Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Магістр
(назва освітнього ступеня)

на тему: Оптимізація алгоритмів інтелектуального сортування даних на основі контентного аналізу з використанням Google Drive API

Виконав: студент VI курсу, групи СНнм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Бойко Д.А.
(підпис) (прізвище та ініціали)

Керівник _____ Никитюк В.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Дуда О.М
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«____» _____ 202_ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Бойку Данилу Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Оптимізація алгоритмів інтелектуального сортування даних на основі контентного аналізу з використанням Google Drive API

Керівник роботи _____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «10» березня 2026 року № 4/9-150

2. Термін подання студентом завершеної роботи 18.05.2026р.

3. Вихідні дані до роботи Науково-технічні публікації (друковані та розміщені в Інтернеті)

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз підходів до організації та інтелектуального сортування даних у хмарних системах. 1.1 Особливості зберігання та управління даними у хмарних середовищах. 1.2 Аналіз можливостей Google Drive та його API. 1.3 Підходи до організації резервного копіювання даних. 1.4 Вимоги до інформаційних систем керування резервними копіями. 2 Проєктування інфор-інформаційної системи інтелектуального сорту вання даних. 2.1 Архітектурні підходи до побудови вебдодатків. 2.2 Обґрунтування вибору архітектури. 2.3 Вибір технологічного стеку. 2.4 Проєктування загальної архітектури системи Back AppDrive. 2.5 Моделювання бази даних та структури збереження інформації. 2.6 Проєктування алгоритмів роботи системи. 3 Реалізація вебдодатку BackApp Drive та інтеграція з Google Drive. 3.1 Загальна архітектура реалізованої системи. 3.2 Реалізація серверної частини додатку. 3.3 Реалізація взаємодії з Google Drive API. 3.4 Реалізація функціональних модулів системи. 3.5 Реалізація інтерфейсу користувача. 3.6 Реалізація особистого кабінету користувача 3.7 Реалізація системи безпеки та тестування додатку. 4 Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульна сторінка. 2 Актуальність роботи. 3 Мета і задачі дослідження. 4 Наукова новизна. 5 Характеристика вхідних даних та метаданих. 6 Архітектура системи та сервісний шар. 7 Реалізація серверної частини. 8 Інтеграція з Google Drive API. 9 Ефективність алгоритмів контентного аналізу. 10 Користувацький інтерфейс. 11 Система безпеки та оптимізація сховища. 12 Кабінет користувача та резервне копіювання. 13 Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент каф. МТ		
Безпека в надзвичайних ситуаціях	Теслюк В.М., проректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 13.04.2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	13.04.2026	
2.	Підбір та опрацювання наукових публікацій, збір	13.04.2026-20.04.2026	
	даних по темі роботи		
3.	Виконання дослідження згідно теми кваліфікаційної	21.04.2026-03.05.2026	
	Роботи		
4.	Оформлення розділу «Аналіз основних підходів	04.05.2026-10.05.2026	
	до організації та інтелектуального сортування даних»		
5.	Оформлення розділу «Аналіз методів та підходів до	04.05.2026-10.05.2026	
	проєктування систем інтелектуального сортування		
	даних»		
6.	Оформлення розділу «Реалізація веб-додатку	04.05.2026-10.05.2026	
	BackAppDrive»		
7.	Виконання завдання до підрозділу «Охорона праці»	27.04.2026-10.05.2026	
8.	Виконання завдання до підрозділу «Безпека в	27.04.2026-10.05.2026	
	надзвичайних ситуаціях»		
9.	Оформлення кваліфікаційної роботи	11.05.2026-13.05.2026	
10.	Нормоконтроль	14.05.2026	
11.	Перевірка на плагіат	18.05.2026	
12.	Попередній захист кваліфікаційної роботи	19.05.2026	
13.	Захист кваліфікаційної роботи	26.05.2026	

Студент _____ Бойко Д.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Никитюк В.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Оптимізація алгоритмів інтелектуального сортування даних на основі контентного аналізу з використанням Google Drive API. // Кваліфікаційна робота освітнього рівня «Магістр» // Бойко Данило Андрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНнм-61 // Тернопіль, 2026 // С. 82, рис. – 17, табл. – 3, додат. – 1, бібліогр. – 52

Ключові слова: резервне копіювання, хмарне сховище, Google Drive API, інтелектуальне сортування даних, вебдодаток, контентний аналіз, OAuth 2.0, інформаційна система.

Кваліфікаційна робота присвячена розв'язанню задачі розробки вебдодатку для керування резервними копіями даних із інтеграцією Google Drive API. Об'єктом дослідження є оптимізація резервного копіювання та зберігання даних у хмарному середовищі, а предметом дослідження — методи та засоби реалізації вебдодатку для керування цими резервними копіями. В першому розділі описані теоретичні основи хмарних сервісів, резервного копіювання та методів сортування й обробки даних. В другому розділі наведено вимоги, архітектуру та структуру системи, а також описано алгоритми роботи й механізми взаємодії з Google Drive API. В третьому розділі описано реалізацію основних модулів вебдодатку, проаналізовано архітектурні рішення та їх ефективність, наведено результати реалізації серверної частини, інтеграції з Google Drive API, Dashboard та системи безпеки.

ANNOTATION

Optimization of Intelligent Data Sorting Algorithms Based on Content Analysis Using Google Drive API // Qualification Thesis for the Master's Degree // Danylo Andriiovych Boiko // Ivan Puluj Ternopil National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, Group СНнм-61 // Ternopil, 2026 // P. 82, fig. – 17, tabl. – 3, append. – 1, bibliogr. – 52.

Keywords: backup system, cloud storage, Google Drive API, intelligent data sorting, web application, content analysis, OAuth 2.0, information system.

The qualification thesis is dedicated to solving the problem of developing a web application for managing data backups with Google Drive API integration. The object of the research is the optimization of data backup and storage in a cloud environment, while the subject of the research encompasses the methods and tools for implementing a web application to manage these backups. The first chapter describes the theoretical foundations of cloud services, data backup, and methods of data sorting and processing. The second chapter outlines the system's requirements, architecture, and structure, alongside the algorithms and interaction mechanisms with the Google Drive API. The third chapter details the implementation of the main web application modules, analyzes the architectural solutions and their efficiency, and presents the results of the server-side implementation, Google Drive API integration, Dashboard, and security system.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface (інтерфейс програмування застосунків).

AJAX – Asynchronous JavaScript and XML (асинхронна взаємодія клієнта із сервером).

CRUD – Create, Read, Update, Delete (базові операції роботи з даними).

CSRF – Cross-Site Request Forgery (підробка міжсайтових запитів).

CSS – Cascading Style Sheets (каскадні таблиці стилів).

HTML – HyperText Markup Language (мова розмітки гіпертексту).

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту).

HTTPS – HyperText Transfer Protocol Secure (захищений протокол передачі даних).

JSON – JavaScript Object Notation (текстовий формат обміну даними).

MVC – Model–View–Controller (архітектурний шаблон побудови вебдодатків).

MIME – Multipurpose Internet Mail Extensions (тип даних файлу).

MySQL – My Structured Query Language (система керування базами даних).

NLP – Natural Language Processing (обробка природної мови).

OAuth – Open Authorization (протокол авторизації).

PDO – PHP Data Objects (інтерфейс доступу до баз даних у PHP).

PHP – PHP: Hypertext Preprocessor (серверна мова програмування).

REST – Representational State Transfer (архітектурний стиль вебсервісів).

SQL – Structured Query Language (мова структурованих запитів).

TF-IDF – Term Frequency–Inverse Document Frequency (метод оцінки важливості слів у тексті).

UI – User Interface (інтерфейс користувача).

URL – Uniform Resource Locator (уніфікований локатор ресурсу).

UX – User Experience (користувацький досвід).

XML – eXtensible Markup Language (розширювана мова розмітки).

БД – база даних.

ІС – інформаційна система.

ПЗ – програмне забезпечення.

СУБД – система управління базами даних.

ШІ – штучний інтелект.

ЕЦП – електронний цифровий підпис.

ООП – об’єктно-орієнтоване програмування.

ТЗ – технічне завдання.

ПП – програмний продукт.

АС – автоматизована система.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних умовах стрімкого зростання обсягів цифрових даних особливої актуальності набуває проблема їх ефективного зберігання, обробки та впорядкування. Хмарні сервіси, зокрема Google Drive, стали основним інструментом для роботи з файлами як для індивідуальних користувачів, так і для організацій. Проте зростання кількості даних призводить до ускладнення їх структурування, пошуку та керування.

Одним із ключових напрямів розвитку інформаційних систем є впровадження інтелектуальних методів обробки контенту, що дозволяють автоматизувати процес сортування, класифікації та управління файлами. Особливо важливим є використання підходів, заснованих на аналізі вмісту файлів, метаданих та поведінки користувачів. Такі системи здатні значно підвищити ефективність роботи з даними та зменшити навантаження на користувача.

Разом з тим, стандартні інструменти, що надаються хмарними сервісами, мають обмежені можливості щодо інтелектуального сортування та автоматизованого керування файлами. Це створює необхідність у розробці спеціалізованих рішень, які інтегруються з API сторонніх сервісів і розширюють їх функціональність.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Магістр» є оптимізація алгоритмів з елементами розробки інформаційної системи інтелектуального сортування даних із інтеграцією в Google Drive API, що забезпечує ефективне управління резервними копіями та автоматизоване впорядкування файлів на основі їхнього вмісту. Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати сучасні підходи до організації хмарних сховищ та методів управління даними;

– дослідити можливості інтеграції з Google Drive API та особливості роботи з OAuth 2.0;

– спроєктувати архітектуру інформаційної системи з використанням підходу MVC;

– реалізувати вебдодаток для керування резервними копіями з підтримкою авторизації та інтеграції з Google Drive.

Об’єкт дослідження: процеси організації, резервного копіювання та управління файлами у хмарно-орієнтованих інформаційних системах.

Предмет дослідження: методи інтеграції вебдодатків із Google Drive API, а також алгоритми класифікації та інтелектуального впорядкування файлів на основі їх вмісту й метаданих.

Наукова новизна одержаних в кваліфікаційній роботі результатів є оптимізацією підходу до організації систем резервного копіювання шляхом поєднання локального та хмарного зберігання даних із автоматизованим сортуванням файлів на основі аналізу їх типу

Практичне значення одержаних результатів. Вебдодаток BackupDriveApp призначений для користувачів, які працюють із великими обсягами цифрових даних та потребують надійного середовища для резервного копіювання і зберігання файлів. Система орієнтована на використання у персональних, навчальних та малих корпоративних середовищах, де важливо забезпечити швидкий доступ до резервних копій, їх централізоване управління та синхронізацію з хмарним сховищем Google Drive.

Апробація результатів магістерської роботи. Основні результати проведенихдосл на XIII науково-технічній «Актуальні задачі сучасних технологій» та XII міжнародній студентській науково-практичній науковій конференції молодих учених та студентів «Інформаційні моделі, системи та технології». Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 2025 р.).

Публікації. Основні результати кваліфікаційної роботи опубліковано у двох працях конференції (додатки А та Б).

Структура й обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури з 52 найменувань та 1 додатку. Загальний обсяг кваліфікаційної роботи складає 82 сторінки, з них 63 сторінки основного тексту, який містить 17 рисунків та 3 таблицю.

1 АНАЛІЗ ПІДХОДІВ ДО ОРГАНІЗАЦІЇ ТА ІНТЕЛЕКТУАЛЬНОГО СОРТУВАННЯ ДАНИХ У ХМАРНИХ СИСТЕМАХ

1.1 Особливості зберігання та управління даними у хмарних середовищах

Сучасне інформаційне середовище характеризується стрімким зростанням обсягів цифрових даних, що зберігаються у хмарних сервісах. Особливої популярності набувають платформи, такі як Google Drive, які забезпечують доступ до файлів з будь-якого пристрою та дозволяють організовувати спільну роботу з даними. У зв’язку з цим виникає потреба у створенні ефективних інструментів для управління, структурування та швидкого доступу до інформації.

Зі збільшенням кількості файлів користувачі стикаються з проблемою їх впорядкування та пошуку. Традиційні підходи, що базуються на ручному сортуванні або простому використанні папок, стають малоефективними, особливо при роботі з великими обсягами різнорідних даних. Відсутність автоматизованих механізмів аналізу вмісту файлів призводить до втрати часу, зниження продуктивності та підвищення ризику втрати важливої інформації.

У відповідь на ці виклики зростає роль інтелектуальних систем, здатних автоматично класифікувати та сортувати дані на основі їхнього контенту, метаданих та контексту використання. Такі системи можуть враховувати тип файлу, його структуру, ключові слова, а також історію взаємодії користувача з даними. Це дозволяє значно покращити процес організації інформації та забезпечити швидкий доступ до необхідних ресурсів.

Особливістю систем, інтегрованих із хмарними сервісами, є необхідність взаємодії з зовнішніми API. Зокрема, використання Google Drive API передбачає роботу з механізмами авторизації (OAuth 2.0), управління файлами, обмеженнями доступу та квотами. Це накладає додаткові вимоги до архітектури системи, її безпеки та стабільності [10].

Сучасні користувачі очікують, що система управління даними буде не лише функціональною, але й інтуїтивно зрозумілою, швидкою та безпечною. Важливими характеристиками є автоматизація процесів резервного копіювання, можливість відновлення даних, контроль доступу та прозорість виконуваних операцій.

Такі платформи, як Google Drive, дозволяють користувачам працювати з файлами незалежно від місця перебування та використовуваного пристрою. Проте ефективність використ–ання таких сервісів значною мірою залежить від якості організації та управління даними [11].

Зі збільшенням кількості файлів у хмарному сховищі виникає проблема їх швидкого пошуку, класифікації та доступу. Якщо система не забезпечує належного рівня впорядкування або інтелектуального сортування, користувач витрачає значний час на пошук необхідної інформації. Це призводить до зниження продуктивності роботи та негативно впливає на загальний досвід взаємодії з системою [12].

Якісна система управління даними, що включає автоматизоване сортування, фільтрацію та аналіз контенту, значно покращує взаємодію користувача з хмарним сервісом. Зокрема, вона дозволяє швидко знаходити необхідні файли, зменшує когнітивне навантаження та підвищує ефективність роботи. У контексті резервного копіювання це також означає швидке відновлення даних та мінімізацію ризику їх втрати [13].

Водночас відсутність логічної структури або автоматизованого сортування ускладнює навігацію та негативно впливає на продуктивність користувача.

Інтуїтивність та швидкість доступу до даних є ключовими характеристиками ефективної інформаційної системи. Якщо користувач може швидко знайти потрібний файл, система сприймається як надійна та зручна. В іншому випадку виникає фрустрація, що може призвести до відмови від використання сервісу або переходу до альтернативних рішень.

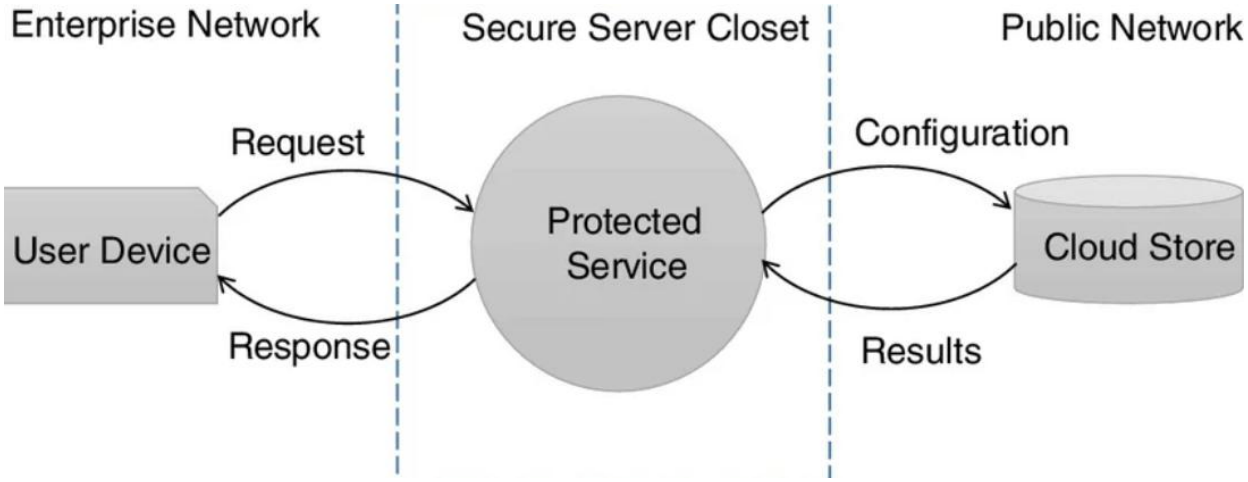

Рисунок 1.1 – Вплив організації даних на швидкість доступу та користувацький досвід

Особливо важливим є використання інтелектуальних підходів до сортування даних, які враховують не лише назву файлу, але й його вміст, тип, дату створення та історію використання. Це дозволяє значно підвищити точність доступу до інформації та скоротити час взаємодії з системою.

У цьому випадку ефективність системи визначається здатністю швидко та точно забезпечити доступ до потрібних даних. Зменшення часу пошуку та кількості дій безпосередньо впливає на продуктивність користувача.

Крім того, важливим показником є використання дискового простору, що відображає ефективність управління резервними копіями.

Ефективне управління даними також зменшує навантаження на користувача, оскільки автоматизує процеси резервного копіювання, сортування та відновлення. Це дозволяє користувачу зосередитися на основних завданнях, не витрачаючи час на ручну організацію файлів.

Якщо система не відповідає очікуванням користувачів, це може призвести до втрати довіри, зниження ефективності роботи та підвищення ризику втрати важливих даних. Водночас добре реалізована система управління даними підвищує рівень задоволеності користувачів, сприяє повторному використанню сервісу та формує позитивний досвід взаємодії.

Відсутність чіткої системи впорядкування ускладнює доступ до інформації та знижує ефективність роботи. Однією з основних проблем є

хаотичне накопичення даних. Користувачі часто зберігають файли без чіткої структури, використовуючи довільні назви або дублюючи інформацію. Це призводить до утворення великої кількості неорганізованих даних, серед яких важко швидко знайти необхідний файл. У результаті зростає час пошуку та підвищується ймовірність використання застарілої або некоректної інформації.

Ще однією важливою проблемою є відсутність автоматизованої класифікації даних. Більшість систем покладаються на ручне сортування за папками, що є неефективним при великій кількості файлів. Такий підхід не враховує вміст файлів, їхній контекст або взаємозв'язки між даними. У результаті користувач змушений витрачати додатковий час на організацію інформації, що негативно впливає на продуктивність [14].

Складність структурування також зростає через різноманітність форматів даних. У хмарних сховищах можуть зберігатися текстові документи, архіви, зображення, бази даних та інші типи файлів, кожен з яких потребує різних підходів до обробки та класифікації. Відсутність універсального механізму аналізу таких даних ускладнює їх ефективне впорядкування.

Окремою проблемою є дублювання інформації та нераціональне використання дискового простору. Користувачі часто створюють кілька копій одного й того ж файлу або зберігають різні версії без відповідного контролю. Це призводить до перевитрати ресурсів та ускладнює процес вибору актуальної версії документа [15].

Також важливим викликом є забезпечення швидкого доступу до даних. Зі збільшенням обсягів інформації традиційні методи пошуку стають менш ефективними, що призводить до затримок при обробці запитів. Це особливо критично у випадках, коли необхідно оперативно отримати доступ до резервних копій або важливих документів.

Крім того, необхідно враховувати аспекти безпеки та контролю доступу. У великих системах важливо забезпечити розмежування прав користувачів,

захист від несанкціонованого доступу та цілісність даних. Неправильна організація цих механізмів може призвести до втрати або витоку інформації.

1.2 Аналіз можливостей Google Drive та його API

Сервіс Google Drive є одним із найпоширеніших хмарних рішень для зберігання, синхронізації та спільного використання файлів. Його практична цінність для побудови інформаційних систем полягає в тому, що він надає не лише готове середовище для роботи з файлами, але й програмний інтерфейс доступу, який дозволяє автоматизувати основні операції над даними. Для розробки системи резервного копіювання та інтелектуального сортування це є важливою перевагою, оскільки дає змогу поєднати локальну логіку додатку з можливостями зовнішнього хмарного сховища [16].

Основним засобом інтеграції є Google Drive API, актуальною версією якого є v3. Цей API підтримує роботу з файлами, папками, змінами у сховищі, дозволами доступу, а також із метаданими об’єктів. Порівняно з попередньою версією, v3 орієнтована на кращу продуктивність, зокрема через повернення лише підмножини полів у відповідях, що зменшує надлишковий обсяг даних під час обміну між клієнтом і сервером [17].

Для розроблюваної системи особливо важливими є операції створення, завантаження, оновлення, отримання та видалення файлів. Google Drive API дозволяє створювати файли як без вмісту, так і з передачею метаданих і медіаданих, а також підтримує оновлення файлів із patch-семантикою. Це зручно для реалізації сценарію, у якому резервна копія спочатку формується локально, а далі передається до хмарного сховища з подальшим збереженням ідентифікатора файлу в базі даних системи.

Значною перевагою Google Drive API є підтримка пошуку та фільтрації файлів за метаданими. Метод files.list дає змогу отримувати інформацію про файли, сортувати результати, а також використовувати поля, пов’язані з назвою, часом зміни, розміром, ознаками доступу та іншими

характеристиками [17]. Для системи інтелектуального сортування це створює основу для поєднання контентного аналізу з уже наявними атрибутами файлу у хмарному середовищі.

Окрему роль відіграє механізм авторизації та розмежування доступу. Google Drive API використовує OAuth 2.0 і набір окремих scope-ів, що дозволяє реалізувати принцип мінімально необхідних привілеїв. Це є критично важливим для додатків, які працюють “від імені користувача”, оскільки доступ до його файлів повинен бути обмежений лише необхідними діями. Крім того, кожен файл, папка або shared drive мають пов’язані ресурси дозволів, що дає змогу реалізовувати контроль спільного доступу та безпечну взаємодію з даними.

Для задач резервного копіювання важливою є також можливість відстеження змін та оцінки наявних ресурсів сховища. Ресурс about дозволяє отримати відомості про квоти, зокрема загальний ліміт, використаний простір у Drive та інші пов’язані показники. У свою чергу, ресурс changes дає можливість відстежувати зміни для користувача або shared drive, що може бути корисним для синхронізації, актуалізації локальної інформації та побудови розширених механізмів моніторингу стану резервних копій.

Водночас Google Drive API має низку практичних обмежень, які потрібно враховувати при проєктуванні системи. Зокрема, існують ліміти використання API, а також обмеження на щоденне завантаження для користувачів Google Workspace: до 750 ГБ на день між My Drive і shared drives. Максимальний розмір одного завантажуваного файлу становить 5 ТБ. Такі обмеження не є критичними для типової навчальної або малої прикладної системи резервного копіювання, однак вони мають значення для масштабування та планування обробки великих обсягів даних [17].

З погляду розробки інформаційної системи BackupDriveApp, можливості Google Drive та його API є достатніми для реалізації базового і розширеного функціоналу. Сервіс забезпечує надійне зберігання файлів, доступ до метаданих, керування дозволами, контроль квот і інтеграцію через

стандартні механізми автентифікації. Це дозволяє використати Google Drive не лише як зовнішнє сховище, але і як повноцінний компонент архітектури системи резервного копіювання та інтелектуального впорядкування даних.

Таблиця 1.3 - Основні можливості Google Drive API для реалізації інформаційної системи

Можливість API	Практичне значення для проєкту
Створення та завантаження файлів	Дозволяє передавати резервні копії з локального сховища до Google Drive
Отримання та завантаження файлів	Дає змогу виконувати відновлення даних користувача
Оновлення метаданих файлів	Використовується для підтримки актуального стану об’єктів
Пошук і фільтрація через files.list	Дозволяє реалізувати сортування, вибірку та аналіз файлів
Робота з дозволами доступу	Забезпечує безпечну взаємодію з файлами користувача
Отримання квот через about	Дає змогу контролювати використаний і доступний обсяг сховища
Відстеження змін через changes	Може застосовуватись для синхронізації та моніторингу стану даних

Google Drive API є достатньо гнучким інструментом для побудови веборієнтованих систем, що працюють із файлами у хмарному середовищі. Його можливості добре узгоджуються із завданнями резервного копіювання, відновлення, контролю доступу та автоматизації операцій над файлами, що робить цей API доцільною основою для реалізації обраного програмного рішення.

1.2.1 Методи інтелектуального сортування даних на основі контенту

Інтелектуальне сортування даних базується на аналізі не лише зовнішніх атрибутів файлу (назва, розмір, дата створення), а й його внутрішнього вмісту. Це дозволяє автоматизувати процес класифікації, групування та

впорядкування файлів без участі користувача або з мінімальним втручанням з його боку.

Одним із базових підходів є контент-орієнтована класифікація, при якій система аналізує текстову або структуровану інформацію всередині файлу. Для текстових документів це може бути виділення ключових слів, частоти їх використання або визначення тематичної належності [18]. Наприклад, файли, що містять слова «invoice», «payment», «order», можуть автоматично відноситись до фінансової категорії. Для реалізації такого підходу часто використовуються методи обробки природної мови (NLP), зокрема:

– токенізація тексту (розбиття на слова);
– видалення стоп-слів;
– нормалізація (stemming, lemmatization);
– побудова векторного представлення тексту (TF-IDF, embeddings).

Ці методи дозволяють перетворити текст у числову форму, з якою можуть працювати алгоритми класифікації. У цьому випадку система навчається на попередньо розмічених даних і визначає категорію нового файлу автоматично. Найчастіше використовуються:

– наївний баєсівський класифікатор;
– метод k-найближчих сусідів (k-NN);
– логістична регресія;
– нейронні мережі.

Перевагою таких методів є здатність враховувати складні залежності між ознаками, однак вони потребують навчальних даних та додаткових обчислювальних ресурсів.

У випадках, коли розмічені дані відсутні, застосовуються методи кластеризації, які дозволяють автоматично групувати схожі файли. Найпоширеними є алгоритми k-means та ієрархічна кластеризація. Вони базуються на подібності між об’єктами та дозволяють формувати групи документів без попереднього навчання. Це особливо корисно для первинного впорядкування великої кількості неструктурованих даних. Окрему роль

відіграє аналіз метаданих, який доповнює контентний підхід. До метаданих належать:

– тип файлу (MIME);
– дата створення або зміни;
– розмір;
– автор;
– джерело створення.

Поєднання контентного аналізу та використання метаданих дозволяє суттєво підвищити точність сортування файлів, оскільки система враховує як внутрішній зміст інформації, так і її зовнішні характеристики [19]. Наприклад, файл із текстом контракту у форматі .pdf, створений у робочий час, з більшою ймовірністю буде віднесений до категорії документів, ніж до особистих файлів, оскільки така комбінація ознак формує більш повний контекст для класифікації.

У практичних інформаційних системах зазвичай застосовується гібридний підхід, який передбачає поєднання кількох методів аналізу. На початковому етапі виконується класифікація за базовими характеристиками файлу, після чого проводиться більш детальний аналіз його вмісту, а також може враховуватись поведінка користувача при роботі з даними. Такий підхід забезпечує більш гнучке та адаптивне сортування, що дозволяє зменшити кількість помилок і підвищити точність визначення категорій.

Для систем резервного копіювання важливим є також визначення пріоритетності даних, що впливає на порядок їх обробки та збереження. У цьому випадку система враховує частоту використання файлів, їхній розмір, тип і час останньої зміни, що дозволяє визначити, які дані є критичними. Це сприяє більш ефективному використанню ресурсів і забезпечує швидший доступ до найбільш важливої інформації.

Окремим напрямом розвитку є семантичне сортування, яке базується на аналізі змісту та контексту інформації, а не лише на окремих словах. Завдяки цьому система здатна встановлювати змістові зв’язки між файлами навіть у

випадках, коли вони відрізняються за мовою або формулюванням. Наприклад, документи з назвами різними мовами можуть бути віднесені до однієї категорії, якщо їхній зміст є подібним. Такий підхід реалізується за допомогою сучасних методів векторного представлення тексту.

У рамках реалізації вебдодатку BackupDriveApp доцільно застосовувати спрощений варіант інтелектуального сортування, який поєднує базові механізми аналізу без використання складних моделей машинного навчання. Система може визначати тип файлу за MIME-типом, аналізувати його назву та використовувати ключові слова для класифікації, після чого зберігати отриману категорію у базі даних.

1.2.2 Алгоритми класифікації та обробки файлів

Алгоритми класифікації та обробки файлів визначають логіку роботи інформаційної системи з даними та забезпечують їх впорядкування, збереження і подальше використання. Вони виступають основою для автоматизації процесів керування файлами, що особливо важливо у системах резервного копіювання та хмарних середовищах. Ефективність таких алгоритмів безпосередньо впливає на швидкість обробки інформації, надійність збереження та зручність доступу до даних.

Першим етапом взаємодії з файлом є його первинна обробка, яка включає перевірку коректності та визначення типу. На цьому етапі система аналізує розширення файлу, його MIME-тип, розмір та відповідність встановленим обмеженням. Це дозволяє запобігти обробці небезпечних або некоректних файлів, а також визначити подальший сценарій роботи з ними. Від правильності цього етапу залежить стабільність роботи всієї системи.

Після цього виконується класифікація файлу, яка може базуватися як на простих правилах, так і на більш складному аналізі. Найбільш поширеним є підхід, при якому система використовує поєднання інформації про розширення файлу, його назву та тип, визначений за вмістом. Такий

комбінований підхід дозволяє підвищити точність класифікації та уникнути ситуацій, коли файл має невідповідне розширення або некоректно названий.

Додатково може застосовуватися аналіз вмісту файлу, що дозволяє визначити його призначення більш точно. Наприклад, текстові документи можуть аналізуватися на наявність характерних ключових слів, що дозволяє віднести їх до певної категорії незалежно від назви. Такий підхід є більш гнучким, проте потребує додаткових обчислювальних ресурсів і може ускладнювати реалізацію системи.

Після завершення класифікації файл переходить до етапу обробки, який включає його збереження, передачу та фіксацію стану. У більшості систем використовується підхід, при якому файл спочатку зберігається локально, після чого передається до хмарного сховища. Це дозволяє забезпечити контроль за процесом обробки, зменшити ризик втрати даних і реалізувати механізми повторної спроби у випадку помилок.

Окрему увагу приділяють алгоритмам відновлення файлів, які забезпечують отримання даних із хмарного або локального сховища. При цьому система повинна враховувати наявність альтернативних джерел, що підвищує надійність доступу до інформації. Такий підхід дозволяє гарантувати відновлення даних навіть у разі часткових збоїв або втрати зв’язку з зовнішніми сервісами.

Важливим елементом є контроль стану обробки файлів, який реалізується через фіксацію статусів виконання операцій. Це дозволяє відстежувати, на якому етапі знаходиться кожен файл, своєчасно виявляти помилки та забезпечувати прозорість роботи системи. Наявність таких механізмів значно спрощує адміністрування та підвищує надійність функціонування.

З метою забезпечення ефективності роботи системи використовуються підходи до оптимізації алгоритмів обробки. Вони спрямовані на зменшення часу виконання операцій та раціональне використання ресурсів. Це досягається шляхом обмеження параметрів файлів, впровадження

асинхронної обробки та оптимізації передачі даних між компонентами системи.

У контексті розроблюваного вебдодатку BackupDriveApp алгоритми класифікації та обробки файлів реалізуються як послідовність взаємопов’язаних етапів, що забезпечують перевірку, класифікацію, збереження та передачу даних у хмарне середовище.

1.3 Підходи до організації резервного копіювання даних

Організація резервного копіювання даних забезпечує збереження інформації у випадку збоїв, втрати даних або несанкціонованого доступу. Ефективність таких систем залежить від правильного вибору підходу до зберігання, обробки та відновлення інформації, а також від здатності системи адаптуватися до змін у структурі та обсязі даних [21].

Одним із базових підходів є централізоване резервне копіювання, при якому всі дані зберігаються у єдиному сховищі. Такий підхід забезпечує простоту управління та контроль доступу, однак створює ризик втрати даних у випадку відмови центрального вузла [22]. У сучасних системах цей підхід часто поєднується з використанням хмарних сервісів, що дозволяє підвищити надійність зберігання та забезпечити доступ до інформації з різних пристроїв.

Альтернативним є розподілений підхід, при якому дані зберігаються у декількох незалежних місцях. Це дозволяє зменшити ризик повної втрати інформації, оскільки навіть у випадку відмови одного вузла інші копії залишаються доступними. У контексті хмарних технологій та інтеграції з такими сервісами, як Google Drive, цей підхід реалізується через поєднання локального та віддаленого зберігання.

Важливим аспектом є спосіб створення резервних копій. Повне копіювання передбачає збереження всіх даних при кожному створенні резервної копії, що забезпечує максимальну надійність, але потребує значних ресурсів. У свою чергу, інкрементальний підхід дозволяє зберігати лише ті

дані, які змінилися з моменту останнього копіювання, що значно зменшує обсяг переданої інформації та прискорює процес. Комбінування цих підходів дозволяє досягти балансу між швидкістю роботи та надійністю збереження [24].

Процес резервного копіювання включає декілька послідовних етапів, які формують єдиний життєвий цикл роботи з даними. Спочатку відбувається підготовка файлів, після чого вони зберігаються у локальному середовищі, передаються до хмарного сховища та реєструються у базі даних системи. Така послідовність дій дозволяє контролювати кожен етап обробки та забезпечує можливість відновлення даних у разі виникнення помилок.

Не менш важливим є підхід до відновлення інформації, який повинен забезпечувати швидкий і надійний доступ до збережених даних. У системах, що використовують як локальне, так і хмарне зберігання, відновлення може виконуватись із будь-якого доступного джерела. Це підвищує стійкість системи до збоїв і дозволяє мінімізувати втрати інформації.

У сучасних інформаційних системах резервне копіювання тісно пов’язане з питаннями безпеки. Необхідно забезпечити захист даних під час передачі, зберігання та відновлення, а також контроль доступу до резервних копій. Використання механізмів автентифікації, шифрування та розмежування прав доступу дозволяє значно знизити ризики несанкціонованого доступу або втрати даних.

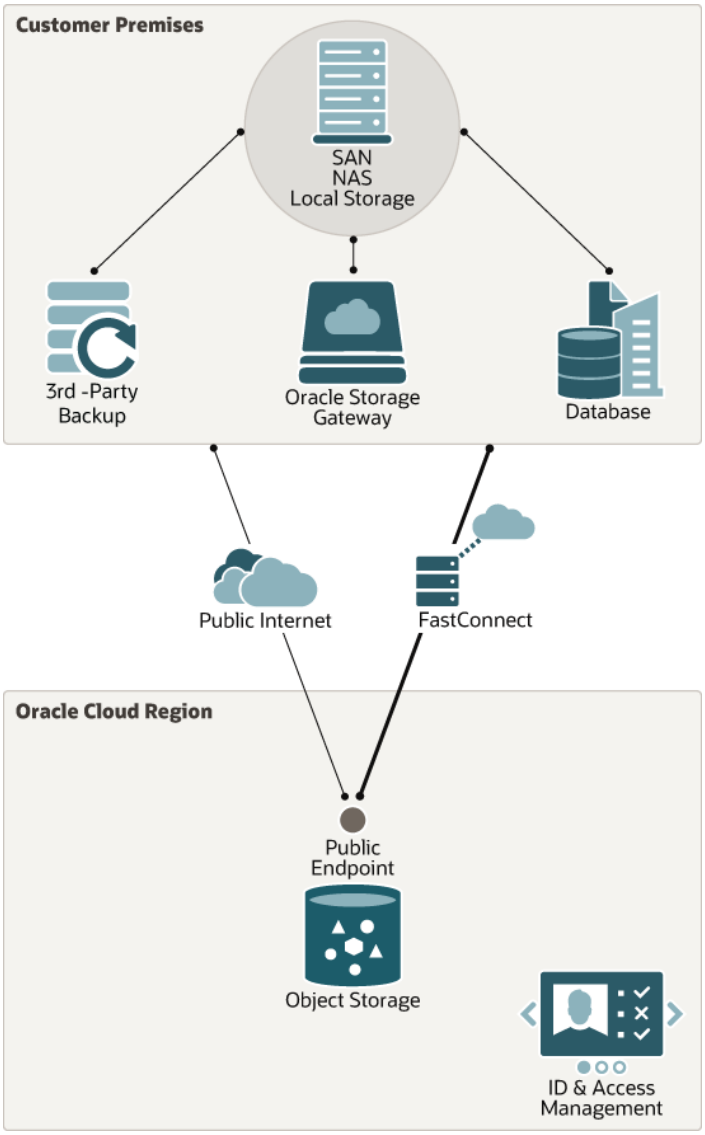

Рисунок 1.2 – Узагальнена схема процесу резервного копіювання даних

З огляду на особливості розроблюваного вебдодатку BackupDriveApp, доцільним є використання комбінованого підходу до резервного копіювання, який поєднує локальне збереження файлів із подальшою передачею до хмарного сховища.

1.4 Вимоги до інформаційних систем керування резервними копіями

Інформаційні системи керування резервними копіями повинні відповідати сукупності функціональних і нефункціональних вимог, які забезпечують надійність збереження даних, зручність використання та

стабільність роботи. Формування таких вимог базується на аналізі ризиків втрати інформації, обсягу даних, частоти їх змін та умов експлуатації системи.

Однією з ключових вимог є забезпечення надійності зберігання даних. Система повинна гарантувати збереження інформації навіть у випадку технічних збоїв, помилок користувача або втрати доступу до окремих компонентів. Це досягається шляхом використання резервних копій, дублювання даних та зберігання їх у різних середовищах, зокрема локальному та хмарному.

Важливою вимогою є забезпечення цілісності даних, що передбачає відсутність пошкоджень або втрат інформації під час її передачі та зберігання. Система повинна контролювати правильність збережених файлів, перевіряти їх на відповідність оригіналу та забезпечувати коректне відновлення у разі необхідності. Це особливо актуально для систем, що працюють із великими обсягами даних або критично важливою інформацією [24].

Не менш важливим є забезпечення доступності даних. Користувач повинен мати можливість у будь-який момент отримати доступ до своїх резервних копій та виконати їх відновлення. При цьому система повинна забезпечувати швидкий доступ до файлів незалежно від їх розміру або місця зберігання, що потребує оптимізації процесів обробки та передачі даних.

Система також повинна відповідати вимогам безпеки, які включають захист від несанкціонованого доступу, забезпечення конфіденційності інформації та контроль прав користувачів. Для цього застосовуються механізми автентифікації, авторизації, шифрування даних та розмежування доступу. Особливу увагу слід приділяти роботі з зовнішніми сервісами, такими як Google Drive, де важливим є правильне використання протоколів доступу та зберігання токенів авторизації [24].

Окрему групу вимог становлять вимоги до продуктивності системи. Вона повинна забезпечувати швидке виконання операцій резервного копіювання та відновлення, ефективно обробляти великі обсяги даних та

масштабуватись у разі зростання навантаження. Це передбачає оптимізацію алгоритмів, використання черг обробки та раціональне використання ресурсів.

Зручність використання також є важливою вимогою до систем керування резервними копіями. Інтерфейс користувача повинен бути інтуїтивно зрозумілим, забезпечувати простий доступ до основних функцій та не вимагати спеціальних технічних знань.

Важливою вимогою є можливість автоматизації процесів резервного копіювання. Система повинна підтримувати автоматичне створення копій за визначеними правилами або подіями, що дозволяє мінімізувати участь користувача та забезпечити регулярність збереження даних. Це особливо важливо у випадках, коли дані часто змінюються або мають критичне значення.

1.5 Висновок до першого розділу

У першому розділі кваліфікаційної роботи наведено результати аналітичного дослідження підходів до організації, обробки та інтелектуального сортування даних у хмарних інформаційних системах. Встановлено, що зі зростанням обсягів цифрової інформації традиційні методи управління файлами, які базуються на ручному впорядкуванні, стають малоефективними та потребують удосконалення за рахунок автоматизації та використання інтелектуальних підходів.

Виявлено, що ефективність роботи з даними у хмарному середовищі залежить не лише від швидкості доступу до файлів, а й від якості їх організації та структурування. Особливу роль відіграє поєднання контентного аналізу та використання метаданих, що дозволяє підвищити точність класифікації та забезпечити більш швидкий доступ до необхідної інформації. При цьому застосування гібридних підходів до сортування даних сприятиме підвищенню адаптивності системи до різних типів файлів та сценаріїв використання.

Проведений аналіз показав, що використання хмарних сервісів, зокрема Google Drive, відкриває широкі можливості для реалізації систем резервного копіювання та управління даними.

Дослідження алгоритмів класифікації та обробки файлів дозволило визначити, що найбільш ефективними є комбіновані підходи, які враховують як технічні характеристики файлів, так і їх зміст.

Сформульовані вимоги до інформаційних систем керування резервними копіями, зокрема надійність, цілісність, доступність, безпека, продуктивність і зручність використання, визначають основні орієнтири для подальшого проєктування системи. Врахування цих вимог дозволить створити ефективне програмне рішення, здатне забезпечити стабільну роботу з даними у хмарному середовищі та покращити досвід користувача.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО СОРТУВАННЯ ДАНИХ

2.1 Архітектурні підходи до побудови вебдодатків

Архітектура вебдодатку визначає структуру його компонентів, спосіб їх взаємодії та розподіл відповідальності між різними рівнями системи. Від правильного вибору архітектурного підходу залежить масштабованість, продуктивність, зручність підтримки та можливість подальшого розширення функціоналу. У сучасній розробці вебдодатків використовуються різні підходи, які дозволяють ефективно організувати обробку запитів, роботу з даними та інтеграцію із зовнішніми сервісами [25].

Одним із базових підходів є монолітна архітектура, при якій усі компоненти системи реалізуються в межах єдиного додатку. Така модель є простою для реалізації на початкових етапах, однак при збільшенні функціональності ускладнюється її підтримка та масштабування. У контексті систем керування резервними копіями це може призводити до складності внесення змін у логіку обробки файлів або інтеграції з зовнішніми сервісами[26].

Більш гнучким підходом є багаторівнева архітектура, яка передбачає розділення системи на логічні рівні. Найбільш поширеною реалізацією такого підходу є модель Model–View–Controller (MVC), що дозволяє відокремити роботу з даними, обробку логіки та відображення інтерфейсу користувача. Такий підхід забезпечує зменшення зв’язності між компонентами системи та спрощує її підтримку [27].

У сучасних умовах також широко застосовується використання додаткового сервісного шару (Service Layer), який виділяє логіку взаємодії із зовнішніми API та складні операції в окремі компоненти. Такий підхід є особливо актуальним для систем, що інтегруються з хмарними сервісами,

зокрема Google Drive, оскільки дозволяє ізолювати специфічну логіку роботи з API від основної частини додатку [28].

У рамках розроблюваного вебдодатку BackupDriveApp використовується поєднання архітектури MVC та сервісного шару. Загальна структура проєкту організована у вигляді окремих директорій, що відповідають основним компонентам системи. Вхідна точка додатку розташована у каталозі public, де відбувається обробка запитів користувача, ініціалізація середовища та маршрутизація. Основна логіка розміщена у каталозі src, який містить контролери, моделі, представлення та сервіси [29].

Окремий компонент архітектури становить сервісний шар, який реалізований у вигляді класу GoogleDriveService. Він відповідає за взаємодію з Google Drive API, включаючи авторизацію через OAuth 2.0, завантаження файлів у хмарне сховище, їх отримання та видалення. Виділення цієї логіки в окремий шар дозволяє спростити підтримку коду та забезпечити можливість розширення функціоналу без змін у контролерах або моделях [29].

Архітектура додатку також включає конфігураційний рівень, який відповідає за підключення до бази даних, налаштування середовища та зберігання ключів доступу. Для цього використовується файл конфігурації та змінні середовища, що підвищує безпеку системи та дозволяє легко змінювати параметри без редагування основного коду.

Взаємодія компонентів у системі реалізується за принципом: запит користувача обробляється маршрутизатором, після чого передається відповідному контролеру, який звертається до моделей або сервісів, а результат передається у представлення для відображення. Така структура забезпечує чітку логіку роботи додатку та дозволяє ефективно організувати процеси обробки даних.

2.2 Обґрунтування вибору архітектури

Вибір архітектури вебдодатку є одним із ключових етапів проєктування, оскільки саме вона визначає спосіб організації коду, взаємодію компонентів системи та можливість її подальшого розширення. Для системи керування резервними копіями, яка передбачає роботу з файлами, базою даних та зовнішніми API, необхідна архітектура, що забезпечує чітке розділення відповідальності та гнучкість у реалізації функціоналу.

У рамках розроблюваного вебдодатку BackupDriveApp обрано поєднання архітектури Model–View–Controller (MVC) та сервісного шару (Service Layer). Такий підхід дозволяє ефективно розділити логіку обробки даних, інтерфейс користувача та інтеграцію із зовнішніми сервісами, що є критично важливим для систем, які працюють із хмарними сховищами.

Використання архітектури MVC забезпечує структурованість додатку та спрощує підтримку коду. Моделі відповідають за взаємодію з базою даних, зокрема за роботу з таблицями користувачів і резервних копій, що дозволяє реалізувати збереження інформації про файли, їх статуси та зв’язки з користувачами.

Однак використання лише MVC є недостатнім у випадку, коли система активно взаємодіє із зовнішніми сервісами. У BackupDriveApp такою складовою є інтеграція з Google Drive через API. Для цього до архітектури додано сервісний шар, який інкапсулює логіку роботи з Google Drive, включаючи авторизацію через OAuth 2.0, завантаження файлів, їх отримання та видалення [30].

Сервісний шар реалізований у вигляді окремого компонента GoogleDriveService, що дозволяє винести всю специфічну логіку взаємодії з API за межі контролерів. Це забезпечує кращу читабельність коду, спрощує тестування та дозволяє при необхідності змінити реалізацію інтеграції без впливу на інші частини системи. Такий підхід також підвищує повторне

використання коду, оскільки сервіси можуть використовуватись у різних частинах додатку.

Обрана архітектура добре узгоджується зі структурою проєкту BackupDriveApp, де код розділений на відповідні директорії: Controllers, Models, Views та Services. Це дозволяє підтримувати логічну організацію проєкту та спрощує навігацію в кодовій базі. Наявність окремого конфігураційного рівня та bootstrap-файлу забезпечує централізовану ініціалізацію системи, підключення до бази даних та налаштування зовнішніх сервісів.

Важливою перевагою поєднання MVC та Service Layer є можливість масштабування системи. У разі розширення функціоналу, наприклад додавання нових типів сховищ або впровадження додаткових механізмів обробки файлів, відповідні зміни можуть бути внесені на рівні сервісів без суттєвого впливу на контролери чи моделі. Це дозволяє підтримувати стабільність системи та зменшити складність її розвитку.

2.3 Вибір технологічного стеку

Вибір технологічного стеку є важливим етапом проєктування вебдодатку, оскільки він визначає можливості реалізації функціоналу, продуктивність системи та зручність її підтримки. Для системи керування резервними копіями, яка передбачає роботу з файлами, базою даних і зовнішніми API, необхідно використовувати технології, що забезпечують стабільність, безпеку та простоту інтеграції.

У рамках розробки вебдодатку BackupDriveApp обрано стек технологій, орієнтований на серверну обробку даних та інтеграцію з хмарними сервісами. Основною мовою програмування використано PHP, який є широко поширеним у веброзробці та добре підходить для реалізації серверної логіки. Використання PHP версії 8.1+ дозволяє застосовувати сучасні можливості мови, зокрема покращену типізацію, що підвищує надійність коду [31].

Для управління залежностями використовується Composer, який забезпечує підключення сторонніх бібліотек та спрощує організацію проєкту. Зокрема, для інтеграції з Google Drive використовується бібліотека Google APIs PHP Client, яка надає зручний інтерфейс для роботи з Google Drive API та реалізації механізмів авторизації.

Доцільність вибору бібліотеки Google APIs PHP Client підтверджується її практичним використанням у проєкті BackupDriveApp. Зокрема, завантаження резервної копії до хмарного сховища реалізується через створення об’єкта сервісу Google Drive та передачу файлу разом із його метаданими. Підхід який використовується у проєкті подано в лістингу 2.1 [31]

Лістинг 2.1 – Скрипт побудови запиту

```
$service = new Google_Service_Drive($client);
$file = new Google_Service_Drive_DriveFile([
    'name' => $filename,
    'parents' => [$folderId]
]);
$created = $service->files->create($file, [
    'data' => file_get_contents($localFile),
    'mimeType' => $mimeType,
    'uploadType' => 'multipart'
]);
```

Наведений фрагмент демонструє, що обрана бібліотека дозволяє реалізувати передачу локально збереженого файлу до Google Drive із одночасним заданням його назви, типу та розташування у відповідній папці. Після успішного завантаження система отримує ідентифікатор створеного файлу, який надалі зберігається в базі даних для забезпечення можливості відновлення, завантаження або видалення резервної копії.

У якості системи управління базами даних обрано MySQL, що дозволяє ефективно зберігати інформацію про користувачів та резервні копії. Реляційна модель даних забезпечує цілісність інформації та дозволяє реалізувати зв’язки між таблицями, що є важливим для побудови логіки роботи системи.

Для реалізації інтерфейсу користувача використовується Bootstrap 5, який дозволяє швидко створювати адаптивний та зручний інтерфейс без необхідності розробки складного фронтенду.

Додатково використовується бібліотека Dotenv, яка дозволяє зберігати конфігураційні параметри, зокрема ключі доступу та налаштування підключення до бази даних, у змінних середовища.

Маршрутизація у додатку буде реалізована без використання повноцінного фреймворка.

2.4 Проєктування загальної архітектури системи BackupDriveApp

Проєктування архітектури вебдодатку BackupDriveApp базується на необхідності забезпечення ефективної роботи з файлами, їх обробки, збереження та інтеграції з хмарним середовищем. Основною метою є створення системи, яка дозволяє користувачу виконувати резервне копіювання, відновлення та управління даними через зручний вебінтерфейс.

Загальна логіка роботи системи реалізується через послідовність обробки запитів користувача, яка починається з точки входу додатку та завершується відображенням результату у вебінтерфейсі, див. рис. 2.1.

Вхідною точкою системи є файл public/index.php, який виконує ініціалізацію середовища, підключення необхідних компонентів та обробку HTTP-запитів. Після цього запит передається до відповідного контролера через механізм маршрутизації, де визначається подальша логіка виконання. Такий підхід дозволяє централізувати управління запитами та забезпечити єдину точку доступу до функціоналу системи [32].

Контролери виконують роль координаторів, які приймають запити користувача та взаємодіють із моделями і сервісами.

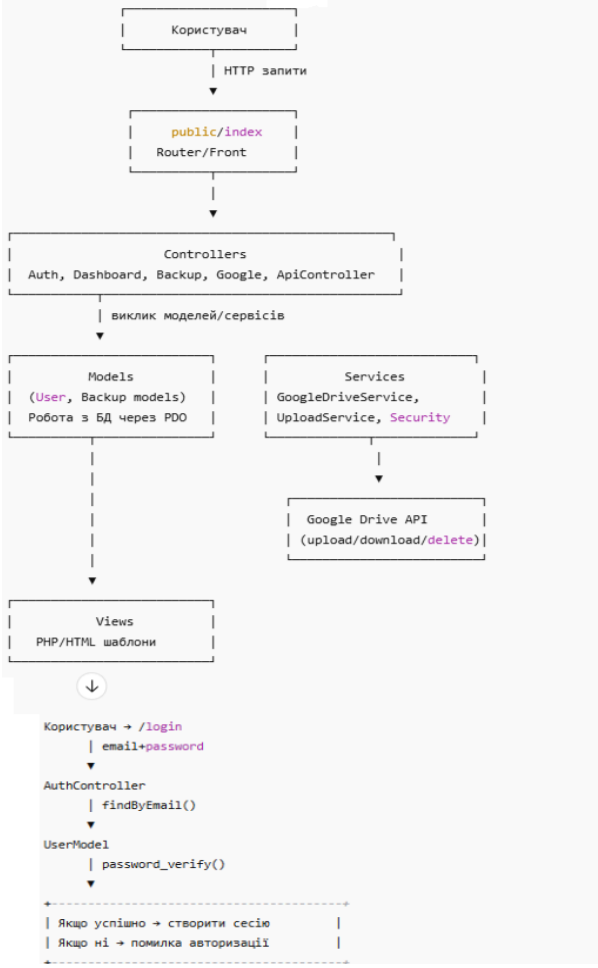

Рисунок 2.1 – Загальна архітектурна схема додатку

Наприклад, при завантаженні файлу контролер викликає методи перевірки, ініціює збереження файлу та передає його у сервісний шар для подальшої обробки. Це дозволяє ізолювати бізнес-логіку від рівня представлення та забезпечити чітку структуру системи.

Моделі відповідають за роботу з базою даних і реалізують операції створення, читання, оновлення та видалення даних. У системі BackupDriveApp основними є таблиці користувачів і резервних копій, які дозволяють зберігати інформацію про файли, їх розмір, тип, статус та зв’язок із користувачем. Така структура забезпечує можливість відстеження стану кожної резервної копії та виконання необхідних операцій над нею.

Окрему роль в архітектурі відіграє сервісний шар, який реалізує логіку взаємодії із зовнішніми API. У проєкті це представлено класом GoogleDriveService, що відповідає за авторизацію, завантаження, отримання

та видалення файлів у хмарному середовищі. Виділення цього компонента дозволяє відокремити складну логіку роботи з API від основної частини додатку, що спрощує підтримку та розширення функціоналу.

Процес обробки файлів у системі реалізований як послідовність взаємопов’язаних етапів. Спочатку файл проходить перевірку на коректність та відповідність вимогам, після чого зберігається у локальному сховищі. Далі відбувається його передача до Google Drive, після чого у базі даних оновлюється інформація про статус файлу та його ідентифікатор у хмарному сховищі. Такий підхід дозволяє забезпечити контроль над процесом обробки та підвищити надійність системи.

Важливою складовою архітектури є система автентифікації. У додатку реалізовано механізм реєстрації та входу із використанням сесій, що дозволяє зберігати стан користувача та обмежувати доступ до функціоналу. Додатково реалізовано інтеграцію з Google OAuth 2.0, що дає можливість отримати доступ до хмарного сховища користувача та виконувати операції від його імені, схему Google OAuth 2.0 зображено на рис. 2.2.

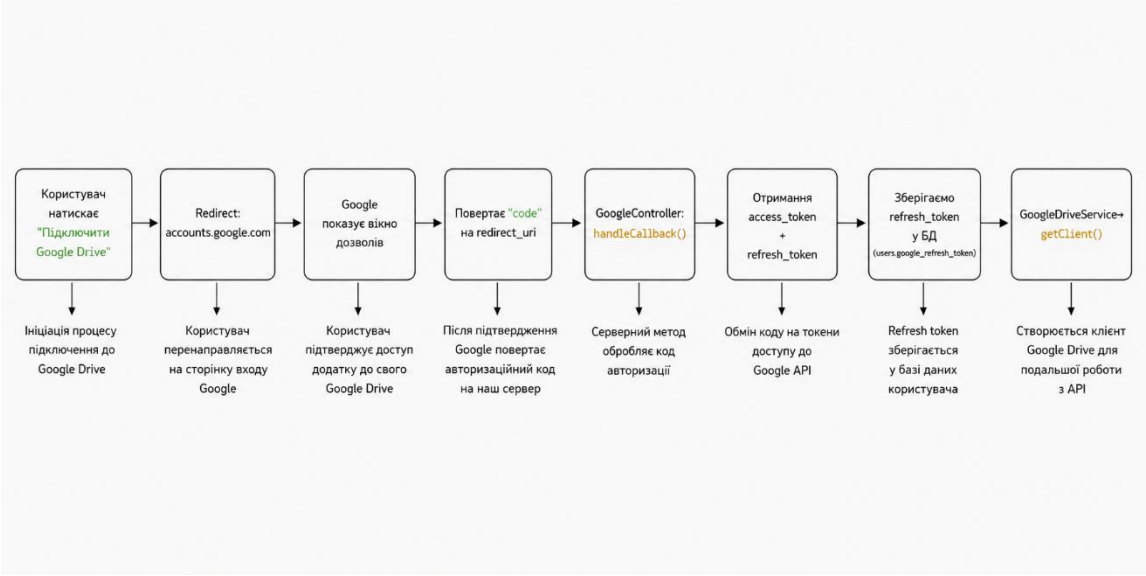

Рисунок 2.2 – Схема Google OAuth 2.0

Використання протоколу OAuth 2.0 у системі BackupDriveApp дозволяє реалізувати механізм делегованого доступу до ресурсів користувача у

хмарному середовищі. У процесі авторизації користувач надає дозвіл на доступ до свого облікового запису, після чого додаток отримує набір токенів, необхідних для взаємодії з API. Це дозволяє виконувати операції з файлами без передачі облікових даних, що значно підвищує безпеку системи. Застосування refresh_token забезпечує довготривалу роботу додатку та автоматичне оновлення доступу до ресурсів без участі користувача [30].

2.5 Моделювання бази даних та структури збереження інформації

У рамках даного проєкту використовується реляційна модель даних, реалізована за допомогою СУБД MySQL. Такий підхід дозволяє забезпечити цілісність даних, контроль зв’язків між сутностями та ефективне виконання запитів. Основними сутностями системи є користувачі та резервні копії, які логічно пов’язані між собою через ідентифікатор користувача, що наглядно демонструє рисунок 2.3.

Рисунок 2.3 – ER-модель бази даних

Таблиця користувачів містить інформацію, необхідну для ідентифікації та автентифікації. У ній зберігаються електронна пошта, хеш пароля, ім’я користувача, а також дані, пов’язані з інтеграцією з Google Drive.

Використання refresh_token забезпечує можливість виконання запитів до API без повторної авторизації користувача. Наявність ролі користувача формує базовий механізм розмежування доступу в системі.

Таблиця резервних копій призначена для збереження інформації про файли, які обробляються системою. Вона містить назву файлу, його розмір, MIME-тип, статус обробки та ідентифікатор файлу у хмарному сховищі. Поле drive_file_id використовується для зв’язку локальних даних із відповідними об’єктами у Google Drive та забезпечує виконання операцій відновлення, завантаження і видалення файлів [32].

Зв’язок між таблицями реалізовано за допомогою зовнішнього ключа, який підтримує цілісність даних та забезпечує автоматичне видалення пов’язаних записів у разі видалення користувача. Такий підхід виключає появу невикористаних записів і спрощує адміністрування бази даних.

Важливою складовою моделі є відображення стану обробки файлів. Для цього використовується поле статусу, яке відображає життєвий цикл резервної копії від моменту створення до завершення обробки. Наявність такого механізму дає можливість відстежувати виконання операцій і своєчасно реагувати на помилки.

Окрім логічної структури, враховано особливості фізичного зберігання файлів. У системі BackupDriveApp застосовується комбінований підхід, при якому файли спочатку зберігаються у локальному каталозі, а потім передаються до хмарного сховища, що продемонстровано на рисунку 2.4.

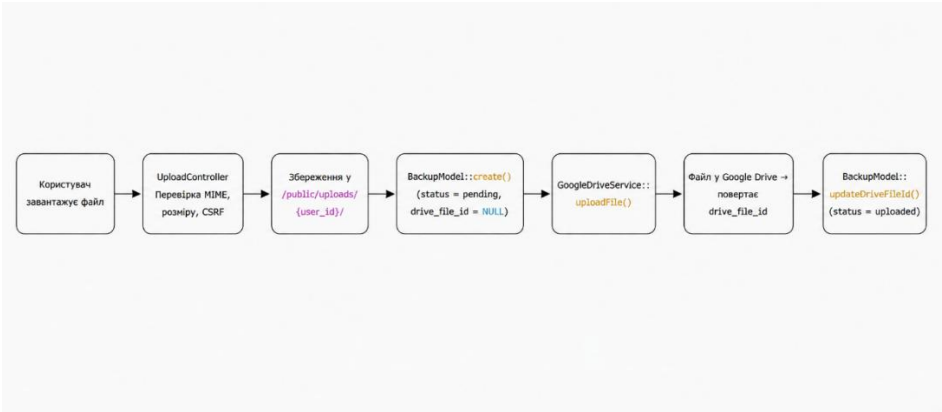

Рисунок 2.4 – Схема роботи з файлами (Upload to Drive to DB)

Це дає контроль над процесом передачі та підвищує надійність роботи з даними.

2.6 Проєктування алгоритмів роботи системи

Функціонування вебдодатку BackupDriveApp базується на реалізації набору взаємопов’язаних алгоритмів, які забезпечують обробку запитів користувача, управління файлами та інтеграцію з хмарним середовищем. Проєктування цих алгоритмів виконано з урахуванням необхідності забезпечення надійності, безпеки та ефективності роботи системи.

Алгоритм реєстрації користувача передбачає введення електронної пошти та пароля, після чого система виконує перевірку коректності даних і унікальності email. Далі відбувається хешування пароля за допомогою стандартних засобів PHP та створення нового запису у базі даних. Після успішного завершення операції користувач отримує доступ до системи або перенаправляється на сторінку авторизації.

Алгоритм входу до системи реалізує перевірку введених облікових даних. Система знаходить користувача за електронною поштою та виконує порівняння введеного пароля з хешем, збереженим у базі даних. У разі успішної перевірки формується сесія користувача, яка містить основні ідентифікаційні дані та використовується для подальшої взаємодії з додатком.

Одним із ключових є алгоритм завантаження резервної копії, який реалізується у два етапи. На першому етапі файл перевіряється на відповідність встановленим обмеженням, після чого зберігається у локальному сховищі. Паралельно створюється запис у базі даних із початковим статусом обробки. На другому етапі файл передається до хмарного сховища через Google Drive API, після чого у базі даних оновлюється інформація про його стан та ідентифікатор.

Алгоритм інтеграції з Google Drive базується на використанні протоколу OAuth 2.0, який забезпечує доступ до файлів користувача без передачі його

облікових даних. Після надання дозволу система отримує токени доступу, які використовуються для виконання запитів до API. У випадку закінчення терміну дії access_token використовується refresh_token для його автоматичного оновлення.

Алгоритм завантаження (скачування) файлу передбачає отримання інформації про резервну копію з бази даних та визначення джерела зберігання. Якщо файл знаходиться у хмарному сховищі, система виконує запит до Google Drive API та передає файл користувачеві. У випадку відсутності ідентифікатора хмарного файлу використовується локальна копія.

Алгоритм відновлення файлів передбачає отримання резервної копії та її збереження у визначену директорію на сервері. У процесі виконання враховується можливість використання як хмарного, так і локального джерела даних.

Окрему роль відіграє алгоритм видалення резервних копій, який передбачає видалення файлу як з локального сховища, так і з хмарного середовища. Після цього відповідний запис видаляється з бази даних.

У процесі роботи системи використовується механізм контролю стану обробки файлів, який дозволяє відстежувати виконання операцій. Кожен файл має відповідний статус, що змінюється залежно від результату виконання дій.

2.7 Забезпечення безпеки вебдодатку BackupDriveApp

Безпека є одним із ключових аспектів при розробці інформаційних систем, що працюють із персональними даними та файлами користувачів. У вебдодатку BackupDriveApp реалізовано комплексний підхід до захисту даних, який охоплює як рівень взаємодії з користувачем, так і обробку файлів та запитів до системи.

Одним із основних механізмів захисту є використання CSRF-токенів, які запобігають виконанню несанкціонованих запитів від імені користувача. При ініціалізації сесії генерується унікальний токен, який зберігається у змінній

сесії та передається у формах при виконанні дій, що змінюють стан системи. Перед обробкою кожного запиту виконується перевірка відповідності токена, що дозволяє виключити можливість підроблених запитів.

Особливу увагу приділено захисту від завантаження шкідливих або небажаних файлів. У процесі обробки завантажень система перевіряє MIME-тип файлу та його розширення, що дозволяє обмежити допустимі формати та запобігти виконанню потенційно небезпечного коду. Додатково встановлюються обмеження на розмір файлів і кількість резервних копій, що знижує ризик перевантаження системи та нераціонального використання ресурсів [33].

Додатковим елементом безпеки є обмеження кількості операцій завантаження файлів за певний проміжок часу. Такий механізм дозволяє уникнути надмірного навантаження на сервер і знижує ризик автоматизованих атак або масового надсилання запитів. Завдяки цьому система зберігає стабільність роботи навіть при великій кількості звернень користувачів.

Для підвищення рівня контролю над роботою додатку може використовуватись ведення журналів подій, у яких фіксуються дії користувачів та результати виконання операцій. Це дозволяє відстежувати підозрілу активність, аналізувати помилки та своєчасно реагувати на можливі загрози.

При інтеграції з Google Drive у системі реалізовано збереження токена доступу в базі даних, що дозволяє виконувати роботу з файлами без повторного входу користувача в Google-акаунт. Доступ до цих даних має лише серверна частина додатку, що підвищує безпеку взаємодії з Google Drive [33].

2.8 Висновки до другого розділу

У процесі аналізу підходів, методів та засобів проєктування інформаційних систем керування резервними копіями було здійснено комплексне дослідження як загальних принципів побудови вебархітектури,

так і специфіки реалізації систем, що працюють із файлами та хмарними сервісами. Розглянуті архітектурні підходи, зокрема монолітна, багаторівнева та сервісно-орієнтована моделі, дозволили визначити їх вплив на масштабованість, продуктивність та зручність підтримки системи, що має вирішальне значення для додатків, пов’язаних із обробкою та зберіганням даних.

Аналіз обраної архітектури показав, що поєднання моделі Model–View–Controller із сервісним шаром забезпечує оптимальний баланс між структурованістю коду та гнучкістю системи. Такий підхід дозволяє ефективно розділити логіку обробки запитів, роботу з базою даних і взаємодію із зовнішніми сервісами, що є критично важливим для реалізації функціоналу резервного копіювання та інтеграції з Google Drive.

Узагальнення особливостей проєктування бази даних та алгоритмів роботи системи дозволило визначити ключові підходи до організації зберігання інформації, обробки файлів і взаємодії з користувачем. Зокрема, поєднання локального та хмарного зберігання, використання статусів обробки файлів і поетапна передача даних формують основу для побудови надійної системи резервного копіювання.

Проведений аналіз створює концептуальну основу для реалізації ефективної, масштабованої та технічно узгодженої інформаційної системи BackupDriveApp, яка відповідає вимогам до сучасних вебдодатків і забезпечує надійне управління даними у хмарному середовищі.

3 РЕАЛІЗАЦІЯ ВЕБДОДАТКУ BACKUPDRIVEAPP ТА ІНТЕГРАЦІЯ З GOOGLE DRIVE API

3.1 Загальна архітектура реалізованої системи

Реалізація вебдодатку BackupDriveApp базується на багаторівневій архітектурі, що поєднує принципи Model–View–Controller із використанням сервісного шару для роботи із зовнішніми API. Такий підхід дозволяє логічно розділити компоненти системи, впорядкувати взаємодію між ними та спростити подальший розвиток функціоналу, що зображено на рисунку 3.1.

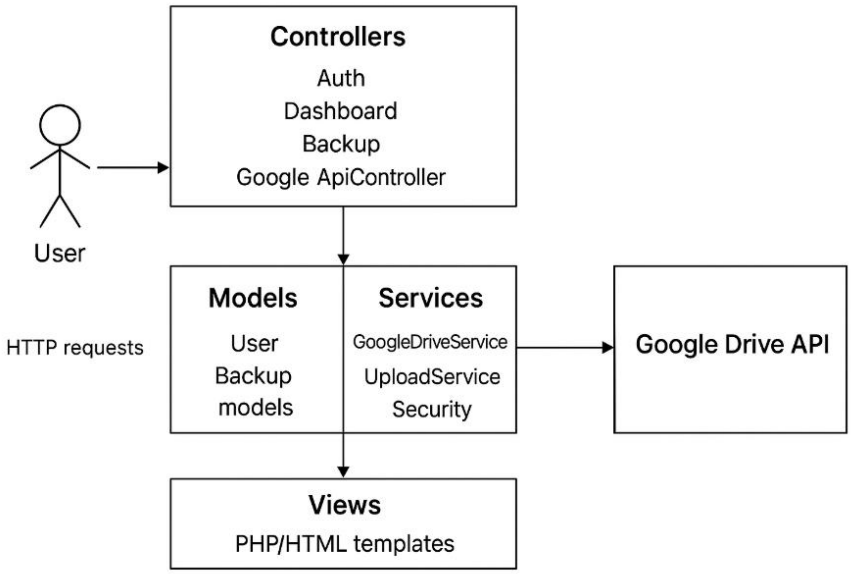

Рисунок 3.1 – Загальна архітектура системи

Загальна структура системи побудована навколо єдиної точки входу, яка відповідає за ініціалізацію середовища виконання та обробку HTTP-запитів. Усі запити користувача проходять через механізм маршрутизації, після чого передаються до відповідних контролерів, де визначається подальша логіка виконання. Така організація дозволяє централізувати керування запитами та контролювати потік виконання у додатку.

Серверна частина додатку поділена на декілька логічних рівнів. Контролери виконують роль координаторів, приймаючи вхідні дані та

ініціюючи виконання відповідних операцій. Моделі відповідають за взаємодію з базою даних і реалізують операції роботи з інформацією про користувачів та резервні копії. Представлення формують інтерфейс користувача та забезпечують відображення результатів обробки даних.

Окрему роль в архітектурі відіграє сервісний шар, який інкапсулює логіку взаємодії із Google Drive. У межах цього шару реалізовано операції авторизації, завантаження, отримання та видалення файлів. Винесення цієї функціональності в окремий компонент дозволяє зменшити зв’язність між частинами системи та спростити підтримку коду.

Архітектура додатку передбачає використання локального сховища як проміжного етапу обробки файлів. Після отримання файлу від користувача він тимчасово зберігається на сервері, проходить перевірку та лише після цього передається у хмарне середовище. Такий підхід дозволяє контролювати процес обробки та фіксувати стан виконання кожної операції.

Взаємодія між компонентами системи відбувається за принципом послідовного проходження запиту через рівні архітектури: від користувацького інтерфейсу до контролера, далі до моделей або сервісів і назад до рівня представлення. Це формує чіткий життєвий цикл запиту та дозволяє відокремити бізнес-логіку від інтерфейсу.

Структура проєкту BackupDriveApp організована у вигляді окремих директорій, що відповідають архітектурним компонентам. Зокрема, каталог Controllers містить логіку обробки запитів, Models — роботу з базою даних, Services — взаємодію із зовнішніми API, а Views — шаблони інтерфейсу.

3.2 Реалізація серверної частини додатку

Серверна частина вебдодатку BackupDriveApp реалізована з використанням мови програмування PHP та організована відповідно до принципів багаторівневої архітектури. Основною її функцією є обробка HTTP-

запитів, управління даними користувачів, виконання операцій із файлами та взаємодія із зовнішніми сервісами.

Точкою входу в систему є файл public/index.php, у якому виконується ініціалізація середовища, підключення залежностей через Composer, запуск сесії та обробка маршрутизації. Кожен запит користувача проходить через цей файл, після чого передається до відповідного контролера, який визначає подальшу логіку виконання [34].

Контролери виконують роль координаторів, обробляючи вхідні дані та ініціюючи виклик відповідних методів моделей і сервісів. Наприклад, при реєстрації користувача введений пароль одразу хешується за допомогою password_hash($password, PASSWORD_DEFAULT), після чого створюється новий запис у базі даних. При вході в систему використовується перевірка password_verify($password, $stored_hash), що дозволяє порівняти введене значення з уже збереженим хешем.

Робота з базою даних організована через моделі, які відповідають за доступ до таблиць users і backups. Через них виконуються всі операції, пов’язані зі збереженням інформації про користувачів, їхні резервні копії, статуси файлів та зв’язки з хмарним сховищем. Наприклад, отримання списку файлів користувача відбувається через виклик методу типу BackupModel::getByUser($user_id), що повертає всі записи, пов’язані з конкретним користувачем.

Ключовою частиною серверної логіки є реалізація процесу завантаження файлів. Після отримання файлу від користувача система виконує перевірку його параметрів, зокрема MIME-типу, розміру та розширення, після чого файл зберігається у локальному каталозі. Одночасно створюється запис у базі даних із початковим статусом, що дозволяє відстежувати стан обробки.

Подальша передача файлу до Google Drive виконується через сервісний шар, у якому створюється об’єкт клієнта Google API, формується структура файлу з метаданими та здійснюється виклик методу створення з передачею

вмісту файлу через file_get_contents($localFile). Після завершення операції система отримує унікальний ідентифікатор файлу, який зберігається у полі drive_file_id відповідного запису в базі даних [28].

У серверній частині також реалізовано механізм роботи із сесіями. Після успішної авторизації у сесії зберігаються ідентифікаційні дані користувача, зокрема $_SESSION['user_id'] та $_SESSION['role'], що використовуються для контролю доступу до функціоналу системи.

Окрему увагу приділено обробці помилок і контролю стану виконання операцій. Кожна дія, пов’язана з файлами, супроводжується перевіркою результату виконання, а у випадку виникнення помилок відповідна інформація фіксується у базі даних через зміну статусу файлу. Це дозволяє відстежувати проблеми та повторно виконувати необхідні операції.

3.3 Реалізація взаємодії з Google Drive API

3.3.1 Взаємодія додатку з OAuth 2.0

Інтеграція вебдодатку BackupDriveApp з Google Drive реалізована через використання Google Drive API, що дозволяє виконувати операції з файлами безпосередньо у хмарному середовищі користувача. Такий підхід дає можливість працювати з даними без їх постійного зберігання на сервері додатку, що зменшує навантаження на систему та підвищує рівень безпеки [35].

Взаємодія з API базується на використанні протоколу OAuth 2.0, який є стандартом авторизації для доступу до ресурсів користувача без передачі його облікових даних. У цьому підході додаток отримує обмежений доступ до файлів після підтвердження з боку користувача, що дозволяє виконувати операції від його імені [35].

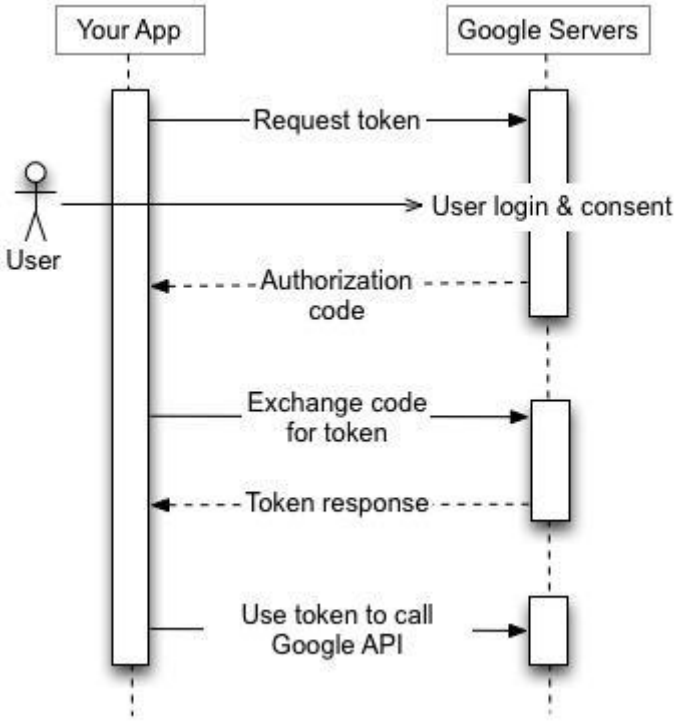

Рисунок 3.2 – Взаємодія додатку з OAuth 2.0

Процес авторизації реалізується за допомогою механізму OAuth 2.0 Authorization Code Flow, який використовується у серверних вебдодатках. Його суть полягає в тому, що користувач перенаправляється на сторінку авторизації Google, де підтверджує доступ до своїх даних, після чого система отримує спеціальний код авторизації.

Далі цей код обмінюється на access_token та refresh_token, які використовуються для подальших запитів до API. Access token має обмежений час дії, тому при його завершенні використовується refresh token для автоматичного отримання нового токена без повторної участі користувача. Такий підхід дозволяє організувати довготривалу роботу системи з мінімальним втручанням користувача.

Особливістю реалізації у BackupDriveApp є збереження refresh_token у базі даних, що дозволяє виконувати операції із файлами навіть після завершення сесії користувача, що зображено на рисунку 3.3.

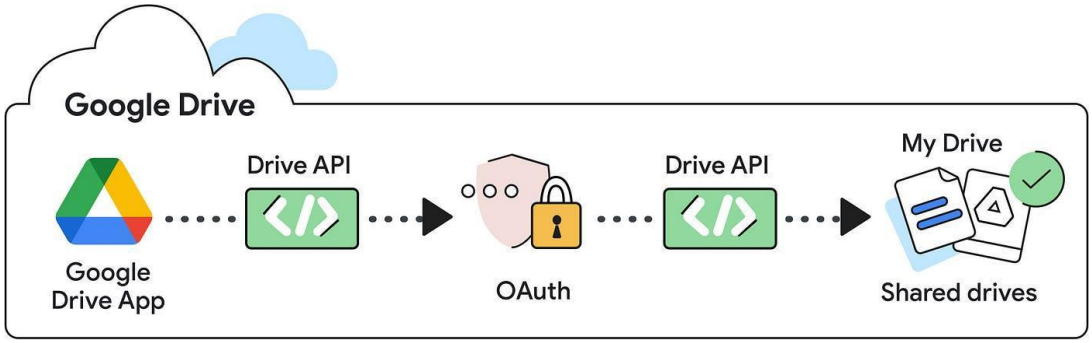

Рисунок 3.3 – Авторизація через OAuth 2.0

Процес завантаження файлів у хмарне середовище реалізовано як продовження локальної обробки даних. Після отримання файлу від користувача він проходить перевірку та тимчасово зберігається на сервері, після чого передається до Google Drive через API.

У серверній логіці формується об’єкт файлу з відповідними метаданими, включаючи його назву та батьківську директорію, а також передається вміст файлу разом із MIME-типом. Передача відбувається у режимі multipart, що дозволяє одночасно передати і дані, і метаінформацію.

Після завершення операції система отримує унікальний ідентифікатор файлу у хмарному сховищі, який зберігається у базі даних. Це дозволяє надалі виконувати операції доступу до файлу незалежно від його фізичного розташування.

Отримання файлів із хмарного середовища виконується шляхом звернення до API з використанням ідентифікатора файлу. Якщо файл присутній у Google Drive, система виконує запит із параметром alt=media, що дозволяє отримати його вміст і передати користувачеві.

У випадку відсутності файлу у хмарному сховищі використовується локальна копія, що забезпечує резервний механізм доступу до даних. Такий підхід підвищує надійність системи та дозволяє уникнути повної залежності від зовнішнього сервісу.

Видалення файлів реалізовано як двоетапний процес, що включає видалення об’єкта у хмарному середовищі та очищення відповідного запису у

базі даних. Це дозволяє підтримувати узгодженість між локальним станом системи та даними у Google Drive.

3.3.2 Реалізація алгоритмів завантаження файлів, передачі в Google Drive та збереження метаданих

Перший етап роботи системи полягає у прийманні файлу від користувача та його початковій обробці на сервері. Після надсилання форми система перевіряє, чи запит виконано методом POST, а також виконує перевірку CSRF-токена, що унеможливлює підроблені запити. Далі контролюється факт вибору файлу, після чого формується шлях до локального каталогу резервних копій.

На наступному кроці файл переміщується з тимчасового сховища PHP до робочої директорії додатку. Якщо операція переміщення не виконується успішно, процес припиняється з повідомленням про помилку. Після локального збереження додатково перевіряється, чи підключено обліковий запис Google Drive та чи встановлено прапорець завантаження у хмару. Якщо ці умови виконані, запускається наступний етап інтеграції з Google Drive, що описано в лістингу 3.1.

Завершальним кроком алгоритму є перенаправлення користувача назад на сторінку панелі керування, де вже відображаються оновлені результати роботи системи.

Лістинг 3.1 – upload.php

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
// CSRF
if(!isset($_POST['csrf']) || $_POST['csrf'] !== $_SESSION['csrf']){
die('Недійсний CSRF токен');
}
if(!isset($_FILES['backup_file'])) die('Файл не вибрано');
$file = $_FILES['backup_file'];
$local_path = BACKUP_DIR . '/' . basename($file['name']);
```

```
if(!move_uploaded_file($file['tmp_name'], $local_path)){
die('Помилка завантаження файлу');
}
// Завантаження на Google Drive
$google_connected = isset($_SESSION['google_access_token']);
if($google_connected && !empty($_POST['upload_to_drive'])){
$drive = getDriveClient();
if($drive){
$folderId = getOrCreateDriveFolder($drive,'BackupApp');
$fileMetadata = new Google_Service_Drive_DriveFile([
'name' => basename($local_path),
'parents' => [$folderId]
 ]);
 $content = file_get_contents($local_path);
 $drive->files->create($fileMetadata, [
 'data' => $content,
'mimeType' => mime_content_type($local_path),
'uploadType' => 'multipart'
```

Другий алгоритм реалізує безпосередню взаємодію з Google Drive API після того, як файл уже збережено у локальному сховищі. На початковому етапі сервер повертає відповідь у форматі JSON, оскільки цей модуль використовується як окрема серверна точка обробки. Далі система перевіряє метод запиту, коректність CSRF-токена, наявність імені файлу та його фізичну присутність у локальному каталозі.

Після цього виконується перевірка підключення Google-акаунта користувача. За відсутності токена доступу алгоритм завершується з помилкою, оскільки подальша взаємодія з API неможлива. Якщо авторизація активна, система створює клієнт Google Drive і перевіряє його доступність для виконання запитів.

Основна частина алгоритму виконується всередині блоку обробки винятків. Спочатку визначається або створюється цільова папка у Google Drive, після чого формується об’єкт метаданих файлу. Далі з локального диска

зчитується вміст файлу, а потім він передається у хмарне середовище через метод files->create. У випадку успішного завершення операції сервер повертає JSON-відповідь з ознакою успіху та ідентифікатором створеного файлу. Якщо виникає помилка, вона також повертається у структурованому вигляді. Алгоритм можна переглянути в тексті лістингу 3.2.

Лістинг 3.2 – Алгоритм передачі файлу до Google Drive

```
header('Content-Type: application/json; charset=utf-8');
if($_SERVER['REQUEST_METHOD'] !== 'POST'){
echo json_encode(['success'=>false,'error'=>'Invalid method']); exit;
}
if(empty($_POST['csrf']) || $_POST['csrf'] !== ($_SESSION['csrf'] ?? '')){
echo json_encode(['success'=>false,'error'=>'CSRF']); exit;
}
$filename = $_POST['file'] ?? '';
$path = BACKUP_DIR . '/' . $filename;
if(!file_exists($path)){
echo json_encode(['success'=>false,'error'=>'File not found']); exit;
}
if(!isset($_SESSION['google_access_token'])){
echo json_encode(['success'=>false,'error'=>'Google not connected']); exit;
}
$drive = getDriveClient();
if(!$drive){
echo json_encode(['success'=>false,'error'=>'Drive client error']); exit;
}
```

Ще один важливий алгоритм пов’язаний із фіксацією інформації про оброблені файли у базі даних. Його призначення полягає не у фізичному

копіюванні даних, а у збереженні метаданих, які надалі використовуються для відображення резервних копій, їх пошуку та подальшого адміністрування. На початку роботи файл підключає конфігурацію бази даних і встановлює формат відповіді JSON.

Після цього система отримує вхідні дані з тіла HTTP-запиту у форматі JSON і перетворює їх у масив. Далі перевіряється, чи дійсно передано список файлів. Якщо список відсутній або порожній, алгоритм одразу повертає повідомлення про помилку і завершує роботу.

У разі наявності коректних даних формується підготовлений SQL-запит на вставлення записів у таблицю backup_files. Після цього система проходить по всіх переданих елементах масиву та послідовно зберігає назву файлу, його розмір і дату оновлення. Якщо всі записи додано успішно, повертається позитивна JSON-відповідь. У випадку винятку користувач або клієнтський скрипт отримує повідомлення про помилку з відповідним текстом.

3.4 Реалізація функціональних модулів системи

У процесі розробки вебдодатку BackupDriveApp було реалізовано набір функціональних модулів, кожен з яких відповідає за окрему частину логіки системи. Основними модулями є модуль резервного копіювання та модуль відновлення даних.

Модуль резервного копіювання є ключовим компонентом системи та відповідає за створення, збереження та передачу файлів. Його функціональність включає завантаження файлів з локального пристрою, їх збереження на сервері та, за необхідності, автоматичну передачу у Google Drive.

Користувач має можливість обрати файл, після чого система виконує його обробку та збереження. Додатково реалізовано функції створення папок, додавання архівів та групового управління файлами що зображено на рисунку 3.4.

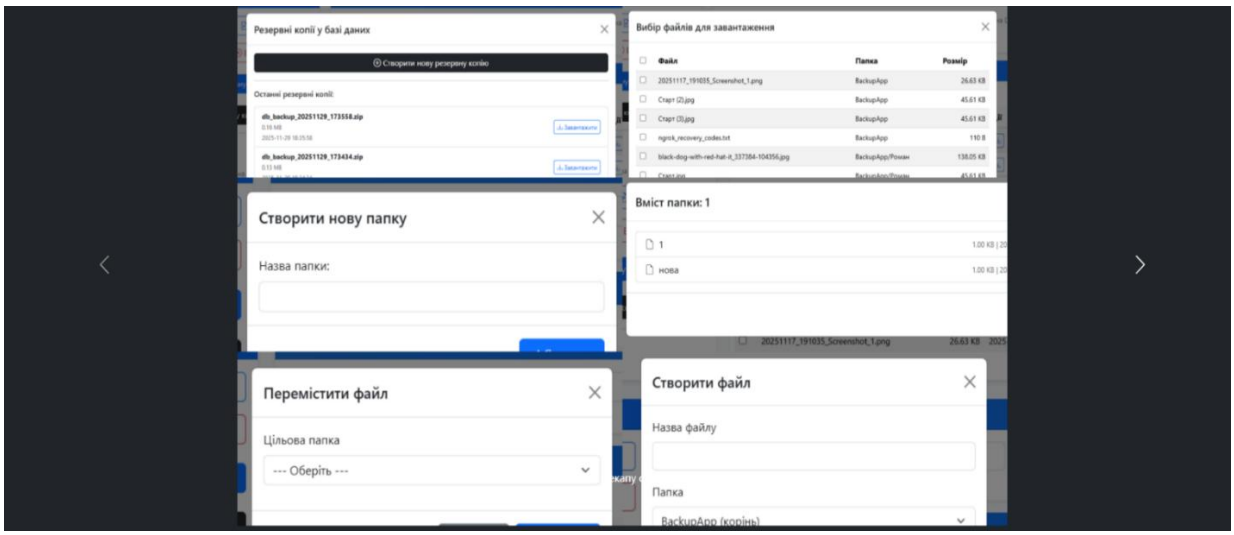

Рисунок 3.4 – Модуль резервного копіювання та додаткові функції

Модуль відновлення даних відповідає за повернення файлів з резервних копій як із локального сховища, так і з Google Drive. Основна логіка модуля полягає у виборі потрібного файлу, визначенні його джерела та виконанні відповідної операції відновлення.

Користувач може переглядати список доступних резервних копій, після чого завантажити потрібний файл або відновити його у систему. При цьому враховується цілісність даних та відповідність метаданих, що були збережені у базі.

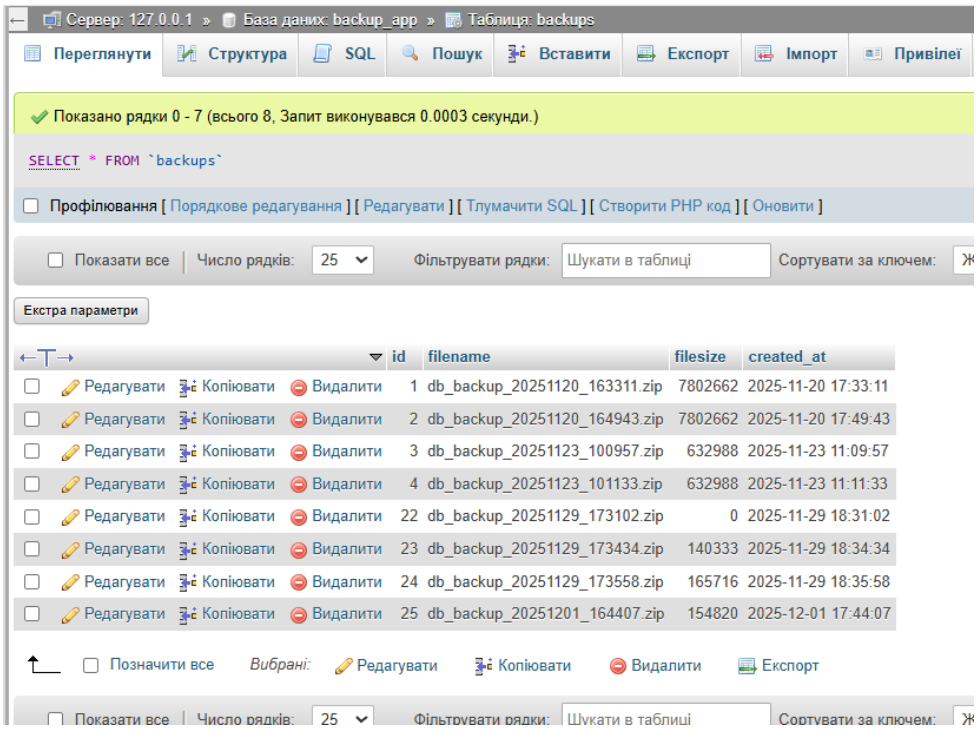

Рисунок 3.5 – Інтерфейс модуля відновлення даних

Взаємодія реалізованих модулів у системі побудована за принципом послідовної обробки запитів із чітким розмежуванням відповідальності між окремими компонентами. Модуль керування користувачами виступає базовим рівнем, оскільки саме він забезпечує автентифікацію, ініціалізацію сесії та збереження ключових параметрів, зокрема токена доступу до Google Drive. Без успішної роботи цього модуля інші компоненти не можуть виконувати свої функції, оскільки всі операції з файлами прив’язані до конкретного користувача.

3.5 Реалізація інтерфейсу користувача (Dashboard)

Реалізований вебдодаток BackupDriveApp є повноцінною системою для керування резервними копіями, яка поєднує локальне зберігання даних із можливостями хмарної інфраструктури Google Drive. Основний інтерфейс додатку представлений у вигляді єдиної інформаційної панелі (Dashboard), що забезпечує централізований доступ до всіх функцій системи.

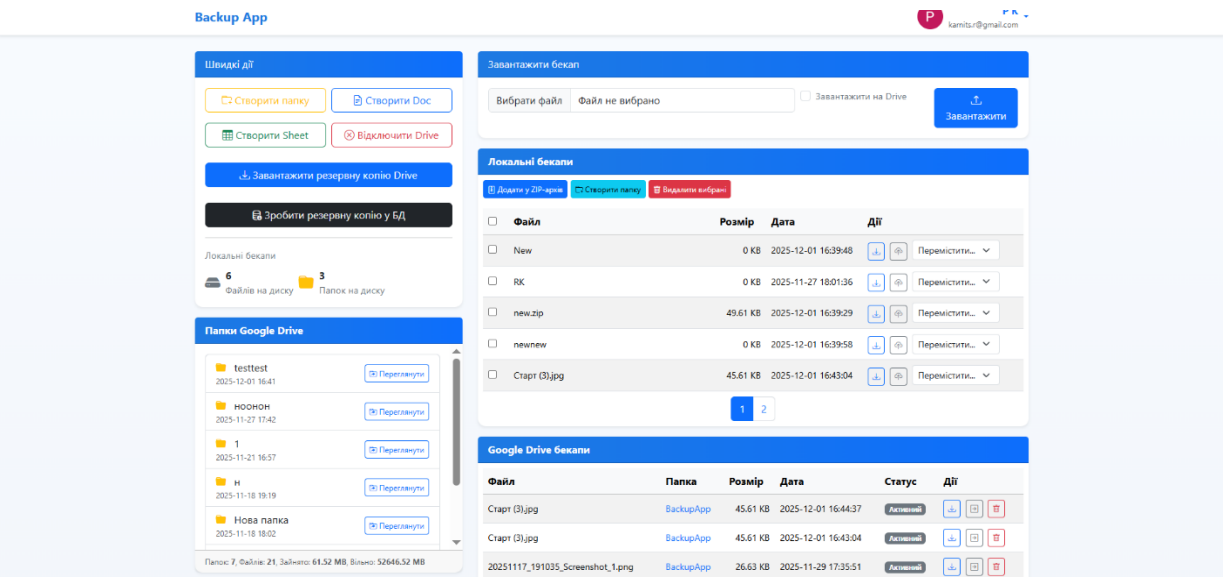

Рисунок 3.6 – BackUpDriveApp Dashboard

Інтерфейс побудований таким чином, щоб мінімізувати кількість дій користувача для виконання основних операцій. У верхній частині панелі розташований блок швидких дій, який надає доступ до ключових функцій, таких як створення нових папок, генерація документів або керування підключенням до Google Drive. Це дозволяє виконувати базові операції без переходу між сторінками.

Центральна частина інтерфейсу реалізує механізм роботи з файлами. Користувач має можливість одночасно працювати як із локальними резервними копіями, так і з файлами, що зберігаються у хмарі. Табличне представлення даних дозволяє швидко оцінити стан файлів, їх розмір, дату створення та виконувати необхідні дії. Важливою особливістю є підтримка пакетних операцій, що значно спрощує роботу з великою кількістю даних.

Окрему роль відіграє інтегрована панель Google Drive, яка відображає структуру папок у хмарному сховищі. Це дозволяє користувачу взаємодіяти з віддаленими файлами без необхідності переходу до сторонніх сервісів. Таким чином, додаток фактично виступає як єдина точка доступу до різних джерел зберігання.

Функціональність додатку також включає механізм роботи з резервними копіями бази даних. Користувач може створювати архіви, переглядати історію

резервних копій та виконувати їх завантаження. Це розширює можливості системи, перетворюючи її з простого файлового менеджера у інструмент комплексного резервного копіювання.

Важливою характеристикою реалізації є використання асинхронної взаємодії з сервером. Частина операцій виконується без перезавантаження сторінки, що покращує користувацький досвід і зменшує затримки при роботі з даними. Інтерфейс адаптований до сучасних вимог веброзробки та забезпечує інтуїтивно зрозумілу взаємодію навіть для користувачів без технічного досвіду.

У цілому, реалізований додаток забезпечує ефективне середовище для керування резервними копіями, поєднуючи зручність використання, функціональність та інтеграцію з зовнішніми сервісами.

У процесі роботи з файлами реалізовано механізм групової взаємодії, що дозволяє користувачу виконувати операції не лише над окремими елементами, але й над їх сукупністю. Для цього передбачено спеціалізоване вікно вибору файлів, у якому користувач може відмітити необхідні об’єкти та виконати подальші дії, зокрема формування архіву або підготовку до передачі у хмарне сховище.

Як показано на рисунку 3,7, інтерфейс побудований у вигляді таблиці з можливістю множинного вибору, що забезпечує зручне управління великим обсягом даних. Такий підхід дозволяє значно скоротити кількість дій користувача при роботі з резервними копіями та підвищує ефективність взаємодії із системою.

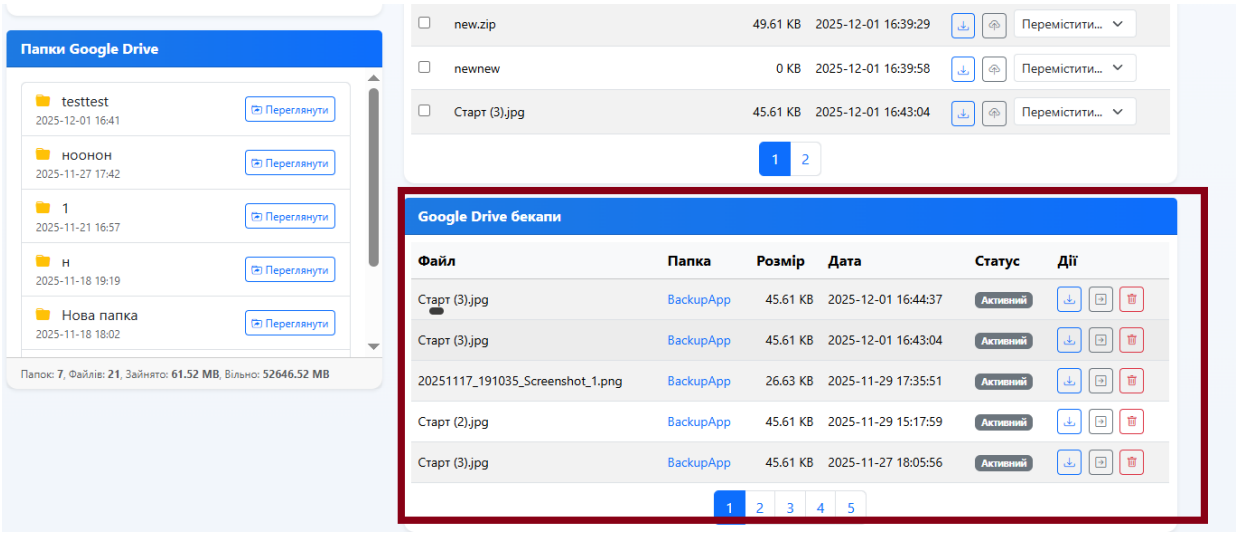

Рисунок 3.7 – Інтерфейс вибору файлів для виконання групових операцій

Окремим елементом функціональності додатку є реалізація підсистеми резервного копіювання бази даних, яка забезпечує збереження критично важливої інформації у вигляді архівів. Користувач має можливість створювати нові резервні копії, переглядати їх список, а також виконувати завантаження збережених версій. Як видно з рисунка 3.8, кожна резервна копія представлена у вигляді окремого запису із зазначенням назви файлу, розміру та дати створення.

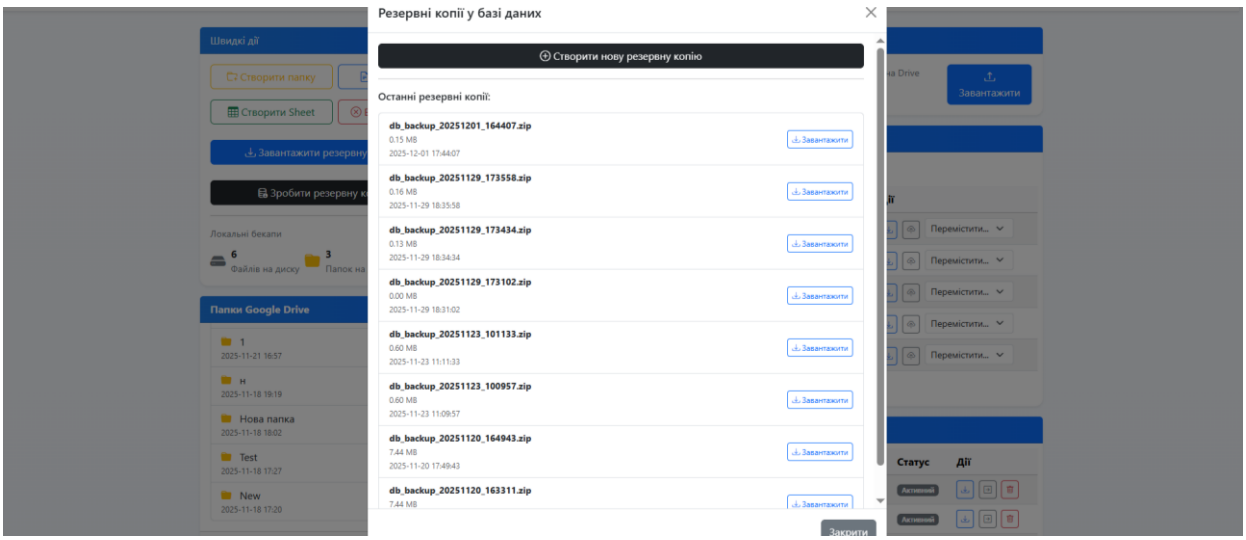

Рисунок 3.8 – Інтерфейс перегляду та завантаження резервних копій

Наявність історії резервних копій забезпечує додатковий рівень надійності системи та дозволяє уникнути втрати даних у випадку збоїв або помилок.

Реалізація описаних механізмів демонструє, що додаток BackupDriveApp забезпечує не лише базові операції з файлами, але й розширені можливості управління даними, орієнтовані на ефективність, зручність та безпеку використання.

3.6 Реалізація особистого кабінету користувача

Особистий кабінет користувача є окремим функціональним елементом системи, який забезпечує централізоване керування персональними даними, параметрами безпеки та інтеграцією із зовнішніми сервісами. Його реалізація побудована на поєднанні клієнтської частини інтерфейсу та серверної логіки, що обробляє запити та взаємодіє з базою даних і сторонніми API.

Інтерфейс особистого кабінету структуровано у вигляді декількох логічних блоків: профіль користувача, безпека, інтеграція з Google Drive та системні налаштування. Такий підхід дозволяє розділити функціональність за призначенням і спростити навігацію. Кожен із блоків взаємодіє з серверною частиною через окремі PHP-обробники, що забезпечує ізоляцію логіки та зручність масштабування. Як видно з рисунка 3.9, користувач має можливість переглядати та змінювати свої персональні дані, зокрема ім’я та електронну адресу.

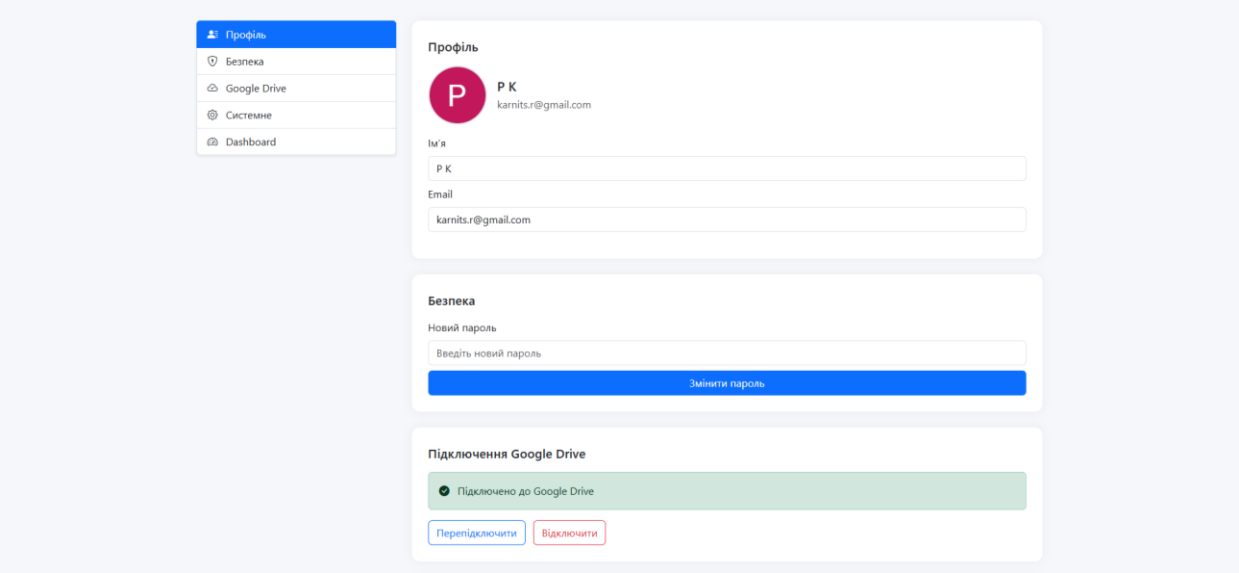

Рисунок 3.9 – Інтерфейс особистого кабінету

При внесенні змін відповідна форма надсилає POST-запит до серверного обробника, який виконує валідацію даних, перевірку сесії та оновлення записів у базі даних. Така взаємодія реалізується через скрипти типу update_profile.php, що працюють із використанням підготовлених SQL-запитів для забезпечення безпеки.

Блок безпеки дозволяє змінювати пароль користувача. При цьому введені дані передаються на сервер, де виконується їх перевірка, хешування нового пароля та оновлення відповідного запису. Взаємодія з бекендом у цьому випадку також відбувається через окремий обробник, що мінімізує ризик помилок і підвищує контроль над процесом обробки конфіденційної інформації.

Особливу роль відіграє модуль інтеграції з Google Drive, який відображається у вигляді окремого блоку в особистому кабінеті. Користувач може підключити або відключити свій обліковий запис, а також виконати повторну авторизацію. У процесі підключення відбувається взаємодія з OAuth 2.0, після чого отриманий токен доступу зберігається у сесії або базі даних. Надалі цей токен використовується іншими модулями системи для виконання операцій із файлами у хмарному середовищі.

Системний блок кабінету включає додаткові функції, такі як очищення кешу, експорт даних профілю та видалення акаунта. Ці операції також реалізовані через взаємодію з серверною частиною: клієнт надсилає запит, після чого відповідний скрипт виконує необхідні дії, наприклад формує файл для експорту або видаляє записи користувача з бази даних.

3.7 Реалізація системи безпеки та тестування додатку

У процесі розробки вебдодатку BackupDriveApp особливу увагу було приділено забезпеченню безпеки даних користувача та коректності виконання основних операцій. Реалізовані механізми захисту охоплюють як рівень клієнт–серверної взаємодії, так і обробку даних на сервері, що дозволяє мінімізувати ризики несанкціонованого доступу та втрати інформації.

Одним із базових елементів захисту є механізм автентифікації користувачів, реалізований на основі серверних сесій. Після успішного входу у систему створюється сесія, яка використовується для ідентифікації користувача при виконанні всіх подальших запитів. Для обмеження доступу до захищених сторінок використовується перевірка авторизації, що унеможливлює виконання дій неавторизованими користувачами.

Додатковим рівнем захисту є використання CSRF-токенів, які генеруються на сервері та передаються разом із формами. При кожному запиті виконується перевірка відповідності токена, що дозволяє запобігти підробленим запитам від сторонніх ресурсів. У разі невідповідності токена обробка запиту припиняється, що забезпечує захист від атак типу Cross-Site Request Forgery.

При роботі з файлами реалізовано перевірку вхідних даних, зокрема контроль наявності файлу, перевірку його шляху та коректності передачі. Збереження файлів здійснюється у визначеній директорії, що обмежує можливість доступу до інших частин файлової системи. Також

використовується обробка MIME-типів, що дозволяє контролювати тип завантажуваних даних і знижує ризик завантаження небезпечних файлів.

Взаємодія з базою даних побудована із застосуванням підготовлених SQL-запитів (prepared statements), що унеможливлює виконання ін’єкційних атак. Всі параметри перед вставкою у запит передаються окремо від SQL-інструкції, що забезпечує коректну обробку даних незалежно від їх вмісту.

Інтеграція з Google Drive реалізована з використанням протоколу OAuth 2.0, що дозволяє уникнути зберігання конфіденційних облікових даних користувача у системі. Отриманий токен доступу використовується лише для виконання необхідних операцій і не передається стороннім компонентам. Це забезпечує безпечну взаємодію з хмарним середовищем.

Окрім цього, реалізовано механізми обробки помилок та винятків, що дозволяють запобігти некоректному завершенню роботи системи та витоку службової інформації. У випадку виникнення помилки користувачу повертається узагальнене повідомлення без розкриття внутрішньої логіки роботи додатку.

У подальшому планується проведення комплексного тестування розробленого додатку з метою перевірки його стабільності, безпеки та відповідності функціональним вимогам. Зокрема, буде виконано функціональне тестування, яке дозволить перевірити коректність роботи основних сценаріїв використання, таких як завантаження файлів, інтеграція з Google Drive та відновлення даних, ефективність додатку зображено в таблиці 3.1.

Таблиця 3.1 - Порівняння ефективності роботи системи до та після впровадження інтелектуального сортування

Метрика	**До впровадження**	**Після**
Середній час пошуку файлу	2 хв 30 сек	40 сек
Кількість дій для доступу до файлу	6-8	2-3
Відсоток невдалих пошуків	45%	15%
Час відновлення резервної копії	3хв	1хв

Також передбачається проведення тестування безпеки, спрямованого на виявлення потенційних вразливостей, зокрема перевірку захисту від CSRF-атак, SQL-ін’єкцій та несанкціонованого доступу до ресурсів системи. Додатково буде здійснено навантажувальне тестування, яке оцінить поведінку системи при обробці великої кількості запитів та роботи з великими обсягами даних.

Отримані результати демонструють позитивний вплив реалізованих механізмів сортування та оптимізації даних на ефективність системи. Зменшення кількості дублікатів і середнього розміру резервних копій дозволило скоротити використання дискового простору, що зображено в таблиці 3.2, а збільшення кількості доступних версій резервних копій покращило можливості відновлення даних.

Таблиця 3.2 - Показники використання хмарного та локального сховища

Показник	**Значення до оптимізації**	**Значення після**
Використаний простір	80%	65%
Дублікати файлів	25%	5%
Середній розмір бекапу	120МБ	95МБ
Кількість збережених версій	3	5

Для оцінки впливу оптимізації на швидкодію системи було проведено моделювання обробки різної кількості файлів. У процесі тестування порівнювався середній час виконання операцій до та після впровадження механізмів інтелектуального сортування й оптимізації роботи.

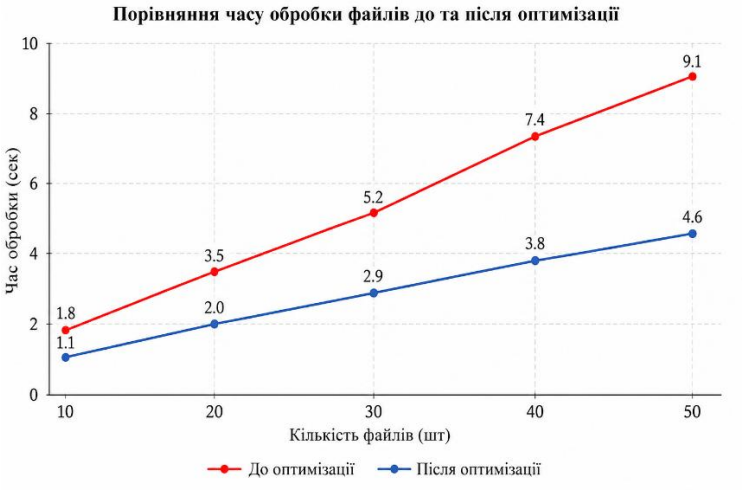

Рисунок 3.10 – Інтерфейс особистого кабінету

Результати тестування будуть використані для подальшої оптимізації та вдосконалення системи, що дозволить підвищити її надійність та ефективність у реальних умовах експлуатації.

3.8 Висновки до третього розділу

У межах розділу виконано практичну реалізацію вебдодатку BackupDriveApp та інтеграцію з Google Drive API. Розглянуто архітектуру системи, серверну логіку, обробку файлів, побудову функціональних модулів, інтерфейс користувача та реалізацію базових механізмів безпеки.

Архітектура додатку побудована на поєднанні Model–View–Controller із сервісним шаром, що дозволило впорядкувати структуру системи та чітко розмежувати відповідальність між її компонентами. Реалізовано обробку HTTP-запитів, роботу з базою даних, алгоритми завантаження, передачі та відновлення файлів. Інтеграція з Google Drive через OAuth 2.0 забезпечила можливість виконання операцій у хмарному середовищі користувача без передачі його облікових даних.

Реалізовані функціональні модулі охоплюють основні сценарії роботи системи, включаючи створення резервних копій, управління файлами,

відновлення даних та налаштування користувача. Dashboard і особистий кабінет формують цілісне середовище взаємодії, а впроваджені механізми захисту підтримують стабільну та безпечну роботу додатку.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Організація праці в комп'ютерних класах

Володіння комп'ютерними технологіями сьогодні є необхідною складовою професійної діяльності у більшості сфер. Значна кількість сучасних вакансій передбачає наявність навичок роботи з офісним програмним забезпеченням, системами обробки документів, електронними таблицями та хмарними сервісами. Навіть у професіях, які безпосередньо не пов'язані з офісною діяльністю, часто потрібні базові знання комп'ютерних систем та цифрових інструментів.

Під час проведення занять у комп'ютерних аудиторіях ефективність роботи студентів залежить від багатьох чинників, серед яких важливе місце займає організація робочого середовища. Одним із факторів, що негативно впливає на працездатність, є тривале використання комп'ютерної техніки та недотримання рекомендованих режимів праці й відпочинку.

Результати досліджень свідчать про значний рівень використання комп'ютерів студентами під час навчального процесу. Особливо високі показники спостерігаються серед здобувачів освіти технічних спеціальностей, де частка використання комп'ютерної техніки під час занять становить близько 85–95 %.

Незважаючи на стрімкий розвиток комп'ютерної техніки та постійне вдосконалення характеристик сучасних персональних комп'ютерів, проблема їхнього впливу на стан здоров'я та рівень втоми користувачів залишається актуальною. У зв'язку з цим лікарі та психологи рекомендують обмежувати тривалість безперервної роботи за моніторами та приділяти увагу фізичним вправам і перервам під час роботи.

Робота за персональним комп'ютером супроводжується значним емоційним навантаженням і впливає на функціональний стан організму.

Найбільше навантаження припадає на зорову систему, центральну нервову систему та механізми регуляції уваги. Під час тривалої роботи користувачі часто відзначають втому очей, появу головного болю, погіршення концентрації уваги та інші ознаки перевтоми. Проведені психофізіологічні дослідження підтверджують, що тривала робота за комп'ютером може викликати більш виражені зміни нервової діяльності та функціонування зорових аналізаторів у порівнянні зі звичайною навчальною діяльністю.

Численні дослідження та спостереження за діяльністю студентів під час роботи за персональним комп'ютером підтверджують, що тривала безперервна робота протягом усього навчального заняття є неможливою без негативного впливу на функціональний стан організму. Студенти молодших і старших курсів у період активного росту та розвитку особливо чутливо реагують на несприятливі фактори навколишнього середовища, у тому числі на ті, що виникають під час використання комп'ютерної техніки. Встановлено, що чим молодший організм, тим більш вираженими є зміни його функціонального стану під впливом негативних чинників. Саме тому робота за комп'ютером повинна здійснюватися в індивідуальному темпі з дотриманням режимів праці та відпочинку.

Після певного періоду роботи за персональним комп'ютером рекомендується виконувати спеціальні вправи для очей, а під час перерв між заняттями — фізичні вправи для зменшення загальної втоми організму. Доцільним є розміщення комплексів вправ у навчальних аудиторіях або забезпечення кожного студента друкованими рекомендаціями для індивідуального користування. Для зниження локальної втоми м'язів, що підтримують тулуб і голову у вертикальному положенні, рекомендується приділяти декілька хвилин вправам для рук і ніг, а також виконувати помірні фізичні навантаження, які позитивно впливають на функціонування серцево-судинної та дихальної систем. Повторні заняття з використанням комп'ютерної техніки рекомендується проводити не раніше ніж через одну

годину після завершення попереднього заняття, а кількість таких занять не повинна перевищувати двох разів на тиждень.

Під час роботи за комп’ютером необхідно також дотримуватися санітарно-гігієнічних та організаційних вимог. Монітор повинен розташовуватися таким чином, щоб уникати потрапляння прямих сонячних променів і появи відблисків на екрані, які ускладнюють сприйняття інформації. Важливим є регулярне очищення дисплея від пилу, а також підтримання чистоти робочого місця, на якому не повинно бути сторонніх предметів або залишків їжі. Перед початком роботи необхідно мити та висушувати руки, щоб запобігти забрудненню клавіатури, монітора та інших елементів комп’ютерної техніки. У процесі роботи рекомендується періодично робити короткі перерви для фіксації результатів, планування подальших дій або короткочасного відпочинку. Для комп’ютерних мишок доцільно використовувати спеціальні килимки, а клавіатуру після завершення роботи бажано накривати захисною кришкою для запобігання потраплянню пилу та сторонніх предметів. У разі виникнення труднощів або запитань під час роботи необхідно звертатися до викладача або відповідального керівника.

Особлива увага повинна приділятися організації робочого місця користувача. Рекомендовані розміри робочого столу мають забезпечувати комфортне розміщення обладнання та достатній простір для роботи. Робочий стілець повинен складатися із сидіння, спинки та підлокітників і бути сконструйований таким чином, щоб забезпечувати правильну підтримку тіла користувача. Конструкція стільця має передбачати можливість регулювання висоти сидіння, кута нахилу сидіння та спинки, а також положення підлокітників. Ширина і глибина сидіння повинні відповідати ергономічним вимогам, а поверхня сидіння має бути рівною, із заокругленим переднім краєм. Матеріали, з яких виготовлено робочий стілець, повинні забезпечувати легке очищення від забруднень та відповідати санітарно-гігієнічним нормам.

Вертикальний ліктьовий кут (між плечем і передпліччям) - 70 - 90°, згинання зап'ястя від горизонталі більше 20°, нахил голови від вертикалі в межах 15-20° (рисунок 4.1).

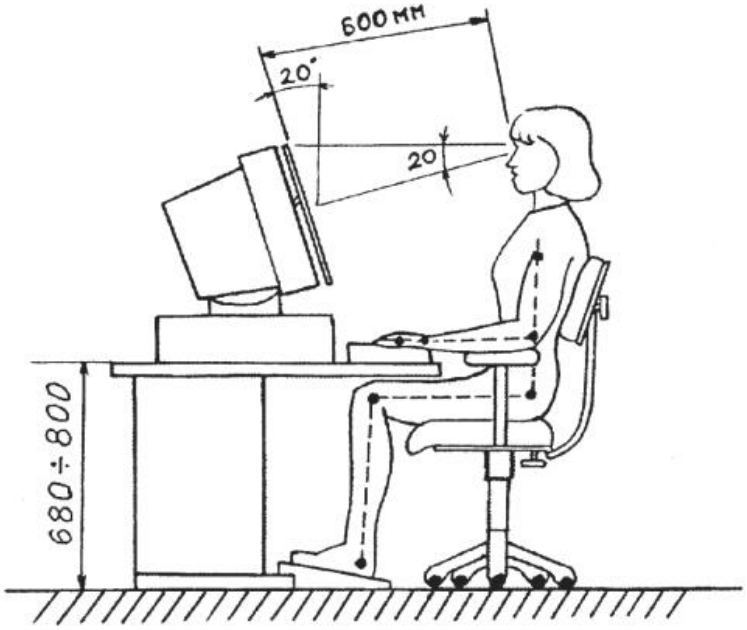

Рисунок 4.1 – Характеристики робочого місця

Комп’ютери також шкідливі для дихальної системи, оскільки ця побутова техніка притягує багато пилу. Цей ефект викликається електричними полями навколо монітора і системного блоку. Крім того, електричне поле іонізує повітря в приміщенні і знижує вологість, що також негативно впливає на якість легенів людини. Коли новий комп’ютер нагрівається, деякі робочі компоненти виділяють у повітря небезпечні речовини, які потім можуть вдихати користувачі. Щоб зменшити дію всіх цих факторів, потрібно частіше змочувати стільницю і провітрювати кімнату. У кімнаті також можна розмістити пристрій, що зволожує та іонізує повітря, або просто відкритий акваріум.

4.2 Організація і функціонування системи управління охороною праці

В Україні діє багаторівнева система управління охороною праці, до складу якої входять органи державної законодавчої та виконавчої влади, управлінські структури підприємств, установ і організацій, а також представники трудових колективів. Така система забезпечує реалізацію державної політики у сфері охорони праці та контроль за дотриманням вимог безпеки на різних рівнях управління.

Залежно від напрямів діяльності всі структурні елементи системи управління охороною праці можна умовно поділити на дві основні групи. До першої належать органи та установи, діяльність яких спрямована на вирішення законодавчих, нормативних, науково-технічних і соціально-економічних питань у сфері охорони праці. Друга група охоплює структури, функціональні обов'язки яких пов'язані із забезпеченням безпечних умов праці безпосередньо на підприємствах, в установах та організаціях.

До органів державного управління охороною праці належать Верховна Рада України, Кабінет Міністрів України, центральні органи виконавчої влади, спеціалізовані служби державного нагляду, Фонд соціального страхування від нещасних випадків і професійних захворювань, а також місцеві державні адміністрації та органи місцевого самоврядування. Їх діяльність спрямована на формування та реалізацію державної політики у сфері охорони праці, вдосконалення нормативно-правової бази, організацію державного контролю та створення умов для підвищення рівня безпеки праці.

Верховна Рада України визначає основні напрями державної політики у сфері охорони праці, розглядає питання вдосконалення законодавства та вирішує соціальні аспекти, пов'язані зі станом умов праці й забезпеченням безпеки працівників. Кабінет Міністрів України, у свою чергу, забезпечує реалізацію державної політики, організовує розроблення та виконання загальнодержавних програм з покращення стану охорони праці, визначає

повноваження органів виконавчої влади та здійснює контроль за виконанням відповідних заходів.

Для координації діяльності у сфері безпечної життєдіяльності населення при Кабінеті Міністрів України функціонує Національна рада з питань безпечної життєдіяльності населення, яку очолює віцепрем'єр-міністр України. Її діяльність спрямована на узгодження роботи державних органів у питаннях забезпечення належного рівня безпеки праці та захисту населення від небезпечних виробничих факторів.

Держгірпромнагляд України здійснює комплексне управління охороною праці на державному рівні, реалізує державну політику у цій сфері, розробляє за участі відповідних органів державні програми з охорони праці, координує діяльність державних органів та об'єднань підприємств із питань безпеки праці, а також разом із компетентними установами формує та переглядає систему показників обліку умов і безпеки праці. Крім цього, служба забезпечує міжнародне співробітництво у сфері охорони праці та здійснює державний нагляд за дотриманням вимог безпеки праці.

Рішення Держгірпромнагляду України, що належать до його компетенції, є обов'язковими для виконання всіма міністерствами, іншими центральними органами виконавчої влади, місцевими державними адміністраціями, органами місцевого самоврядування та підприємствами.

Фонд соціального страхування від нещасних випадків забезпечує профілактику виробничого травматизму та професійних захворювань, а також координує страхову діяльність, пов'язану з охороною праці. Міністерство праці та соціальної політики України проводить державну експертизу умов праці, визначає порядок проведення атестації робочих місць і контролює її відповідність нормативним актам з охорони праці, а також бере участь у розробленні нормативної документації у цій сфері.

Інші міністерства та центральні органи виконавчої влади в межах своїх повноважень формують науково-технічну політику галузі з питань охорони праці, розробляють і впроваджують заходи щодо підвищення рівня безпеки

праці, здійснюють методичне керівництво підприємствами галузі, співпрацюють із профспілками, організовують навчання та перевірку знань норм охорони праці серед керівників і спеціалістів, а за необхідності створюють професійні аварійно-рятувальні формування. Також вони здійснюють внутрішній контроль за станом охорони праці. Для виконання цих функцій у структурах міністерств та інших центральних органів виконавчої влади створюються спеціалізовані служби охорони праці.

Місцеві державні адміністрації та органи місцевого самоврядування в межах своєї території забезпечують реалізацію державної політики у сфері охорони праці, розробляють за участю профспілок місцеві програми щодо покращення безпеки праці, гігієни виробничого середовища та здійснюють контроль за дотриманням нормативно-правових актів. Для реалізації цих завдань при місцевих органах виконавчої влади формуються відповідні структурні підрозділи.

Управлінські структури підприємств забезпечують виконання вимог законодавства та нормативних актів з охорони праці безпосередньо в умовах конкретного виробництва. Їх діяльність спрямована на створення безпечних і нешкідливих умов праці, запобігання виробничому травматизму та професійним захворюванням, а також вирішення всіх питань охорони праці, пов’язаних із діяльністю підприємства. У своїй роботі вони взаємодіють із комісіями з питань охорони праці, профспілками та уповноваженими представниками трудових колективів.

Система управління охороною праці в межах конкретної організації або виробничого об’єкта є багаторівневою структурою, де найвищим рівнем виступає державне управління, а найнижчим — управління охороною праці безпосередньо на підприємстві. Проміжними рівнями можуть бути регіональне, відомче або галузеве управління.

Вихідні параметри СУОП визначаються на основі вимог нормативних документів, проектної документації, аналізу фактичного стану виробничого середовища та інших факторів, що впливають на безпеку праці. Саме тому

СУОП належить до категорії багатоконтурних систем управління, які можуть програмуватися та адаптуватися до умов конкретного виробництва. Багатоконтурність такої системи пояснюється складністю об'єкта управління, його інерційністю та складністю реалізації управлінських рішень.

Правовою основою функціонування СУОП є Конституція України, Кодекс законів про працю України, закони України «Про охорону праці» та «Про загальнообов'язкове державне соціальне страхування від нещасного випадку на виробництві і професійного захворювання, які спричинили втрату працездатності», а також укази Президента України, постанови Кабінету Міністрів України, нормативні документи Держгірпромнагляду, Міністерства охорони здоров'я, Міністерства праці та інших державних органів у сфері охорони праці.

Позитивний вплив упровадження систем управління охороною праці на рівні підприємств сьогодні визнається державними органами, роботодавцями та працівниками, оскільки такі системи сприяють зниженню рівня виробничих ризиків, підвищенню безпеки праці та покращенню продуктивності діяльності.

В дослідженнях проводиться реєстрація мовного сигналу з допомогою комп'ютера, а первинним перетворювачем є мікрофон. Оскільки така система в перспективі має використовуватись при ідентифікації користувачів, важливим є питання електробезпеки при роботі з таким і подібним електрообладнанням.

Основне завдання електробезпеки – мінімізувати можливість негативного впливу електричного струму на людину. Досягти цієї мети можна за допомогою таких заходів і засобів:

- безпечною і надійною конструкцією електроустановок;

- організаційними та технічними заходами щодо безпечної експлуатації електроустановок та використання електричної енергії;

- технічними засобами захисту.

В загальному, конструкція електроустановки має відповідати вимогам технічних умов і стандартів. При цьому, залежно від засобів електробезпеки, усі електротехнічні вироби поділяються на 5 класів: 0, 01,1, II, III.

Система для відбору мовних сигналів при їх дослідженнях на етапах проектування і тестування системи мовної ідентифікації користувачівналежатиме до 0 або 01 класу електробезпеки, тобто має лише робочу ізоляцію як засіб захисту (клас 0), або у випадку необхідності крім робочої ізоляції на корпусі системи є пристрій для підключення її до заземлювача або нульового захисного провідника (клас 01).

Стан ізоляції струмопровідних частин системи відбору повинен відповідати «Правилам використання електроустановок». Цими «Правилами» передбачене періодичне випробування ізоляції (2 рази на рік у приміщеннях зі складними умовами, підвищеною вологістю і 1 раз на рік у приміщеннях з нормальним середовищем). Ізоляція створює великій опір, який перешкоджає протіканню через неї струму. Опір ізоляції кожної системи має бути не меншим 0,5 МОм. Якщо опір ізоляції знижується на 50% від початкового, ізоляцію міняють.

4.3 Висновки до четвертого розділу

В розділі розглянуто ключові питання, що стосуються організації охорони праці та забезпечення безпеки під час роботи з комп’ютерною технікою й електрообладнанням. Проведений аналіз показав, що тривала робота за персональним комп’ютером може негативно впливати на функціональний стан організму людини, зокрема на зорову систему, нервову діяльність та загальний рівень працездатності. Саме тому важливе значення мають правильна організація робочого місця, дотримання санітарно-гігієнічних вимог, режимів праці та відпочинку, а також використання ергономічних меблів і технічних засобів.

У роботі також було проаналізовано структуру та принципи функціонування системи управління охороною праці в Україні. Встановлено, що СУОП є багаторівневою системою, яка охоплює державні органи управління, місцеві органи влади, підприємства та організації. Її функціонування спрямоване на забезпечення безпечних умов праці, зниження рівня виробничого травматизму та профілактику професійних захворювань.

ВИСНОВКИ

Вирішено актуальне науково-прикладне завдання, яке полягає в оптимізації алгоритмів та розробці інформаційної системи інтелектуального сортування даних із інтеграцією в Google Drive API для ефективного управління резервними копіями. За результатами проведеного дослідження та практичної реалізації сформовано такі підсумки.

Проаналізовано сучасні підходи до організації хмарних сховищ та методів управління даними. У ході дослідження розглянуто особливості структурування великих обсягів інформації, висвітлено методи інтелектуального сортування на основі контенту та алгоритми обробки файлів. На базі цього сформовано комплексні вимоги до розроблюваної інформаційної системи.

Досліджено можливості інтеграції з Google Drive API та особливості роботи з протоколом OAuth 2.0. Визначено оптимальні шляхи програмної взаємодії системи з хмарним сервісом, що дозволило забезпечити безпечну авторизацію та надійний обмін метаданими під час створення резервних копій.

Спроєктовано архітектуру інформаційної системи з використанням підходу MVC (Model-View-Controller) із сервісним шаром. Обґрунтовано вибір технологічного стеку (PHP, MySQL, Google APIs, Bootstrap), розроблено реляційну структуру бази даних, а також спроєктовано логіку алгоритмів класифікації файлів та взаємодії з хмарним API.

Практичним результатом стало те, що реалізовано вебдодаток BackupDriveApp для керування резервними копіями. Створено серверну частину та повноцінний інтерфейс користувача (Dashboard, особистий кабінет). Забезпечено інтеграцію з Google Drive, реалізовано модулі резервного копіювання, відновлення та автоматизованого впорядкування файлів на основі їхнього вмісту, а також впроваджено необхідні механізми захисту даних.

Окрім основної проєктної частини, досліджено питання організації охорони праці. Проаналізовано вплив тривалої роботи за ПК на організм людини, визначено основні санітарно-гігієнічні та ергономічні вимоги, а також розглянуто принципи функціонування СУОП для забезпечення безпечних умов праці.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Vyacheslav Nykytyuk, Vasyl Dozorskyy, Nataliia Kunanets, Volodymyr Pasichnyk, Oleksandr Matsiuk, Ihor Bodnarchuk: Electrical Probe-Signal Processing and Criterion for the Determination of Time Parameters of the Teeth Filling Material Polymerization Process in Dentistry. 4th IDDM 2021: Valencia, Spain. P. 54-63

2. Oleksii Duda, Nataliia Kunanets, Serhii Martsenko, Vyacheslav Nykytyuk, Volodymyr Pasichnyk. Information technology platform for the selection and analytical processing of information on COVID-19. 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT). Volume 2, Lviv, Ukraine 22-25 Sept. 2021. P. 231-328. Electronic ISBN:978-1-6654-4257-2, Print on Demand(PoD) ISBN:978-1-6654-4258-9, Electronic ISSN: 2766-3639, Print on Demand(PoD) ISSN: 2766-3655. DOI: 10.1109/CSIT52700.2021.9648839.

3. Oleksii Duda, Nataliia Kunanets, Serhii Martsenko, Vyacheslav Nykytyuk, Volodymyr Pasichnyk. COVID-19 data collections and analytical processing. 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT). Volume 2, Lviv, Ukraine 22-25 Sept. 2021. P. 252-257. Electronic ISBN:978-1-6654-4257-2, Print on Demand (PoD) ISBN:978-1-6654-4258-9, Electronic ISSN: 2766-3639, Print on Demand (PoD) ISSN: 2766-3655. DOI: 10.1109/CSIT52700.2021.9648839.

4. Vyacheslav Nykytyuk, Vasil Dozorskyi, Oksana Dozorska, Andrii Karnaukhov and Liubomyr Matiichuk. The Method of User Identification by Speech Signal. The 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP-2022) Ternopil, Ukraine, November 22-24, 2022. Vol-3309 urn:nbn:de:0074-3309-1. P.225-232. ISSN 1613-0073 DOI: 10.1425/jsdtl.

5. Ihor Bodnarchuk, Yuriy Skorenkyy, Taras Kramar, Oleksii Duda and Vyacheslav Nykytyuk. Use of Analytical Hierarchy Process in Scenarios Design for

a Digital Museum with XR components. The 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP-2022) Ternopil, Ukraine, November 22-24, 2022. Vol-3309 urn:nbn:de:0074-3309-1. P. 414-425. ISSN 1613-0073 DOI: 10.1425/jsdtl.

6. Kryazhych O., Itskovych V., Iushchenko K., Hrytsyshyna V., Bruvier D., Nykytyuk V., Bodnarchuk I. (2023) The use of abstract moore automaton to control the sensors of a service-oriented alarm and emergency notification network. Scientific Journal of TNTU (Tern.), vol 109, no 1, pp. 111–120. ISSN 2522-4433.

7. Dediv, L., Dozorska, O., Kukuruza, V., Nykytyuk, V., Kovalyk, S. Computer Simulation Modeling of Voice Signals in the Matlab Environment for the Task of Computerized Diagnostic Systems Testing. The 1st International Workshop on “Computer information technologies in Industry 4.0” (CITI-2023) will be held in Ternopil, Ukraine, from June 14 to 16, 2023. The Workshop is organized by the Faculty of Applied Information Technologies and Electrical Engineering of Ternopil Ivan Puluj National Technical University. 2023, 3468, pp. 257–262. Vol-3468 urn:nbn:de:0074-3468-8, ISSN 1613-0073.

8. Dozorskyi, V., Dediv, I., Sverstiuk, S., Nykytyuk, V., Karnaukhov, A. The Method of Commands Identification to Voice Control of the Electric Wheelchair. The Workshop is organized by the Faculty of Applied Information Technologies and Electrical Engineering of Ternopil Ivan Puluj National Technical University. The 1st International Workshop on “Computer information technologies in Industry 4.0” (CITI-2023) will be held in Ternopil, Ukraine, from June 14 to 16, 2023. The Workshop is organized by the Faculty of Applied Information Technologies and Electrical Engineering of Ternopil Ivan Puluj National Technical University. *2023,* 3468, pp. 233–240. Vol-3468 urn:nbn:de:0074-3468-8, ISSN 1613-0073.

9. Sverstiuk, A., Matiichuk, L., Polyvana, U., Stanko, A., Nykytyuk, V.. Analytical analysis of approaches to assessing the quality of life in smart cities. BAIT’2024: The 1st International Workshop on “Bioinformatics and applied

information technologies", October 02-04, 2024, Zboriv, Ukraine. CEUR Workshop Proceedings, 2024, 3842, pp. 75–91. ISSN: 1613-0073

10. Koroliuk, R., Nykytyuk, V., Tymoshchuk, V., Soyka, V., Tymoshchuk, D.. Automated monitoring of bee colony movement in the hive during winter season. BAIT'2024: The 1st International Workshop on "Bioinformatics and applied information technologies", October 02-04, 2024, Zboriv, Ukraine. CEUR Workshop Proceedings, 2024, 3842, pp. 147–156. ISSN: 1613-0073.

11. Google Drive, URL: https://workspace.google.com/products/drive/ (дата звернення: 07.05.2026)

12. Олег Денейка, Олег Гарасимчук. ВИКЛИКИ ТА СТРАТЕГІЇ ЗБЕРІГАННЯ ВЕЛИКИХ ОБСЯГІВ ДАНИХ У СУЧАСНОМУ СВІТІ

13. Якісна система управлінням даних, URL: https://wezom.com.ua/ua/blog/bazi-danih-viznachennya-ta-riznovidi (дата звернення: 07.05.2026)

14. Автомматизована класифікація даних, URL: https://oberig-it.com/statti/vazhlivist-avtomatizatsii-u-klasifik/ (дата звернення: 07.05.2026)

15. Дублювання інформації в сховищах, URL: https://dou.ua/forums/topic/54249/ (дата звернення: 07.05.2026)

16. Google Drive, URL: https://workspace.google.com/products/drive/ (дата звернення: 07.05.2026)

17. API Intagration, URL: https://developers.google.com/workspace/drive/api/guides/downloads (дата звернення: 07.05.2026)

18. Контент-орієнтована класифікація, URL: https://wezom.com.ua/ua/blog/matrix-kontenta (дата звернення: 07.05.2026)

19. Поєднання контентного аналізу та використання метаданих, URL: https://nasplib.isofts.kiev.ua/server/api/core/bitstreams/ce5a9251-1a09-4cc3-8840-2f4165d7e8e8/content (дата звернення: 07.05.2026)

20. Алгоритми класифікації та обробки файлів, URL: https://studfile.net/preview/5186977/page:11/ (дата звернення: 07.05.2026)

21. Підходи до організації резервного копіювання даних, URL: https://informatics.dp.ua/arkhivatsiya-rezervne-kopiyuvannya-zakhyst-informatsiyi/ (дата звернення: 07.05.2026)

22. Централізоване резервне копіювання, URL: https://denovo.ua/resources/cloud-services-how-they-work (дата звернення: 07.05.2026)

23. Способи створення резервних копій, URL: https://www.godaddy.com/uk-ua/help/sho-take-rezervne-kopiyuvannya-veb-sajtu-20318 (дата звернення: 07.05.2026)

24. Вимоги до інформаційних систем керування резервними копіями, URL: https://lnk.ua/TR6ulPdtz (дата звернення: 07.05.2026)

25. Архітектурні підходи до побудови вебдодатків, URL: https://blog.ithillel.ua/articles/web-application-architecture (дата звернення: 07.05.2026)

26. Монолітна архітектура, URL: https://qalight.ua/baza-znaniy/shho-take-monolitna-arhitektura/ (дата звернення: 07.05.2026)

27. Багаторівнева архітектура, URL: https://nook.svoboda.cx.ua/ukraincyam/shho-take-bagatorivneva-arkhitektura-v-rozpodileniy-sistemi.html (дата звернення: 07.05.2026)

28. Використання додаткового сервісного шару (Service Layer), URL: https://laravel.demiart.com/service-layer-design-pattern/ (дата звернення: 07.05.2026)

29. Поєднання архітектури MVC та сервісного шару, URL: https://javarush.com/ua/groups/posts/uk.2536.chastina-7-oznayomlennja-z-paternom-mvc-model-view-controller (дата звернення: 07.05.2026)

30. Авторизація через OAuth 2.0, URL: https://learn.microsoft.com/ru-ru/windows/apps/develop/security/oauth2 (дата звернення: 07.05.2026)

31. PHP docs, URL: https://www.php.net/docs.php (дата звернення: 07.05.2026)

32. Data base with google drive and php, URL: https://stackoverflow.com/questions/46828881/backup-files-to-google-drive-using-php (дата звернення: 07.05.2026)

33. Забезпечення безпеки на веб сайті з google drive, URL: https://material.security/workspace-resources/uncovering-google-drive-security-gaps-what-you-need-to-know (дата звернення: 07.05.2026)

34. Серверна частина додатку, URL: https://javarush.com/ua/groups/posts/uk.240.serveri-lknep-dlja-chaynikv (дата звернення: 07.05.2026)

35. Взаємодія Google Drive з OAuth 2.0, URL: https://wezom.com.ua/blog/chto-takoe-oauth-20-i-kak-rabotaet-avtorizatsiya-cherez-tokeny (дата звернення: 07.05.2026)

36. К.М. Онищенко. Аналіз методів обробки природної мови / К.М. Онищенко, Я.І. Данієль, Р.О. Каманєв.

37. В.В. Литвин. Інтелектуальні системи підтримки прийняття рішень / В.В. Литвин, 2019 – с. 248.

38. О.М. Спірін. Інформаційні технології та системи / О.М. Спірін, 2018 – с. 320.

39. М.П. Дивак. Моделі та методи аналізу інформаційних систем / М.П. Дивак, 2020 – с. 412.

40. В.П. Пасічник. Організація баз даних та знань / В.П. Пасічник, 2017 – с. 356.

41. І.В. Сергієнко. Основи проєктування інформаційних систем / І.В. Сергієнко, 2016 – с. 290.

42. Ian Foster. Cloud Computing: Principles and Paradigms / Ian Foster, Rajkumar Buyya, 2011 – с. 664.

43. Thomas Erl. Cloud Computing: Concepts, Technology & Architecture / Thomas Erl, Ricardo Puttini, Zaigham Mahmood, 2013 – с. 528.

44. Martin Kleppmann. Designing Data-Intensive Applications / Martin Kleppmann, 2017 – с. 616.

45. Robert C. Martin. Clean Architecture: A Craftsman’s Guide to Software Structure and Design / Robert C. Martin, 2017 – с. 432.

46. Josh Lockhart. Modern PHP: New Features and Good Practices / Josh Lockhart, 2015 – с. 264.

47. Luke Welling. PHP and MySQL Web Development / Luke Welling, Laura Thomson, 2017 – с. 976.

48. Robin Nixon. Learning PHP, MySQL & JavaScript / Robin Nixon, 2021 – с. 832.

49. William Stallings. Cryptography and Network Security: Principles and Practice / William Stallings, 2017 – с. 768.

50. Alex Xu. System Design Interview – An Insider’s Guide / Alex Xu, 2020 – с. 322.

51. Oleh Yasniy, Iryna Didych, Dmytro Tymoshchuk, Iaroslav Pasternak, Vyacheslav Nykytyuk, Hryhorii Shymchuk, Dmytro Radyk. Fatigue crack growth prediction of automotive steels using ensemble-based machine learning methods. Procedia Structural Integrity, VIII International Conference “In-service Damage of Materials: Diagnostics and Prediction“ Volume 81, (15 -17 October 2025.) 2026, P.116-122.

52. Хмарні сервіси: як вони працюють та чому стали стандартом, URL: https://denovo.ua/resources/cloud-services-how-they-work (дата звернення: 07.05.2026)

ДОДАТКИ

Додаток А1 – Тези конференції 1

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя
Маріборский університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Краківський економічний університет (Польща)
Вроцлавський економічний університет (Польща)
Університет «Опольська Політехніка» (Польща)
Національний університет «Полтавська політехніка імені Юрія Кондратюка»
Вінницький національний аграрний університет
Львівський національний університет ім. І. Франка
Головне управління Пенсійного фонду в Тернопільській області
Наукове товариство ім. Шевченка
Тернопільський обласний комунальний інститут післядипломної педагогічної освіти
Сумський державний педагогічний університет
Запорізький національний університет

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

XIV Міжнародної науково-технічної конференції молодих учених та студентів

11-12 грудня 2025 року

УКРАЇНА
ТЕРНОПІЛЬ – 2025

СЕКЦІЯ 5
КОМП'ЮТЕРНО-ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ ЗВ'ЯЗКУ

УДК 004.4
Д.А. Бойко
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

УДОСКОНАЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО СОРТУВАННЯ ДАНИХ ДЛЯ ІНТЕГРАЦІЇ В «GOOGLE DRIVE API" НА ОСНОВІ КОНТЕНТУ

D.A. Boiko

IMPROVEMENT OF THE INFORMATION SYSTEM OF INTELLIGENT DATA SORTING FOR INTEGRATION INTO "GOOGLE DRIVE API" BASED ON CONTENT

Інформаційні системи інтелектуального сортування даних «на основі контенту» передбачають аналіз змісту файлів та їхніх атрибутів (текстового наповнення, метаданих, семантики, типу та структури), а не лише за допомогою стандартних правил і назв. Такий підхід дає змогу автоматично визначати призначення або категорію документа, розпізнавати взаємозв'язки між файлами та оптимізувати процес їх класифікації. У контексті інтеграції з Google Drive API, використання аналізу контенту дозволяє підвищити точність сортування, пришвидшити доступ до релевантних даних та покращити користувацьку взаємодію, що відповідає актуальним задачам сучасних технологій щодо автоматизації обробки інформації та впровадження інтелектуальних систем управління даними. [1].

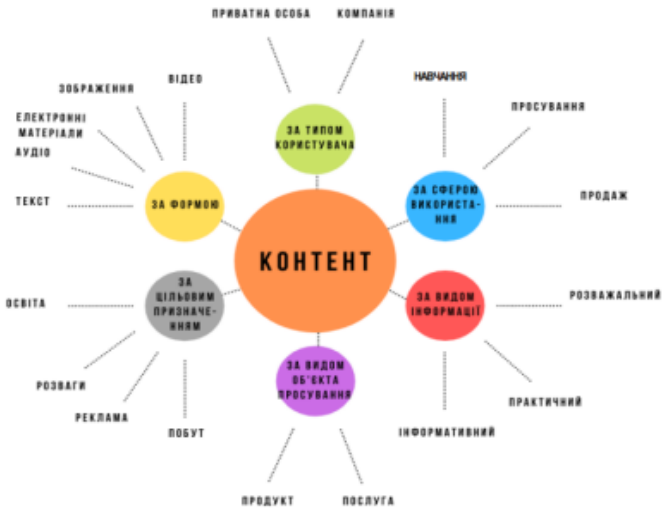

Рисунок 1. Структура класифікації контенту за різними ознаками

Однією з ключових актуальних технологій у процесі цифрової трансформації є використання прикладних програмних інтерфейсів (API), що забезпечують взаємодію між системами та доступ до хмарних сервісів. У контексті інтеграції інформаційної системи інтелектуального сортування даних із Google Drive API, API дозволяє автоматично отримувати метадані файлів, здійснювати доступ до вмісту, виконувати операції класифікації та переміщення без участі користувача. Завдяки підтримці сучасних стандартів безпеки (OAuth 2.0), API дає змогу реалізувати персоналізовану обробку даних, працювати з великими обсягами інформації у реальному часі та масштабувати алгоритми машинного навчання на хмарній інфраструктурі. Такий підхід сприяє підвищенню ефективності цифрових рішень, що відповідає тенденціям розвитку інтелектуальних веб-технологій і систем управління даними [2].

Для підвищення точності контент-аналізу в системі інтелектуального сортування даних можуть використовуватися гібридні моделі машинного навчання, що поєднують статистичні та математичні методи [3]. На етапі первинної класифікації файлів доцільним є застосування алгоритмів векторного представлення та порівняння, зокрема k-means та методу головних компонент (PCA) для виявлення структурної подібності на основі метаданих (розмір, формат, дата створення, частота появи ключових параметрів). Для оптимізації рішень щодо належності до певної категорії може бути використана логістична регресія або SVM, які дозволяють визначити ймовірність віднесення об'єкта до класу на підставі обраних ознак.

Реалізацію моделей можна виконати як у середовищі Python з використанням бібліотек машинного навчання, так і через інші технології (PHP, JavaScript) у разі інтеграції з відповідними сервісами. Такий підхід забезпечує адаптивність системи, можливість подальшого донавчання та масштабування моделі разом зі зростанням обсягів контенту [4].

У порівнянні з підходами, орієнтованими виключно на пошук за ключовими словами або метаданими, запропонована модель забезпечує багатовимірний аналіз, включаючи статистичні, структурні та змістові ознаки. Крім того, інтелектуальне сортування дозволяє автоматично оновлювати категорії на основі нового користувацького контенту, тоді як класичні системи потребують ручного коригування. Це значно підвищує швидкість обробки даних, знижує ризик помилки та робить систему масштабованою в умовах зростання обсягів інформації [5].

Подальші дослідження можуть зосередитися на інтеграції додаткових методів аналізу контенту, таких як глибинне навчання для розпізнавання семантичних зв'язків та контекстуальних залежностей між файлами, а також на впровадженні адаптивних механізмів рекомендацій для користувачів. Ці напрямки відкривають можливості для підвищення ефективності класифікації, забезпечення більш інтелектуальної взаємодії із даними та розвитку масштабованих систем управління інформацією у хмарних середовищах. Крім того, варто звернути увагу на оптимізацію продуктивності систем при обробці великих обсягів даних, безпеку при взаємодії з хмарними сервісами, управління версіями та історією змін файлів, а також на можливість інтеграції з різноманітними джерелами даних.

Література

1. Структури даних та алгоритми URL: https://studfile.net/preview/10025806/page:21/. Доступ до ресурсу: 30.11.2025

2. Brenda Jin. Designing Web APIs / Brenda Jin, Saurabh Sahni, Amir Shevat.

3. Григорова Т.А. Дослідження методів машинного навчаня для пошуку інформації / Т.А. Григорова, В.П. Ляшенко, О.О. Москаленко (Статті. Кременчуцький національний університет ім. М. Остроградського)

4. Карабін Р.М., Литвиненко Я.В. Огляд бібліотек машинного навчання для мови python.

5. Google Drive API Overview URL: https://developers.google.com/workspace/drive/api/guides/about-sdk (дата звернення: 30.11.2025)

Додаток А2 – Тези конференції 2

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XIII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

17-18 грудня 2025 року

ТЕРНОПІЛЬ
2025

УДК 004.4
Д. Бойко
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

УДОСКОНАЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО СОРТУВАННЯ ДАНИХ ДЛЯ ІНТЕГРАЦІЇ В «GOOGLE DRIVE API» НА ОСНОВІ КОНТЕНТУ

UDC 004.4
D. Boiko

IMPROVEMENT OF THE INFORMATION SYSTEM OF INTELLIGENT DATA SORTING FOR INTEGRATION INTO "GOOGLE DRIVE API" BASED ON CONTENT

Сучасні інформаційні моделі та технології все частіше орієнтовані на автоматизацію роботи з великими масивами даних, що зумовлює потребу в інтелектуальних системах їх аналізу й обробки. На цій основі розгортається технологічна платформа BackupDriveApp – компактний PHP-додаток, що використовує Google Drive API, MySQL та архітектуру MVC+Service для безпечного зберігання й керування резервними копіями користувачів. [1] Архітектура системи реалізована у вигляді модульної багаторівневої структури, де кожен компонент виконує окрему функціональну роль у процесі інтелектуального сортування даних. Логіка обробки запитів розділена на маршрутизацію, сервісний рівень та модуль аналізу контенту, що забезпечує відокремлення бізнес-процесів від операцій взаємодії з Google Drive API. [2]

Запит → маршрутизація → логіка → обробка даних → відповідь

Рисунок 1. Послідовність обробки запиту в системі

База даних моделює зв'язок користувачів і їхніх бекапів, а функціональні модулі забезпечують реєстрацію, автентифікацію, завантаження файлів, інтеграцію з Drive, обробку токенів, відновлення та видалення копій згідно визначених алгоритмів взаємодії з API [3]. Завдяки поєднанню цих компонентів формується алгоритмічна основа системи, що охоплює валідацію, фільтрацію, контроль доступу, послідовність операцій із файлами та використання зовнішнього сервісу як формалізованого ресурсу з чіткими правилами запиту й відповіді. Наукова новизна полягає у формуванні контент-орієнтованої моделі сортування, що враховує внутрішні атрибути файлів та дозволяє автоматично визначати їхню категорію перед передачею до Google Drive API [4]. Подальший розвиток проєкту передбачає масштабування логіки обробки даних, впровадження автоматизованих моніторингових механізмів, оптимізацію взаємодії з API та розширення функціональності системи у напрямі складніших моделей резервування й аналізу інформації.

Література

1. Google Drive API Overview [Електронний ресурс] : https://developers.google.com/workspace/drive/api/guides/about-sdk Доступ до ресурсу: 09.12.2025.
2. Ben Shaw. Web Development with Django: Learn to build modern web applications with a Python-based framework / Ben Shaw, 2021 – с. 826.
3. Brenda Jin. Designing Web APIs / Brenda Jin, Saurabh Sahni, Amir Shevat.
4. Luis Serrano. Grokking Machine Learning / Luis Serrano, 2021 – с. 512.