DOI: https://doi.org/10.36910/6775-2524-0560-2026-62-26
УДК 004.65
Загвойський Ростислав Юрійович, аспірант
https://orcid.org/0009-0005-2090-4255
Казимира Ірина Ярославівна, к.т.н., доцент
https://orcid.org/0000-0003-1597-5647
Данько Юрій Ігорович, магістр
Національний університет «Львівська політехніка», м. Луцьк, Україна

МОДЕЛЬ ТА ЗАСОБИ ЗБЕРІГАННЯ ЦИФРОВИХ РЕСУРСІВ КОРИСТУВАЧА З ВИКОРИСТАННЯМ ХМАРНИХ ТЕХНОЛОГІЙ

Загвойський Р.Ю., Казимира І.Я., Данько Ю. І. Модель та засоби зберігання цифрових ресурсів користувача з використанням хмарних технологій. Стрімке зростання обсягів цифрової інформації, яку генерують користувачі та організації в умовах цифрової трансформації, актуалізує потребу в надійних, масштабованих і безпечних інструментах її зберігання, структуризації та контрольованого доступу. У роботі розглянуто побудову приватно орієнтованого хмарного сховища, що забезпечує повний контроль над цифровими ресурсами, політиками безпеки та життєвим циклом даних, а також зменшує ризики, пов'язані з передачею даних третім сторонам. Запропоновано концептуальну модель і мікросервісну архітектуру системи «Cloud Drive», реалізовану з використанням сервісів AWS, де функціональність декомпозовано на Auth, User і Storage сервіси з можливістю незалежного масштабування. Програмний прототип створено на базі Spring Boot та React і він підтримує ієрархічну організацію файлів і папок, рольову модель контролю доступу (RBAC) з гранулярними правами, автентифікацію на основі JWT та механізм відкликання токенів. Для зберігання метаданих використано PostgreSQL, а бінарний вміст делеговано Amazon S3, що відповідає гібридному підходу розділення метаданих і об'єктів. Безпечне завантаження реалізовано через попередньо підписані URL, які дозволяють виконувати пряму передачу даних між клієнтом і S3 без проксіювання через бекенд, знижуючи мережеве навантаження. Антивірусну перевірку організовано асинхронним пайплайном із карантинним бакетом, подіями S3 та чергою SQS із використанням ClamAV. Результати експериментальних і навантажувальних випробувань підтверджують ефективність рішень щодо продуктивності, безпеки та масштабованості системи. Зокрема, під час stress test сценарію завантаження 10 000 запитів зафіксовано нуль помилок і середній час відповіді близько 6 мс.
Ключові слова: хмарні технології, автоматизоване управління, мікросервісна архітектура, AWS S3, RBAC, Cloud Drive, обчислювальні системи.

Zahvoiskyi R., Kazymyra I., Danko Y. Model and tools for storing user digital resources using cloud technologies. The rapid growth of digital information generated by users and organizations in the context of digital transformation actualizes the need for reliable, scalable, and secure tools for its storage, structuring, and controlled access. This paper examines the design of a private cloud storage system that ensures full control over digital resources, security policies, and the data lifecycle, while reducing risks associated with transferring data to third parties. A conceptual model and microservice architecture of the «Cloud Drive» system are proposed, implemented using AWS services, where functionality is decomposed into Auth, User, and Storage services with the possibility of independent scaling. A software prototype was developed based on the Spring Boot and React stack; it supports hierarchical organization of files and folders, a role-based access control (RBAC) model with granular permissions, JWT-based authentication, and a token revocation mechanism. PostgreSQL is used for metadata storage, while binary content is delegated to Amazon S3, reflecting a hybrid approach that separates metadata from file objects. Secure upload is implemented through presigned URLs, enabling direct data transfer between the client and S3 without backend proxying, thereby reducing network load. Malware scanning is organized through an asynchronous pipeline with a quarantine bucket, S3 events, and an SQS queue integrated with ClamAV. Experimental and load testing results confirm the effectiveness of the proposed solutions in terms of performance, security, and scalability. In particular, during a stress test scenario involving 10,000 upload requests, zero errors were recorded with an average response time of approximately 6 ms.
Keywords: cloud technologies, automated control, microservice architecture, AWS S3, RBAC, Cloud Drive, compute systems.

Постановка проблеми у загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями

Цифрова трансформація суспільства призвела до небувалого зростання обсягів даних. Приватні особи та організації генерують масиви цифрових ресурсів, забезпечення надійного зберігання яких та зручного доступу до них є критичним завданням. Хмарні технології, що поєднують розподілені обчислення та віртуалізацію, стали домінуючим підходом до організації відмовостійких сервісів зберігання.

Хоча ринок пропонує потужні рішення, такі як Google Drive або Dropbox, існує значний попит на приватні хмарні сховища, які забезпечують організаціям повний контроль над даними та відповідність специфічним політикам безпеки. Створення власної моделі хмарного сховища дозволяє нівелювати ризики, пов'язані з передачею даних третім сторонам, та адаптувати функціонал під конкретні бізнес-процеси.

Формулювання мети дослідження (постановка завдання)

Метою роботи є розроблення моделі, архітектури та програмна реалізація системи зберігання цифрових ресурсів, що забезпечує надійність, масштабованість, безпеку та сучасний користувацький досвід.

Аналіз останніх досліджень і публікацій, в яких започатковано розв'язання даної проблеми і на які спирається автор

Предметна область охоплює системи зберігання, де дані розташовані на віддалених серверах і доступні через мережу Інтернет. Головними викликами у цій сфері є забезпечення надійності (гарантія цілісності), масштабованості (здатність працювати при зростанні навантаження) та безпеки даних.

Аналіз існуючих рішень, таких як Google Drive, Dropbox, OneDrive та Nextcloud, дозволив виділити ключові архітектурні патерни (Таблиця 1).

Таблиця 1. Порівняння існуючих рішень хмарного зберігання / Table 1. Comparison of existing cloud storage solutions

Рішення	**Особливості та фокус**	**Архітектура (ключові риси)**
Google Drive	Інтеграція з Google Workspace; AI-пошук; спільне редагування	Розподілена інфраструктура (Colossus FS), глобальні БД (Spanner)
Dropbox	Надійна синхронізація; блокове зберігання	Власна система Magic Pocket (блокове зберігання, реплікація)
OneDrive	Інтеграція з Windows/Office 365	Базується на Azure Blob Storage та SharePoint
Nextcloud	Повний контроль (Self-hosted); Open-source	Класична веб-архітектура (PHP+SQL), підтримка зовнішніх сховищ

Більшість сучасних систем використовують об'єктні сховища або розподілені файлові системи для забезпечення надійності ("11 дев'яток" у Amazon S3). Важливим аспектом є відокремлення метаданих від безпосередньо вмісту файлів, що дозволяє оптимізувати пошук та структурування.

Виділення невирішених раніше частин загальної проблеми, котрим присвячується означена стаття

Сучасні дослідження демонструють еволюцію хмарних систем зберігання від монолітних архітектур до складних розподілених систем з використанням патернів мікросервісів [1,2]. Junfeng et al. [3] запропонували модель PGCE, що базується на частковій георепликації та співпраці хмарних і периферійних вузлів для зниження затримок у розподілених системах зберігання. Kontodimas et al. [4] розробили фреймворк оркестрації для безпечного розподіленого зберігання з балансуванням цілісності, довговічності та вартості через еразурне кодування. Дослідження продуктивності S3 API показують, що стандартні реалізації не завжди задовольняють вимоги високопродуктивних навантажень, проте оптимізовані клієнтські бібліотеки та попередньо підписані URL дозволяють досягти значних покращень [1,5]. Питання безпеки залишаються критичними: Gupta et al. [6] провели систематичний огляд технік захисту даних у хмарі, виділивши компроміси між криптографічним забезпеченням та продуктивністю. Alattab et al. [7] продемонстрували ефективність RBAC-моделей з рольовим шифруванням для контролю доступу до хмарного сховища. Tang & Li [8] запропонували багаторівневу модель безпеки, що поєднує шифрування ChaCha20, k-анонімізацію та розподілене зберігання на основі кодів Коші. Незважаючи на прогрес, існує потреба в інтегрованих рішеннях, що одночасно забезпечують високу продуктивність, надійність та безпеку при використанні мікросервісної архітектури [4,6].

Виклад основного матеріалу з повним обґрунтуванням отриманих результатів

Методи та засоби дослідження

Методологічну основу дослідження становить системний підхід до проєктування розподілених систем зберігання даних. Застосовано методи декомпозиції предметної області, об'єктно-орієнтованого аналізу та архітектурного моделювання з використанням нотації C4 (Context, Container, Component, Code). Архітектурні рішення базуються на принципах Domain-Driven Design та мікросервісних патернах, що забезпечують слабку зв'язність компонентів і високу

когезію бізнес-логіки.

Для реалізації серверної частини обрано платформу Spring Boot 3 (Java 17), що надає готові абстракції для побудови RESTful API (Spring Web), механізмів автентифікації та авторизації (Spring Security), роботи з реляційними базами даних через об'єктно-реляційне відображення (Spring Data JPA), а також реактивного програмування для асинхронної обробки подій (Spring WebFlux). Клієнтська частина реалізована з використанням бібліотеки React 18 та мови TypeScript, що забезпечує типобезпечність та підтримку сучасних патернів розробки односторінкових застосунків (SPA).

Як реляційну систему управління базами даних для зберігання метаінформації про користувачів, файлову структуру та права доступу обрано PostgreSQL, що відзначається підтримкою ACID-властивостей та розширеними можливостями індексування. Фізичне зберігання файлів делеговано об'єктному сховищу Amazon S3, що гарантує довговічність даних на рівні 99.999999999% завдяки внутрішнім механізмам реплікації та еразурного кодування. Асинхронна взаємодія між компонентами системи організована через чергу повідомлень Amazon SQS, що забезпечує надійну доставку подій для тригерів обробки завантажених файлів. Для кешування списків відкликаних токенів доступу використано in-memory сховище Redis. Автентифікація реалізована на основі стандарту JSON Web Tokens (JWT) з роздільним використанням токенів доступу та оновлення. Антивірусна перевірка завантажених файлів здійснюється засобами ClamAV через модель "карантинного" бакету з подальшим переміщенням верифікованих об'єктів до продуктивного сховища.

Архітектура та модель системи

Архітектура системи «Cloud Drive» побудована за принципами мікросервісного підходу, що передбачає декомпозицію функціональності на слабко зв'язані, незалежно розгортувані сервіси (Рисунок 1). Така організація забезпечує горизонтальну масштабованість окремих компонентів залежно від навантаження, спрощує супровід та дозволяє застосовувати гетерогенні технологічні стеки для різних підсистем.

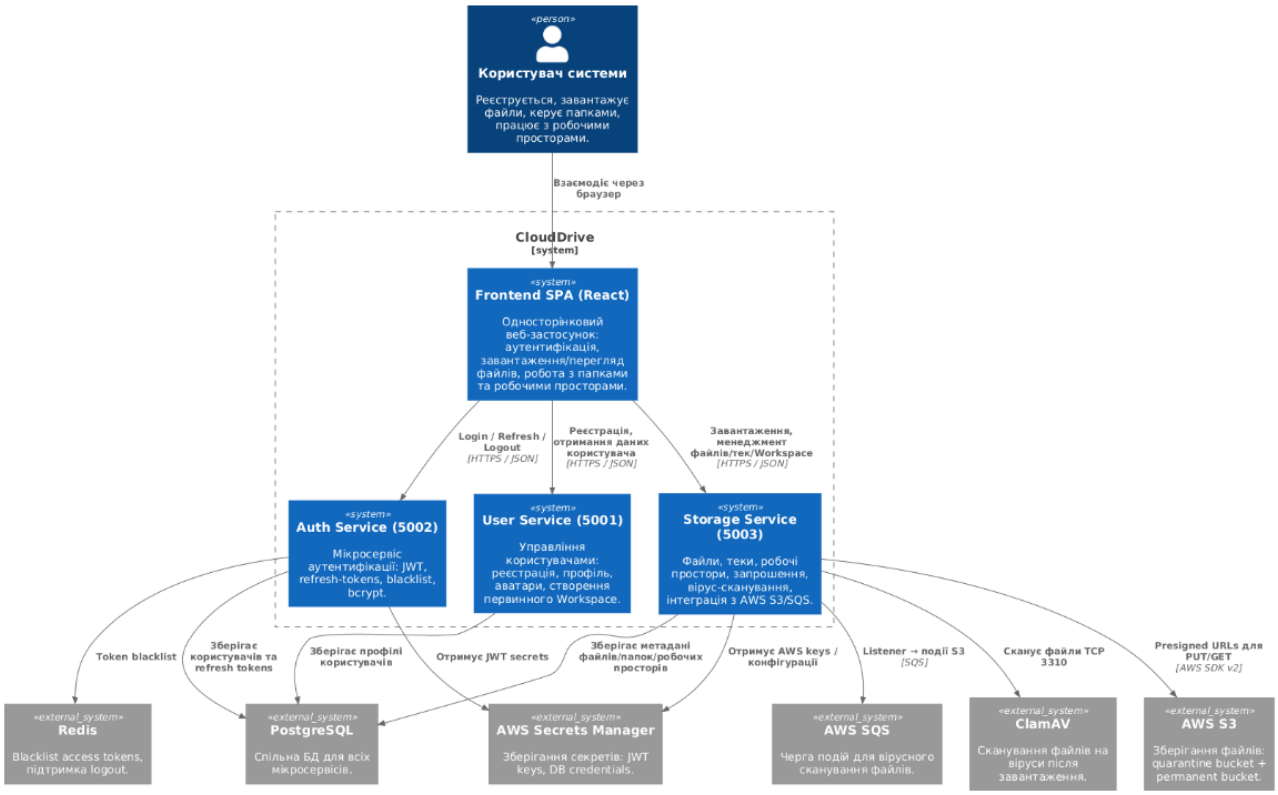

Рис. 1. Контекстна діаграма системи (C4 Context)

Система складається з трьох основних мікросервісів. Auth Service відповідає за управління життєвим циклом ідентифікаційних сутностей: реєстрацію користувачів, автентифікацію через перевірку облікових даних, генерацію пари JWT-токенів (access і refresh) та підтримку механізму

відкликання токенів через Redis-based blacklist. Цей підхід дозволяє реалізувати безпечний вихід користувача з системи навіть за умови stateless-природи JWT. User Service інкапсулює бізнес-логіку управління профілями користувачів та ініціалізації робочих просторів (Workspaces) — логічних контейнерів для організації файлової структури з можливістю колаборації. Storage Service є центральним компонентом, що обробляє операції створення, читання, оновлення та видалення файлів і папок (CRUD), забезпечує рольову модель контролю доступу (RBAC) з гранулярними правами (Власник, Редактор, Глядач) та координує взаємодію з об'єктним сховищем AWS S3 через механізм попередньо підписаних URL (Presigned URLs). Останній дозволяє делегувати операції передачі даних безпосередньо між клієнтом та S3, мінімізуючи навантаження на серверну інфраструктуру та усуваючи вузьке місце мережевого вводу-виводу [1,5]. Міжсервісна комунікація здійснюється через синхронні RESTful API з автентифікацією на основі JWT, що забезпечує єдину модель безпеки в розподіленому середовищі.

Модель даних та організація зберігання

Організація зберігання в системі «Cloud Drive» базується на гібридному підході, що поєднує реляційну модель для метаданих та об'єктне сховище для бінарного вмісту файлів (Рисунок 2). Таке розділення відповідає сучасним практикам побудови масштабованих систем зберігання [3,9] і дозволяє незалежно оптимізувати швидкість пошуку структурованої інформації та надійність зберігання великих об'єктів.

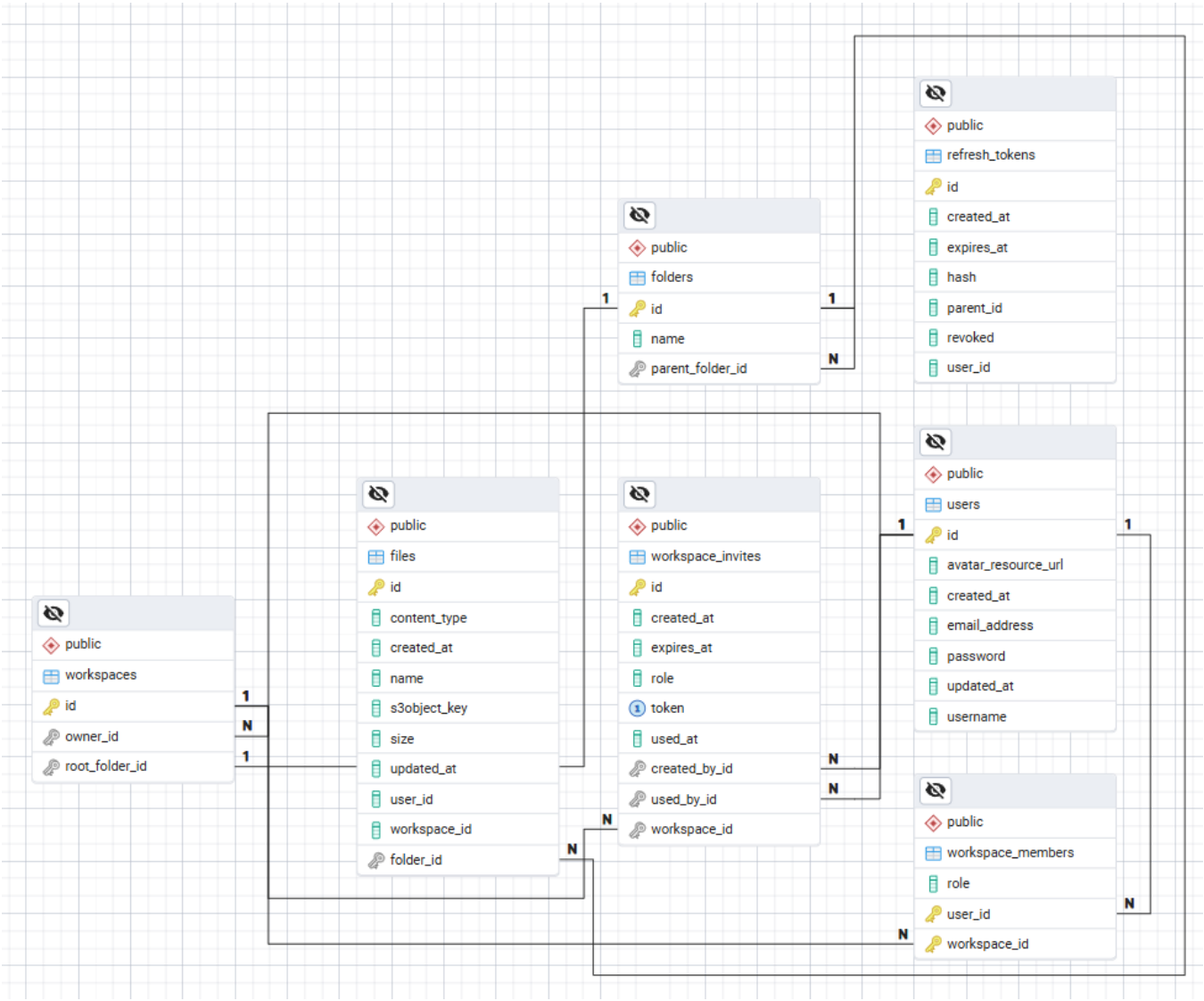

Рис. 2. ER-діаграма бази даних системи

Реляційна схема PostgreSQL включає сутності для користувачів (users), робочих просторів

(workspaces), папок (folders), файлів (files) та прав доступу (workspace_members, folder_permissions, file_permissions). Таблиця files зберігає метаатрибути: ім'я файлу, MIME-тип, розмір, контрольну суму (для перевірки цілісності), часові мітки створення та модифікації, статус антивірусної перевірки, а також критичне поле s3_object_key — унікальний ідентифікатор об'єкта в сховищі S3. Таблиця folders реалізує ієрархічну структуру каталогів через самопосилання (parent_folder_id), що дозволяє рекурсивно обходити дерево папок засобами SQL (Common Table Expressions). Права доступу моделюються через асоціативні таблиці з полями role (enum: OWNER, EDITOR, VIEWER), що забезпечує гранулярний контроль на рівні робочих просторів, папок та окремих файлів. Фізичне зберігання делеговано S3, де кожен файл ідентифікується через s3_object_key у форматі {workspace_id}/{folder_path}/{file_uuid}, що забезпечує логічну ізоляцію даних різних робочих просторів та уникнення колізій імен. Така архітектура дозволяє виконувати швидкі запити до метаданих (пошук за іменем, фільтрація за типом, сортування за датою) без необхідності звертання до об'єктного сховища, а також забезпечує масштабованість зберігання завдяки горизонтальному партиціонуванню S3.

Алгоритм безпечного завантаження даних

Процес завантаження файлів до системи організовано за схемою прямої передачі (direct upload) з використанням попередньо підписаних URL, що генеруються сервером на основі тимчасових облікових даних AWS STS (Security Token Service). Цей підхід дозволяє уникнути проксіювання бінарних даних через серверну інфраструктуру, знижуючи навантаження на мережеві та обчислювальні ресурси backend-сервісів [1,5]. Діаграма послідовності процесу наведена на Рисунку 3.

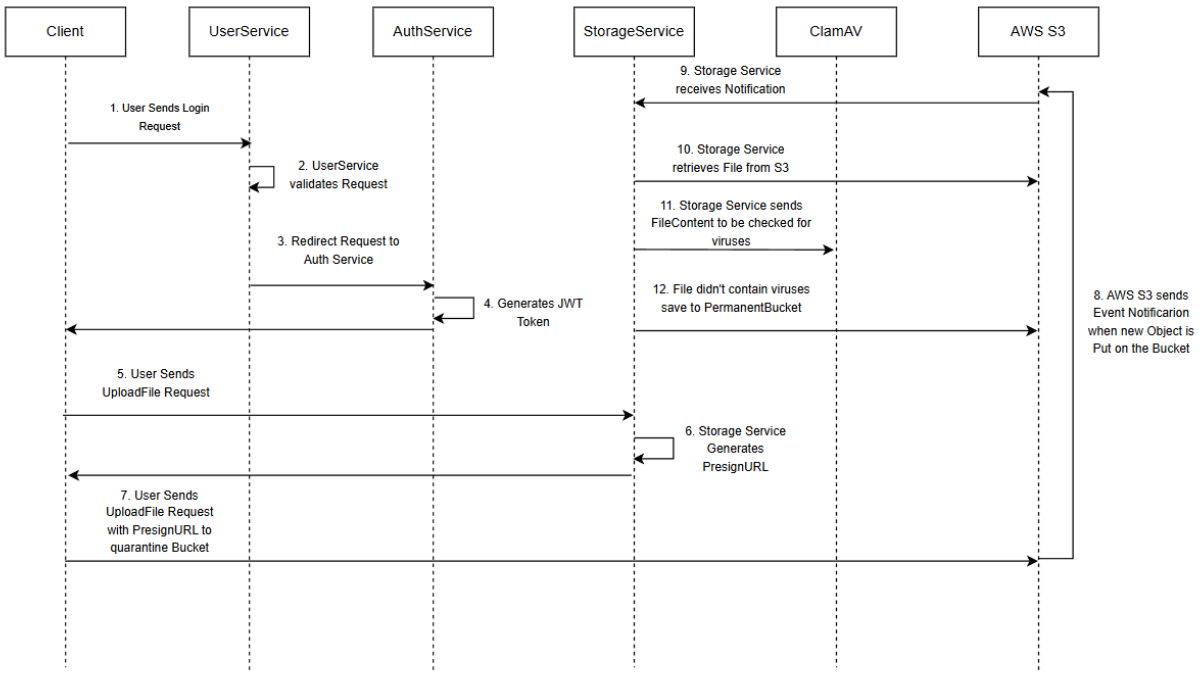

Рис. 3. Діаграма послідовності процесу завантаження файлу

Алгоритм складається з наступних етапів. На першому кроці клієнтський застосунок надсилає HTTP POST запит до Storage Service з метаданими файлу (ім'я, розмір, MIME-тип) та ідентифікатором цільової папки. Сервіс перевіряє права доступу користувача до вказаної папки згідно з RBAC-політиками. У разі успішної авторизації на другому кроці генерується унікальний ключ об'єкта S3 та створюється запис у таблиці files зі статусом PENDING. Далі Storage Service викликає AWS SDK для отримання presigned URL з обмеженим терміном дії (типово 15 хвилин) та правами PUT для запису в "карантинний" бакет (quarantine bucket). Цей URL повертається клієнту у відповіді. На третьому кроці клієнт виконує HTTP PUT запит безпосередньо до S3 за отриманим URL, передаючи бінарний вміст файлу. S3 приймає дані, зберігає їх у карантинному бакеті та генерує подію s3:ObjectCreated:Put. На четвертому кроці ця подія через налаштований Event Notification надходить до черги Amazon SQS, яку слухає асинхронний обробник (worker) з

інтегрованим антивірусним сканером ClamAV. Worker завантажує файл з карантинного бакету, виконує сканування та, у разі відсутності загроз, переміщує об'єкт до продуктивного бакету (production bucket) операцією S3 CopyObject з подальшим видаленням з карантину. Паралельно оновлюється статус файлу в базі даних на AVAILABLE та фіксується фінальний s3_object_key у продуктивному бакеті. У випадку виявлення шкідливого вмісту файл залишається в карантині, статус встановлюється в INFECTED, а користувач отримує повідомлення про блокування. Такий пайплайн забезпечує безпеку зберігання без затримки інтерфейсу користувача, оскільки перевірка виконується асинхронно [8].

Результати реалізації та експериментальні дослідження

В результаті роботи створено повнофункціональний прототип системи «Cloud Drive». Інтерфейс користувача (Рисунок 4) реалізовано у вигляді SPA, що забезпечує швидку реакцію на дії користувача.

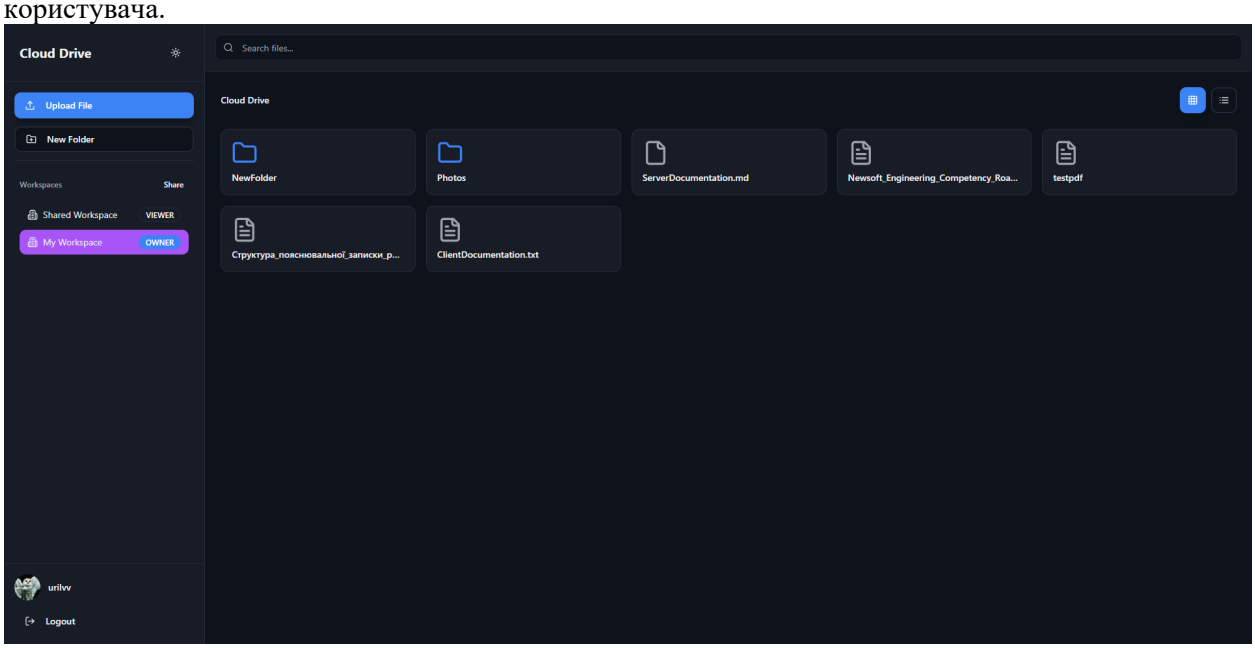

Рис. 4. Головний екран веб-застосунку «Cloud Drive»

Система підтримує створення робочих просторів, рольову модель (Власник/Редактор/Глядач), попередній перегляд файлів та контекстний пошук.

Аналіз продуктивності

Для перевірки ефективності запропонованих архітектурних рішень було проведено комплексне навантажувальне тестування із використанням інструменту Postman, який дозволяє моделювати інтенсивні сценарії взаємодії клієнтів із серверною частиною системи. Зокрема, реалізовано сценарій Stress Test для операцій завантаження файлів (Рисунок 5), орієнтований на оцінювання стабільності сервісу, швидкодії обробки запитів та поведінки системи в умовах пікового навантаження. У межах експерименту було виконано 10 тисяч послідовних і паралельних HTTP-запитів на завантаження об'єктів різного розміру. Зафіксовано нульовий рівень помилок, що свідчить про відмовостійкість реалізованої архітектури та коректність обробки транзакцій. Середній час відповіді склав близько 6 мс, що пояснюється делегуванням ресурсоємної операції передачі бінарних даних безпосередньо інфраструктурі AWS S3 через механізм попередньо підписаних URL. Такий підхід дозволив мінімізувати навантаження на backend-компоненти та усунути вузькі місця мережевого вводу-виводу. Система продемонструвала здатність до горизонтального масштабування, стабільно обробляючи паралельні завантаження без деградації продуктивності. Додатково проведене тестування безпеки підтвердило надійність механізмів автентифікації та авторизації: спроби доступу з невалідними JWT-токенами або недостатніми правами доступу коректно відхилялися сервером із поверненням статусу 403 Forbidden.

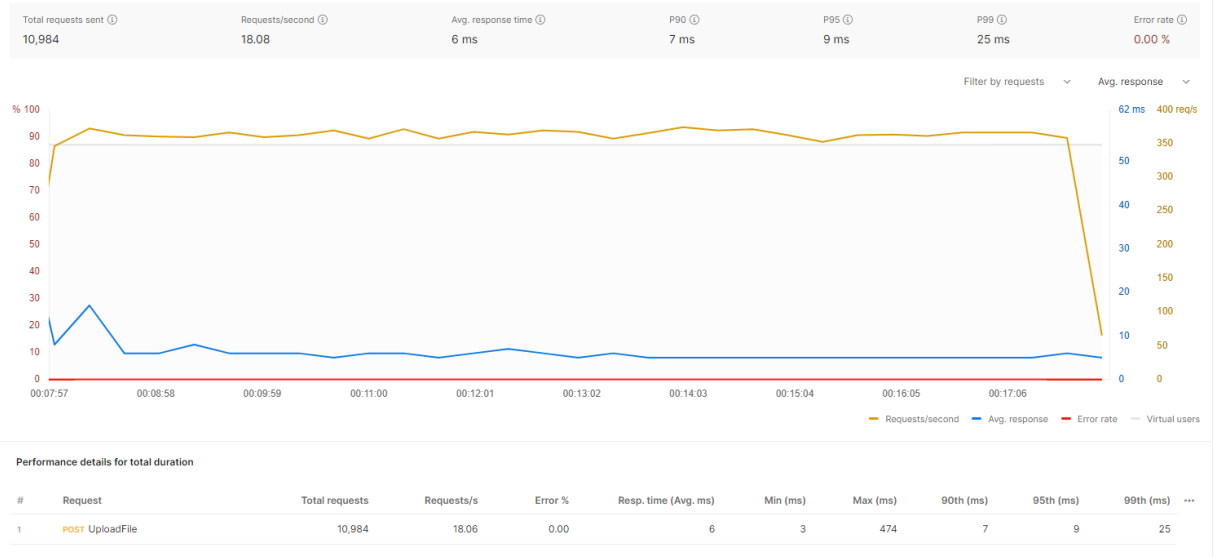

Рис. 5. Результати навантажувального тестування (Stress Test) завантаження файлів

Обговорення результатів

Отримані результати підтверджують ефективність запропонованого підходу до побудови хмарних систем зберігання на основі мікросервісної архітектури та об'єктного сховища AWS S3. Експериментальні дослідження продемонстрували, що використання механізму presigned URL дозволяє досягти середнього часу відповіді на рівні 6 мс при обробці 10 000 паралельних запитів завантаження без жодної помилки, що узгоджується з результатами [1] щодо оптимізації S3 API через делегування операцій передачі даних. Порівняно з традиційними підходами, де бінарні дані проксіюються через серверну інфраструктуру, запропонована архітектура усуває вузьке місце мережевого I/O та дозволяє горизонтально масштабувати обчислювальні компоненти незалежно від обсягу переданих даних [5].

Наукова новизна роботи полягає в інтеграції трьох архітектурних патернів у єдину систему: (1) формуванні мікросервісної архітектурної основи, що передбачає ізоляцію сервісів Auth, User та Storage для досягнення незалежного масштабування та диференційованого забезпечення показників продуктивності; (2) у розробленні гібридної моделі зберігання з розділенням метаданих (PostgreSQL) та бінарного вмісту (S3), що забезпечує швидкий пошук структурованої інформації при збереженні високої надійності об'єктного сховища [3]; (3) вдосконаленні асинхронного пайплайну безпеки з карантинним сховищем та інтеграцією ClamAV через SQS, що дозволяє верифікувати файли без блокування користувацького інтерфейсу [8]. На відміну від існуючих рішень, таких як Nextcloud (монолітна PHP-архітектура з обмеженою масштабованістю) або публічних хмар (обмежений контроль над даними), запропонована система поєднує переваги self-hosted розгортання з хмарною інфраструктурою AWS, забезпечуючи повний контроль над метаданими при делегуванні відмовостійкого зберігання об'єктів провайдеру.

Реалізована RBAC-модель з гранулярними правами на рівні робочих просторів, папок та файлів відповідає сучасним вимогам до контролю доступу в багатокористувацьких системах [7] і дозволяє гнучко налаштовувати політики колаборації. Тестування безпеки підтвердило коректність механізму JWT з blacklist-підходом для відкликання токенів: спроби доступу з невалідними або відкликаними токенами, а також запити з недостатніми правами, коректно відхилялися сервером зі статусом 403 Forbidden. Використання Redis для кешування списків відкликаних токенів забезпечує швидку перевірку (O(1)) без навантаження на реляційну базу даних.

Обмеженням запропонованого рішення є залежність від стабільного інтернет-з'єднання для доступу до даних, що характерно для всіх хмарних систем. Проте, як показують дослідження Barron et al. [2], впровадження ієрархічного кешування на рівні клієнта або периферійних вузлів може пом'якшити цю проблему для часто використовуваних файлів. Іншим аспектом є вартість зберігання в S3, яка зростає пропорційно обсягу даних, однак використання lifecycle policies для автоматичного переміщення рідко доступних об'єктів до класів зберігання S3 Glacier або Glacier Deep Archive дозволяє суттєво знизити експлуатаційні витрати при збереженні довговічності даних.

Практична цінність роботи полягає у можливості використання розробленої системи як

референсної архітектури для побудови приватних корпоративних сховищ, що поєднують зручність публічних хмар (масштабованість, надійність) з повним контролем над метаданими та політиками безпеки. Відкритість використаних технологій (Spring Boot, React, PostgreSQL) та документованість архітектурних рішень дозволяють адаптувати систему під специфічні вимоги організацій різних галузей. Подальші дослідження можуть бути спрямовані на інтеграцію механізмів end-to-end шифрування на стороні клієнта [6], впровадження інтелектуального пошуку з використанням векторних баз даних для семантичного аналізу вмісту файлів, а також оптимізацію витрат через динамічне переміщення об'єктів між класами зберігання S3 на основі аналізу патернів доступу [4].

Висновки (включно з науковою новизною) та перспективи подальших досліджень

У роботі вирішено задачу створення моделі та програмних засобів для хмарного зберігання цифрових ресурсів. Розроблена архітектура, що базується на мікросервісах та використанні хмарної платформи AWS, довела свою ефективність під час експериментальних досліджень.

1. Запропоновано комплексну модель, яка поєднує об'єктне зберігання даних з реляційною структурою метаданих, що забезпечує гнучкість управління файлами та папками.
2. Реалізовано механізм безпечної взаємодії клієнта з хмарним сховищем через попередньо підписані URL, що значно підвищило продуктивність системи при роботі з великими файлами.
3. Впроваджена система контролю доступу на основі ролей (RBAC) та ізоляція файлів через карантинний бакет забезпечують високий рівень безпеки даних користувачів.

Практична цінність роботи полягає у можливості використання розробленого рішення як основи для розгортання приватних корпоративних сховищ, що поєднують зручність публічних хмар з повним контролем над даними.

Список бібліографічного опису

1. Gadban, F., & Kunkel, J. M. (2021). Analyzing the performance of the S3 object storage API for HPC workloads. Applied Sciences, 11(18), 8540. https://doi.org/10.3390/APP11188540
2. Barron, A., Sanchez-Gallegos, D. D., Carrizales-Espinoza, D., Gonzalez-Compean, J. L., & Morales-Sandoval, M. (2022). On the efficient delivery and storage of IoT data in edge–fog–cloud environments. Sensors, 22(18), 7016. https://doi.org/10.3390/s22187016
3. Junfeng, T., Bai, W., & Jia, H. (2022). PGCE: A distributed storage causal consistency model based on partial geo-replication and cloud-edge collaboration architecture. Computer Networks, 213, 109065. https://doi.org/10.1016/j.comnet.2022.109065
4. Kontodimas, K., Soumplis, P., Kretsis, A., Kokkinos, P., & Feher, M. (2023). Secure distributed storage orchestration on heterogeneous cloud-edge infrastructures. IEEE Transactions on Cloud Computing, 11(3), 2909–2924. https://doi.org/10.1109/tcc.2023.3287653
5. Lee, K., Kim, J., Kwak, J. R., & Kim, Y.-T. (2023). Dynamic multi-resource optimization for storage acceleration in cloud storage systems. IEEE Transactions on Services Computing, 16(3), 2058–2071. https://doi.org/10.1109/TSC.2022.3173333
6. Gupta, I., Singh, A. K., Lee, C.-N., & Buyya, R. (2022). Secure data storage and sharing techniques for data protection in cloud environments: A systematic review, analysis, and future directions. IEEE Access, 10, 71247–71277. https://doi.org/10.1109/access.2022.3188110
7. Alattab, A. A., Irshad, R. R., Yahya, A. A., & Al-Awady, A. (2022). Privacy protected preservation of electric vehicles' data in cloud computing using secure data access control. Energies, 15(21), 8085. https://doi.org/10.3390/en15218085
8. Tang, T., & Li, M. (2025). Enhanced secure storage and data privacy management system for big data based on multilayer model. Scientific Reports, 15(1), 2345. https://doi.org/10.1038/s41598-025-16624-y
9. Chikhaoui, A., Lemarchand, L., Boukhalfa, K., & Boukhobza, J. (2021). Multi-objective optimization of data placement in a storage-as-a-service federated cloud. ACM Transactions on Storage, 17(3), Article 19. https://doi.org/10.1145/3452741

References

1. Gadban, F., & Kunkel, J. M. (2021). Analyzing the performance of the S3 object storage API for HPC workloads. Applied Sciences, 11(18), 8540. https://doi.org/10.3390/APP11188540
2. Barron, A., Sanchez-Gallegos, D. D., Carrizales-Espinoza, D., Gonzalez-Compean, J. L., & Morales-Sandoval, M. (2022). On the efficient delivery and storage of IoT data in edge–fog–cloud environments. Sensors, 22(18), 7016. https://doi.org/10.3390/s22187016
3. Junfeng, T., Bai, W., & Jia, H. (2022). PGCE: A distributed storage causal consistency model based on partial geo-replication and cloud-edge collaboration architecture. Computer Networks, 213, 109065. https://doi.org/10.1016/j.comnet.2022.109065
4. Kontodimas, K., Soumplis, P., Kretsis, A., Kokkinos, P., & Feher, M. (2023). Secure distributed storage orchestration on heterogeneous cloud-edge infrastructures. IEEE Transactions on Cloud Computing, 11(3), 2909–2924. https://doi.org/10.1109/tcc.2023.3287653
5. Lee, K., Kim, J., Kwak, J. R., & Kim, Y.-T. (2023). Dynamic multi-resource optimization for storage acceleration in cloud storage systems. IEEE Transactions on Services Computing, 16(3), 2058–2071.

https://doi.org/10.1109/TSC.2022.3173333
6. Gupta, I., Singh, A. K., Lee, C.-N., & Buyya, R. (2022). Secure data storage and sharing techniques for data protection in cloud environments: A systematic review, analysis, and future directions. IEEE Access, 10, 71247–71277. https://doi.org/10.1109/access.2022.3188110
7. Alattab, A. A., Irshad, R. R., Yahya, A. A., & Al-Awady, A. (2022). Privacy protected preservation of electric vehicles' data in cloud computing using secure data access control. Energies, 15(21), 8085. https://doi.org/10.3390/en15218085
8. Tang, T., & Li, M. (2025). Enhanced secure storage and data privacy management system for big data based on multilayer model. Scientific Reports, 15(1), 2345. https://doi.org/10.1038/s41598-025-16624-y
9. Chikhaoui, A., Lemarchand, L., Boukhalfa, K., & Boukhobza, J. (2021). Multi-objective optimization of data placement in a storage-as-a-service federated cloud. ACM Transactions on Storage, 17(3), Article 19. https://doi.org/10.1145/3452741

Історія статті:
Отримано: 10.02.2026 Доопрацьовано: 13.03.2026 Прийнято до друку: 23.03.2026 Опубліковано: 29.03.2026