ресурс] // Google Cloud Documentation. URL: https://docs.cloud.google.com/storage/docs/reference/libraries (дата звернення: 12.02.2026).

29. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98 [Електронний ресурс] : правила : затв. наказом МОЗ України від 10.12.1998 № 7. URL: https://zakon.rada.gov.ua/go/v0007282-98 (дата звернення: 12.02.2026).

30. Про затвердження Правил пожежної безпеки в Україні [Електронний ресурс] : наказ МВС України від 30.12.2014 № 1417. URL: https://zakon.rada.gov.ua/go/z0252-15 (дата звернення: 12.02.2026).

31. Загальні вимоги стосовно забезпечення роботодавцями охорони праці працівників [Електронний ресурс] : наказ МНС України від 25.01.2012 № 67. URL: https://zakon.rada.gov.ua/go/z0226-12 (дата звернення: 12.02.2026).

32. ДБН В.2.5-28:2018. Природне і штучне освітлення [Електронний ресурс]. Київ, 2018. URL: https://zakon.isu.net.ua/sites/default/files/normdocs/dbn_v_2.5-28_2018.pdf (дата звернення: 12.02.2026).

33. Пістун І. П., Тимочко В.О., Городецький І. М.. Березовецький А. П. Охорона праці (гігієна праці та виробнича санітарія): навч. посібн. / за ред. І.П. Пістуна. Ч. ІІ. Львів: Тріада плюс, 2011. 224 с.

34. Пістун І. П., Тимочко В.О., Городецький І. М.. Березовецький А. П. Охорона праці (гігієна праці та виробнича санітарія): навч. посібн. / за ред. І.П. Пістуна. Ч. ІІ. Львів: Тріада плюс, 2011. 224 с.

ДОДАТКИ

Фрагмент коду головного вікна інформаційної системи

```
from PyQt6.QtWidgets import (
    QMainWindow, QWidget, QHBoxLayout, QVBoxLayout,
    QPushButton, QLabel, QFrame
)

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.resize(920, 540)

        central_widget = QWidget()
        self.setCentralWidget(central_widget)

        root_layout = QHBoxLayout(central_widget)

        self.sidebar = QFrame()
        self.sidebar_layout = QVBoxLayout(self.sidebar)

        self.info_button = QPushButton("Інфо-панель")
        self.ml_button = QPushButton("ML деталі")
        self.settings_button = QPushButton("Налаштування")

        self.sidebar_layout.addWidget(self.info_button)
        self.sidebar_layout.addWidget(self.ml_button)
        self.sidebar_layout.addWidget(self.settings_button)
        self.sidebar_layout.addStretch()
```

```
self.content = QFrame()
self.content_layout = QVBoxLayout(self.content)

self.token_label = QLabel("Токен неактивний")
self.backup_button = QPushButton("Резервне копіювання")

self.content_layout.addWidget(self.token_label)
self.content_layout.addStretch()
self.content_layout.addWidget(self.backup_button)
self.content_layout.addStretch()

root_layout.addWidget(self.sidebar, 1)
root_layout.addWidget(self.content, 3)
```