НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
_____комп'ютерних наук_____
(назва кафедри)

_____ /_____Голуб Б.Л._____/
(підпис) (ПІБ)

"___" _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Розробка та створення автоматизованого робочого місця працівника архіву»

Спеціальність 122 – «Комп'ютерні науки»

Гарант освітньої програми
_____д.е.н., професор_____ _____ _____Руденський Р.А.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи
_____к.е.н., доц._____ _____ ___Саяпін С.П._
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _____Ромашов Дмитро Сергійович_____
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____комп'ютерних наук_____
(назва кафедри)

___к.т.н., доцент_____Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"___" _____ 2025 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту
Ромашов Дмитро Сергійович

Спеціальність 122 – «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Розробка та створення автоматизованого робочого місця працівника архіву» затверджена наказом ректора НУБіП України від 16.12.2024 № 2246с
2. Термін подання завершеної роботи на кафедру_____
(рік, місяць, число)
3. Вихідні дані до роботи: надання інформації про облік документів в архіві приватної організації в електронному вигляді, створення візуальних звітів у вигляді графіків.
4. Перелік питань, що розглядаються:
   - Аналіз проблемної області
   - Моделювання предметної області
   - Проєктування програмної системи
   - Впровадження та експлуатація системи

Дата видачі завдання "___"_____ 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____Саяпін С.П._
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання:_____/ Ромашов Д.С. /
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У цій бакалаврській роботі представлено розробку та впровадження автоматизованого робочого місця працівника архіву. Проект передбачає створення веб-інформаційної системи, яка оптимізує процеси реєстрації, пошуку, видачі та звітності документів. Система розроблена для спрощення повсякденних завдань для архівного персоналу, покращення доступу до даних та підвищення ефективності роботи.

Веб-додаток розроблено з використанням PHP і фреймворку Symfony, а реляційною базою даних є MySQL. Додаток підтримує створення та класифікацію документів, розширені можливості пошуку та фільтрації, можливість видачі документів співробітникам, формування різноманітних аналітичних звітів. Ролі користувачів і механізми контролю доступу забезпечують безпечне поводження з архівними даними.

Система була розроблена відповідно до сучасної практики розробки програмного забезпечення та фокусується на зручності використання, масштабованості та безпеці. Це рішення сприяє цифровій трансформації управління документами та забезпечує надійний інструмент для працівників архіву як у державному, так і в приватному секторах.

ABSTRACT

This bachelor’s thesis presents the development and implementation of an automated workplace for an archive worker. The project involves the creation of a web-based information system that streamlines the processes of document registration, search, issuance, and reporting. The system is designed to simplify daily tasks for archive personnel, improve data accessibility, and enhance operational efficiency.

The web application is developed using PHP and the Symfony framework, with MySQL serving as the relational database. The application supports the creation and classification of documents, advanced search and filtering capabilities, the ability to issue documents to employees, and the generation of various analytical reports. User roles and access control mechanisms ensure secure handling of archival data.

The system was developed in accordance with modern software development practices and focuses on usability, scalability, and security. This solution contributes to the digital transformation of document management and provides a reliable tool for archive staff in both public and private sectors.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання бакалаврської роботи	Строк виконання етапів бакалаврської роботи	Примітка
1	Отримання завдання	21 лютого 2025	
2	Аналіз предметної області	березень 2025	
3	Моделювання предметної області	березень 2025	
4	Проектування програмної системи	квітень 2025	
5	Розгортання та експлуатація програмного забезпечення	квітень- травень 2025	
6	Економічне дослідження розробки та експлуатації програмної системи	квітень- травень 2025	
7	Оформлення записки	травень 2025	
8	Перевірка на плагіат	2 червня 2025	
9	Проходження нормо контроля	29 травня – 4 червня 2025	
10	Проходження передзахисту	5-7 червня 2025	
11	Захист роботи	14-16 червня 2025	

Студент _____ Ромашов Д.С.

(підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

доцент к.т.н. _____ _____

(науковий ступінь та вчене звання) (підпис) (ПІБ)

ЗМІСТ

ВСТУП

У сучасних організаціях ефективне управління архівними документами відіграє вирішальну роль у забезпеченні безперебійного функціонування внутрішніх процесів. Архіви служать сховищем важливих записів, включаючи юридичні документи, контракти, листування та історичні дані, які мають бути легкодоступними та надійно зберігатися. Однак багато установ все ще покладаються на ручні або частково оцифровані методи організації та керування своїми архівами, що часто призводить до неефективності, втрати даних та обмеженого доступу.

Перехід на цифрові рішення став радше необхідністю, ніж можливістю. Автоматизація архівних процесів допомагає зменшити людські помилки, заощадити час і підвищити загальну продуктивність персоналу архіву. Автоматизоване робоче місце для архівного працівника може значно покращити обробку документів за рахунок інтеграції таких функцій, як створення документів, структуроване зберігання, ефективний пошук, видача працівникам і статистична звітність.

Ця бакалаврська робота присвячена розробці та створенню веб-інформаційної системи, яка слугує автоматизованим робочим місцем для архівного працівника. Основна **мета** – спростити повсякденні завдання, пов’язані з керуванням документами, забезпечивши зручним інтерфейсом для взаємодії з архівними даними.

Система розроблена з використанням сучасних веб-технологій, зокрема PHP і фреймворку Symfony для бекенд-розробки, а також MySQL як системи керування базами даних. Програма забезпечує рольовий доступ, підтримує повний життєвий цикл обробки документів і надає інструменти для створення звітів на основі використання документів.

Актуальність цієї теми полягає у зростаючому попиті на цифрові рішення в адміністративному середовищі та необхідності заміни застарілих,

неефективних ручних процесів надійними та масштабованими системами. Розробка цієї системи не тільки сприяє цифровій трансформації документообігу, але й служить практичним прикладом застосування сучасних методологій розробки програмного забезпечення для вирішення реальних проблем.

Об’єктом дослідження в цій роботі є процес організації та ведення архівів документів в інституційному та корпоративному середовищах. Це включає створення, каталогізацію, зберігання, пошук і контрольований доступ працівників до документів. У роботі розглядаються типові робочі процеси та виклики, з якими зустрічаються архівні працівники, і спрямовано на розробку технічного рішення, яке задовольнить ці потреби шляхом автоматизації та оцифрування.

Основні завдання цього дослідження включають:

- аналіз існуючих практик управління документами в архівах;
- визначення функціональних і технічних вимог;
- проектування архітектури системи та структури бази даних;
- впровадження основних функціональних можливостей;
- перевірка рішення шляхом тестування.

Отримана система призначена для підвищення ефективності роботи, забезпечення безпечного доступу до даних і забезпечення масштабованої основи для майбутніх удосконалень.

До складу цієї роботи входить аналіз існуючих систем, формулювання функціональних і технічних вимог, проектування та впровадження програмного продукту, тестування та оцінка результатів.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Опис предметної області

Архіви документів є фундаментальною частиною інфраструктури будь-якої установи чи організації. Вони зберігають записи, важливі для юридичних, історичних, адміністративних та оперативних цілей. Ці записи можуть включати в себе особові справи, договори, звіти, листування, фінансові документи та різні форми внутрішньої документації. Належна організація, зберігання, пошук і захист цих документів безпосередньо впливають на ефективність і відповідність установи.

У багатьох випадках архівами все ще керують застарілими або ручними методами. Паперові системи та прості файлові каталоги без індексації чи категоризації є звичайними. Ці підходи не тільки неефективні, але й створюють такі ризики, як фізичне пошкодження документів, неправильне розташування та несанкціонований доступ. Відсутність структурованих механізмів пошуку та метаданих ще більше ускладнює пошук, що часто призводить до значних втрат часу для працівників, які намагаються знайти документи.

Крім того, багато організацій, які почали перехід на цифрові рішення, часто використовують фрагментовані інструменти — окремі бази даних, електронні таблиці чи загальне хмарне сховище, — які не призначені спеціально для архівних потреб. Ці інструменти можуть не мати дуже важливих функцій, таких як керування ролями користувачів, відстеження видачі документів, журналу аудиту та функції звітності. Як наслідок, працівники архіву зустрічаються з дедалі більшими труднощами щодо підтримки порядку, забезпечення своєчасного доступу та формування звітів для керівників чи аудиторів [3].

Відсутність уніфікованого та спеціалізованого цифрового рішення перешкоджає продуктивності архівних відділів і призводить до

неузгодженості даних, обмеженої можливості відстеження та загальної неефективності робочого процесу. Існує очевидна потреба в системі, адаптованій до архівної області, системі, яка включає найкращі практики в управлінні даними, контролі доступу та автоматизації.

Крім того, нормативні вимоги щодо конфіденційності даних, періодів зберігання та прав доступу стають серйознішими як у державному, так і в приватному секторах. Відповідність таким правилам важко підтримувати в традиційних системах. Автоматизоване робоче місце для співробітника архіву може вирішити ці проблеми шляхом впровадження стандартизованих процесів, структурованого зберігання даних і механізмів безпечного доступу.

Тому розробка веб-автоматизованого робочого місця працівників архіву є актуальною та потрібною. Результат пропонує багатообіцяюче рішення для модернізації архівних операцій, зменшення людських помилок, спрощення доступу до документів і підвищення загальної продуктивності організації.

Предметом цього дослідження є процеси, завдання та технології організації та управління архівною документацією в установі. Архіви відіграють вирішальну роль у збереженні інституційної пам’яті, підтримці прийняття рішень, виконанні вимог законодавства та забезпеченні доступу до історичних та оперативних записів. Працівник архіву (або архіваріус) відповідає за систематичний облік, класифікацію, зберігання, пошук, видачу та захист документів протягом усього їх життєвого циклу.

Архівна робота зазвичай включає як адміністративні, так і технічні завдання. В адміністративному плані архіваріус веде облік вхідних і вихідних документів, забезпечує дотримання правил зберігання документів і координує доступ до файлів між співробітниками. Технічно архівіст наглядає за системою зберігання, керує індексами або каталогами, контролює фізичний стан документів (у випадку паперових архівів) і забезпечує належну цифрову організацію в електронних архівах [3].

Зі збільшенням обсягу документів і зростаючою потребою у швидкому та безпечному доступі до інформації традиційні методи архівування стають менш ефективними. Це особливо вірно в установах з великим документообігом або різноманітними потребами користувачів. Щоб вирішити ці проблеми, тематична область розширюється за рахунок автоматизованих інформаційних систем, які підтримують основні архівні функції.

Автоматизоване робоче місце працівника архіву – спеціалізоване програмне рішення, призначене для полегшення всіх етапів роботи з документами: від створення та класифікації до пошуку, контролю доступу, видачі та звітності. Такі системи, як правило, інтегровані з центральною базою даних і пропонують доступ на основі ролей, щоб гарантувати, що лише авторизовані користувачі можуть переглядати або змінювати певну інформацію.

Ключові компоненти предметної області включають:

- класифікація документів і керування метаданими;
- контроль користувачів і доступу;
- видача та відстеження документів;
- механізми пошуку та фільтрації;
- інструменти архівної звітності;
- захист даних і резервне копіювання.

Предметна область також включає відповідні стандарти та правові норми, пов’язані з обробкою документів, такі як графіки зберігання, закони про конфіденційність даних та інституційні політики. Розуміючи та аналізуючи ці елементи, розробка автоматизованої системи може ефективно задовольнити реальні потреби працівників архіву та забезпечити довгострокову стійкість і відповідність.

1.2 Огляд існуючих рішень

На сьогоднішній день існує чимало програмних рішень для електронного документообігу та архівування, які використовуються у державних установах, медичних закладах, освітніх організаціях та приватних компаніях. Проте більшість із них є універсальними системами або комерційними продуктами з широким функціоналом, який не завжди враховує специфіку роботи саме архівного працівника.

Серед сучасних програмних засобів, що забезпечують електронний документообіг в Україні, особливу популярність здобула система M.E.Doc (Мій електронний документ). Вона створена з акцентом на автоматизацію бухгалтерських процесів та податкової звітності, що робить її незамінним інструментом для фінансових підрозділів підприємств. Програмне забезпечення надає можливість формувати, зберігати та обмінюватися електронними документами, використовуючи електронний підпис, який надає юридичну силу всім діям [1].

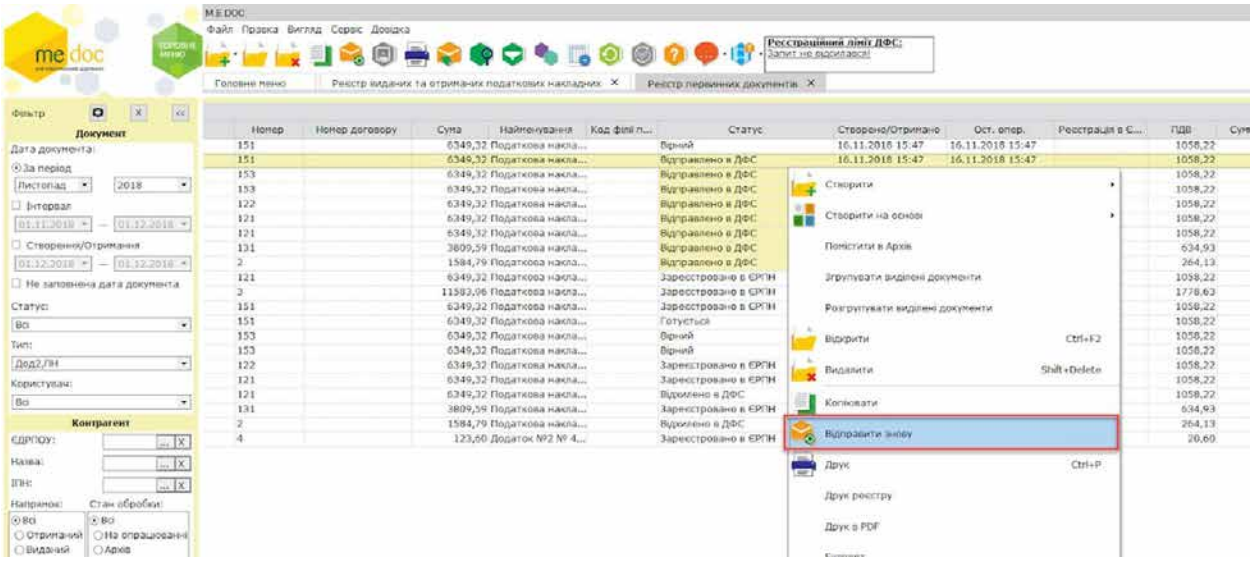

Рис. 1 Інформаційна система “M.E.Doc”

Однією з ключових переваг цієї системи є інтеграція з державними установами — податковою службою, пенсійним фондом, органами статистики — що дозволяє підприємствам подавати звітність в електронному форматі безпосередньо з інтерфейсу програми. Крім того, платформа забезпечує електронний обмін первинними документами між контрагентами, зберігаючи їх у захищеному електронному архіві з доступом до історії змін і журналу дій.

Проте попри потужний функціонал у сфері звітності, можливості M.E.Doc залишаються обмеженими, коли мова йде про організацію внутрішнього архіву установи. Система не передбачає спеціалізованих інструментів для ведення обліку видачі документів, побудови внутрішньої структури архіву чи створення кастомізованих звітів для аналізу документообігу. Також варто зазначити, що закритість коду та платна ліцензія обмежують можливість адаптації системи під специфічні потреби установ, де архівування є ключовим елементом повсякденної діяльності [1].

Загалом, M.E.Doc добре справляється зі своїми завданнями в галузі податкового та фінансового обліку, проте для повноцінної автоматизації архівного процесу, особливо в контексті створення індивідуального робочого місця архівіста, ця система не є оптимальним рішенням. Це підкреслює актуальність розробки більш гнучких та адаптивних веб-рішень, здатних повністю відповідати специфіці архівної справи.

Розглянемо інше програмне рішення на ринку.

Zoho Docs — це хмарна платформа для керування документами, яка є частиною екосистеми офісних сервісів компанії Zoho Corporation. Вона орієнтована на корпоративних користувачів і надає широкий набір інструментів для зберігання, спільної роботи, редагування та обміну документами в режимі реального часу. Основна мета сервісу — забезпечити централізоване, безпечне та зручне середовище для роботи з цифровими файлами [2].

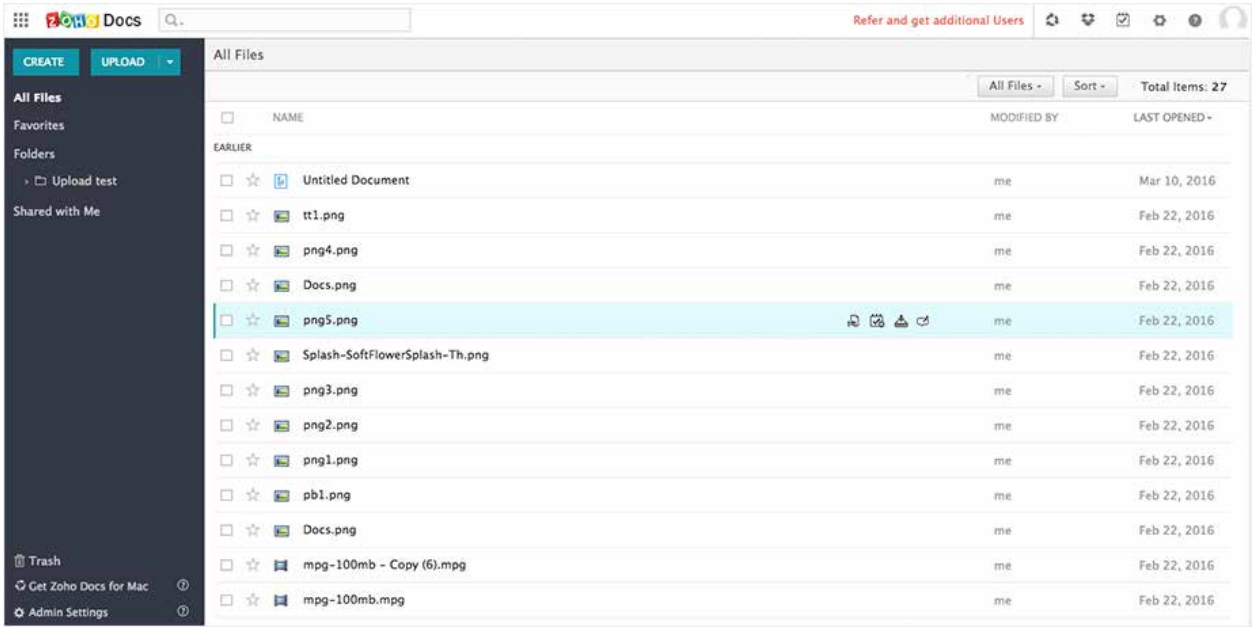

Рис. 2 Інформаційна система “Zoho Docs”

Серед сильних сторін платформи варто виділити повну інтеграцію з іншими продуктами Zoho, такими як CRM, Projects, Mail та інші бізнес-інструменти. Користувачі можуть створювати документи, таблиці та презентації безпосередньо в браузері, використовуючи вбудовані редактори. Завдяки підтримці ролей і прав доступу, система дозволяє ефективно розподіляти повноваження між співробітниками та контролювати всі дії з файлами. Хмарна природа сервісу дає змогу працювати з документами з будь-якої точки світу, використовуючи лише підключення до Інтернету.

Водночас, попри універсальність функціоналу, Zoho Docs не є вузькоспеціалізованим рішенням для архівної діяльності. Система не підтримує повноцінну модель обліку фізичних або службових документів, не містить можливостей реєстрації видачі архівних одиниць або побудови детальних статистичних звітів щодо їх використання. Крім того, певним бар’єром для державних установ або організацій зі специфічними вимогами до зберігання даних може стати іноземне походження платформи та обмежений контроль над інфраструктурою [2].

Таким чином, Zoho Docs добре підходить для організації спільної роботи над цифровими документами, однак її функціонал не охоплює повного спектру задач, які стоять перед архівним працівником. Це робить її менш

ефективною у контексті автоматизації архівних процесів, де важливою є не лише зручність зберігання, а й дотримання внутрішніх регламентів, облік видачі документів та збереження історичних записів.

Серед інших готових рішень можна виділити OpenText.

OpenText — цеплатформа для управління корпоративною інформацією (Enterprise Information Management, EIM), яка спеціалізується на автоматизації документообігу, а також довгостроковому зберіганні та архівуванні даних. Рішення компанії OpenText активно використовуються великими організаціями, державними установами та фінансовими структурами, де особливо важливими є безпека, контроль доступу та підтримка стандартів збереження документів.

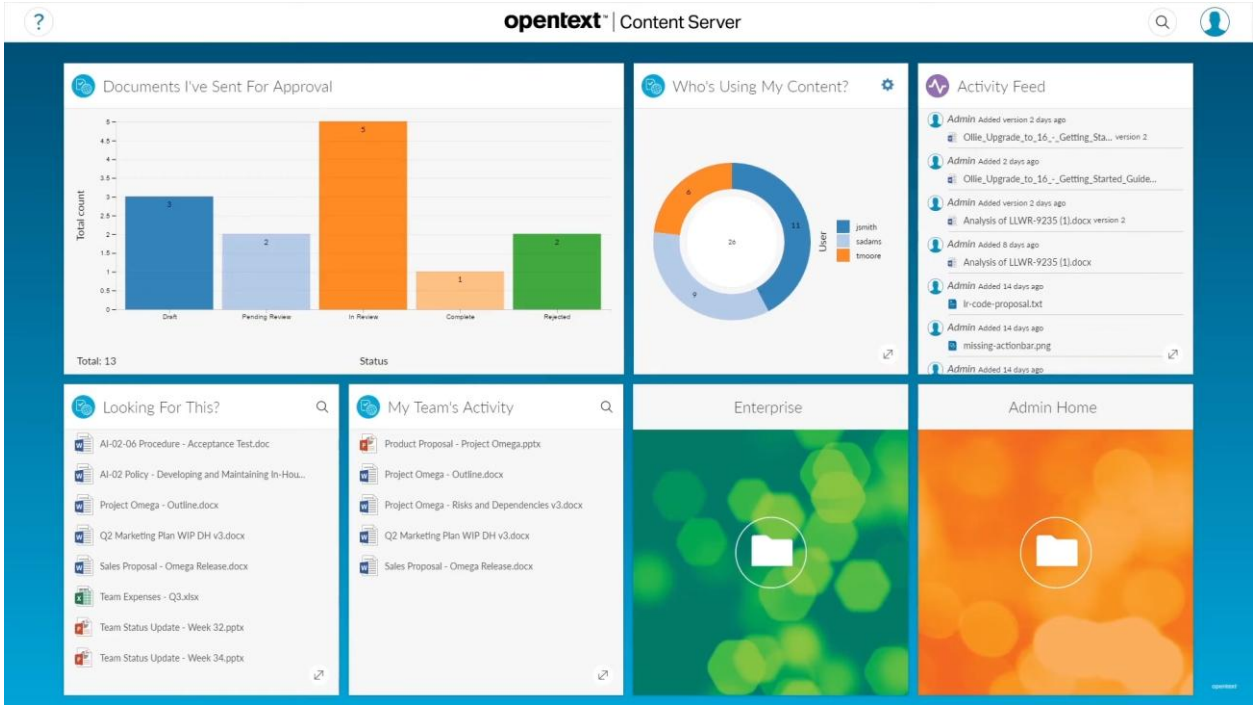

Рис. 3. Інформаційна система “OpenText Content Suite”

Однією з ключових переваг OpenText є її архітектура, яка дає адаптувати систему під конкретні бізнес-потреби. Основні функцій системи, які мають значення у архівній справі є централізоване сховище з підтримкою метаданих, класифікації та тегування архівних об’єктів. Також є автоматизоване керування життєвим циклом документів — від їх створення та

затвердження до архівування та видалення. Серед переваг OpenText можна виділити управління правами доступу на рівні ролей, користувачів, категорій документів та здатність ведення обліку видачі архівних документів, журналів запитів та дій з файлами. Можна виділити функції пошуку, які охоплюють як зміст документів, так і пов’язані метадані.

Крім того, OpenText підтримує інтеграцію з зовнішніми системами (ERP, CRM, державними реєстрами), що дозволяє побудувати цілісний цифровий архів у межах єдиного середовища.

Система OpenText є вузьконаправленим рішенням, орієнтованим на високий рівень документального контролю. Вона підтримує електронні підписи та сертифікацію дій, що має особливе значення для юридичних документів, масштабованість, що дозволяє використовувати систему у великих архівах з мільйонами документів.

Попри свої широкі можливості, впровадження OpenText потребує значних ресурсів таких як: складність налаштування та адміністрування системи, висока вартість ліцензії та підтримки, що може виключає можливість придбання невеликими організаціями. Також інтерфейс не є інтуєтивно-зрозумілим, що може вимагати спеціального навчання для архівних працівників.

На відміну від універсальних хмарних рішень, таких як **Zoho Docs, Google Workspace** або **Microsoft OneDrive, OpenText** є професійною системою керування електронним архівом, яка спеціально розроблена для забезпечення повного циклу обробки документів у відповідності до архівних і регуляторних вимог. Якщо Zoho Docs або Google Workspace здебільшого орієнтовані на спільну роботу з офісними файлами, базове зберігання та онлайн-редагування, то OpenText реалізує **повноцінний життєвий цикл**

документа: від створення, реєстрації та класифікації – до архівування, обмеженого доступу, тривалого зберігання й знищення згідно з політиками.

1.3 Постановка завдання

Працівник архівного відділу несе відповідальність за облік, збереження та видачу документів, що мають юридичну, адміністративну чи історичну цінність. Щоб ефективно виконувати ці задачі, необхідно мати інструмент, який дозволяє не лише зберігати документи в структурованому вигляді, але й відстежувати їх використання, контролювати переміщення, а також своєчасно формувати звіти для керівництва або інших підрозділів.

Для цього передбачається створення автоматизованого робочого місця архівіста, яке ґрунтується на веб-додатку з єдиною базою даних. Система повинна забезпечити можливість введення, редагування та перегляду інформації про кожен документ, включаючи його категорію, дату створення, автора, місце зберігання та статус. Крім того, програма повинна вести облік операцій із видачі документів працівникам підприємства, із зазначенням дати, одержувача та терміну повернення.

Основні функції розроблюваної системи зосереджені на швидкому доступі до інформації про:

- місцезнаходження та статус документів;
- історію видачі документів співробітникам;
- документи, що очікують на повернення;
- статистику за категоріями або періодами використання.

Інтерфейс програми реалізується у вигляді діалогового меню, що дозволяє користувачу зручно перемикатися між основними функціями. У разі помилкового введення даних або необхідності скасування операції система повинна передбачати механізм відміни або корекції запису. Таким чином,

архіваріус отримує інструмент, що значно полегшує рутинну роботу, мінімізує кількість помилок та забезпечує прозорість архівних процесів.

1.4 Функціональні та нефункціональні вимоги

Функціональні вимоги:

1. система повинна підтримувати створення облікових записів користувачів із рольовим доступом (архівіст, співробітник, адміністратор) і забезпечувати безпечну функцію входу;
2. користувачі повинні мати можливість додавати нові документи до бази даних, вказуючи такі метадані, як назва, автор, дата створення, категорія та опис. Він також має підтримувати долучення сканованих копій або цифрових файлів;
3. документи слід класифікувати за категоріями (наприклад, накази, контракти, службові записки) і зберігати в структурованому вигляді, щоб можна було легко переглядати та керувати ними;
4. система повинна запропонувати потужну функцію пошуку з фільтрами на основі ключових параметрів, таких як назва, дата, тип, статус або призначений користувач;
5. архіваріус повинен вміти реєструвати видачу документів із зазначенням імені одержувача, мети використання, дати видачі та передбачуваної дати повернення;
6. система повинна автоматично формувати звіти про використання документів, періодичність видачі, популярні категорії, прострочені повернення тощо;
7. для прозорості всі дії користувача (наприклад, додавання, редагування, видалення, видача документів) повинні бути записані та доступні для перегляду системним адміністраторам.

Нефункціональні вимоги:

1. система повинна швидко реагувати на дії користувача, особливо при пошуку або завантаженні документів, навіть при зростаючій базі даних;
2. система має бути розроблена таким чином, щоб підтримувати майбутнє зростання кількості користувачів і документів без значного зниження продуктивності;
3. дані повинні надійно зберігатися та передаватись. Дозволи на основі ролей повинні запобігати несанкціонованому доступу до конфіденційних документів і функцій системи [4];
4. інтерфейс повинен бути інтуїтивно зрозумілим і зручним, дозволяючи співробітникам архіву швидко навчатися та ефективно взаємодіяти з системою;
5. програма має бути стабільною, з мінімальним часом простою та здатністю відновлюватися після неочікуваних збоїв;
6. Код і архітектура системи повинні передбачати майбутні оновлення, виправлення помилок і вдосконалення функцій з мінімальними затримками;
7. Слід підтримувати регулярне резервне копіювання бази даних документів, а система повинна підтримувати легке відновлення у разі втрати даних.

1.5 Вимоги до інтерфейсу користувача

Інтерфейс користувача (UI) автоматизованого робочого місця для співробітника архіву має бути розроблений з урахуванням простоти та зрозумілості, щоб забезпечити легкість використання та ефективний пошук. Він повинен відповідати всім ролям користувача (архівіст, співробітник, адміністратор), пропонуючи чіткий та інтуїтивно зрозумілий макет. Основна мета полягає в тому, щоб скоротити час навчання для нових користувачів,

одночасно надаючи доступ до необхідних інструментів для керування та отримання архівних документів.

Після входу в систему користувачі повинні зустріти інформаційну панель, яка представляє огляд важливої інформації. Це може включати кількість документів, що очікують на випуск, прострочені елементи та останні дії (журнал аудиту). Така інформаційна панель допоможе користувачам бути в курсі стану системи, не перевантажуючись зайвою інформацією.

Інтерфейс користувача повинен підтримувати доступ на основі ролей, надаючи кожному користувачеві персоналізований досвід. Наприклад, архівісти повинні мати доступ до функцій керування документами та інструментів створення звітів, тоді як працівникам може бути дозволено лише шукати, переглядати та запитувати документи. З іншого боку, адміністратори повинні мати додаткові можливості, такі як керування користувачами та налаштування системи.

Ключовою особливістю інтерфейсу є функція пошуку, яка має бути легкодоступною на всіх сторінках. Цей інструмент пошуку повинен дозволити користувачам швидко знаходити документи, вводячи ключові слова або застосовуючи фільтри на основі метаданих, таких як назва, категорія, автор або дата створення. Слід також запропонувати розширені фільтри для уточнення результатів пошуку на основі конкретних критеріїв, таких як статус документа або призначений користувач.

Управління документами має бути простим. Система повинна дозволяти користувачам створювати нові записи, без зайвих зусиль, включаючи такі метадані, як назва, категорія та опис. Користувачі також повинні мати можливість прикріплювати цифрові файли або скановані копії документів. Добре організований список документів повинен дозволяти користувачам легко переглядати, редагувати або видаляти записи [5].

Що стосується видачі документів, інтерфейс має представляти зрозумілу та просту форму, де користувачі можуть вводити такі деталі, як ім’я одержувача, мета документа та очікувана дата повернення. Система повинна

автоматично оновлювати статус кожного документа (наприклад, «видано», «повернено») на основі дій користувачів.

Окрім керування документами, система має запропонувати надійні можливості звітування. Інтерфейс звітності повинен дозволяти користувачам створювати різні типи звітів, наприклад кількість виданих документів, прострочені позиції та категорії документів. Користувачі повинні мати можливість налаштовувати параметри звіту, наприклад діапазони дат або типи документів, і експортувати результати у такі формати, як PDF або Excel.

Щоб тримати користувачів в курсі, система повинна надсилати сповіщення про важливі події, такі як терміни повернення документів або їх прострочення. Ці сповіщення мають бути видимими на інформаційній панелі та, якщо необхідно, надсилатися електронною поштою, щоб користувачі не пропустили важливу інформацію.

Перевірка введених даних також має вирішальне значення для забезпечення цілісності даних. Система повинна включати вбудовані перевірки підтвердження, які запобігають введенню неправильних або неповних даних. Наприклад, дати мають бути у правильному форматі, а користувачам не можна дозволяти вводити повторювані записи. Будь-які помилки, виявлені під час введення даних, мають бути чітко виявлені за допомогою корисних повідомлень про помилки.

Узгодженість інтерфейсу є ключем до створення бездоганної взаємодії з користувачем. Це означає збереження однакових колірних схем, шрифтів і стилів кнопок, а також дотримання усталених практик веб-розробки, таких як адаптивний дизайн і стандарти доступності. Система також має бути доступною для користувачів з обмеженими можливостями, підтримувати такі функції, як програми зчитування з екрана, навігація з клавіатури та режими високої контрастності.

Для користувачів, які розмовляють різними мовами, система повинна пропонувати багатомовну підтримку, щоб гарантувати, що вони можуть взаємодіяти з системою мовою, яку вони люблять. Також має бути доступним

розділ довідки, який містить вказівки щодо використання різних функцій, наприклад підказки або посилання на розділ із поширеними запитаннями.

Враховуючи ці принципи дизайну, користувальницький інтерфейс допоможе працівникам архіву ефективно керувати своїм робочим навантаженням, скорочуючи час і зусилля, витрачені на процеси вручну, забезпечуючи при цьому, що система залишається доступною, безпечною та зручною для користувача.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальні відомості

Уніфікована мова моделювання (UML) — це стандартизований інструмент, який використовується для визначення, візуалізації, побудови та документування компонентів і поведінки програмних систем. Він служить комплексною структурою, яка допомагає розробникам програмного забезпечення, системним архітекторам і аналітикам проектувати та повідомляти структуру своїх систем. UML особливо важливий в об’єктно-орієнтованому проектуванні та функціонує як цінний проект протягом усього процесу розробки програмного забезпечення [5].

Діаграми UML широко поділяються на структурні та поведінкові типи, кожен з яких служить різним цілям у зображенні різних аспектів системи. Структурні діаграми в першу чергу стосуються статичної структури системи, представляючи, як організовані компоненти та як вони взаємодіють один з одним. Ключові діаграми цієї категорії включають діаграму класів, яка описує класи системи, атрибути та зв’язки; діаграма компонентів, яка зображує компоненти та їх залежності; і діаграму розгортання, яка описує фізичну інфраструктуру системи. Інші діаграми, такі як діаграми об’єктів і складених структур, надають уявлення про екземпляри об’єктів і внутрішні

структури відповідно. Між тим, діаграми пакетів зосереджені на організації системних компонентів у пакети, пропонуючи більш чітке уявлення про модульні зв’язки.

З іншого боку, діаграми поведінки зосереджені на динамічних взаємодіях у системі, детально описуючи, як компоненти поводяться та спілкуються протягом певного часу. Діаграма варіантів використання має важливе значення для відображення функціональних вимог системи з точки зору користувача, тоді як діаграми послідовності показують взаємодію між об’єктами в певній послідовності. Діаграми діяльності використовуються для моделювання робочих процесів, фіксуючи потік дій у системі. Діаграми станів відстежують стани об’єктів і переходи між ними, що конче важливо для систем, де поведінка об’єкта залежить від його стану. Інші діаграми поведінки, такі як діаграми зв’язку, діаграми огляду взаємодії та часові діаграми, надають додаткові рівні деталей щодо того, як об’єкти контактують, як розгортаються взаємодії та як часові обмеження впливають на поведінку системи.

Використання UML пропонує кілька переваг. Він створює універсальну мову, яка сприяє ефективній комунікації між зацікавленими сторонами, включаючи розробників, архітекторів і клієнтів. Візуальний характер діаграм UML полегшує документування системи, забезпечуючи чітке представлення архітектури та проектних рішень. Крім того, ці діаграми сприяють кращому розумінню системи, розбиваючи складні взаємодії та структури на більш засвоювані елементи. UML також цінний на етапах проектування та планування, допомагаючи командам координувати свої зусилля, завчасно виявляти потенційні проблеми та гарантувати, що система відповідає вимогам. Крім того, інструменти UML часто дозволяють здійснювати зворотне проектування, дозволяючи розробникам генерувати код із діаграм і навпаки, таким чином спрощуючи процес розробки [5].

Загалом, UML є вкрай важливим інструментом для забезпечення ясності та узгодженості проектування системи. Використовуючи широкий спектр діаграм, розробники можуть документувати, зображати та ефективно

передавати свої системи, полегшуючи керування як малими, так і великими проектами. Незалежно від того, чи йдеться про планування архітектури чи розуміння динамічної поведінки системи, UML забезпечує структурований підхід, який допомагає забезпечити успіх ініціатив з розробки програмного забезпечення.

2.2 Об’єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Діаграма варіантів використання — це візуальне представлення, яке використовується в уніфікованій мові моделювання (UML) для ілюстрації функціональних вимог системи з точки зору її користувачів, також відомих як актори. Ці діаграми особливо корисні для відображення взаємодії між користувачами (або іншими системами) і системою, що проектується. Вони допомагають забезпечити високорівневе уявлення про функціональність системи та те, як різні користувачі взаємодіють з нею, висвітлюючи, що система робитиме, без детального опису того, як це буде зроблено.

Однією з головних переваг діаграм прецедентів є їх простота. Вони забезпечують чіткий і зрозумілий спосіб візуалізації взаємодії користувачів із системою, що робить їх чудовим інструментом для спілкування між зацікавленими сторонами, розробниками та клієнтами. Вони також служать важливим інструментом для визначення ключових характеристик системи та забезпечення виконання вимог користувачів.

Підсумовуючи, діаграми варіантів використання пропонують суттєвий спосіб відобразити взаємодію між користувачами та системами, надаючи огляд функціональних вимог системи. Вони служать важливим першим кроком у проектуванні системи, допомагаючи всім залученим сторонам узгодити своє розуміння цілей і функцій системи.

Розроблена діаграма прецедентів представлена на рис. 4.

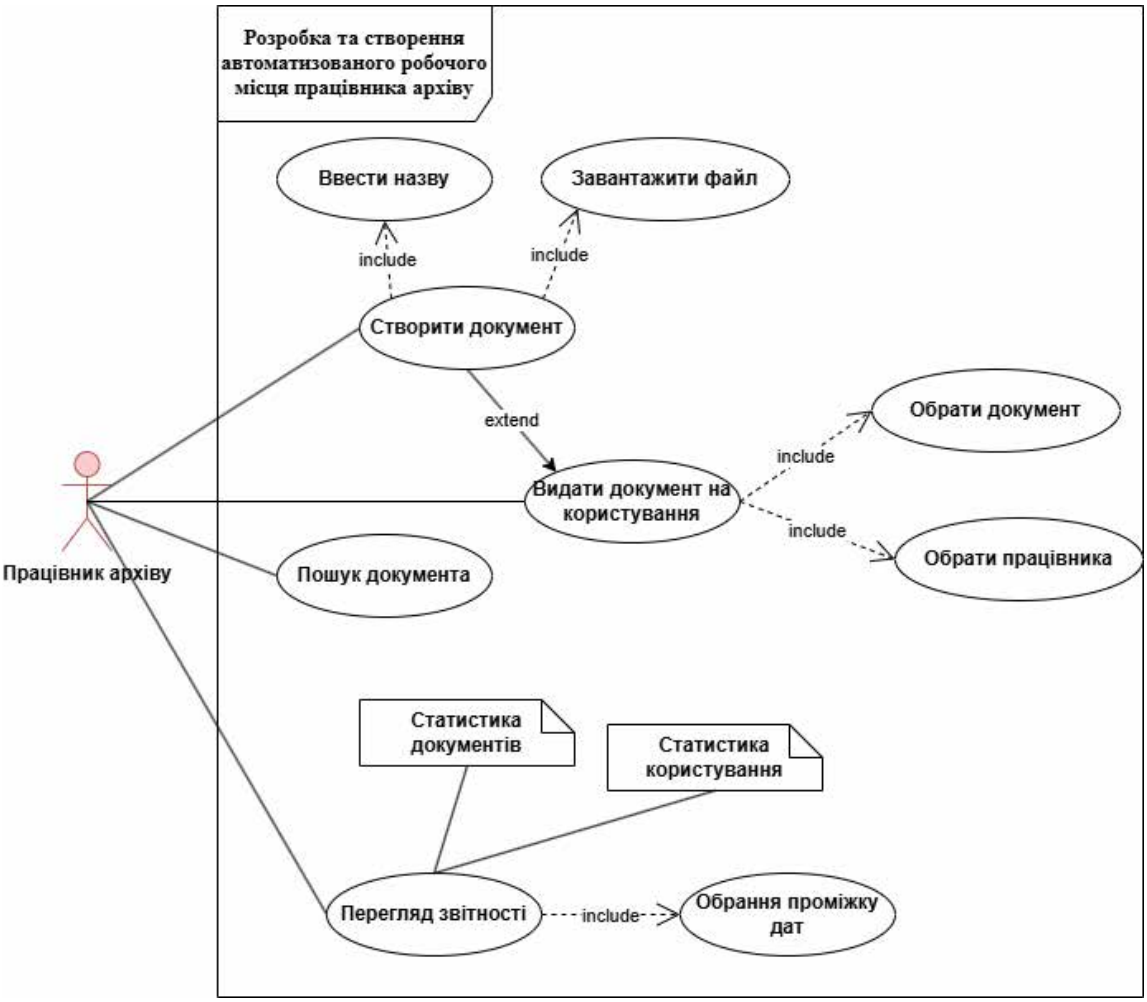

Рис. 4 Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Працівник архіву”.

Актор «Працівник архіву» включає такі прецеденти:

- створити документ;
- видати документ на користування;
- пошук документа;
- ввести назву;
- завантажити файл;
- обрати документ;
- обрати працівника;
- перегляд звітності;
- обрання проміжку дат.

Прецеденти певним чином залежать одне від одного.

Розглянемо детальніше вищеописані прецеденти.

Сценарій: створити документ

Актор: Працівник архіву

Передумови: працівник архіву увійшов у систему та має відповідні права на створення документів. Система має активне підключення до бази даних, де будуть зберігатися документи.

Основний потік:

1. працівник архіву отримує доступ до опції «Створити документ» із головного меню або інформаційної панелі системи;
2. система надає працівнику архіву форму для введення необхідних реквізитів для документа;
3. після введення всіх реквізитів працівник архіву перевіряє введену інформацію на предмет точності. Працівник може ввести будь-які необхідні зміни, перш ніж продовжити;
4. після підтвердження даних працівник архіву натискає кнопку «Зберегти» або «Створити». Система обробляє дані та створює запис для нового документа в базі даних;
5. система зберігає файл документа разом із його метаданими (наприклад, заголовком, типом, конфіденційністю тощо) у відповідному місці бази даних;
6. система виводить повідомлення про успішне створення документа. Тепер документ доступний для пошуку, пошуку або подальших дій (таких як видача чи оновлення).

Альтернативні потоки:

1. якщо працівник архіву вибирає недійсний тип документа або залишає обов’язкове поле порожнім, система запропонує користувачеві виправити помилку, перш ніж продовжити;
2. якщо завантажити файл не вдається (наприклад, через непідтримуваний формат або розмір файлу), система сповіщає працівника архіву та дозволяє йому повторно завантажити файл;

3. якщо працівник архіву вирішить скасувати процес створення документа, він може вийти з форми без збереження, і документ не буде створено.

Цей сценарій надає детальний погляд на процес створення документа, гарантуючи, що працівник архіву може ефективно створювати, класифікувати та зберігати документи в системі, покращуючи загальний робочий процес у середовищі архіву.

Розглянемо ще один сценарій використання.

Сценарій: видати документ для використання

Актор: Працівник архіву

Передумови: Співробітник архіву авторизується в системі з відповідними правами доступу для видачі документів. Документ уже зберігся в системі (тобто існує в архіві). Система має актуальний облік наявності та використання документів.

Основний потік:

1. співробітник архіву обирає опцію «Видати документ» в основному інтерфейсі системи або на панелі керування документами;
2. система запропонує працівнику архіву пошук документа для видачі. Пошук можна здійснювати за різними критеріями;
3. система пропонує працівнику архіву ввести або вибрати одержувача документа;
4. співробітник архіву перевіряє введену інформацію для підтвердження реквізитів запиту на видачу. Система показує наявність документа, інформацію про одержувача та термін видачі;
5. коли працівник архіву підтвердить дані, він натискає кнопку «Видати» або «Підтвердити». Потім система записує видачу в історію транзакцій документа, оновлюючи статус документа на «Видано» або «Видано»;

6. система формує квитанцію або повідомлення як для працівника архіву, так і для одержувача, що підтверджує видачу документа;
7. працівник архіву надає одержувачу фізичний або цифровий документ залежно від формату документа та можливостей системи. Якщо це фізичний документ, його можна передати вручну; якщо він цифровий, система може надіслати посилання або надати доступ через інтерфейс користувача;
8. на цьому процес видачі завершено, і документ вважається «виписаним» вказаним користувачем. Система відстежує використання документа, включаючи статус повернення та будь-які затримки.

Альтернативні потоки:

1. якщо документ уже виданий іншому користувачеві або позначений як недоступний, система повідомляє про це працівника архіву. Працівнику може бути надано можливість затримати документ для майбутньої видачі або вибрати альтернативний документ;
2. якщо дані одержувача неправильні або неповні (наприклад, не знайдено жодного працівника з наданим іменем), система запропонує працівнику архіву перевірити або повторно ввести інформацію про одержувача;
3. якщо працівник архіву вирішить не продовжувати видачу документа, він може скасувати дію. Потім система повертається до інтерфейсу пошуку документів, і запис про видачу не створюється.

Цей сценарій гарантує, що працівник архіву може ефективно видавати документи, зберігаючи чіткий запис про те, хто мав доступ до документа, з якою метою та протягом якого часу. Система допомагає оптимізувати процес видачі документів, зменшує ймовірність помилок і забезпечує підзвітність в документообігу.

2.2.2 Діаграма послідовності. Діаграма послідовності — це тип діаграми взаємодії в уніфікованій мові моделювання (UML), яка демонструє, як об’єкти в системі взаємодіють протягом певного часу. Ця діаграма спеціально розроблена, щоб показати порядок повідомлень, якими обмінюються різні об’єкти або компоненти в системі, як правило, для виконання певного завдання або функції [6].

На діаграмі послідовності актори представляють користувачів або системи, які взаємодіють із системою, що моделюється. Це можуть бути такі ролі, як «Працівник архіву» або «Користувач системи», залежно від контексту. Об’єкти на діаграмі є екземплярами системних компонентів або класів, які беруть участь у взаємодії, наприклад «Система керування документами», «База даних» або «Документ». Кожен із цих об’єктів представлений лінією життя, вертикальною пунктирною лінією, яка вказує на існування цього об’єкта протягом усієї взаємодії.

Повідомлення зображені у вигляді стрілок між лініями життя, що показує, як об’єкти спілкуються один з одним. Ці стрілки позначені діями чи операціями, які виконуються, наприклад створенням документа чи отриманням інформації. Коли об’єкт активно бере участь у виконанні завдання, використовується панель активації, яка є прямокутним блоком, що з’являється на лінії життя, щоб позначити період, протягом якого об’єкт задіяний в обробці запиту.

Після надсилання повідомлення зворотне повідомлення зазвичай відображається у вигляді пунктирної стрілки, яка вказує на вихідний об’єкт, що вказує на результат операції. Наприклад, після надсилання повідомлення про створення документа система може повернути повідомлення про підтвердження, яке вказує на успіх або невдачу операції.

Діаграма послідовності починається з того, що актор або система ініціює запит. Потім він ілюструє ряд взаємодій між об’єктами, показуючи потік повідомлень, які виникають під час обробки запиту системою.

Діаграма завершується кінцевими повідомленнями або операціями, що сигналізують про успішне виконання завдання [6].

Діаграма послідовності, зображена на рис. 5, включає наступні об'єкти:

- “Працівник архіву”;
- “Документи”;
- “Видача документів”;
- “Звітність”.

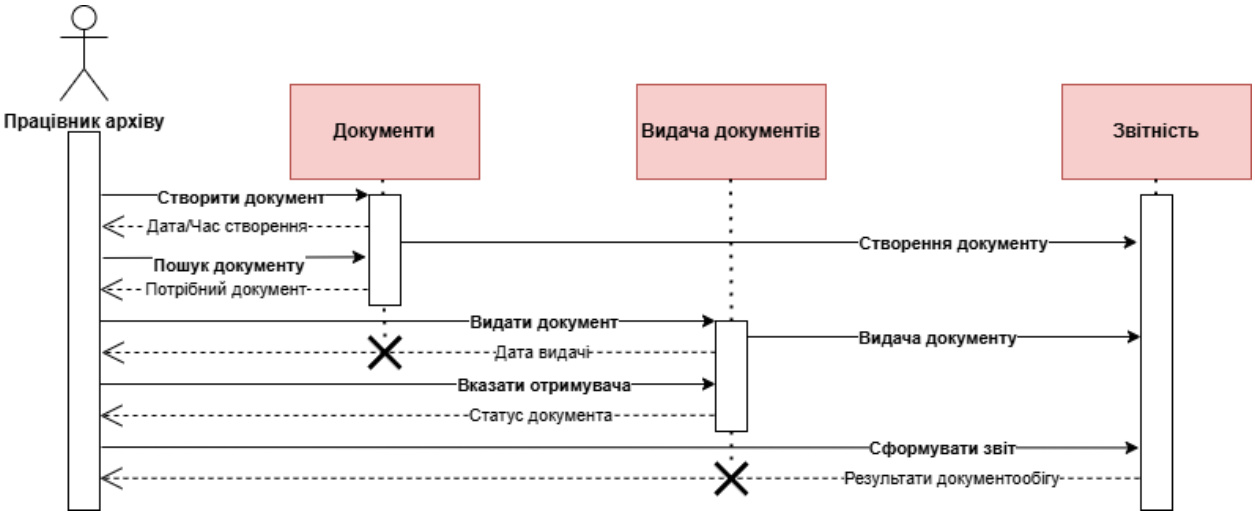

Рис. 5 Діаграма послідовності

Працівник архіву управляє документообігом, ініціюючи створення нових документів, які отримують відповідний часовий запис. Коли необхідний документ вже існує, він надсилає запит на його пошук. Після знаходження потрібної інформації працівник звертається до системи видачі документів, вказуючи дату отримання. Видача документів, своєю чергою, реєструє отримувача та передає відповідні дані.

Для контролю руху документів архіваріус перевіряє статус, уточнюючи їхнє поточне місце знаходження та використання. Завершальним етапом є формування звітності, де система аналізує отримані результати документообігу та узагальнює інформацію про всі проведені операції. Такий підхід забезпечує злагоджений і зрозумілий процес управління документацією.

2.2.3 Діаграма активності. Діаграма діяльності — це тип діаграми в уніфікованій мові моделювання (UML), яка візуалізує потік керування та послідовність дій у системі чи процесі. Це особливо корисно для моделювання динамічної поведінки системи, показуючи виконувані завдання чи дії та те, як вони взаємопов’язані в рамках конкретного робочого процесу. Зосереджуючись на етапах завершення процесу, діаграми діяльності забезпечують чітке уявлення про те, як різні дії, рішення та умови обробляються в системі [7].

Діаграма зазвичай починається з початкового вузла, який позначає початок робочого процесу. Звідти серія вузлів діяльності представляє окремі завдання або дії, які відбуваються під час процесу. Ці вузли з’єднані потоками керування, які є стрілками, які вказують напрямок виконання від однієї дії до іншої. У деяких випадках можуть виникати точки прийняття рішень, представлені вузлами прийняття рішень, які є ромбами. Ці вузли вводять розгалуження в потік на основі умов, дозволяючи процесу слідувати різними шляхами залежно від результату рішення.

Окрім вузлів прийняття рішень, вузли злиття можна використовувати для об’єднання кількох потоків назад у єдиний шлях після прийняття рішення. Це гарантує, що робочий процес продовжується впорядкованим чином. Коли процеси включають паралельні завдання, вузли розгалуження використовуються для поділу потоку на кілька шляхів, які можуть виконуватися одночасно, тоді як вузли об’єднання використовуються для синхронізації цих паралельних шляхів назад у єдиний потік, коли завдання завершені.

Наприкінці процесу останній вузол вказує на завершення робочого процесу, сигналізуючи про те, що подальших дій виконувати не потрібно.

На всій діаграмі потоки об’єктів представляють рух даних або об’єктів між діями, ілюструючи, як інформація передається під час виконання завдань.

Розроблена діаграма активності представлена на рис. 6

Рис. 6 Діаграма активності

Ця діаграма активності ілюструє процес роботи працівника архіву, охоплюючи всі ключові етапи взаємодії з документами. Все починається зі створення документа, де користувач вводить його назву та завантажує відповідний файл. Далі, якщо необхідний документ вже існує, система підтримує можливість його пошуку. Якщо пошук успішний, користувач може переглянути файл, ввести зміни або, якщо необхідно, видалити документ.

У разі, якщо потрібний файл не був знайдений, передбачено процес видачі документа на користування. Працівник архіву визначає отримувача, обирає необхідний документ, після чого має можливість переглянути звітність, що фіксує всі операції. Для контролю ефективності документообігу система також генерує статистику документів і видач, дозволяючи аналізувати загальну активність архіву. Завершення роботи позначає фінальний етап, після якого можна підбити підсумки всіх виконаних дій.

Ця діаграма допомагає структурувати робочий процес, роблячи управління документами чітким і логічним.

2.3 Абстракції предметної області

Абстракції предметної області стосуються процесу спрощення та узагальнення складних систем реального світу в керовані моделі або концепції, які фіксують основні характеристики та взаємодії, не загрузаючи в непотрібних деталях. У контексті розробки програмного забезпечення абстракції мають вирішальне значення, оскільки вони дозволяють чіткіше зрозуміти систему, її компоненти та те, як вони взаємодіють. Вони зосереджені на основних аспектах системи, приховуючи складність певних процесів і дозволяючи розробникам, користувачам і зацікавленим сторонам працювати з більш зрозумілими представленнями [8].

При розробці автоматизованого робочого місця для архівного працівника абстракції допомагають моделювати ключові дії, пов’язані з керуванням документами, такі як створення, зберігання, пошук і видача

документів. Ці абстракції дозволяють визначати такі важливі сутності, як документи, працівники та дії, які вони виконують, і зосереджуються на тому, що потрібно зробити, а не на технічних тонкощах того, як це робиться.

Однією з основних абстракцій у цьому випадку є документ, який представляє центральний об’єкт, яким керують в архіві. Замість того, щоб мати справу зі складністю файлових систем, структур метаданих і окремих форматів, абстракція документа може спростити процес до основних операцій, таких як створення, зберігання, отримання та редагування.

Ще однією важливою абстракцією є архіватор або користувач, який взаємодіє з системою. Ця абстракція зосереджена на ролях і дозволах користувача, його здатності керувати документами та взаємодії, яку вони можуть мати з системою. Замість того, щоб турбуватися про конкретні елементи інтерфейсу чи технічні деталі, абстракція працівника допомагає зосередитися на завданнях, які користувач має виконати, як-от видача документів, пошук записів або створення звітів.

Крім того, процеси робочого процесу є абстрагованими, щоб проілюструвати послідовність завдань і рішень, пов’язаних з керуванням документами. Розбиваючи такі процеси, як затвердження, видача та зберігання документів, на спрощені кроки, систему можна спроектувати з чіткими обов’язками та шляхами потоку, що полегшує розробку та оптимізацію взаємодії з користувачем [8].

Підсумовуючи, абстракції є фундаментальними для спрощення складності системи, полегшуючи її проєктування, реалізацію та розуміння. У контексті автоматизованого робочого місця архівного працівника абстракції зосереджуються на ключових компонентах і діях, таких як документи, користувачі та робочі процеси, одночасно усуваючи непотрібну технічну складність, що допомагає створити ефективнішу та зручнішу систему.

Рис. 7 Абстракції предметної області

Абстракція "Документ" включає базові характеристики, такі як дата та час створення, назва, файл, опис і статус. Її функціональні можливості дозволяють створювати, редагувати, шукати та видаляти документи, забезпечуючи керування інформацією та її доступність.

Абстракція "Видача документа" описує процес передачі документа. Серед її властивостей—дата видачі, працівник архіву, отримувач, сам документ і статус операції. Основні функції включають видачу документа, його повернення до архіву та можливе знищення.

Ці абстракції формують чітку модель документообігу, дозволяючи ефективно керувати інформацією, відстежувати її рух і забезпечувати контроль над кожним етапом.

2.4 Діаграма класів

Діаграма класів є фундаментальним елементом уніфікованої мови моделювання (UML), який використовується для ілюстрації статичної структури програмної системи. Він забезпечує високорівневе уявлення про архітектуру системи шляхом моделювання ключових класів, їхніх атрибутів, методів і зв’язків, які існують між ними. Цей тип діаграми важливий на ранніх

етапах розробки програмного забезпечення, оскільки допомагає визначити план системи до того, як почнеться фактичний процес кодування.

Кожен клас на діаграмі представлено у вигляді прямокутного блоку, розділеного на три секції: у верхній частині відображається ім’я класу, у середній міститься його атрибути, а в нижній — перелік його операцій або функцій. Цей формат чітко передає структуру та обов’язки кожного класу в системі. Організовуючи інформацію таким чином, діаграми класів спрощують розуміння об’єктно-орієнтованих зв’язків і сприяють кращому спілкуванню між розробниками.

Відносини відіграють важливу роль у діаграмах класів, показуючи, як різні класи взаємодіють один з одним. Наприклад, асоціації представляють загальні зв’язки між класами, такими як користувач, який керує набором документів. Успадкування вказує на те, що один клас походить від іншого, що дозволяє спільну поведінку та ієрархічну організацію. Композиція та агрегація відображають зв’язки між цілими частинами, коли один клас складається з інших класів або містить їх. Залежності припускають, що зміни в одному класі можуть впливати на поведінку іншого [9].

При розробці автоматизованого робочого місця для співробітника архіву діаграма класів служить для моделювання ключових сутностей, таких як документи, користувачі, працівники архіву, звіти та системні модулі, відповідальні за обробку взаємодії та збереження даних. Наприклад, клас Document може містити такі поля, як заголовок, ім’я файлу, дата створення та статус доступу, а також операції для редагування, зберігання або отримання документа. Клас ArchiveWorker керує автентифікацією користувачів і надає доступ до певних функцій документа, тоді як клас Report може відповідати за збір даних про використання документів або журнали активності.

Надаючи структуроване візуальне представлення, діаграми класів допомагають прояснити організацію системи та оптимізувати розробку. Вони також спрощують виявлення потенційних проблем, таких як дубльована логіка або незрозумілі залежності, на етапі проєктування. Загалом цей

інструмент моделювання підтримує більш ефективний процес розробки, забезпечуючи послідовність, заохочуючи модульність і сприяючи глибшому розумінню того, як компоненти системи взаємопов’язані та функціонують.

Для цього проекту було створено спеціальну діаграму класів з використанням об’єктно-орієнтованого підходу (рис. 8).

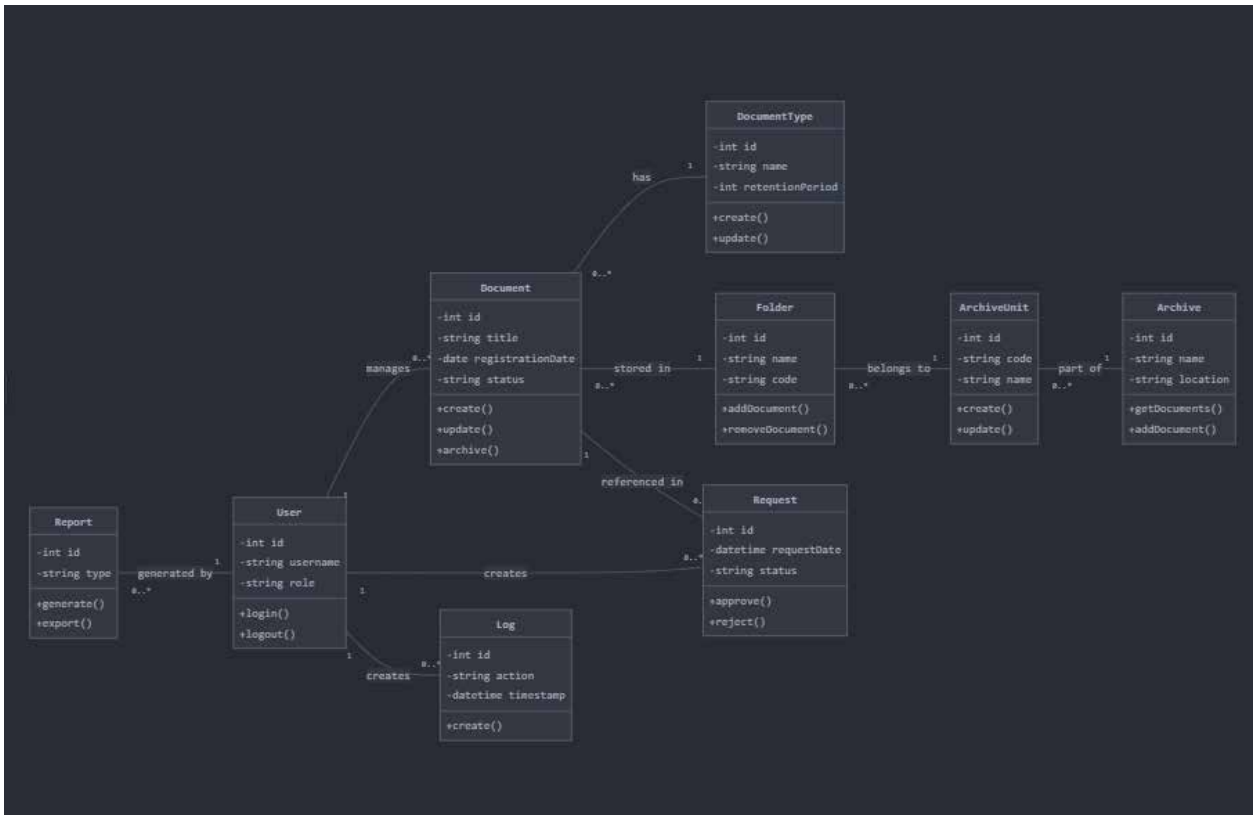

Рис. 8 Діаграма класів

У центрі системи знаходиться клас User, що моделює працівників архіву з різними рівнями доступу. Користувачі взаємодіють з документами через авторизацію (методи login/logout), створюючи та опрацьовуючи архівні записи.

Клас Document слугує основною інформаційною одиницею системи і містить важливі метадані – заголовок, дату реєстрації та статус. Документи категоризуються за типами (DocumentType), які визначають властивості зберігання, зокрема термін утримання в архіві.

Організаційна структура архіву представлена кількома взаємопов'язаними класами. Folder виступає контейнером для групування

документів, ArchiveUnit об'єднує папки в логічні блоки (фонди), а Archive представляє загальне сховище документації.

Операційна складова системи забезпечується класами Request, Log та Report. Запити дозволяють оформлювати доступ до архівних матеріалів з можливістю схвалення чи відхилення. Журнал автоматично фіксує всі дії в системі, створюючи аудиторський слід. Клас звітів забезпечує аналітичні можливості, дозволяючи генерувати та експортувати статистичні дані.

Взаємозв'язки сутностей

Архітектура системи побудована на логічних зв'язках між сутностями, що відображають бізнес-процеси архівної справи. Користувачі керують документами, створюють запити та генерують записи в журналі активності. Документи належать до визначених типів та зберігаються в ієрархічній структурі: папка → архівна одиниця → архів.

Система запитів забезпечує контрольований доступ до документів, при цьому кожен запит пов'язаний як з користувачем-ініціатором, так і з конкретним документом. Формування звітів також прив'язане до користувачів, які їх генерують.

Така структура класів забезпечує гнучку основу для розробки функціонального веб-додатку, що покриває всі основні потреби працівників архіву: документообіг, організацію зберігання, контроль доступу та аналітичну звітність. Система спроєктована з урахуванням специфіки архівної роботи та забезпечує належне управління документацією впродовж усього життєвого циклу.

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Логічна модель даних

Логічна модель даних представляє абстрактну структуру даних в інформаційній системі, зосереджуючись на організації інформації та зв’язках між об’єктами даних без урахування деталей фізичної реалізації. Він слугує сполучною ланкою між бізнес-вимогами та технічним дизайном, переводячи процеси реального світу в структурований і зрозумілий формат для розробників, аналітиків і зацікавлених сторін [10].

На відміну від фізичних моделей, які стосуються того, як дані зберігаються в базах даних, логічна модель підкреслює, які дані зберігаються, як вони пов’язані з іншими даними та правилами, які регулюють ці відносини. Він включає сутності (наприклад, «Документ», «Користувач», «Архів»), їхні атрибути (як-от назва документа, роль користувача або дата створення) і зв’язки між ними (наприклад, з’єднання «один до багатьох» або «багато до багатьох»). Ці зв’язки зазвичай візуалізуються за допомогою діаграми сутності та зв’язку (ERD), яка допомагає прояснити залежності та потоки даних.

Однією з основних цілей логічної моделі даних є забезпечення узгодженості та повноти структур даних перед переходом до впровадження. Він забезпечує системно-агностичний погляд, який дозволяє розробникам зосередитися на тому, як дані повинні логічно взаємодіяти в контексті бізнес-операцій. Наприклад, в автоматизованому робочому місці для працівника архіву модель визначала б, як документи пов’язуються з користувачами, як керується доступом, як генеруються звіти та як архівуються та витягуються історичні записи.

Логічна модель також визначає обмеження цілісності даних, такі як первинні ключі (унікальні ідентифікатори), зовнішні ключі (посилання на інші сутності) і правила перевірки (наприклад, документ повинен мати статус і дату

створення). Ці обмеження допомагають підтримувати надійність і точність даних у всій системі.

Створивши логічну модель даних на початку циклу розробки, команди можуть запобігти проблемам, пов’язаним з даними, зменшити надмірність і покращити зв’язок між технічними та нетехнічними учасниками. Він відіграє важливу роль у узгодженні структури даних програмного забезпечення з реальними бізнес-потребами та закладає основу як для фізичного дизайну бази даних, так і для майбутньої масштабованості системи.

Логічна модель системи представлена на рисунку 9.

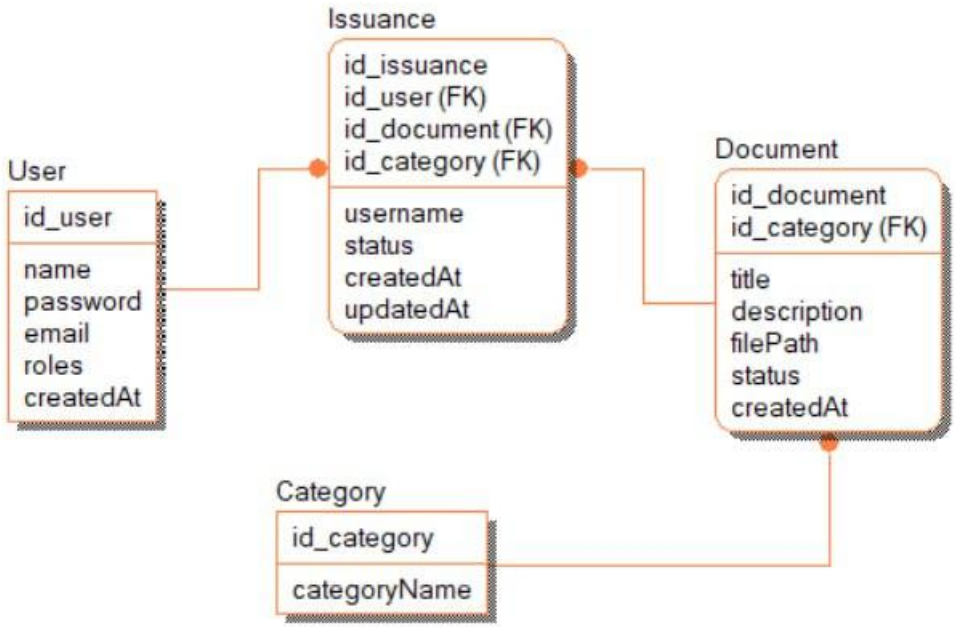

Рис. 9 ER-діаграма

Ця ER-діаграма відображає структуру бази даних, де основними сутностями є User, Issuance, Document і Category. Вона демонструє зв'язки між таблицями через зовнішні ключі, що дозволяє ефективно управляти документами та їхнім обігом.

- user містить інформацію про користувача, включаючи його ім'я, електронну пошту, пароль та роль у системі;

- document зберігає характеристики документів, такі як назва, опис, шлях до файлу та статус;
- issuance фіксує факт видачі документа, пов'язуючи його з користувачем та категорією, а також включає статус і часові мітки;
- category класифікує документи за певними типами, що допомагає у швидкому пошуку та систематизації.

Діаграма ілюструє взаємодію між цими сутностями, дозволяючи зрозуміти, як документи створюються, зберігаються та передаються між користувачами.

3.2 Вибір системи управління базою даних та її реалізація

У процесі розробки автоматизованого робочого місця для архівного працівника вибір відповідної системи керування базами даних (СУБД) відіграє вирішальну роль у забезпеченні надійності даних, масштабованості та ефективного доступу до збереженої інформації. Оскільки система призначена для обробки різних типів документів, ролей користувачів, журналів використання та даних звітів, вибрана СУБД повинна пропонувати надійну функціональність для зберігання структурованої інформації, забезпечення цілісності даних і підтримки одночасного доступу.

Після оцінки кількох варіантів СУБД для цього проєкту було обрано MySQL. MySQL — це широко використовувана система реляційних баз даних з відкритим вихідним кодом, відома своєю стабільністю, продуктивністю та сумісністю з сучасними веб-додатками. Він бездоганно інтегрується з фреймворком Symfony на основі PHP, який використовується для створення серверної частини програми. MySQL пропонує повну підтримку мови структурованих запитів (SQL), що робить її ідеальною для визначення та керування реляційними структурами даних, необхідних у системі архівування.

Однією з ключових переваг MySQL є його підтримка складних зв’язків між таблицями, що важливо для моделювання реальних об’єктів, таких як користувачі, документи, відділи та журнали доступу. Він також забезпечує узгодженість даних завдяки використанню первинних і зовнішніх ключів, контролю транзакцій і обмежень перевірки даних. Це гарантує, що всі взаємодії з базою даних зберігають цілісність і логіку системи.

Етап впровадження передбачав створення реляційної схеми на основі логічної моделі даних. Таблиці були визначені для основних сутностей, таких як документи, користувачі, видані_документи, журнали та звіти. Кожна таблиця містить ретельно відібрані поля з відповідними типами даних і обмеженнями. Індексування було застосовано до ключових стовпців, щоб прискорити запити, зокрема ті, що стосуються пошуку та фільтрації документів.

Вбудовані інструменти фреймворку Symfony, такі як Doctrine ORM (Object-Relational Mapping), були використані для спрощення взаємодії з базою даних і забезпечення чіткого поділу між рівнями бізнес-логіки та доступу до даних. Це не тільки прискорює розробку, але й полегшує підтримку та масштабування програми в майбутньому.

Рішення вибрати MySQL як систему управління базами даних (СУБД) для розробки автоматизованого робочого місця співробітника архіву було обумовлено кількома ключовими факторами, які відповідають як технічним, так і функціональним вимогам проекту.

По-перше, MySQL є широко визнаною та зрілою системою реляційної бази даних, відомою своєю надійністю, продуктивністю та масштабованістю. Враховуючи природу архівної системи, яка включає в себе керування структурованими даними, такими як документи, користувачі, журнали доступу та звіти, реляційна модель була важливою для підтримки цілісності даних і забезпечення того, щоб усі зв’язки між об’єктами (наприклад, документами та користувачами) були чітко визначені та послідовно дотримані. Потужна підтримка MySQL SQL, який є галузевим стандартом для

реляційних баз даних, зробила його ідеальним вибором для ефективного керування такими структурованими даними [11].

Іншою важливою причиною вибору MySQL була легкість інтеграції з обраною структурою веб-розробки Symfony. Symfony побудовано на PHP, а MySQL забезпечує відмінну підтримку програм на основі PHP. Крім того, сумісність Symfony з Doctrine ORM (Object-Relational Mapping) забезпечує безперебійну взаємодію між бізнес-логікою програми та базою даних, дозволяючи розробникам зосередитися на основній функціональності програми, а не мати справу з операціями з базою даних низького рівня. Це зробило MySQL природною придатністю для проекту, спрощуючи розробку та зменшуючи потенційну складність.

MySQL також пропонує надійні функції цілісності даних, такі як підтримка первинних і зовнішніх ключів, які є вирішальними для встановлення зв’язків між різними таблицями, наприклад зв’язування документів із користувачами та відстеження використання документів. Використання цих ключів гарантує, що всі дані в системі залишаються точними та узгодженими, запобігаючи таким проблемам, як втрати записів або неузгоджені зв’язки між об’єктами.

Крім того, важливою мірою була масштабованість. У міру зростання системи та збільшення обсягу архівованих документів MySQL надає ефективні механізми для індексування даних та оптимізації запитів, гарантуючи, що операції пошуку та отримання даних залишаються швидкими та оперативними навіть із великим набором даних. Можливість горизонтального та вертикального масштабування разом із розширеними функціями, такими як кластеризація та реплікація, дає системі гнучкість для майбутнього зростання без шкоди для продуктивності.

Іншою практичною причиною вибору MySQL є його природа з відкритим кодом, що знижує загальні витрати на розробку та забезпечує широку спільноту розробників і ресурсів. Будучи відкритим кодом, MySQL також уникає ліцензійних зборів, що робить його економічно вигідним

варіантом для проєктів будь-якого розміру. Це відповідає бюджетним обмеженням проєкту, одночасно забезпечуючи доступ до потужної та повністю підтримуваної системи баз даних.

Крім того, безпека була ключовим фактором у процесі прийняття рішень. MySQL пропонує широкий спектр функцій безпеки, включаючи надійну автентифікацію користувача, шифрування та контроль доступу, які необхідні для керування конфіденційними документами в архівній системі. Ці функції допомагають захистити систему від несанкціонованого доступу та гарантують, що лише авторизовані користувачі можуть виконувати певні дії, наприклад створювати, редагувати або видавати документи.

MySQL відомий своїм широким впровадженням і підтримкою в спільноті розробників. Завдяки обширній документації, онлайн-ресурсам і потужній базі розробників було легко знайти рішення потенційних технічних проблем у процесі розробки. Крім того, популярність MySQL гарантує, що він підтримуватиметься і в майбутньому, що робить його стабільним вибором для довгострокових проєктів [11].

Підсумовуючи, вибір MySQL як СУБД пропонує надійну основу для рівня даних системи. Його інтеграція з інфраструктурою Symfony, підтримка реляційного моделювання та перевірений досвід розробки веб-додатків роблять його придатним і ефективним вибором для автоматизованого робочого місця співробітника архіву.

Таким чином було створено базу даних в середовищі MySQL Workbench (Рис. 10).

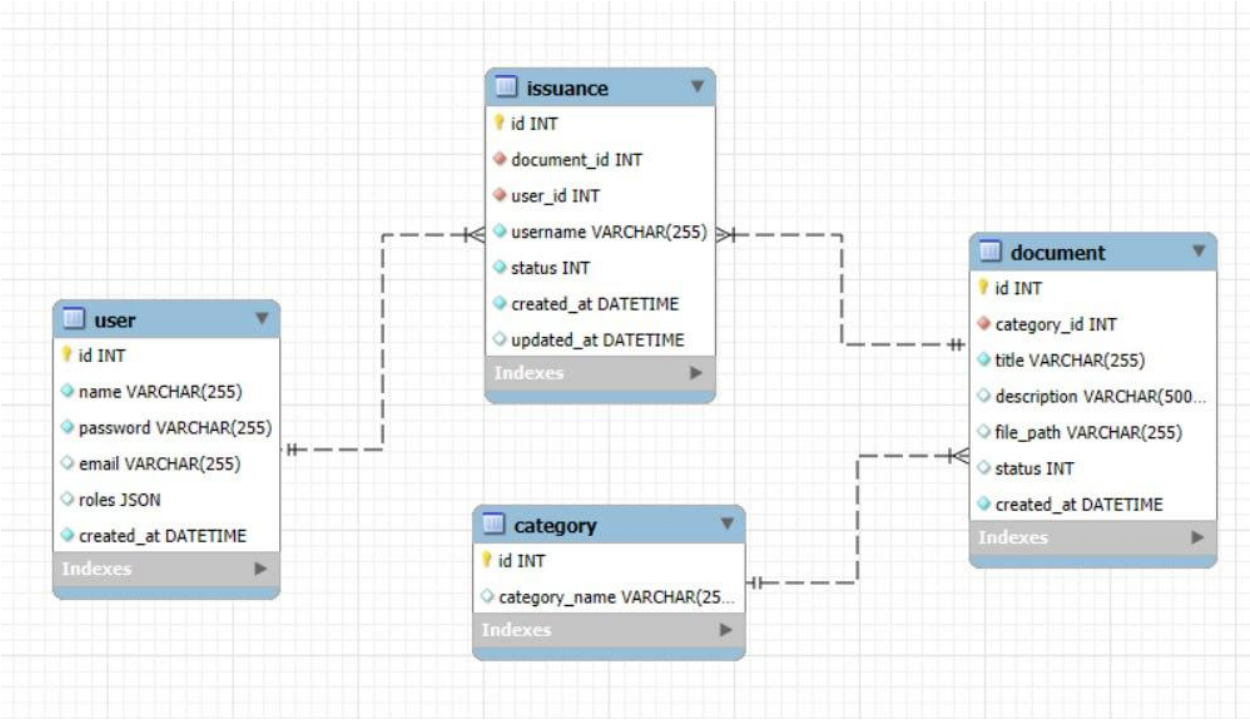

Рис. 10 База даних

Ця база даних складається з чотирьох таблиць: user, issuance, document та category, кожна з яких має власні атрибути та зв'язки.

Таблиця user містить інформацію про користувача, включаючи id, name, password, email, roles (збережені у форматі JSON) та часову мітку створення created_at.

Таблиця issuance зберігає дані про видачу документів, містячи id, document_id (зв’язок із таблицею document), user_id (зв’язок із таблицею user), username, status, а також часові мітки created_at та updated_at.

Таблиця document описує документи, що зберігаються в системі. Вона включає id, category_id (зв’язок із таблицею category), title, description, file_path, status та created_at.

Таблиця category класифікує документи за категоріями і містить id та category_name, що допомагає в упорядкуванні документів.

Зв'язки між таблицями:

- issuance.user_id пов’язаний із user.id, що дозволяє відстежувати, хто отримав документ;

- issuance.document_id пов’язаний із document.id, що забезпечує зв’язок між операцією видачі та самим документом;
- document.category_id пов’язаний із category.id, що допомагає категоризувати документи.

Така структура забезпечує чітке та ефективне управління документами в системі.

3.3 Архітектура програмного забезпечення

Програмна архітектура автоматизованого робочого місця працівника архіву розроблена таким чином, щоб забезпечити ефективність, ремонтопридатність і масштабованість. Завдяки багаторівневому підходу система розділяє різні проблеми, що полегшує оновлення, зміни та доповнення, не впливаючи на загальну структуру. Ця модульна конструкція забезпечує плавну інтеграцію нових функцій з часом, адаптацію до мінливих потреб і покращення загальної продуктивності системи.

На передньому плані презентаційний рівень відповідає за інтерфейс користувача, забезпечуючи інтуїтивно зрозумілий і зручний досвід для працівників архіву. Створений за допомогою таких технологій, як HTML, CSS, JavaScript і механізм шаблонів Symfony Twig, інтерфейс забезпечує ефективну взаємодію з системою. Співробітники можуть створювати, шукати, випускати документи та звітувати про них за допомогою зручних для навігації форм і фільтрів. Інтерфейс динамічно спілкується з сервером через RESTful API або виклики AJAX, забезпечуючи плавний обмін даними та оновлення в реальному часі без необхідності перезавантажувати сторінку.

Прикладний рівень служить ядром системи, обробляючи бізнес-логіку. Написаний на PHP з використанням фреймворку Symfony, цей рівень керує запитами з інтерфейсу, обробляє дані та взаємодіє з базою даних для виконання необхідних завдань. Контролери відповідають за маршрутизацію запитів користувачів, а служби інкапсулюють основні функції системи,

включаючи керування документами, адміністрування користувачів і створення звітів. Безпека також є невід'ємною частиною цього рівня, гарантуючи, що лише авторизовані користувачі можуть отримати доступ до певних документів і виконувати певні дії, що забезпечується механізмами автентифікації та авторизації. Крім того, перевірка даних гарантує, що вся інформація, введена в систему, є точною та відповідає необхідним критеріям перед обробкою.

Рівень даних, створений за допомогою MySQL, служить основою для зберігання та отримання даних. Структуровані дані, такі як інформація про користувача, записи документів і журнали транзакцій, обробляються за допомогою системи керування реляційною базою даних. Використання бібліотеки Doctrine ORM спрощує взаємодію з базою даних, дозволяючи об'єктам PHP безпосередньо відображатися в таблицях бази даних. Цей рівень абстракції спрощує операції з базою даних і робить запити, оновлення та керування записами більш ефективними. Ключові сутності в базі даних включають таблиці для документів, користувачів, випусків і журналів, гарантуючи, що кожна частина даних надійно зберігається та може бути швидко відновлена за потреби [12].

Крім того, рівень інтеграції забезпечує безперебійну взаємодію із зовнішніми системами або сторонніми службами, дозволяючи системі розширювати свої можливості. Незалежно від того, чи йдеться про інтеграцію з інструментами сканування документів, зовнішніми системами керування чи створення звітів у різних форматах, як-от PDF чи Excel, рівень інтеграції забезпечує гнучкість і майбутнє розширення. Будучи модульним, цей рівень може легко включати нові інструменти або системи за потреби.

Безпека є важливою складовою архітектури. Він захищає цілісність системи та забезпечує конфіденційність конфіденційних даних. Функції безпеки вбудовані в прикладний рівень і рівень бази даних, захищаючи від несанкціонованого доступу та гарантуючи, що дані шифруються як під час передачі, так і в стані спокою. Автентифікація та авторизація контролюють

доступ до системи, а шифрування забезпечує безпеку конфіденційної інформації.

Продуктивність і масштабованість також є ключовими факторами в цій архітектурі. Система оптимізована для обробки потенційного зростання та збільшення навантаження даних за допомогою ефективних запитів до бази даних і стратегій кешування. Це гарантує, що система може масштабуватися в міру додавання нових документів або зростання бази користувачів. Модульна конструкція системи гарантує впровадження нових функцій з мінімальним впливом на існуючу структуру, зберігаючи систему адаптивною та готовою до майбутніх вимог.

Для підтримки поточного обслуговування та забезпечення безперебійної роботи система включає компоненти журналювання та моніторингу. Журнали відстежують дії користувачів, продуктивність системи та помилки, надаючи уявлення про стан і безпеку системи. Адміністратори можуть переглядати ці журнали, щоб визначити проблеми або області, які потрібно покращити. Інструменти моніторингу постійно оцінюють продуктивність системи, надаючи сповіщення в реальному часі про такі проблеми, як повільні запити до бази даних або спроби неавторизованого доступу.

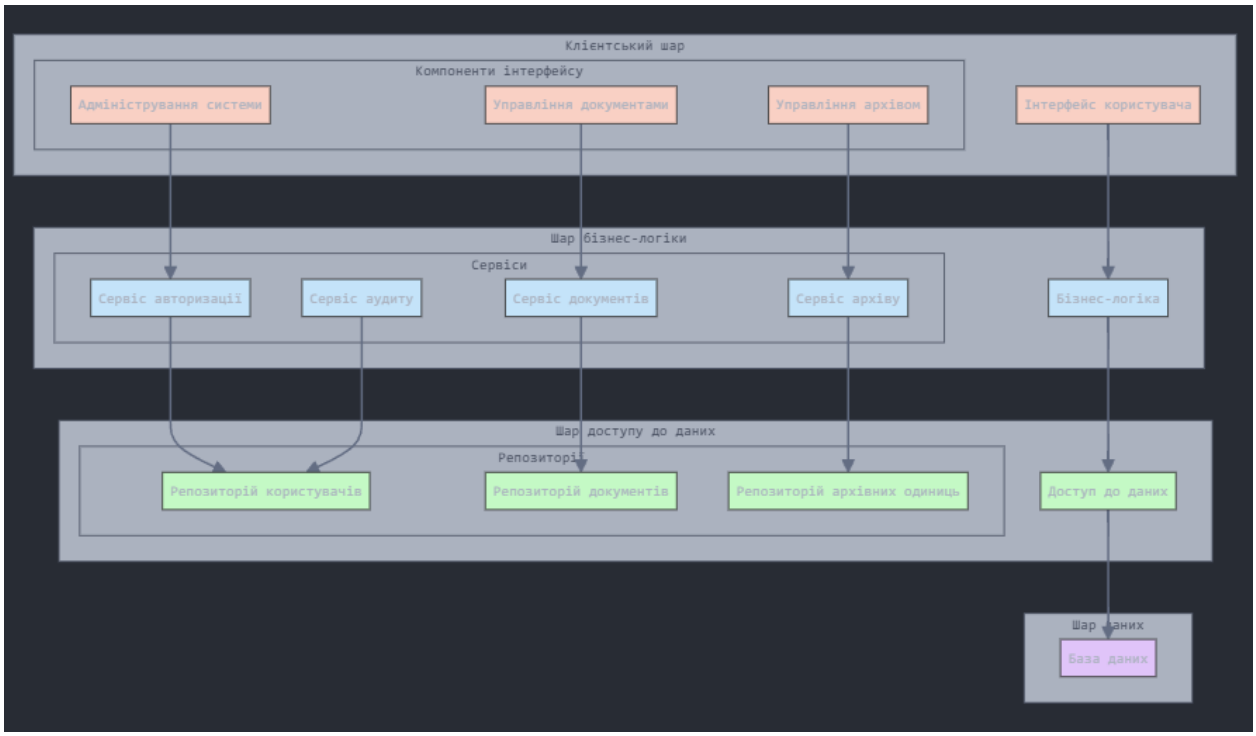

Рис. 11 Багатошарова архітектура

На вершині архітектурної піраміди розташовується клієнтський шар з інтерфейсом користувача. Він включає панелі для роботи з документами, структурними елементами архіву та адміністративними функціями. Цей рівень забезпечує інтуїтивно зрозумілу взаємодію працівників архіву з системою через графічні компоненти.

Під клієнтським шаром функціонує шар бізнес-логіки, який втілює основні функціональні можливості системи. Тут розміщені сервіси документообігу, керування архівною структурою, автентифікації користувачів та аудиту дій. Саме в цьому шарі реалізуються бізнес-правила та процеси, специфічні для архівної діяльності.

Далі йде шар доступу до даних, що слугує посередником між бізнес-логікою та сховищем інформації. Репозиторії цього рівня абстрагують механізми взаємодії з базою даних для різних типів сутностей: користувачів системи, документів та структурних одиниць архіву. Цей шар забезпечує уніфікований інтерфейс для операцій зберігання та отримання даних.

В основі архітектури знаходиться шар даних, представлений базою даних. Вона виступає централізованим сховищем усієї інформації, необхідної для функціонування системи.

3.4 Організаційна структура програмного забезпечення

3.4.1 Діаграма пакетів. Пакетна діаграма — це структурна модель на UML (Unified Modeling Language), яка використовується для організації компонентів системи в логічні групи, відомі як пакети. Він пропонує високорівневий огляд архітектури системи шляхом групування пов’язаних класів, компонентів або модулів разом, що допомагає керувати складністю системи. Така організація не тільки спрощує розробку, але й робить систему більш модульною, полегшуючи її підтримку, масштабування та оновлення [14].

Для автоматизованого робочого місця, призначеного для архівного працівника, пакетна схема буде важливою для зображення ключових модулів системи, кожен з яких відповідає за певні завдання. Цей підхід оптимізує процес розробки, дозволяючи обробляти кожен модуль окремо, водночас показуючи, як вони взаємодіють один з одним. Це також робить майбутні вдосконалення або модифікації більш простими.

Пакет інтерфейсу користувача містить усе, що пов’язано з інтерфейсом системи, як-от представлення даних, шаблони та контролери, які безпосередньо взаємодіють із архіватором. Саме тут співробітники створюють, шукають і видають документи, пропонуючи безперебійну та зручну роботу.

Тим часом пакет бізнес-логіки займається основними функціями системи. Це включає в себе керування створенням документів, пошуком, видачею та обробкою необхідної логіки для перевірки та контролю доступу користувачів. Це гарантує, що такі операції, як обробка документів, виконуються правильно та безпечно.

Пакет взаємодії з базою даних служить мостом між системою та основною базою даних MySQL. Він містить компоненти, які полегшують зберігання та отримання даних, наприклад сутності та сховища. Цей пакет забезпечує узгодженість даних і ефективний зв’язок із базою даних, гарантуючи правильне керування документами та даними користувача та легкий доступ.

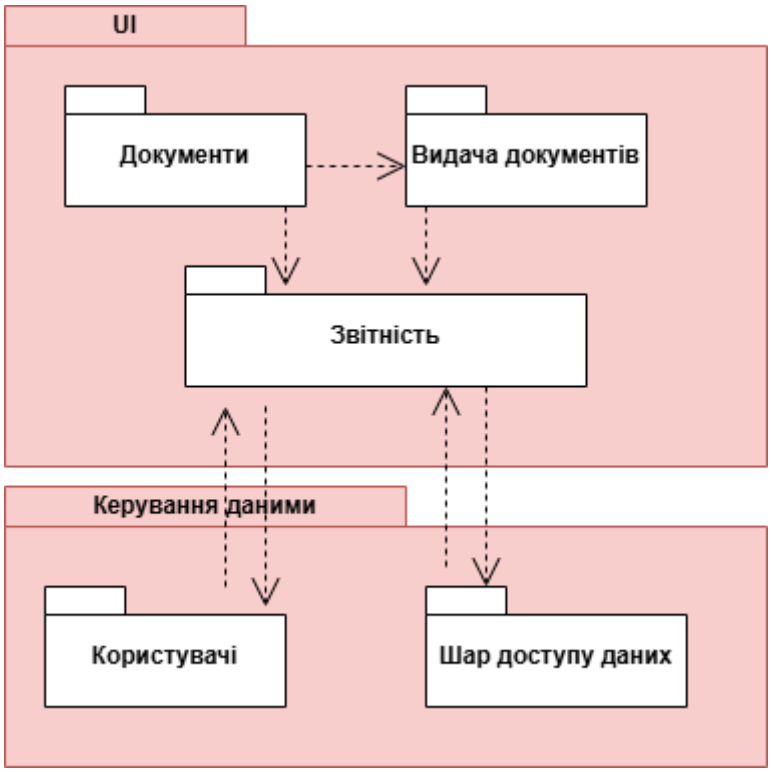

Рис. 12 Діаграма пакетів

Ця діаграма пакетів представляє структуру програмного забезпечення, розподілену на логічні модулі. Вона складається з кількох ключових пакетів, кожен із яких виконує свою роль у функціонуванні системи.

Пакет UI відповідає за взаємодію з користувачем. Він містить підмодулі, які керують документами та процесом їх видачі, забезпечуючи зручний інтерфейс для перегляду, пошуку й управління інформацією.

Пакет звітності займається формуванням та обробкою аналітичних даних, дозволяючи генерувати звіти про документообіг та отримані документи. Він інтегрується з іншими пакетами, щоб отримувати необхідну інформацію.

Пакет керування даними включає дві складові: модуль для роботи з користувачами та шар доступу даних. Перша частина відповідає за автентифікацію, управління ролями та правами доступу, а друга—за безпечне зберігання та обробку документів у базі даних.

Зв'язки між пакетами демонструють логічну взаємодію між компонентами. Наприклад, UI працює з пакетом видачі документів і звітності, а керування даними забезпечує доступ та контроль над користувачами та їхніми діями.

3.5 Вибір інструментарію для створення програмного забезпечення

Для розробки автоматизованого робочого місця працівника архіву було обрано різноманітні інструменти та технології, які забезпечують надійність, масштабованість та ефективність системи. Основні технології, обрані для цього проєкту, включають PHP, Symfony, MySQL, Doctrine ORM, Git, HTML, CSS і JavaScript. Кожен із цих інструментів відіграє вирішальну роль у різних аспектах розробки та впровадження системи.

PHP обрано як основну мову програмування для серверної частини програми. Він широко використовується для створення сценаріїв на стороні сервера та має розгалужену екосистему бібліотек, фреймворків і підтримки спільноти, що робить його надійним вибором для розробки веб-додатків. PHP забезпечує гнучкість і продуктивність, необхідні для обробки складної серверної логіки, такої як керування документами, ролі користувачів і контроль доступу, що є важливим для системи архівування.

Symfony, потужний фреймворк PHP, обрано за його здатність оптимізувати та прискорити розробку. Symfony надає повний набір повторно використовуваних компонентів та інструментів для створення веб-додатків. Він відповідає найкращим практикам, таким як шаблон MVC (Model-View-Controller), що робить кодову базу більш зручною для обслуговування,

організованою та легшою для масштабування. Вбудовані функції Symfony, такі як маршрутизація, створення шаблонів і безпека, створюють міцну основу для реалізації функціональності програми, одночасно забезпечуючи високий рівень безпеки та продуктивності.

В якості системи управління базою даних для проєкту обрано MySQL. MySQL — це надійна, широко використовувана система реляційної бази даних, яка забезпечує ефективне зберігання, керування та пошук даних. Його підтримка складних запитів, транзакцій і узгодженості даних робить його ідеальним для роботи з архівними даними та забезпечення цілісності документів і записів. Структурований характер реляційної моделі MySQL також забезпечує ефективну організацію даних архівної системи, полегшуючи зберігання документів і доступ до них за потреби.

Doctrine ORM використовується як інструмент об’єктно-реляційного відображення (ORM) для взаємодії з базою даних. Doctrine ORM спрощує процес роботи з базою даних, дозволяючи розробникам використовувати PHP-об’єкти замість написання складних SQL-запитів. Він автоматично обробляє такі операції з базою даних, як отримання, збереження, оновлення та видалення записів, оптимізуючи взаємодію з базою даних і забезпечуючи узгодженість між об’єктами програми та базою даних. Doctrine ORM також пропонує такі функції, як кешування та автоматичне створення схем, які покращують продуктивність і скорочують час розробки.

Для контролю версій обрано Git. Він забезпечує ефективну співпрацю між розробниками, відстежуючи зміни в кодовій базі, керуючи різними версіями та дозволяючи бездоганну інтеграцію нових функцій. Розподілений характер Git гарантує, що кожен розробник має повну копію репозиторію, забезпечуючи гнучкість і полегшуючи роботу в автономному режимі або повертаючись до попередніх версій, коли це необхідно. Git також полегшує перевірку коду та спільну розробку, що робить його важливим інструментом для керування життєвим циклом розробки програмного забезпечення.

Для інтерфейсу користувача вибрано HTML, CSS і JavaScript. HTML використовується для структурування вмісту веб-сторінок, гарантуючи, що дані системи представлені в організованому та доступному вигляді. CSS використовується для стилізації інтерфейсу користувача, забезпечуючи візуально привабливий і зручний досвід. JavaScript використовується для реалізації інтерактивних елементів, таких як динамічне оновлення вмісту, перевірка форм та інші функціональні можливості на стороні клієнта, які покращують взаємодію з користувачем і швидкість реакції. Разом ці зовнішні технології гарантують, що автоматизоване робоче місце інтуїтивно зрозуміле, швидко реагує та легко орієнтується для працівників архіву.

Підсумовуючи, поєднання PHP, Symfony, MySQL, Doctrine ORM, Git, HTML, CSS і JavaScript забезпечує надійний і ефективний набір інструментів для розробки автоматизованого робочого місця для співробітника архіву. Ці технології вибрано через їхню надійність, масштабованість і здатність сприяти розробці серверної та зовнішньої частини, гарантуючи, що кінцева система є функціональною та зручною для користувача. Цей набір інструментів дозволить створити безпечну, ефективну та добре структуровану систему, здатну виконувати всі завдання, пов’язані з керуванням документами та архівними процесами.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Для розробки та впровадження автоматизованого робочого місця працівника архіву вкрай важливо встановити відповідні вимоги до апаратного та програмного забезпечення, щоб забезпечити ефективну роботу системи та безперебійну роботу користувача. Ці вимоги закладають основу для побудови надійної інфраструктури, здатної виконувати завдання системи та забезпечувати безперебійну взаємодію працівників архіву.

Сервер, на якому розміщена система, повинен мати достатню обчислювальну потужність, пам’ять і ємність для зберігання. Щоб забезпечити безперебійне виконання внутрішніх процесів, таких як створення документів, пошук і керування, рекомендується багатоядерний процесор із тактовою частотою не менше 2,5 ГГц. Мінімум 8 ГБ оперативної пам’яті, бажано 16 ГБ для масштабованості, необхідні для забезпечення ефективної роботи під час великих навантажень даних. Для зберігання пропонується твердотільний накопичувач ємністю не менше 500 ГБ, щоб забезпечити високу швидкість читання/запису та підтримувати резервне копіювання. Крім того, ефективне рішення для резервного копіювання та резервування через зовнішнє сховище або хмарні системи допоможе захистити важливі дані від втрати.

Клієнтські машини, які працівники архіву будуть використовувати для взаємодії з системою, повинні бути оснащені сучасним двоядерним процесором (наприклад, Intel Core i3), не менше 4 ГБ оперативної пам’яті та не менше 100 ГБ пам’яті. Клієнтські машини також повинні мати дисплей із роздільною здатністю принаймні 1280x1024 для полегшення чіткого перегляду документів. Периферійні пристрої, такі як клавіатура та миша, є

стандартними, а додаткове обладнання, наприклад сканери, можна використовувати для оцифрування фізичних документів.

Надійне та стабільне мережеве з'єднання має важливе значення для зв'язку між клієнтськими машинами та сервером. Рекомендована швидкість мережі становить мінімум 10 Мбіт/с як для завантаження, так і для завантаження, що забезпечує безперебійний потік даних, особливо під час отримання чи видачі документів.

З точки зору програмного забезпечення, на сервері має працювати стабільна та безпечна операційна система. Зазвичай надають перевагу дистрибутивам на базі Linux, таким як Ubuntu Server або CentOS, через їх безпеку, стабільність і функції, орієнтовані на сервер. Однак Windows Server можна використовувати як альтернативу залежно від потреб інфраструктури. На стороні клієнта система повинна бути сумісна з сучасними операційними системами, такими як Windows 10/11, macOS або популярними дистрибутивами Linux.

Основні серверні технології для системи включають PHP, широко використовувану мову сценаріїв для веб-розробки. Додаток використовуватиме фреймворк Symfony для створення надійної серверної архітектури з MySQL як системою керування реляційною базою даних (RDBMS) для зберігання даних, таких як записи документів та інформація про користувачів. Система використовуватиме Doctrine ORM для ефективної взаємодії з базою даних, забезпечуючи безперебійну обробку даних.

На стороні інтерфейсу HTML5, CSS3 і JavaScript будуть використовуватися для структурування веб-програми, дизайну її зовнішнього вигляду та реалізації інтерактивності відповідно. Для оформлення користувальницького інтерфейсу можна інтегрувати зовнішні фреймворки, такі як Bootstrap або Tailwind CSS, для швидшого та більш чуйного впровадження дизайну.

Для контролю версій і спільної розробки Git використовуватиметься для відстеження змін у кодовій базі та забезпечення безперебійної командної

роботи. GitHub або GitLab розмістять репозиторій, забезпечуючи хмарну платформу для керування кодом. Шифрування SSL/TLS використовуватиметься для захисту передачі даних, забезпечуючи конфіденційність конфіденційної інформації документів. Система запровадить контроль доступу на основі ролей для керування дозволами користувачів і доступністю даних.

Нарешті, такі інструменти, як phpMyAdmin або MySQL Workbench, полегшать керування базами даних, тоді як IDE, такі як PhpStorm або Visual Studio Code, використовуватимуться для розробки програмного забезпечення. Для тестування та доступу до веб-програми будуть підтримуватися сучасні браузери, такі як Google Chrome, Mozilla Firefox або Microsoft Edge.

Ці вимоги до обладнання та програмного забезпечення створюють міцну основу для створення автоматизованого робочого місця для працівників архіву. Вибрані технології та інфраструктура забезпечать ефективність, надійність і масштабованість системи, дозволяючи працівникам архіву легко керувати та отримувати документи, зберігаючи безпеку та доступність даних.

Також було зроблено діаграму розгортання для візуального огляду деплою системи (Рис. 13).

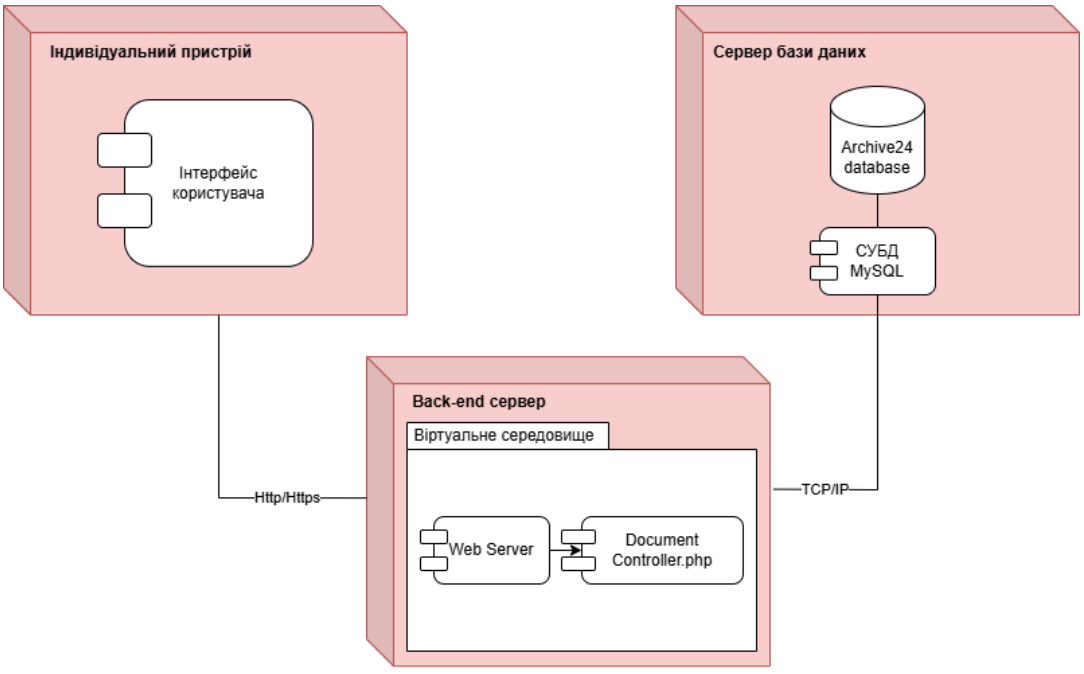

Рис. 13 Діаграма розгортання

4.2 Тестування системи

Запустивши систему, потрапляємо на форму авторизації, тут нам треба ввести логін та пароль користувача (рис. 14).

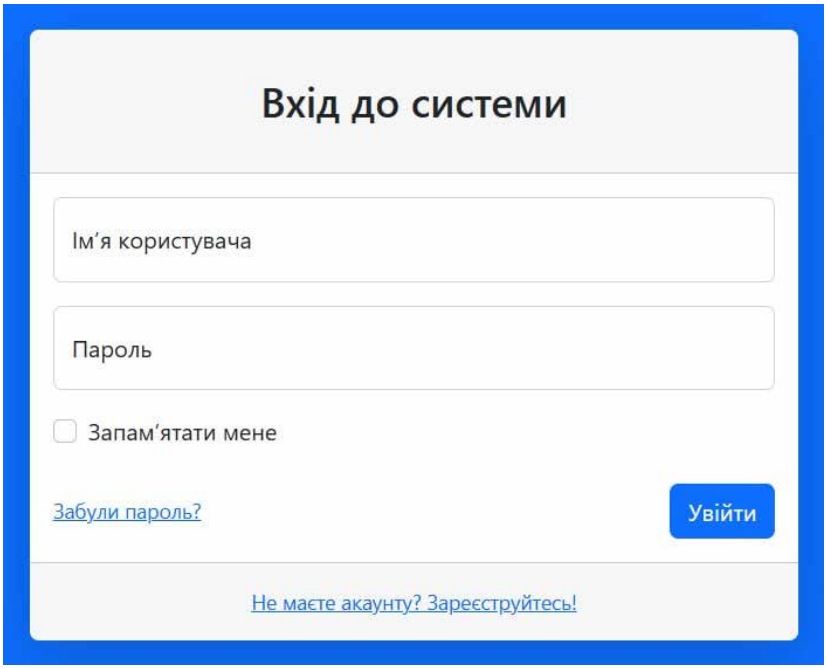

Рис. 14 Форма авторизації

Спочатку нас зустрічає сторінка з інформаційною панеллю. Тут показуються актуальні дані щодо використання документів у вигляді карток (Рис. 15).

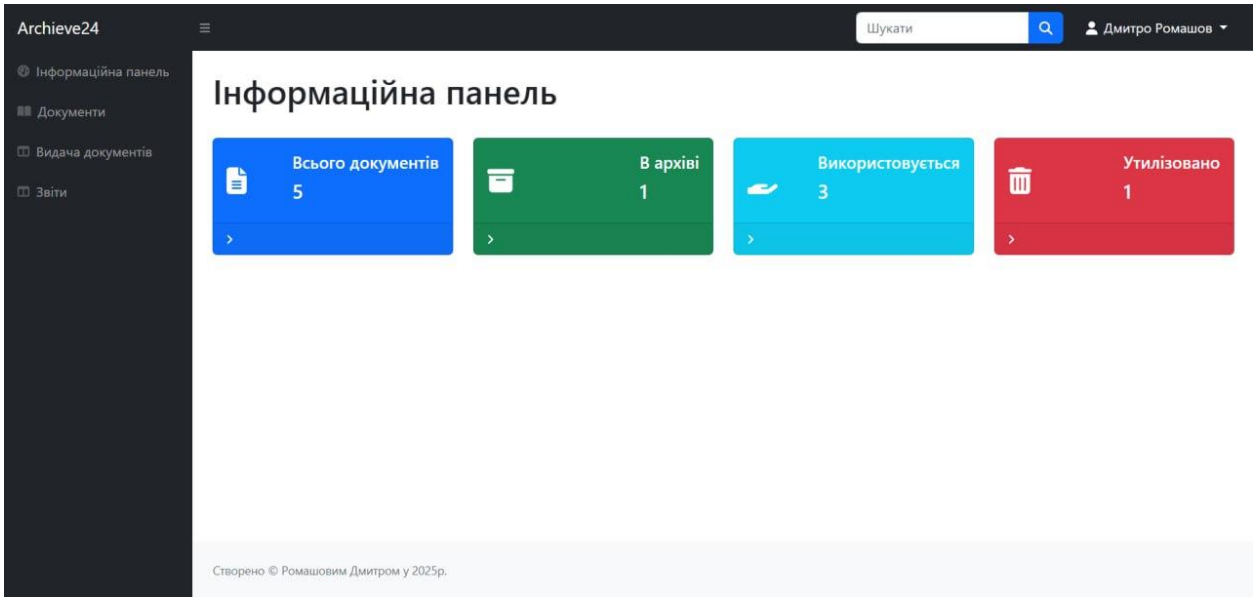

Рис. 15 Інформаційна панель

Переходимо на сторінку "Документи". Тут ми маємо список усіх доданих документів в архіві. Тут у нас є можливість пошуку документа за словами, кожен документ має можливість його подивитися, відредагувати або видалити (Рис. 16).

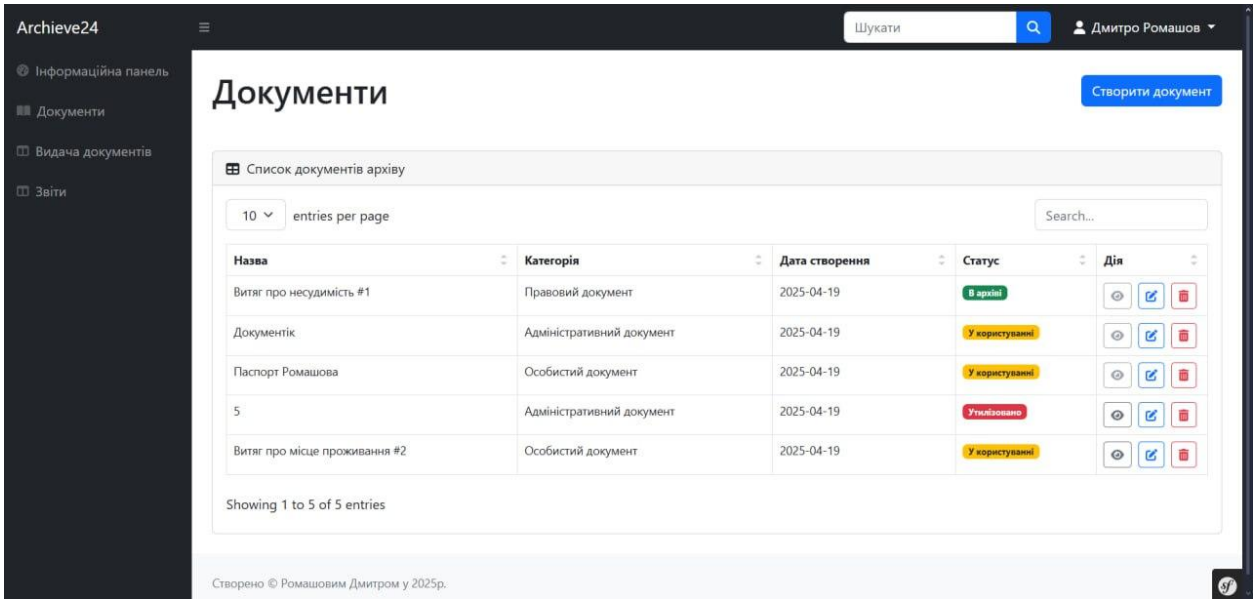

Рис. 16 Сторінка “Документи”

Якщо ми натиснемо на створення документа, то перед нами відкриється форма, де нам потрібно внести всі потрібні дані та завантажити файл, після чого документ буде успішно доданий до архіву (Рис. 17).

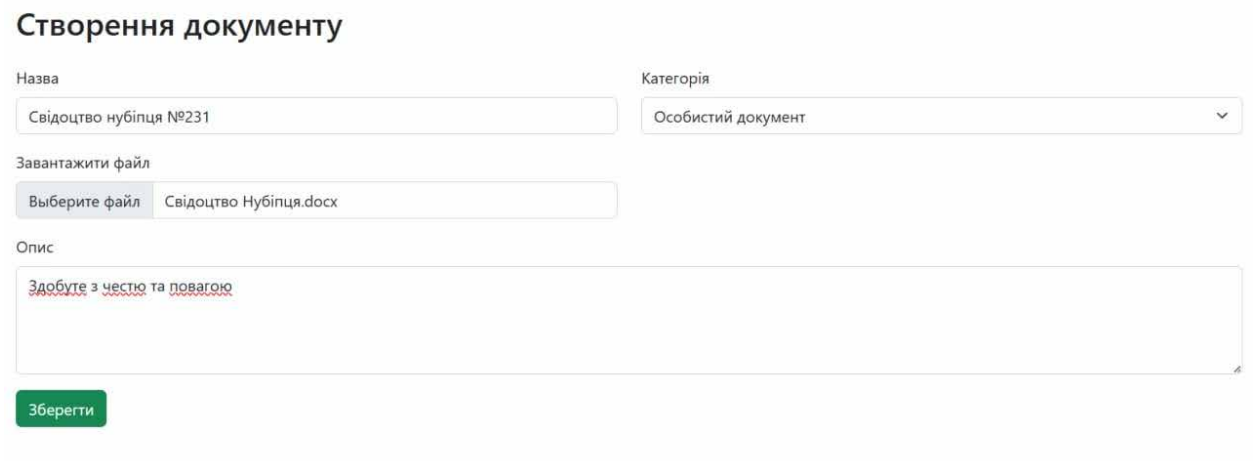

Рис. 17 Форма створення документу

Це наш новостворений документ зі статусом "В архіві" (Рис. 18).

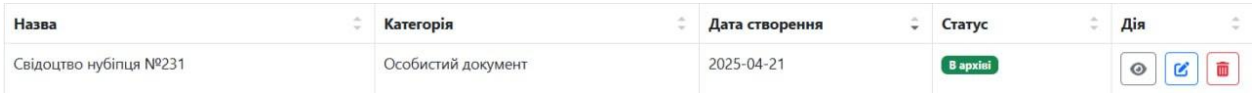

Рис. 18 Новостворений документ

Перейдемо на сторінку "Видача документів". Тут ми бачимо останні дії з документами: хто ними користувався, дата коли документ було вилучено, статус документа. Також тут є кнопки за допомогою яких можна повернути документ в архів або утилізувати його (Рис. 19).

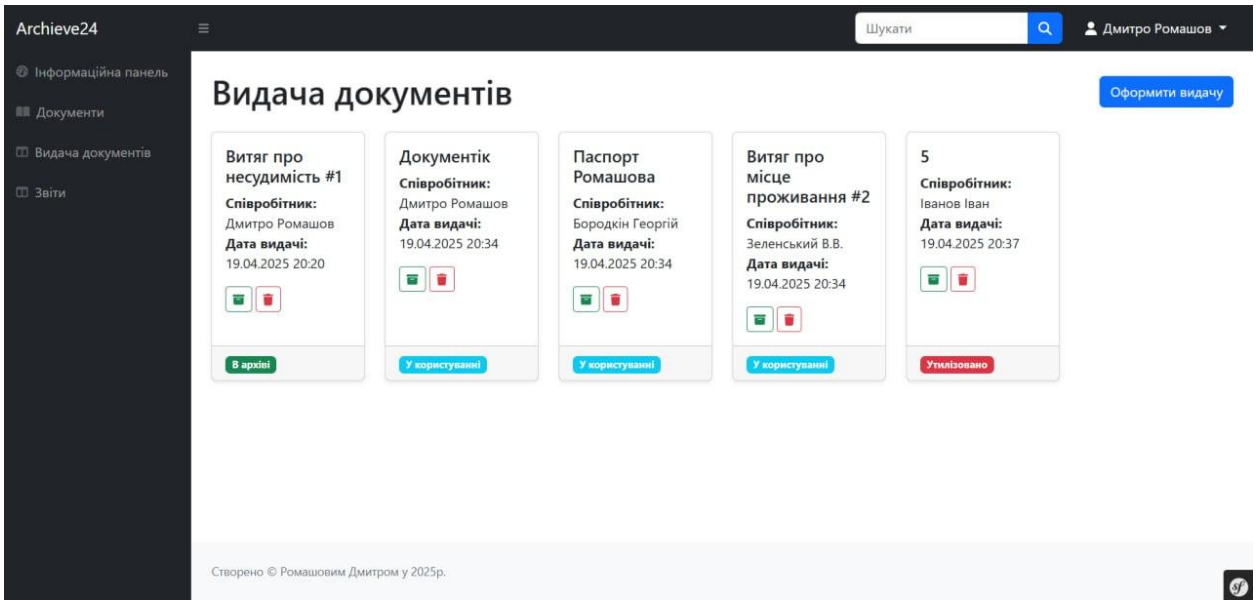

Рис. 19 Сторінка “Видача документів”

Спробуємо видати документ потрібному робітнику на користування, у нас відкривається форма видачі документа. Тут ми вибираємо певний документ і вписуємо потрібного співробітника, після чого підтверджуємо відправку форми (Рис. 20).

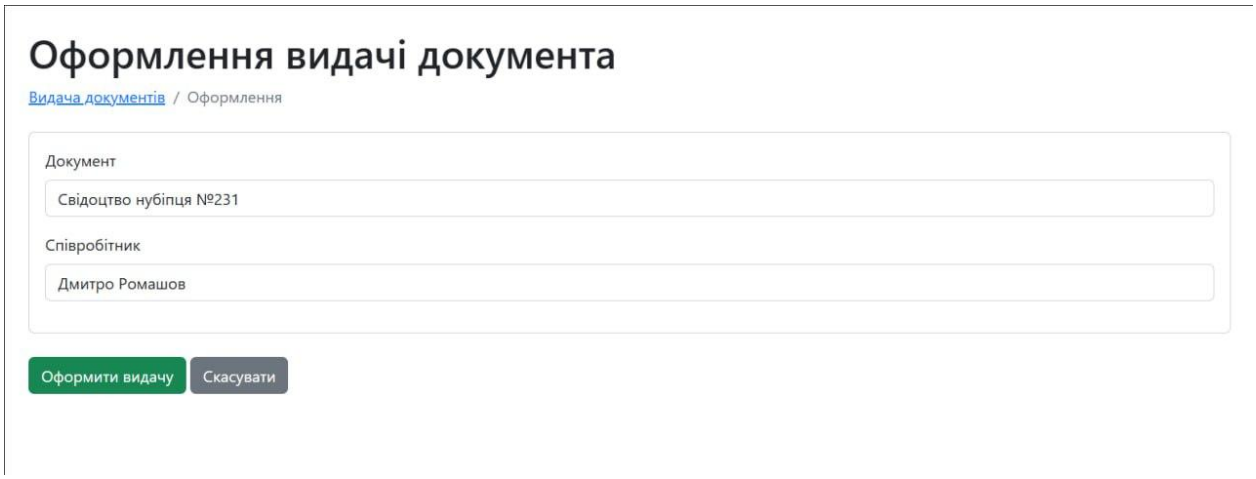

Рис. 20 Форма видачі документа

Після того, як оформили видачу, у нас з'явиться відповідний запис із цим документом (Рис. 21).

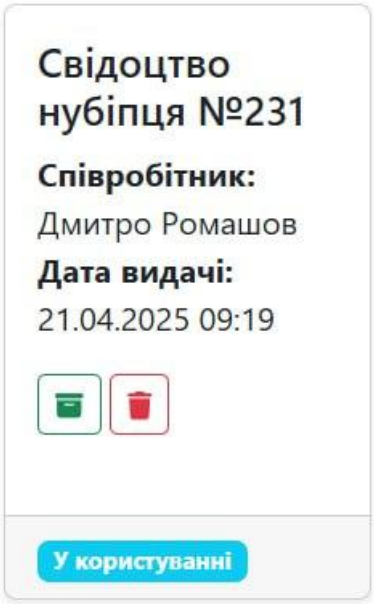

Рис. 21 Новостворена видача

Якщо ж ми натиснемо на зелену кнопку, ми повернемо документ назад до архіву, але перед цим у нас з'явиться модальне вікно з підтвердженням (Рис. 21).

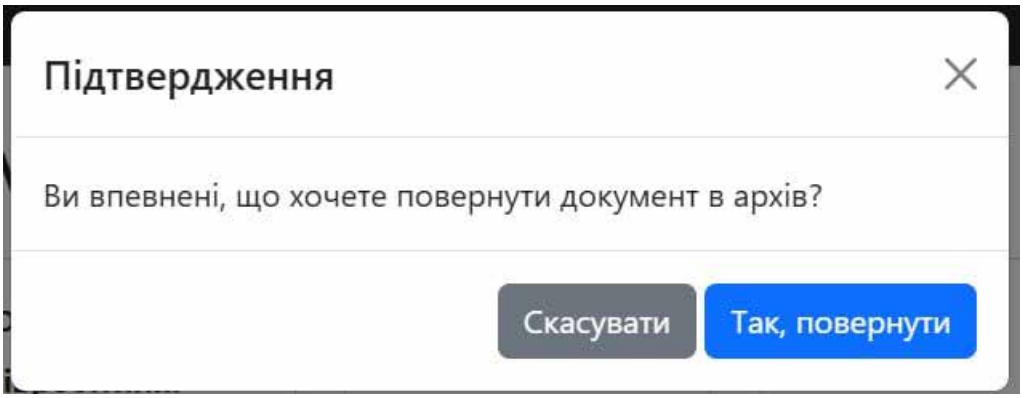

Рис. 22 Модальне вікно підтвердження дій з документом

Зайшовши на сторінку "Звіти", перед нам відкриється форма вибору періоду дат, вибираємо дати, що цікавлять нас, і перед нам з'явиться якась статистична інформація щодо використання документів в архіві у вигляді тексту та графіків (Рис. 23).

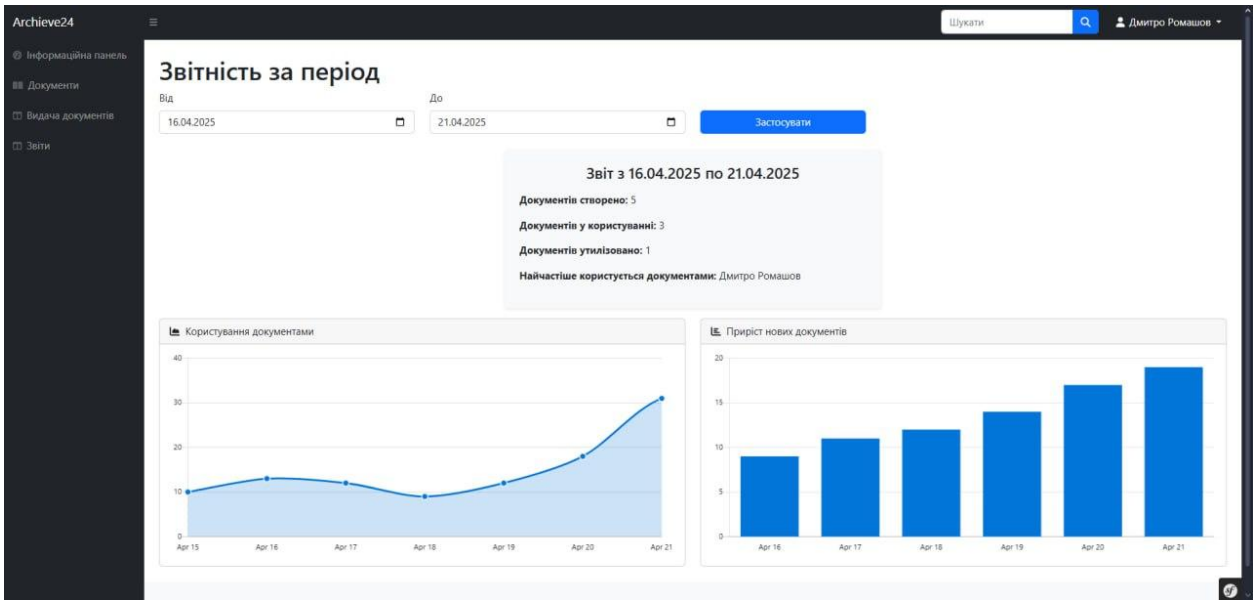

Рис. 23 Сторінка “Звіти”

Якщо в обраному періоді не знайдеться даних, ми отримаємо відповідне попередження (Рис. 24).

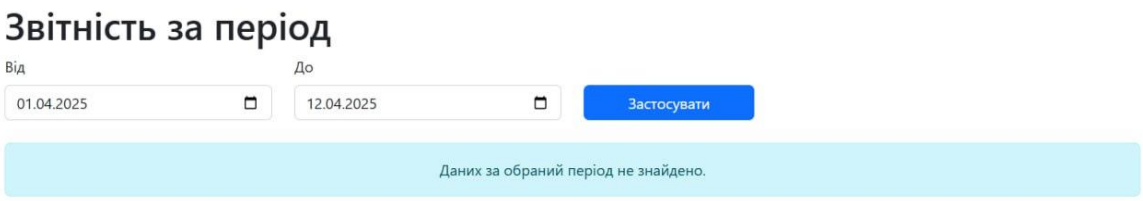

Рис. 24 Відсутність даних

ВИСНОВКИ

Підсумовуючи, розробка та створення автоматизованого робочого місця працівника архіву є вирішальним кроком до модернізації архівних процесів в організаціях. Завдяки переходу від ручних або частково оцифрованих методів до повністю автоматизованої системи можна досягти значних покращень щодо ефективності, точності та доступності даних. Система, розроблена для цієї мети, об’єднує основні функції, такі як створення документів, зберігання, пошук, видача та звітність, і все це в рамках раціоналізованого, зручного інтерфейсу.

Рішення використовувати PHP і фреймворк Symfony для розробки серверної частини, а також MySQL і Doctrine ORM для керування базами даних гарантує, що система є надійною, масштабованою та здатною ефективно обробляти великі обсяги даних. Завдяки впровадженню сучасних веб-технологій, таких як HTML, CSS і JavaScript, програма забезпечує чутливий та інтуїтивно зрозумілий інтерфейс для користувачів. Крім того, використання Git для контролю версій разом із шифруванням SSL/TLS для безпечної передачі даних гарантує надійність і безпеку системи.

Автоматизація архівних завдань не тільки заощадить дорогоцінний час і ресурси, але й зменшить ризик людської помилки, забезпечуючи цілісність і доступність важливих записів. Завдяки контролю доступу на основі ролей система захищатиме конфіденційну інформацію, надаючи авторизованому персоналу належний рівень доступу відповідно до їхніх ролей. Крім того, гнучкість і здатність до адаптації системи дозволяють її масштабувати та вдосконалювати відповідно до потреб організації з часом.

Крім підвищення ефективності, автоматизоване робоче місце для архівного працівника пропонує численні переваги з точки зору відповідності та перевірки. Завдяки цифровим системам організації можуть легко відстежувати зміни, контролювати доступ до документів і створювати

вичерпні звіти, які забезпечують дотримання правових і нормативних вимог. Ця прозорість допомагає підтримувати цілісність архівних даних, полегшуючи проведення аудитів і перевірку автентичності записів. Здатність системи зберігати докладні журнали взаємодії з документами забезпечує додатковий рівень підзвітності, що є вкрай важливим для організацій, які обробляють конфіденційні або юридично обов’язкові документи.

Зрештою, впровадження такої системи сприяє цифровій трансформації архівної практики, підвищуючи продуктивність і роблячи управління документами більш ефективним і безпечним. Ця робота служить практичним прикладом того, як сучасні методи розробки програмного забезпечення можуть вирішувати проблеми реального світу, покращуючи робочий процес і допомагаючи організаціям краще керувати своїми архівними даними. Знання, отримані в рамках цього проекту, дадуть цінну інформацію про розвиток подібних систем в інших секторах, які покладаються на управління величезними обсягами даних і документації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. M.E.Doc – [Електронний ресурс] – Режим доступу: https://medoc.ua
2. Zoho – [Електронний ресурс] – Режим доступу: https://www.zoho.com/mail/help/webmail-interface.html
3. Microsoft Docs. (2021). Layered architecture pattern – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/azure/architecture/patterns/layered
4. What is Component Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/
5. Deployment Diagram in UML: Definition, Examples & Components – [Електронний ресурс] – Режим доступу: https://study.com/academy/lesson/deployment-diagram-in-uml-definition-examples-components.html
6. What is a data flow diagram? – [Електронний ресурс] – Режим доступу: https://www.lucidchart.com/pages/data-flow-diagram
7. Entity Relationship Diagram - Data Modeling – [Електронний ресурс] – Режим доступу: https://www.visualparadigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html
8. UML 2 Tutorial - Package Diagram - Sparx Systems – [Електронний ресурс] – Режим доступу: https://sparxsystems.com/resources/tutorials/uml2/package-diagram.html
9. Документація Bootstrap (офіційний сайт) – [Електронний ресурс] – Режим доступу: www.getbootstrap.com/documentation.
10. W3Schools – [Електронний ресурс] – Режим доступу: www.w3schools.com

11. What is Sequence Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/
12. UML - Activity Diagrams – Tutorialspoint – [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm
13. Побудова діаграми класів – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ru/gpg_class-diagram.html
14. UML-діаграми класів – [Електронний ресурс] – Режим доступу: https://prog-cpp.ru/uml-classes/
15. Багаторівнева архітектура – [Електронний ресурс] – Режим доступу: https://simpleone.ru/glossary/mnogourovnevaya-arhitektura/
16. The Clean Architecture – [Електронний ресурс] – Режим доступу: https://medium.com/clean-code-channel/clean-architecture-the-solution-to-have-a-reusable-flexible-and-testable-code-ac7e296d1a75
17. Типи архітектури програмного забезпечення – [Електронний ресурс] – Режим доступу: https://medium.com/nuances-of-programming/4-типа-архитектуры-программного-обеспечения-917133174724
18. What is Unified Modeling Language (UML)? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
19. ПРОЄКТУВАННЯ ER-ДІАГРАМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ru/skills/proektirovanie-er-diagrammy/
20. Діаграма варіантів використання (UseCase diagram) – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ru/fd_use-case-diagram.html
21. ПРОЄКТУВАННЯ USE CASE ДІАГРАМИ. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СИСТЕМИ – [Електронний ресурс] – Режим доступу:

https://nationalteam.worldskills.ru/skills/proektirovanie-use-case-diagrammy-opredelenie-funktsionalnykh-vozmozhnostey-sistemy/

22. Прессман, Р. С. та Максим, Б. Р. (2015). Software Engineering: A Practitioner's Approach (8-е видання). Освіта McGraw-Hill.
23. Соммервіль І. (2011). Інженерія програмного забезпечення (9-е видання). Аддісон-Уеслі.
24. Піхлер, Р. (2010). Гнучке управління продуктами за допомогою Scrum: створення продуктів, які подобаються клієнтам. Аддісон-Уеслі.
25. Хоффер, Дж. А., Джордж, Дж. Ф., і Валачіч, Дж. С. (2013). Сучасний системний аналіз і проєктування (7-е видання). Pearson Education.
26. Дістель, Р. (2017). Теорія графів (5-е видання). Спрингер.
27. Бернерс-Лі, Т., Фінкельштейн, Л., і Трелл, Г. (1996). Гіпертекст і гіпермедіа (2-е видання). Аддісон-Уеслі.
28. Консорціум W3C (2019). Специфікація HTML5. W3C. Доступно за адресою: https://www.w3.org/TR/html5/
29. Документація MySQL. (2020). Довідковий посібник MySQL 8.0. Оракул. Доступно за адресою: https://dev.mysql.com/doc/
30. Гамма, Е., Хелм, Р., Джонсон, Р., і Вліссайдс, Дж. (1994). Патерни проєктування: елементи багаторазового об'єктно-орієнтованого програмного забезпечення. Аддісон-Уеслі.
31. Фаулер, М. (2002). Шаблони архітектури корпоративних додатків. Аддісон-Уеслі.
32. Ларман К. (2004). Застосування UML і шаблонів: Вступ до об'єктно-орієнтованого аналізу та проєктування та ітераційної розробки (3-є видання). Прентіс Холл.
33. Хенсел, М. (2017). Об'єкти PHP, шаблони та практика. Apress.
34. Грейді, Р. Б. (2011). Показники програмного забезпечення: строгий і практичний підхід (3-е видання). Видавнича компанія PWS.
35. Крухтен, П. (2003). Раціональний уніфікований процес: Вступ (3-є видання). Аддісон-Уеслі.

ДОДАТОК А

Фрагменти програмного коду. Функція створення документу

```
<?php

class DocumentController extends AbstractController
{
    #[Route('/documents', name: 'app_documents')]
    public function index(EntityManagerInterface $entityManager): Response
    {
        $documents = $entityManager->getRepository(Document::class)->findAll();

        return $this->render('documents/index.html.twig', [
            'documents' => $documents
        ]);
    }

    #[Route('/documents/create', name: 'app_document_create')]
    public function create(Request $request, EntityManagerInterface $em, SluggerInterface $slugger): Response
    {
        $categories = $em->getRepository(Category::class)->findAll();

        if ($request->isMethod('POST')) {
            dump($request->getMethod()); // должен быть POST
            dump($request->request->all()); // обычные поля
            dump($request->files->all()); // файл

            $title = $request->request->get('title');
            $description = $request->request->get('description');
            $categoryId = $request->request->get('category');
            $file = $request->files->get('filePath');

            $category = $em->getRepository(Category::class)->find($categoryId);

            $document = new Document();
            $document->setTitle($title);
            $document->setDescription($description);
            $document->setStatus(Document::STATUS_ARCHIVED);
            $document->setCategory($category);
            $document->setCreatedAt(new \DateTimeImmutable());
```

```
            if ($file) {
                $originalFilename = pathinfo($file->getClientOriginalName(), PATHINFO_FILENAME);
                $safeFilename = $slugger->slug($originalFilename);
                $newFilename = $safeFilename . '-' . uniqid('', true) . '.' . $file->guessExtension();

                try {
                    $file->move(
                        $this->getParameter('documents_directory'),
                        $newFilename
                    );
                } catch (FileException $e) {
                }

                $document->setFilePath($newFilename);
            }

            $em->persist($document);
            $em->flush();

            return $this->redirectToRoute('app_documents');
        }

        return $this->render('documents/create.html.twig', [
            'categories' => $categories
        ]);
    }

    #[Route('/documents/edit/{id}', name: 'app_document_edit')]
    public function edit(int $id, Request $request, EntityManagerInterface $em, SluggerInterface $slugger): Response
    {
        $document = $em->getRepository(Document::class)->find($id);
        $categories = $em->getRepository(Category::class)->findAll();

        if (!$document) {
            throw $this->createNotFoundException('Документ не знайдено');
        }

        if ($request->isMethod('POST')) {
            $title = $request->request->get('title');
            $description = $request->request->get('description');
            $categoryId = $request->request->get('category');
```

ДОДАТОК Б

Фрагменти програмного коду. Функціонал видачі документів на користування

```
<?php

namespace App\Controller;

use App\Entity\Document;
use App\Entity\Issuance;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;

class IssuanceController extends AbstractController
{
    #[Route('/issuance', name: 'app_issuance')]
    public function index(EntityManagerInterface $entityManager): Response
    {
        $issuances = $entityManager->getRepository(Issuance::class)->findAll();

        return $this->render('issuances/index.html.twig', [
            'issuances' => $issuances,
        ]);
    }

    #[Route('/issuance/create', name: 'app_issuance_create')]
    public function create(Request $request, EntityManagerInterface $em): Response
    {
        $documents = $em->getRepository(Document::class)->findAll();

        if ($request->isMethod('POST')) {
            $issuance = new Issuance();

            $document = $em->getRepository(Document::class)->find($request->request->get('document_id'));
            $currentUser = $this->getUser();

            $issuance->setDocument($document);
            $issuance->setUser($currentUser);
```

```
            $issuance->setUsername($request->get('username'));
            $issuance->setStatus(Document::STATUS_ISSUANCE);
            $issuance->setCreatedAt(new \DateTimeImmutable());
            $document->setStatus(Document::STATUS_ISSUANCE);

            $em->persist($issuance);
            $em->flush();

            return $this->redirectToRoute('app_issuance');
        }

        return $this->render('issuances/create.html.twig', [
            'documents' => $documents,
        ]);
    }

    #[Route('/issuance/{id}/archive', name: 'app_issuance_archive', methods: ['POST'])]
    public function archive(Issuance $issuance, EntityManagerInterface $em): Response
    {
        $issuance->setStatus(Document::STATUS_ARCHIVED);
        $issuance->getDocument()->setStatus(Document::STATUS_ARCHIVED);
        $issuance->setUpdatedAt(new \DateTimeImmutable());

        $em->flush();

        return $this->redirectToRoute('app_issuance');
    }

    #[Route('/issuance/{id}/utilize', name: 'app_issuance_utilize', methods: ['POST'])]
    public function utilize(Issuance $issuance, EntityManagerInterface $em): Response
    {
        $issuance->setStatus(Document::STATUS_DELETED);
        $issuance->getDocument()->setStatus(Document::STATUS_DELETED);
        $issuance->setUpdatedAt(new \DateTimeImmutable());

        $em->flush();

        return $this->redirectToRoute('app_issuance');
    }
}
```