Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Розробка автоматизованої розподіленої системи обліку документів адміністрації міського району

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-21-2
_____ /М. В. Кондратенко /

Керівник
кваліфікаційної роботи _____ / І. А. Котов /

Завідувач кафедри _____ / А. М. Стрюк /

Кривий Ріг
2025

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: бакалавр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А. М. Стрюк

«___» _____ 20__ р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ-21-2 Кондратенко Максиму Вадимовичу

1. Тема: Розробка автоматизованої розподіленої системи обліку документів адміністрації міського району

затверджено наказом по КНУ № 200 від «10» квітня 2025 р.

2. Термін подання студентом закінченої роботи: «15» червня 2025р.

3. Вихідні дані по роботі: розроблювана система повинна стабільно працювати в умовах високого навантаження на сервери та захищати дані від пошкоджень.

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): виконати аналіз існуючих методів розв’язання задачі, спроектувати систему з якомога витривалою архітектурою, розробити механізми захисту даних від пошкоджень та кіберзлочинців, здійснити програмну реалізацію розробленої системи, провести тестування розробленої системи.

5. Перелік ілюстративного матеріалу: графіки впливу навантажень на роботу системи, блок-схеми розроблених алгоритмів, схема взаємодії модулів системи, знімки екранних форм.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Огляд літератури за тематикою та збір даних	15.02.2025 – 18.02.2025
2	Проведення порівняльного аналізу прикладних методів розробки стресостійких розподілених систем обліку документів адміністрації міського району	19.02.2025 – 01.03.2025
3	Підготовка матеріалів першого розділу роботи	02.03.2025 – 13.03.2025
4	Розробка математичного забезпечення для розподіленої системи обліку документів адміністрації міського району	14.03.2025 – 23.03.2025
5	Підготовка матеріалів другого розділу роботи	24.03.2025 – 03.04.2025
6	Підготовка матеріалів третього розділу роботи	04.04.2025 – 17.05.2025
7	Розробка програмного забезпечення для розподіленої системи обліку документів адміністрації міського району	18.05.2025 – 29.05.2025
8	Оформлення пояснювальної записки	30.05.2025 – 09.06.25

Дата видачі завдання: «15» лютого 2025 р.

Студент _____ / М. В. Кондратенко/

Керівник роботи _____ / І. А. Котов /

РЕФЕРАТ

СИСТЕМА ОБЛІКУ ДОКУМЕНТІВ, DJANGO, JWT, АНАЛІТИКА НАВАНТАЖЕННЯ, ПОГОДЖЕННЯ ДОКУМЕНТІВ, РОЗПОДІЛЕНА СИСТЕМА, API, FRONTEND-ІНТЕРФЕЙС.

Пояснювальна записка: 71 с., 32 рис., 2 дод., 15 джерел.

Метою кваліфікаційної роботи є розробка інформаційної системи для автоматизованого обліку, погодження та зберігання службових документів у межах адміністрації міського району з можливістю інтерактивного моніторингу навантаження на сервер та адміністрування доступу.

У роботі проведено аналіз підходів до ведення електронного документообігу в публічному секторі. Оцінено технічні вимоги до розподілених систем та критерії вибору технологій. Обрано стек Django REST Framework для побудови backend-частини, React.js для frontend-інтерфейсу та SQLite3 як систему керування базами даних. Реалізовано механізм авторизації та автентифікації користувачів на основі JWT-токенів.

Особливу увагу приділено реалізації аналітичного модуля, що відображає завантаження процесора, кількість запитів до сервера та затримки відповідей у реальному часі. Для симуляції навантаження використано інструмент Locust. Система підтримує створення, погодження, архівацію документів, розподіл ролей між користувачами, адміністрування доступу до дій та повнофункціональний журнал подій. Усі критичні дії логуються, дані зберігаються відповідно до політик безпеки Django. Розроблену систему протестовано у тестовому середовищі. За результатами випробувань встановлено її працездатність, масштабованість та зручність використання.

Основні положення кваліфікаційної роботи опубліковано у вигляді тез доповіді на Всеукраїнській науково практичній WEB конференції КІСМ 2025.

ABSTRACT

DOCUMENT ACCOUNTING SYSTEM, DJANGO, JWT, LOADING ANALYTICS, DOCUMENT APPROVAL, DISTRIBUTED SYSTEM, API, FRONTEND INTERFACE.

Thesis in: 71 p., 32 fig, 2 app, 15 references.

The goal of the qualification work is to develop an information system for automated accounting, approval and storage of official documents within the administration of the city district with the ability to interactively monitor the server load and administer access.

The thesis analyzes approaches to electronic document management in the public sector. The technical requirements for distributed systems and criteria for selecting technologies are evaluated. The Django REST Framework stack was chosen to build the backend part, React.js for the frontend interface and SQLite3 as a database management system. A mechanism for authorization and authentication of users based on JWT tokens was implemented.

Particular attention was paid to the implementation of an analytical module that displays the CPU load, the number of requests to the server, and response delays in real time. The Locust tool was used to simulate the load. System All critical actions are logged, data is stored in accordance with Django security policies. The developed system has been tested in a test environment. The results of the tests showed its performance, scalability, and usability.

The main ideas of the qualification work were published at the All-Ukrainian Scientific and Practical WEB Conference KISM 2025.

ЗМІСТ

ВСТУП

У світі швидким темпом набирає популярності загальна цифрова трансформація державного управління та автоматизація процесів документообігу. Стабільність та коректність роботи таких систем є критично зажливими для великих підприємств, відомств та обласних установ. Ефективність роботи адміністративних структур безпосередньо залежить від здатності швидко обробляти, зберігати та знаходити велику кількість управлінських документів. Традиційні паперові системи вже не відповідають сучасним вимогам: вони малоефективні, ненадійні та піддаються впливу людського фактора. Автоматизовані системи обліку документів можуть значно підвищити продуктивність працівників, зменшити ризики втрати інформації, а також забезпечити прозорість і контроль за управлінськими рішеннями. Особливо важливими є розподілені системи, які дозволяють кільком структурним підрозділам працювати над єдиною базою даних одночасно, незалежно від їхнього фізичного розташування. Проте, впровадження таких систем супроводжується низкою технічних викликів, серед яких одна з найгостріших - це проблема перевантаження серверів. Якщо не звертати достатньої уваги на архітектуру, масштабованість і продуктивність, це може призвести до збоїв у системі. Метою цієї кваліфікаційної роботи є створення автоматизованої розподіленої системи обліку документів, яка відповідає потребам адміністрації міського району, враховуючи вимоги до масштабованості, надійності, безпеки та зручності використання. Під час роботи було проведено аналіз існуючих рішень, виявлено їхні сильні та слабкі сторони, сформульовано вимоги до нової системи, а також реалізовано її прототип, враховуючи особливості організаційної структури адміністративної установи.

1. ПРОБЛЕМА ПЕРЕНАВАНТАЖЕННЯ СЕРВЕРІВ РОЗПОДІЛЕНИХ СИСТЕМ ОБЛІКУ ДОКУМЕНТІВ

1.1. Аналіз проблеми перенавантаженням серверів розподілених систем обліку документів

Сьогодні розподілені інформаційні системи обліку документів стали основою для роботи сучасних органів державної влади, включаючи адміністрації міських районів. Ці системи дозволяють ефективно зберігати, обробляти, шукати та обмінюватися документами між різними відділами, службами та користувачами. Проте, зростання кількості користувачів, обсягів даних і складності запитів створює значне навантаження на сервери таких систем, що, в свою чергу, може призводити до проблем з продуктивністю, доступністю та надійністю.

Перенавантаження серверів означає ситуацію, коли ресурси серверів (процесорний час, оперативна пам’ять, дисковий простір, пропускна здатність мережі) використовуються на рівні, що перевищує їх проєктні або рекомендовані межі. Це може призвести до зниження швидкості роботи системи, збільшення часу відгуку, збоїв у функціонуванні або навіть тимчасової недоступності серверів [1].

Розглянемо основні причини виникнення перенавантажень:

- підвищене навантаження користувачів. Виникає, коли одночасно працює велика кількість співробітників або користувачів системи, які створюють, редагують або переглядають документи;
- некоректна архітектура системи. Виникає, коли система не має належного розподілу навантаження та навіть невелике збільшення кількості транзакцій може призвести до її краху;
- відсутність масштабування. Системи, які не можуть підтримувати горизонтальне масштабування або балансування навантаження,

швидко вичерпують свої ресурси, коли обсяги обробки даних зростають.

У контексті теми кваліфікаційної роботи система обліку документів складається з декількох компонентів.

Електронний документообіг у сфері адміністративної діяльності є цифровою платформою для створення, реєстрації, погодження, підписання та надсилання документів між структурними підрозділами. Він забезпечує фіксацію маршруту проходження кожного документа, автоматизує процеси делегування завдань, дозволяє відслідковувати терміни виконання доручень та зберігає юридичну силу документів завдяки кваліфікованому електронному підпису.

Система внутрішнього контролю інтегрується в документообіг з метою забезпечення відповідності управлінських процесів внутрішнім регламентам, посадовим інструкціям та вимогам законодавства. Вона дозволяє фіксувати відповідальних осіб за виконання конкретних дій, контролює строки виконання, надсилає автоматичні нагадування, генерує звіти для керівництва та допомагає запобігати порушенням у внутрішній організаційній діяльності.

Реєстраційно-довідкові бази виконують роль централізованих сховищ службової інформації, такої як дані про кореспонденцію, контрагентів, службові листи, постанови, акти тощо. Вони дозволяють швидко ідентифікувати вхідні та вихідні документи, здійснювати пошук за різними параметрами (дата, тип, автор, номер) та формувати відповідні статистичні звіти. Такі бази служать також джерелом інформації для аналітичних модулів.

Архів документів являє собою електронну підсистему довгострокового зберігання завершених справ і документів, які втратили оперативну актуальність, але підлягають збереженню відповідно до нормативів діловодства. Архів забезпечує можливість доступу до історичних даних, зберігає документи у форматах, придатних для довгострокового зберігання

(наприклад, PDF/A), та дотримується вимог до збереження метаданих, цілісності й незмінності документів.

Механізми взаємодії з іншими державними інституціями реалізовані у вигляді інтеграційних модулів, які дозволяють автоматично обмінюватися даними з зовнішніми системами – зокрема, з реєстрами територіальних громад, державними кадастрами, системами судового документообігу, порталами електронних петицій та запитів. Ці механізми забезпечують синхронізацію даних, дозволяють уникати дублювання інформації та підвищують загальну ефективність адміністративних процесів через міжвідомчу цифрову координацію.

Особливістю цих систем є складна організаційна структура з великою кількістю користувачів, які працюють з системою паралельно, нерідко з обмеженими технічними ресурсами (наприклад, застарілими ПК або слабкою мережею). Часто система обліку документів не має єдиного дата-центру, а розміщена на декількох фізичних серверах або хмарних майданчиках, що ускладнює управління навантаженням.

Такі фактори, як надмірне надсилання запитів до одних і тих самих ресурсів (наприклад, централізованої бази даних), імпорт та експорт великих обсягів файлів (PDF, Word, Excel), активне сканування та архівація документів, зумовлюють пікові навантаження, що можуть призводити до затримок і збоїв у роботі системи .

Регулярне перевантаження системи може призвести до низки негативних наслідків. Одним з таких є зниження швидкості обробки документів. Коли кількість запитів перевищує технічну здатність системи їх обробляти, сервери переходять у режим деградації продуктивності. Це проявляється у збільшенні часу відгуку – наприклад, операції, які в нормальному режимі виконуються за кілька секунд (реєстрація вхідного документа, відкриття справи, накладення підпису), можуть тривати декілька хвилин. Затримки в роботі створюють

труднощі в дотриманні строків реагування на звернення громадян або виконання управлінських рішень.

Більш серйозною загрозою є втрата або пошкодження даних, що може виникати внаслідок критичних перевантажень системи управління базами даних (СУБД). У пікові моменти, коли обсяг транзакцій досягає межі пропускної здатності, підвищується ризик виникнення помилок під час запису або оновлення інформації. Такі помилки можуть призводити до часткової або повної втрати документів, неправильного збереження службових реквізитів (дат, номерів, авторів), порушення логіки обліку або неможливості відновлення певної версії документа.

У разі серйозного навантаження можлива ситуація повної недоступності системи. Це відбувається тоді, коли сервери перестають відповідати на запити користувачів через повне вичерпання ресурсів або падіння ключових сервісів. У випадках, коли документообіг є критично важливим (наприклад, під час підготовки розпоряджень, проведення нарад або звітності перед вищими органами), така недоступність призводить до зупинки ділових процесів, зниження ефективності адміністрації та можливого порушення нормативно визначених строків обробки інформації.

Ще одним важливим аспектом є висока вартість підтримки системи у таких умовах. Перенавантаження часто вимагає негайного реагування від технічного персоналу, який змушений вручну відновлювати стабільність роботи – перезапускати служби, очищувати черги запитів, шукати та виправляти помилки в конфігураціях або логах. Це не лише забирає час, але й супроводжується фінансовими витратами на утримання додаткового IT-персоналу, придбання обладнання або впровадження аварійних рішень. У довгостроковій перспективі такі витрати можуть значно перевищити вартість своєчасного впровадження масштабованої інфраструктури.

За роки розвитку серверної діяльності було розроблено безліч методів боротьби з перенавантаженням. Деякі з них є дуже специфічними та

направленими на конкретні випадки. Основними ж методами вирішення проблеми є:

- вертикальне масштабування – збільшення потужностей існуючих серверів (додавання оперативної пам’яті, процесорів тощо). Метод є швидким, але має обмеження;
- горизонтальне масштабування – додавання нових серверів та балансувальників навантаження для розподілу запитів. Це дозволяє будувати більш стійкі архітектури;
- оптимізація баз даних – запровадження індексації, розділення баз даних, кешування запитів та асинхронної обробки транзакцій;
- використання хмарних технологій – перенесення частини навантаження в хмару, що забезпечує гнучке масштабування та резервування [2].

1.2. Загальний аналіз функціонування розподілених систем обліку документів

З огляду на зростання обсягів адміністративної документації та розширення мережі взаємодіючих структурних підрозділів, виникає об’єктивна потреба у впровадженні гнучких, масштабованих і надійних інформаційних систем. Однією з таких технологічних концепцій є розподілена архітектура, яка дозволяє не лише централізовано зберігати документи, а й забезпечувати ефективний обмін інформацією між усіма задіяними ланками організаційної структури [3].

Розподілені системи становлять собою інтегровані інформаційно-технологічні комплекси, що функціонують на основі децентралізованого розміщення обчислювальних ресурсів. Вони забезпечують автоматизовану реєстрацію, маршрутизацію, збереження, контроль проходження та пошук інформації або документів, які стосуються певних ділових процедур. Завдяки

поділу функціональних навантажень між окремими вузлами досягається підвищена стресостійкість системи, стабільність її роботи при пікових навантаженнях і можливість масштабування без зупинки загального процесу документообігу.

Головна ідея розподіленої системи полягає в тому, що різні компоненти програмного забезпечення реалізуються на окремих фізичних або віртуальних вузлах, які взаємодіють між собою через захищені канали зв’язку. Такі вузли можуть бути географічно розподілені – наприклад, розташовані у різних районах міста, у відокремлених структурних підрозділах адміністрації чи в зовнішніх організаціях-партнерах. Кожен вузол має автономний набір функціональних можливостей, зберігає частину або повну копію бази даних та здатен опрацьовувати користувацькі запити навіть за тимчасової втрати зв’язку з іншими елементами системи.

Головною перевагою такої архітектури є висока гнучкість адаптації до зміни організаційної структури або адміністративно-територіального устрою. Наприклад, при створенні нового відділу або об'єднанні адміністративних одиниць не виникає потреби в повному переоснащенні інфраструктури, достатньо підключити новий вузол до існуючої системи із заданими правами доступу та обміну даними. Крім того, зберігається можливість централізованого управління загальною політикою збереження, шифрування та видалення документів, що забезпечує відповідність вимогам інформаційної безпеки та законодавства про захист персональних даних.

Функціонування розподіленої системи неможливе без налагодженого механізму синхронізації даних між усіма вузлами. Це особливо актуально в умовах, коли документи можуть редагуватися або маршрутизуватися одночасно кількома користувачами. Для підтримки цілісності інформації застосовуються механізми контрольованої реплікації баз даних, часових міток, черг подій і журналів транзакцій. У разі виникнення конфлікту версій система

автоматично визначає актуальний запис або передає запит на ручне вирішення відповідальному працівнику.

Сфера розподілених систем не є новою у світі інформаційних технологій. За роки її існування було розроблено багато компонентів для оптимізації системи, маршрутизації запитів, зберігання та шифрування даних. У складі сучасних систем завжди наявні компоненти, кожен з яких виконує окрему, але критично важливу функцію у загальному технологічному циклі обробки адміністративних документів.

Клієнтська частина є основним інструментом взаємодії кінцевого користувача з системою. Вона реалізується у вигляді веб-застосунків, які функціонують у браузерах або у вигляді десктопних застосунків, що встановлюються безпосередньо на комп'ютери користувачів. Головна мета клієнтського застосунку – забезпечення інтуїтивно зрозумілого доступу до функціональності системи: створення, редагування, пошуку, маршрутизації та перегляду документів. Інтерфейс має підтримувати механізми авторизації та автентифікації, надавати можливості персоналізації та адаптуватися до ролей користувача у системі, надаючи відповідні права доступу. Принцип роботи системи «клієнт-сервер» зображено на рисунку 1.1.

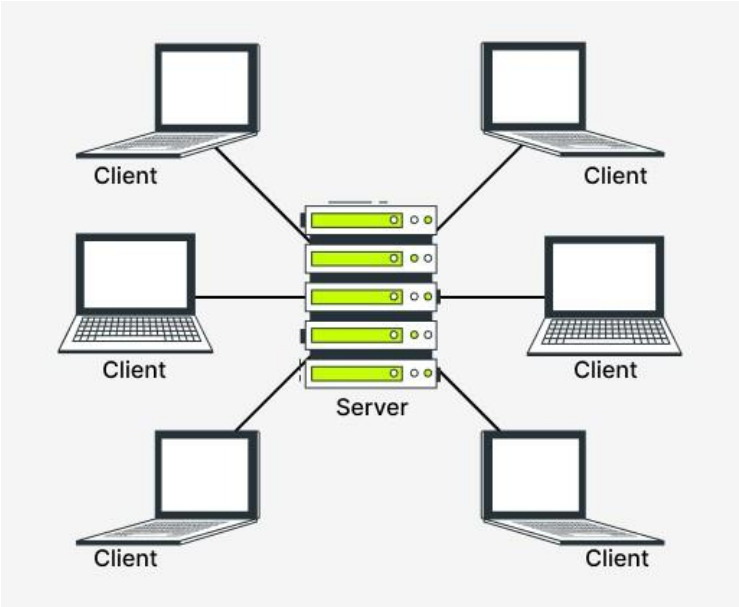

Рисунок 1.1 – Клієнт-серверна архітектура

Прикладні сервери відповідають за реалізацію бізнес-логіки системи – тобто за опрацювання сценаріїв, які визначають правила обробки документів. Наприклад, саме серверна логіка визначає, який маршрут має пройти певний документ, кому і в якій послідовності він має бути переданий, чи дотримано строки виконання, чи коректно накладено електронний підпис. Прикладні сервери працюють з отриманими від клієнтського інтерфейсу запитами, опрацьовують їх згідно із внутрішніми алгоритмами, і повертають відповідь або змінюють стан об’єктів у базі даних. Важливою характеристикою таких серверів є підтримка розподіленої обробки запитів і можливість масштабування горизонтальним способом – тобто через додавання нових екземплярів серверів у міру зростання навантаження. Сервери також можуть бути контейнеризовані за допомогою Docker або оркестровані через Kubernetes, що забезпечує швидке розгортання й оновлення системи.

Бази даних у розподілених системах можуть функціонувати як у централізованому, так і в децентралізованому режимі. У першому випадку єдине централізоване сховище обслуговує всі вузли системи, що вимагає високої продуктивності та відмовостійкості. У другому – дані можуть бути частково дубльовані або розділені між вузлами, що дозволяє зменшити навантаження та забезпечити локальний доступ до інформації. Для забезпечення надійності застосовуються механізми реплікації, резервного копіювання та контролю транзакцій. Крім того, у системах, де зберігаються великі обсяги сканованих копій документів, застосовуються окремі файлові сховища або об’єктні сховища даних на основі технологій типу MinIO або Amazon S3-сумісних API.

Одним з основних аспектів правомірного функціонування системи є підтримка електронного цифрового підпису (ЕЦП), який має юридичну силу згідно із законодавством України. Модулі ЕЦП інтегруються із системами кваліфікованого електронного підпису через відповідні криптографічні бібліотеки. Ці модулі забезпечують підписання документів посадовими

особами, перевірку справжності підпису, визначення часу накладання підпису та зберігання сертифікатів. Разом із підписом застосовуються алгоритми шифрування даних, що унеможливлює доступ до вмісту документа сторонніми особами під час його передачі або зберігання. Також важливим є реалізація журналу підписів, який дозволяє простежити історію внесення змін та авторів цих дій.

Ефективний та безпечний обмін даними між компонентами має критичне значення у розподіленій системі. Для цього використовуються різні типи комунікаційних протоколів: HTTPS – для передачі даних через інтернет або внутрішню мережу з використанням SSL/TLS-шифрування; SFTP – для передачі файлів у зашифрованому вигляді між серверами; VPN-тунелі – для організації безпечного з’єднання між територіально віддаленими вузлами. Окрім цього, часто впроваджуються механізми черг повідомлень, які дозволяють асинхронно обробляти події, балансувати навантаження та забезпечувати надійність передачі повідомлень у випадку збою одного з компонентів.

1.3. Аналіз системи організації бізнес-процесів обліку документів адміністрації міського району

Організація обліку документів є важливим елементом внутрішнього налаштування органів місцевого самоврядування. Вона забезпечує стабільне функціонування, прозорість прийняття рішень та належне документальне підтвердження дій посадових осіб. Система організації бізнес-процесів у цій сфері охоплює всі етапи життєвого циклу документа: від його реєстрації та первинного опрацювання до архівного зберігання або видалення згідно з нормативами.

У типовій міській адміністрації бізнес-процеси обліку документів реалізуються у межах діловодної служби та охоплюють як внутрішні

документи, так і ті, що надходять від громадян, підприємств, установ, органів державної влади тощо. Процеси організовано з урахуванням функціонального поділу відповідальності між діловодами, керівниками, виконавцями, архіваріусами та системними адміністраторами [4].

Перший етап - реєстрація документа. На цьому етапі відповідальний працівник створює у системі електронний запис, де фіксуються основні реквізити: дата надходження, відправник, короткий зміст, кількість аркушів, супровідна документація, пріоритет, категорія доступу. Якщо документ має форму паперового носія, він може бути оцифрований через сканування.

Другий етап – первинний розподіл. Визначається відповідальний виконавець, до документа додається резолюція, яка може бути сформована автоматично або вручну посадовою особою. У деяких адміністраціях на цьому етапі використовується система електронних резолюцій, що дозволяє фіксувати вказівки керівництва в електронному вигляді з використанням кваліфікованого електронного підпису.

Третій етап – контроль виконання. У рамках процесу передбачене автоматичне або ручне встановлення термінів виконання завдання, нагадування виконавцю про наближення граничної дати, та моніторинг виконання через систему звітності. Деякі адміністрації впроваджують систему кольорового кодування статусів документа (наприклад: червоний – прострочено, жовтий – виконується, зелений – виконано), що дозволяє оперативно оцінити навантаження та ефективність праці підрозділів.

Після завершення роботи над документом здійснюється етап його завершення та архівування. У випадку, якщо документ передбачає відповідь, вона також реєструється і зв’язується з оригіналом, утворюючи документальний ланцюжок. Архівування може відбуватися як в електронному вигляді, так і у вигляді передавання документа на зберігання в архівний відділ у фізичній формі. На рисунку 1.2 зображені усі ключові процеси, що виникають під час надання адміністративних послуг.

Також, у процесі організації обліку документів значну роль відіграють реєстраційно-довідкові бази, які формуються автоматично в ході реєстрації документа. До них входять: журнал вхідної/вихідної кореспонденції, картотека службових листів, перелік доручень, база виконавців, класифікатори типів документів. Саме ці бази дозволяють реалізувати швидкий пошук, аналітику, формування статистичних звітів та історичних оглядів потоків документів.

Рисунок 1.2 – Схема автоматизації процесів надання адміністративних послуг

Система внутрішнього контролю у документальному обліку виконує роль аудиту процесів документообігу. Це включає в себе логування всіх дій з документами (створення, редагування, ознайомлення, пересилання), моніторинг активності користувачів, формування логів доступу та перевірку дотримання строків виконання. У багатьох адміністраціях функції контролю покладено на уповноважених осіб, які за графіком формують звіти про виконання доручень.

Важливою складовою бізнес-процесів є механізми взаємодії з іншими державними інституціями, зокрема: обласними адміністраціями, судами,

територіальними підрозділами ДПС, Пенсійного фонду, Нацполіції тощо. Обмін документацією із цими установами здійснюється як у фізичному вигляді, так і в електронному. Відповідно, у системі обліку мають бути реалізовані модулі або API для прийому, верифікації та реєстрації вхідних електронних повідомлень.

Важливим чинником, що впливає на ефективність є ступінь автоматизації окремих операцій. У багатьох адміністраціях сьогодні спостерігається часткова автоматизація (електронна реєстрація та ведення журналів поєднується із паперовим листуванням, що ускладнює загальну логістику документа). Також зустрічаються випадки дублювання дій. Наприклад, документ реєструється в системі, але резолюція додається у вигляді рукописного напису. Така нерівномірність свідчить про недостатній рівень інтеграції та недосконаліть існуючої інфраструктури.

Проведений аналіз свідчить, що наявна система організації бізнес-процесів у сучасних адміністративних системах є функціональною, але потребує подальшої оптимізації. Основними проблемами є: низький рівень повної автоматизації, відсутність наскрізної цифрової взаємодії з зовнішніми організаціями, неоднорідність інтерфейсів облікових систем та недостатня масштабованість у разі зростання кількості документів або збільшення кількості користувачів. Враховуючи зазначене, модернізація системи шляхом упровадження автоматизованої розподіленої системи обліку документів є об’єктивною необхідністю для підвищення якості адміністративних процесів.

Варто також окремо виділити інформаційні ризики, що притаманні системам документального обліку. В умовах сучасних кіберзагроз адміністрації повинні забезпечувати захист конфіденційної інформації – персональних даних, внутрішньої службової інформації, бюджетної документації тощо. Це вимагає наявності систем резервного копіювання, обмежень на доступ до певних категорій документів, реалізації багаторівневої аутентифікації користувачів та застосування криптографічних засобів захисту

інформації. Такі засоби повинні бути інтегровані в бізнес-процеси організації, а не існувати паралельно.

В умовах воєнного стану, пандемії, евакуації працівників або аварійної втрати доступу до сервера виникає потреба в адаптації документального процесу до умов кризових ситуацій. Для її вирішення необхідне створення резервних каналів доступу, можливостей для дистанційної роботи з документами, мобільних інтерфейсів та процедур екстреного дублювання ключових функцій. У багатьох адміністраціях такі механізми реалізовані лише частково або відсутні взагалі, що становить значний ризик у разі надзвичайної ситуації.

1.4. Аналіз програмних комплексів – аналогів для обліку адміністративних документів

1.4.1. СЕДО Megapolis.DocNet

Для розробки актуальної та стабільної системи обліку документів необхідно провести ґрунтовний аналіз наявних інформаційних систем, що вже використовуються у публічному управлінні. Це дозволить не лише уникнути повторення наявних рішень, але й перейняти кращі практики в архітектурі, безпеці, функціоналі та зручності використання. Проаналізовано декілька ІС розроблених в Україні. Деякі з них не надають необхідної інформації без придбання підписки, тому у ході роботи буде розглянуто лише ті з них, що надають прикладну демонстрацію роботи своїх сервісів.

Система електронного документообігу Megapolis.DocNet, розроблена компанією «Інфосистеми Intecracy Group», є одним із найбільш поширених рішень для органів публічного управління, зокрема органів місцевого самоврядування в Україні. Цей програмний продукт позиціонується як універсальна платформа для управління життєвим циклом документів, що

дозволяє забезпечити централізований контроль та зручне адміністрування діловодства в установах будь-якого рівня.

Однією з головних переваг Megapolis.DocNet є модульний підхід до побудови системи. Завдяки цьому кожна установа може конфігурувати систему відповідно до власної організаційної структури та специфіки документообігу. Наприклад, можна реалізувати як простий лінійний облік вхідної/вихідної кореспонденції, так і багаторівневу маршрутизацію з автоматичним розподілом відповідальних осіб, часовими обмеженнями на виконання та контролем усіх стадій погодження.

Функціональні можливості системи включають:

- реєстрацію вхідних, вихідних і внутрішніх документів із подальшим формуванням реєстрів;
- організацію маршрутів погодження та візування документів за заздалегідь заданими схемами;
- контроль строків виконання документів і доручень, включаючи автоматичні нагадування виконавцям;
- збереження історії змін, контроль версій і ведення журналів подій;
- можливість накладення електронного цифрового підпису (ЕЦП) та інтеграцію з державними сервісами електронної ідентифікації;
- підтримку доступу до системи через веб-інтерфейс, що дозволяє працювати з документами в режимі реального часу незалежно від фізичного місцезнаходження користувача.

Megapolis.DocNet має потужний інструментарій для адміністрування доступу, де можлива деталізація прав до рівня операцій з конкретними видами документів, що відповідає вимогам інформаційної безпеки в державному секторі. Крім того, передбачено можливість інтеграції з Active Directory, що спрощує керування обліковими записами персоналу та забезпечує гнучкість у роботі з особистими даними працівників.

Особливістю системи є її здатність функціонувати як у локальному середовищі, так і в умовах розподіленої архітектури — наприклад, при використанні в мережі між кількома філіями або адміністративними підрозділами. Водночас, для реалізації повноцінної розподіленої інфраструктури потрібне впровадження додаткових технічних рішень, зокрема налаштування VPN-з'єднань, розгортання кластеризованих баз даних або реплікованих сервісів синхронізації.

Ще однією сильною стороною системи є розширені аналітичні можливості: формування звітності за запитом, аналітика по виконавській дисципліні, аналіз навантаження на структурні підрозділи, тощо. Ці інструменти дозволяють керівництву приймати обґрунтовані керівницькі рішення, спираючись на фактичні дані, отримані при роботі з документами. На рисунку 1.3 зображено інтерфейс системи «Megapolis.DocNet» [5].

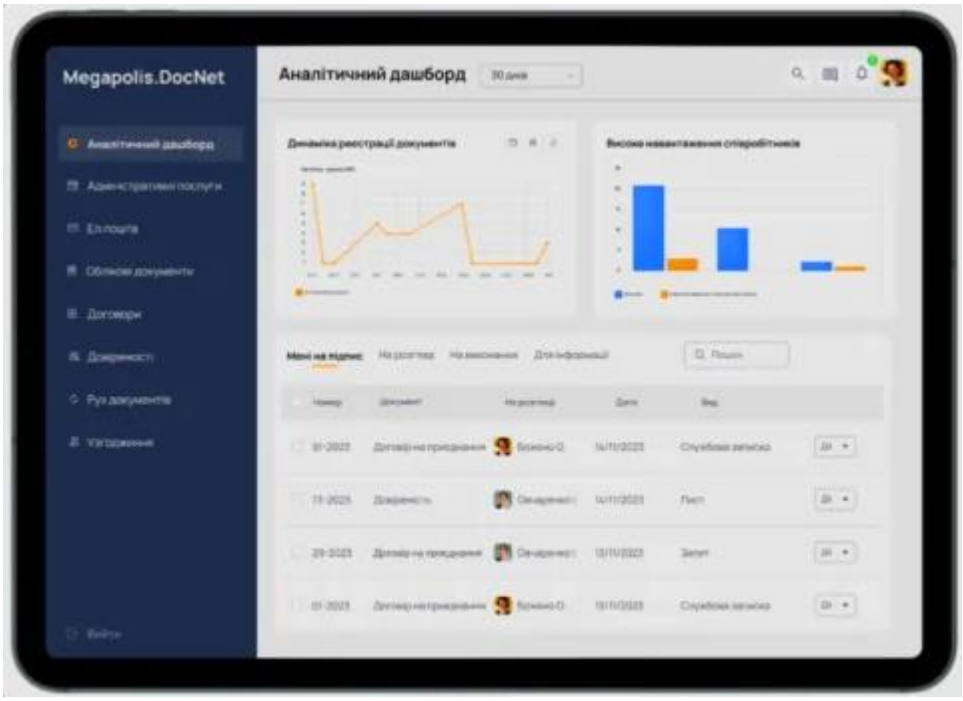

Рисунок 1.3 – Система «Megapolis.DocNet»

Щодо недоліків, то, з програмної точки зору, – вони відсутні. Система дотримується усіх стандартів та виконана за дуже високому рівні, на що слід

зауважити під час розробки. Megapolis.DocNet є зрілою, функціонально насиченою та адаптованою до українського діловодства платформою, яка може стати зразком для розробки нової автоматизованої розподіленої системи обліку документів адміністрації міського району.

1.4.2. Автоматизована Система Контролю та Обліку Документів

Система АСКОД є однією з найвідоміших в Україні платформ для автоматизації процесів діловодства та документообігу в органах виконавчої влади, включаючи міністерства, обласні державні адміністрації, міські ради та інші державні установи. Її функціональність охоплює повний життєвий цикл документа — від реєстрації та погодження до передачі в архів і довготривалого зберігання. Розробником системи є українська компанія ТОВ «Інститут Програмних Систем».

Важливою особливістю АСКОД є розвинена система контролю виконавської дисципліни. Інтерфейс та логіка роботи дозволяють здійснювати моніторинг кожного етапу обробки документа: від моменту його надходження до остаточного виконання або закриття, що особливо важливо в умовах державного управління, де необхідно дотримуватися жорстких строків реагування. АСКОД підтримує формування звітності про стан виконання доручень, оцінку ефективності роботи окремих підрозділів або співробітників, аналіз затримок у виконанні задач.

З технічної точки зору система забезпечує:

- багаторівневу систему прав доступу до документів і функцій — від обмеженого перегляду до повноцінного адміністрування;
- використання електронного цифрового підпису згідно з українськими нормами, а також шифрування документів на рівні передачі та зберігання;

- підтримку централізованого й децентралізованого зберігання документів, з можливістю конфігурування реплікації;
- можливість інтеграції з іншими інформаційними системами установи, зокрема CRM-системами, кадровими системами, реєстрами, за допомогою API та веб-сервісів;
- ведення електронного архіву з повнотекстовим пошуком і класифікацією документів за типами, категоріями, тематиками та іншими атрибутами.

АСКОД відповідає вимогам нормативно-правових актів України щодо ведення електронного документообігу в державному секторі, включаючи стандарти інформаційної безпеки та вимоги до архівного зберігання. Система дозволяє автоматизувати вхідну, вихідну та внутрішню документацію, створення розпорядчих документів, а також забезпечує фіксацію маршруту погодження і візування з точністю до конкретного працівника. На рисунку 1.4 зображено інтерфейс системи «АСКОД» [6].

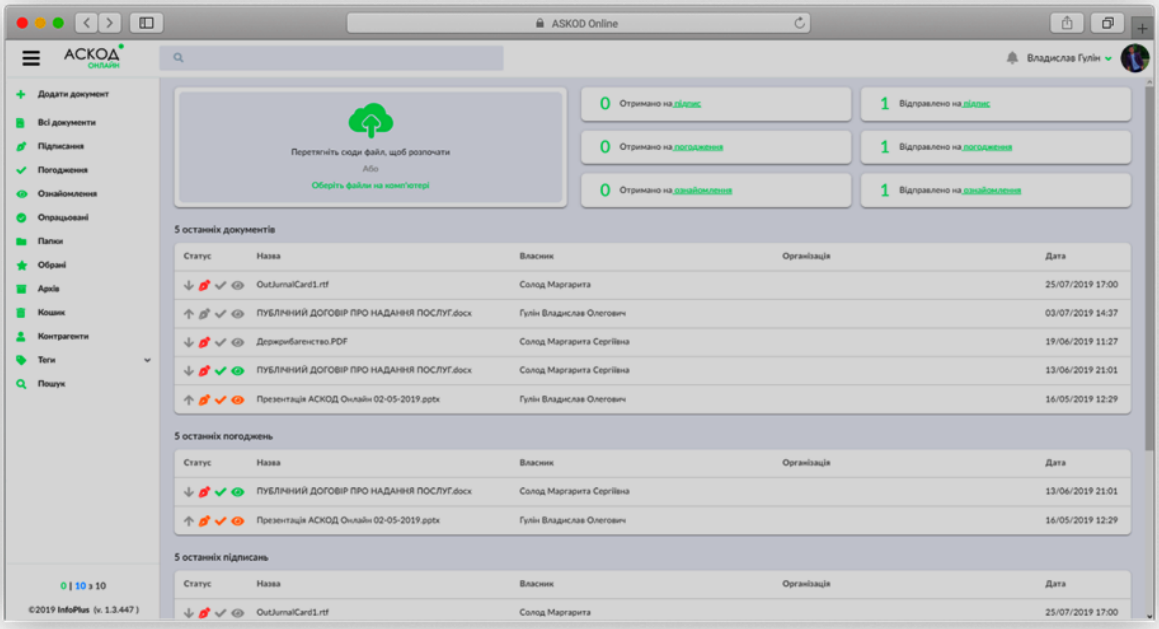

Рисунок 1.4 – Система «АСКОД»

Нажаль, система має й певні недоліки. Система передбачає досить централізовану модель керування, що робить її менш гнучкою для установ із

розподіленою структурою, де важлива автономність підрозділів. Також інтерфейс системи не відповідає сучасним вимогам юзабіліті, і потребує підготовки персоналу для повноцінного використання. АСКОД залишається одним із базових рішень у сфері державного електронного документообігу і може розглядатися як надійна основа для розробки нових розподілених рішень, орієнтованих на підвищену масштабованість і гнучкість.

Проведений аналіз дозволяє зробити висновок, що хоча сучасні системи електронного документообігу здатні покривати основні процеси діловодства, вони залишаються обмеженими у контексті побудови справді ефективної розподіленої архітектури. Однією з найважливіших проблем є значні витрати на впровадження та подальше обслуговування. Багато з існуючих систем мають складну інсталяційну процедуру, потребують спеціалізованого апаратного забезпечення, а також супроводу з боку розробника, що унеможливлює їх ефективне масштабування у децентралізованих структурах, що робить їх неуніверсальними.

Було виявлено, що автономність окремих елементів таких платформ часто обмежується. Це ускладнює роботу в умовах, коли необхідно забезпечити гнучку взаємодію між окремими підрозділами або територіально віддаленими структурами. Проблемним залишається також питання інтеграції. Навіть ті програмні комплекси, що заявляють про підтримку API або інших інтерфейсів обміну даними, не завжди забезпечують їх належну документацію чи підтримку, що значно ускладнює взаємодію з іншими державними або внутрішніми інформаційними системами.

Також слід зауважити, що інтерфейс деяких систем виявляється занадто громіздким або незручним для звичайних користувачів, що не мають спеціальної ІТ-підготовки. Це призводить до додаткових витрат часу на навчання персоналу, а також до зниження ефективності документообігу через людський фактор.

У світлі цих проблем постає потреба у створенні нового типу системи, яка б інтегрувала кращі властивості існуючих рішень, але водночас відповідала викликам сучасного інформаційного середовища. Така система повинна передбачати не лише можливість хмарного розгортання, а й підтримку гібридних інфраструктур, що дозволить гнучко адаптувати її до потреб конкретної адміністративної структури. Окрім цього, вона має бути здатною до інтелектуального керування потоками документів, включаючи динамічну маршрутизацію залежно від типу документа, його змісту чи організаційної ролі виконавця. Не менш важливим є забезпечення мобільного доступу, який дозволить працювати з документами з будь-якої точки, використовуючи планшети або смартфони, що особливо актуально для керівного складу та виїзних підрозділів. Водночас архітектура безпеки повинна гарантувати не лише автентифікацію користувачів, а й захист переданої інформації, можливість обмеження доступу на основі ролей, ведення журналів дій та інші інструменти контролю. Окрема увага має бути приділена налагодженню ефективного обміну даними з іншими державними інформаційними системами, що дозволить створити єдине інформаційне середовище, де документи не потребуватимуть дублювання, а їх обробка та перевірка здійснюватиметься у режимі реального часу.

1.5. Завдання розробки автоматизованої розподіленої системи обліку документів адміністрації міського району

Підсумовуючи проаналізовані проблеми та аналоги необхідно скласти повноцінне технічне завдання для розробки розподіленої системи обліку документів адміністрації міського району. Слід зазначити, що сфера електронного документообігу в Україні не є досконалою та має деякі розбіжності з європейськими системами, наприклад, паралельне існування паперових на електронних баз даних, що створює багато проблем у роботі з

документами. Необхідно розробити систему, яка буде здатна працювати в умовах нестабільності та великої кількості користувачів.

Основне призначення такої системи полягає в забезпеченні повного життєвого циклу документа в електронному середовищі від його створення або надходження, до архівного зберігання або знищення. Це включає не лише класичні функції діловодства, як-от реєстрація, класифікація, маршрутизація та зберігання, а й підтримку гнучких бізнес-процесів, що відображають реальні потреби управлінської діяльності. При цьому система має забезпечувати повноцінне функціонування навіть у випадках територіальної віддаленості структур, перебоїв у зв'язку чи необхідності тимчасової автономної роботи окремих підрозділів.

Необхідним є створення модульної, масштабованої архітектури, яка дозволить адміністрації розгортати систему поступово від окремих департаментів до повноцінного міжвідомчого рівня. Такий підхід сприятиме оптимальному розподілу навантаження, запобіганню перевантаженню серверів, а також підвищенню стресостійкості системи. Розподіленість має враховуватись не лише на рівні збереження та обробки даних, але й на рівні прав доступу, маршрутизації документів та налаштування локальних регламентів.

Особливу увагу слід приділити механізмам безпеки як технічної, так і організаційної. Передбачається реалізація комплексної системи ідентифікації та автентифікації користувачів із використанням електронного цифрового підпису, а також шифрування даних під час збереження та передачі. Журнали дій, система аудиту та контроль за доступом до документів мають гарантувати юридичну значущість та підзвітність усіх операцій у системі.

Також важливим є побудова інтуїтивно зрозумілого та адаптивного інтерфейсу користувача, що дозволить ефективно працювати в системі як висококваліфікованому ІТ-персоналу, так і рядовим співробітникам без глибоких технічних знань. Інтерфейс повинен бути доступним з різних

пристроїв, включаючи мобільні платформи, із забезпеченням повного функціоналу незалежно від типу клієнтського середовища. Це дозволить зекономити багато часу та коштів на навчання працівників.

Критичним завданням є забезпечення сумісності з іншими державними інформаційними системами, такими як Єдиний державний реєстр, інформаційні платформи органів юстиції, фіскальної служби тощо. Для цього передбачається створення відкритих і задокументованих API, що дозволять інтегрувати нову систему в єдиний інформаційний простір електронного урядування.

З огляду на зазначене, можна виділити такі ключові функціональні та технічні орієнтири для розробки:

- розробка математичної моделі оцінки навантаження серверів для виявлення та запобігання критичним станам у роботі системи;
- створення структурної моделі програмної системи, яка дозволить організувати логічну взаємодію між її компонентами та визначити ключові елементи архітектури;
- формування функціональної моделі системи, що забезпечить реалізацію повного життєвого циклу документа (від реєстрації до архівного зберігання або знищення);
- проєктування схеми бази даних, оптимізованої для роботи у розподіленому середовищі з можливістю масштабування та резервування даних;
- розробка алгоритмів програмних модулів, зокрема механізмів маршрутизації документів, контролю доступу та інтеграції з державними реєстрами;
- побудова моделі інтерфейсу користувача, адаптивного до різних рівнів технічної підготовки персоналу та різних типів пристроїв;
- програмна реалізація бази даних, модулів і інтерфейсу, із забезпеченням кросплатформеності та мобільного доступу;

- реалізація інструментів контролю навантаження серверів, що дозволить забезпечити стійкість та продуктивність системи в умовах пікових навантажень;
- розробка засобів адміністрування доступу, включаючи підтримку електронного цифрового підпису, логування дій та шифрування даних.

2.РОЗРОБКА МАТЕМАТИЧНИХ, СТРУКТУРНИХ І ФУНКЦІОНАЛЬНИХ МОДЕЛЕЙ РОЗПОДІЛЕНОЇ СИСТЕМИ ОБЛІКУ ДОКУМЕНТІВ АДМІНІСТРАЦІЇ МІСЬКОГО РАЙОНУ

2.1. Розробка математичних моделей оцінки ступеня навантаження серверів обліку документів

Оцінка та налаштування навантаження на сервери в розподілених системах електронного документообігу забезпечує стабільної та безперебійної роботи всієї інформаційної інфраструктури систем документообігу. Надмірне навантаження може призводити до затримок в обробці документів, втрати даних або навіть повної недоступності сервісів. Тому на етапі проєктування системи необхідно передбачити математичну модель, яка дозволяє об’єктивно оцінити навантаження на сервери та забезпечити ефективне масштабування.

Для оцінки ступеня навантаження використовується така система показників:

- середня кількість запитів до сервера за одиницю часу (λ);
- середній час обслуговування одного запиту (t_s);
- кількість активних користувачів (N);
- пропускна здатність сервера (μ).

Один з базових підходів до моделювання навантаження ґрунтується на теорії масового обслуговування. Зокрема, застосовуються моделі M/M/1 та M/M/m (де m – кількість серверів) для нерозподілених та розподілених систем відповідно. Коефіцієнт завантаження сервера в моделі нерозподіленої системи визначається за формулою:

$$\rho = \frac{\lambda}{\mu} \tag{2.1}$$

де $\rho \in [0,1]$. Значення $\rho = 1$ свідчить про критичне навантаження, коли система перебуває на межі перевантаження. У такому випадку середній час очікування та довжина черги стрімко зростають.

Середній час очікування запиту в черзі (W_s) та середня довжина черги (L_s) визначаються за формулами:

$$W_s = \frac{\rho}{\mu(1-\rho)}; \qquad L_s = \frac{\rho^2}{1-\rho} \tag{2.2}$$

Однак для систем з кількома серверами або обробниками слід використовувати розширену модель. У цьому випадку обчислення здійснюється через формулу Ерланга-С [7], що враховує ймовірність того, що всі m серверів зайняті:

$$P_{\text{очікування}} = \frac{\frac{(\lambda/\mu)^m}{m!} * \frac{m\mu}{m\mu-\lambda}}{\sum_{k=0}^{m-1}\frac{(\lambda/\mu)^k}{k!} + \frac{(\lambda/\mu)^m}{m!} * \frac{m\mu}{m\mu-\lambda}} \tag{2.3}$$

При розробці розподіленої системи документообігу адміністрації міського району доцільно застосовувати модель на базі M/M/m. Це дозволить не лише розраховувати поточне навантаження, але й адаптивно реагувати на зміну інтенсивності обробки запитів у реальному часі, автоматично перенаправляючи навантаження між вузлами системи.

Застосування математичних моделей такого типу дозволяє формалізувати завдання оптимального розміщення серверних ресурсів, розробити правила балансування навантаження, а також формувати обґрунтовані рішення щодо масштабування системи. В подальших етапах розробки система повинна мати у собі модулі моніторингу, які в реальному часі обраховують значення λ, μ, ρ та інші метрики, автоматично попереджаючи адміністратора про потенційні перевантаження.

2.2. Розробка структурної моделі програмної системи

Проєктування розподіленої автоматизованої системи обліку документів потребує створення чіткої структурної моделі, яка відображає логічну організацію її компонентів, взаємозв’язки між ними та основні напрями обміну даними. Така модель є основою для подальшої реалізації, масштабування та технічної підтримки системи. Структурна модель системи описується у

вигляді розподіленої архітектури, що забезпечує гнучкість, модульність та розмежування відповідальностей між окремими підсистемами. Основними складовими такої архітектури є: клієнтська частина, серверна частина, сервіси безпеки та модуль зберігання й обробки даних.

Клієнтська частина відповідає за взаємодію кінцевого користувача з системою. Користувачами є працівники районної адміністрації, секретарі, керівники відділів, архівісти, а також працівники віддалених підрозділів. Клієнтський інтерфейс реалізується через:

- веб-застосунок (браузерна версія системи);
- десктоп-клієнт (окремий застосунок для Windows-середовища);
- мобільний застосунок для операційних систем Android та iOS.

Клієнт забезпечує виконання операцій введення даних, перегляду реєстрів, запуску пошуку, візування документів, створення шаблонів тощо. Через нього передаються запити на обробку даних на серверну частину системи.

Сервер містить основну бізнес-логіку системи. Він реалізується у вигляді набору сервісів та мікросервісів, кожен з яких відповідає за окрему функціональність:

- реєстрація документів;
- маршрутизація документів між виконавцями;
- контроль виконання та звітність;
- електронний підпис і валідація;
- генерація номерів, штампів та реєстрів;
- керування доступом та ролями.

Важливою перевагою цієї архітектури є можливість горизонтального масштабування – у разі зростання навантаження окремі модулі можуть дублюватися на інших вузлах або серверах, забезпечуючи високу стресостійкість та мінімізацію часу відповіді. [8]

Модулі сервісів безпеки виконують функції шифрування, автентифікації та авторизації користувачів, а також логування подій. Його компоненти включають:

- модулі підтримки електронного цифрового підпису (ЕЦП);
- механізми двофакторної автентифікації;
- інструменти контролю доступу на основі ролей (RBAC);
- сервіси захищеної передачі даних (HTTPS, VPN, TLS).

Інтеграція з державними реєстрами автентифікації дозволяє забезпечити відповідність вимогам законодавства України щодо використання електронного документообігу в органах влади [9].

Модуль бази даних використовується для зберігання, резервного копіювання та обробки даних. Бази даних побудовані з урахуванням принципів кластеризації та реплікації для забезпечення стійкості до збоїв. Дані розміщуються у розподіленому середовищі з можливістю зберігання локальних копій документів у підрозділах. Система має використовувати реляційну СУБД, а також передбачати додаткові індексаційні та кешуючі механізми для прискорення пошуку та доступу до метаданих. Резервне копіювання виконується автоматично з регламентованою періодичністю, при цьому збереження архівів здійснюється на окремих захищених носіях. Для цього використовуються RESTful API, SOAP або інші стандартизовані протоколи, що забезпечують сумісність систем на рівні обміну повідомленнями та документами.

У структурній моделі важливим є також наявність сервісів інтеграції, що забезпечують взаємодію з іншими державними або внутрішніми системами, зокрема:

- системами керування фінансами;
- системами реєстрації звернень громадян;
- зовнішніми архівними сервісами;
- національними реєстрами електронної взаємодії.

Для візуалізації розробленої моделі було створено схему (рис. 2.1), яка ілюструє зв’язки між основними рівнями та компонентами. Схема допомагає зрозуміти логіку взаємодії частин системи, а також є невід'ємною частиною документації для працівників.

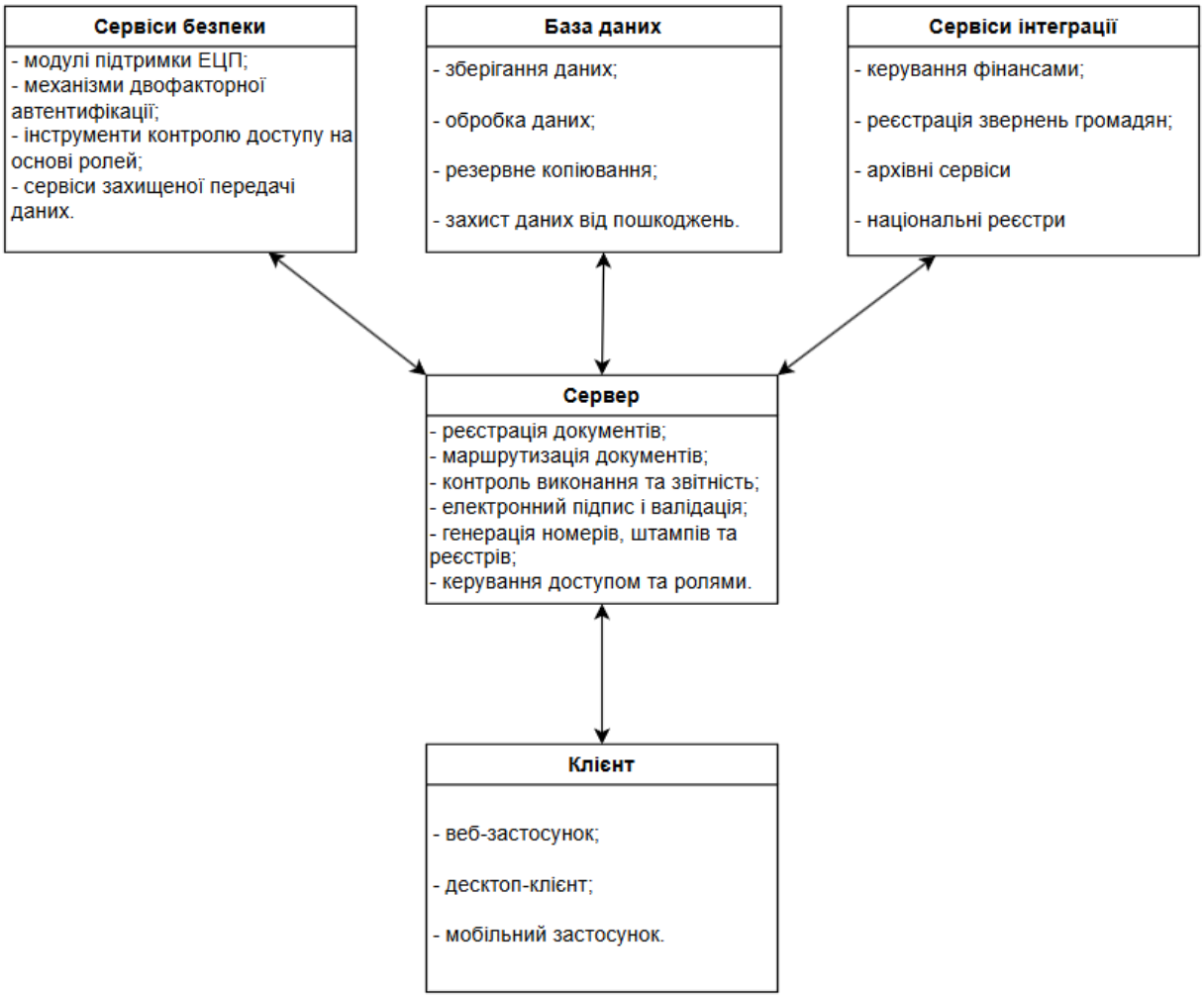

Рисунок 2.1 – Структура розподіленої системи обліку документів

Також, для більш детального розуміння структури взаємодії об’єктів системи було створено діаграму зв’язків сутностей (рис. 2.2). На схемі зображені основні діючі елементи системи, їх ключові компоненти та типи зв’язків. Центром системи є сервер, який виконує операції з документами, отриманими з бази даних, на запит користувача. Після виконання операції відбувається логування, що забезпечує підвищений рівень безпеки даних.

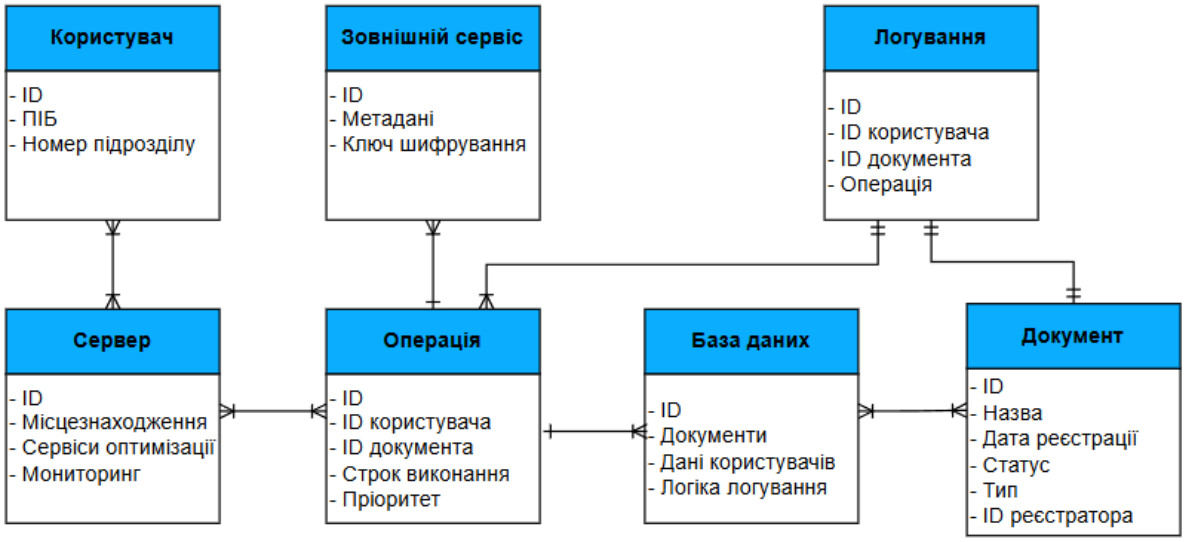

Рисунок 2.2 – ER-діаграма розподіленої системи обліку документів

На рисунку 2.3 зображено діаграму контекстного рівня функціональної моделі розподіленої системи обліку документів. Вона демонструє основну взаємодію користувача, системи, зовнішніх інтерфейсів та бази даних.

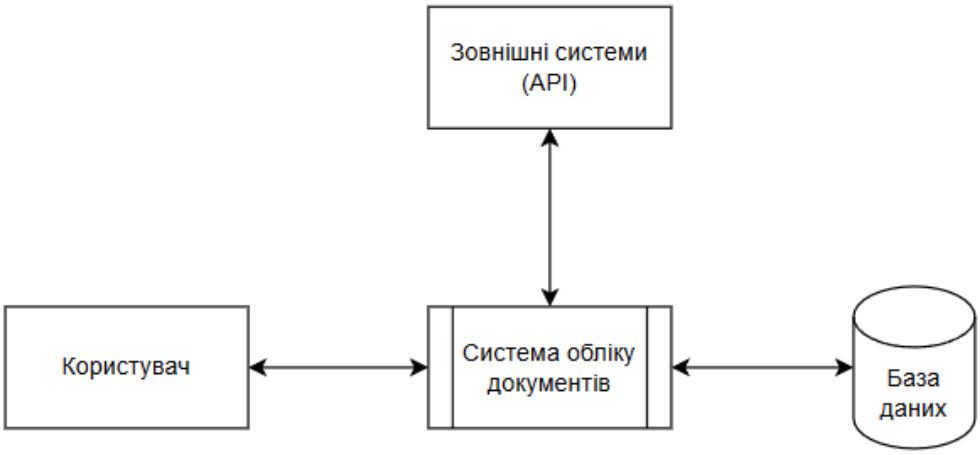

Рисунок 2.3 – Контекстна діаграма розподіленої системи обліку документів

На рисунку 2.4 зображено діаграму першого рівня деталізації системи обліку документів. На цій діаграмі було деталізовано основні функціональні підсистеми: інтерфейс користувача, модулі реєстрації, маршрутизації та

контролю виконання документів, а також взаємодію з сервером бази даних і зовнішніми системами, що надає змогу більш досконало зрозуміти принцип роботи системи обліку документів адміністрації міського району.

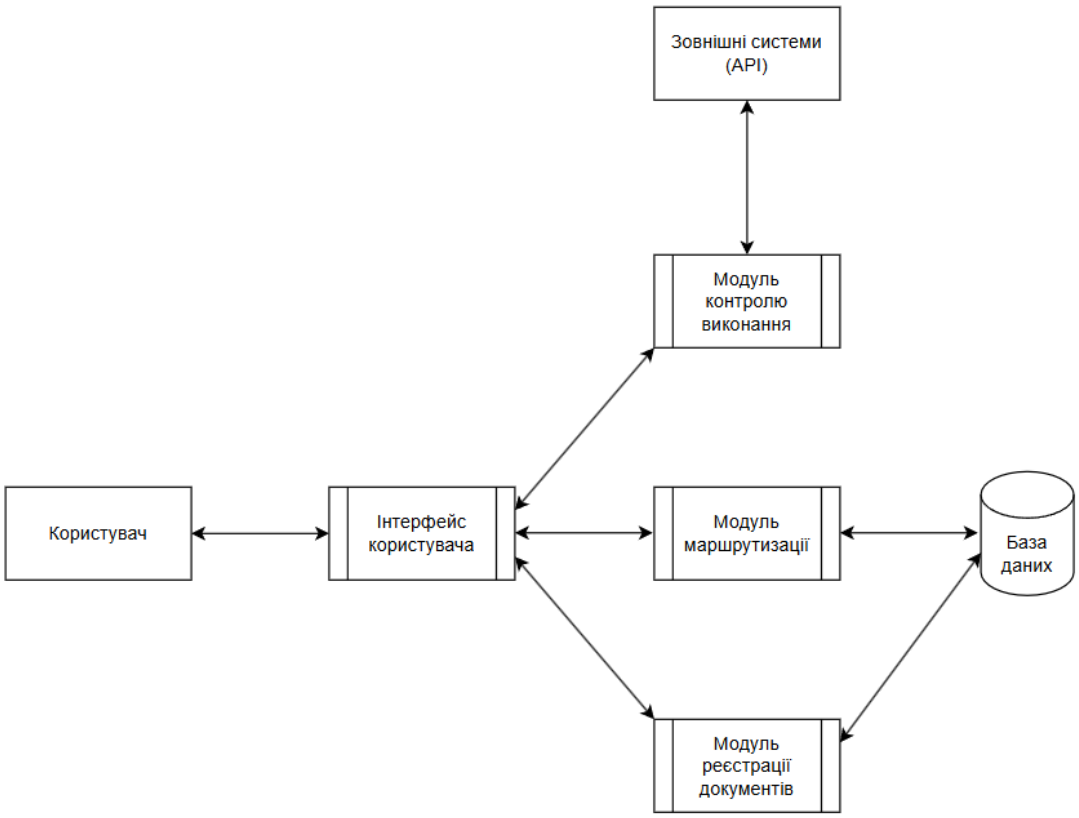

Рисунок 2.4 – DFD-діаграма першого рівня функціональної моделі розподіленої системи обліку документів

Розроблена структурна модель також враховує особливості територіальної розподіленості районної адміністрації, де окремі структурні підрозділи можуть працювати в умовах нестабільного або обмеженого інтернет-з'єднання. Для таких випадків передбачається локальне кешування документів з подальшою синхронізацією змін при відновленні каналу зв’язку. Такий підхід забезпечує безперервність обробки документів навіть у віддалених районах, а також мінімізує ризики втрати даних під час короткочасних перебоїв мережі. Окремо слід відзначити можливість гнучкого масштабування системи. В залежності від навантаження, нові сервери обробки

можуть бути додані до інфраструктури без зупинки системи. Така масштабована модель дозволяє адаптувати систему до зростаючих обсягів документообігу без необхідності повної модернізації IT-інфраструктури.

Для досягнення високої продуктивності в архітектуру системи можуть бути інтегровані технології балансування навантаження, які автоматично розподіляють потоки запитів між наявними серверами. Це не лише зменшує ризик перевантаження окремих вузлів, а й підвищує загальну стабільність системи в умовах пікового навантаження, наприклад, під час масового обміну документами або реєстрації великих обсягів вхідної кореспонденції.

2.3. Розробка функціональної моделі розподіленої системи обліку документів

Розробка системи обліку документів потребує детального опису її внутрішніх функціональних процесів. Для цього слід створити модель, яка є концептуальним відображенням основних операційних процесів системи для забезпечення повного життєвого циклу документа. Така модель дозволяє чітко ідентифікувати функції, взаємозв’язки між компонентами, а також вхідні та вихідні дані кожного модуля. Вона побудована з урахуванням принципів модульності, розподіленості та сервіс-орієнтованого підходу. Кожен функціональний блок виконує певну роль у загальному процесі документообігу. У складі цієї моделі є кілька ключових функціональних підсистем.

Підсистема реєстрації документів відповідає за первинне введення документів до системи. Реалізується можливість реєстрації вхідних, вихідних та внутрішніх документів з автоматичним або ручним присвоєнням номеру, зазначенням автора, дати, типу документа, а також відправника або одержувача.

Підсистема маршрутизації формує маршрут проходження документа на основі заздалегідь визначених правил або індивідуального сценарію. Ця підсистема визначає порядок розгляду, візування, погодження та підпису документів між посадовими особами. Може підтримувати як послідовну, так і паралельну маршрутизацію.

Підсистема контролю виконання забезпечує контроль за строками виконання документів та доручень. Включає механізми нагадувань, трекінгу статусів та формування звітів про виконання. Забезпечує прозорість та підзвітність дій виконавців.

Підсистема роботи з електронним підписом відповідає за підписання та перевірку документів за допомогою КЕП (кваліфікованого електронного підпису). Інтегрується з зовнішніми сервісами центральних засвідчувальних центрів.

Підсистема пошуку та доступу до документів реалізує механізми повнотекстового та фільтрованого пошуку документів за реквізитами, тегами, контекстом. Надає інтерфейси для обмеження доступу до документів на основі ролей користувачів і рівнів допуску.

Підсистема архівування відповідає за перенесення завершених документів до електронного архіву з відповідним рівнем захисту та довгострокового зберігання. Передбачає функції резервного копіювання, експорту та аудиту змін.

Підсистема аналітики та звітності формує узагальнені та детальні звіти щодо обсягів документообігу, навантаження на підрозділи, дотримання строків, ефективності обробки тощо. Це дозволяє керівництву приймати обґрунтовані рішення.

На рисунку 2.5 зображено функціональну схему роботи системи обліку документів. Створена діаграма враховує усі зазначені функціональні підсистеми, окрім звітності. Це пов’язано з тим, що система аналітики використовується для збору статистики усієї системи та не є частиною точкової

роботи з документом. Підсистеми пошуку, підпису та маршрутизації було спрощено та включено до відповідного за порядком блоку діаграми.

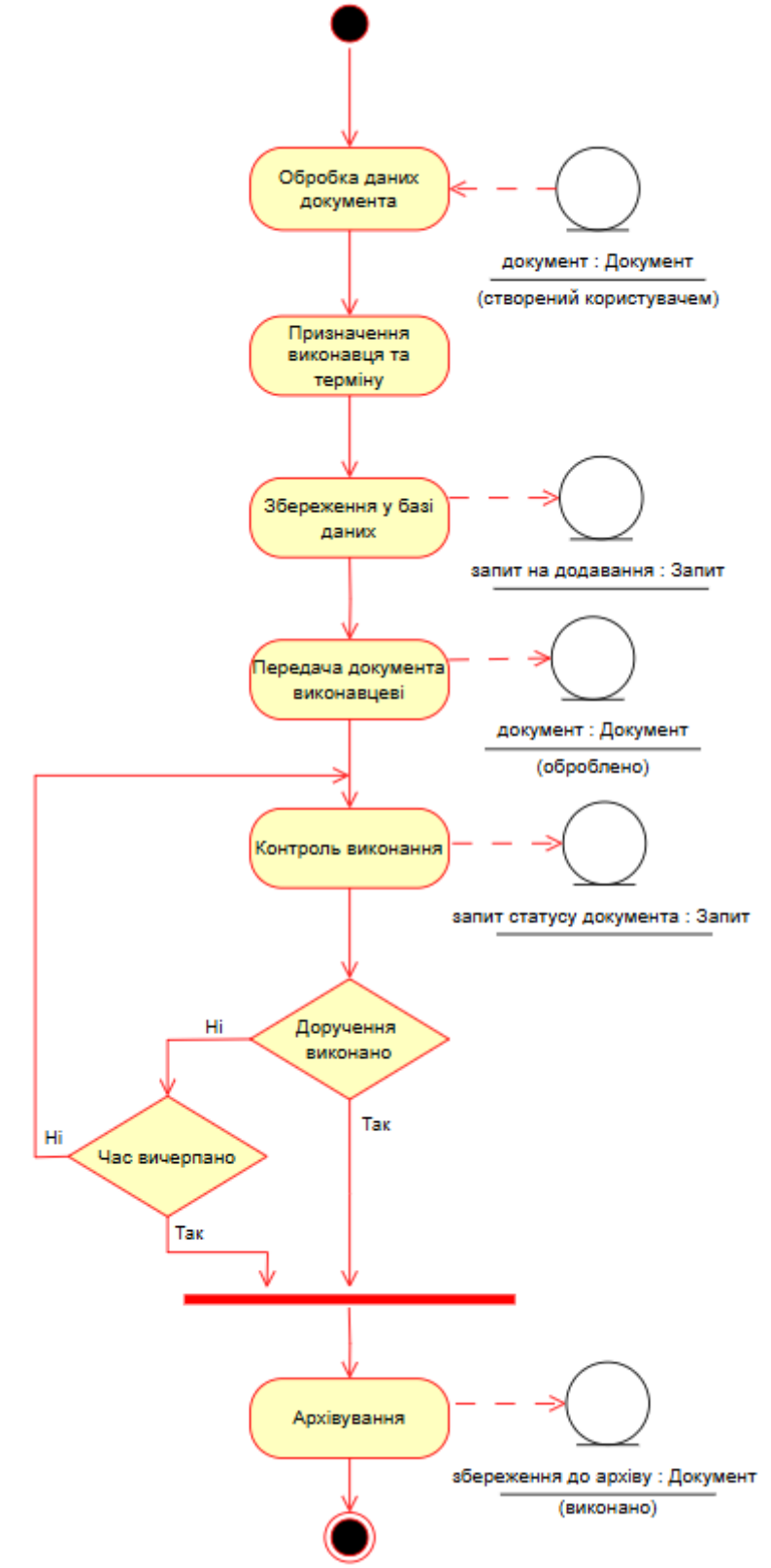

Рисунок 2.5 – Функціональна діаграма системи обліку документів

Розподілена система має підтримувати типові бізнес-сценарії документообігу в межах адміністрації:

- сценарій обробки вхідного листа – прийом кореспонденції, реєстрація, розподіл відповідальності, погодження відповідей, підпис та відправлення;
- сценарій створення внутрішнього документа – ініціація службової записки, направлення на погодження, формування доручення, контроль виконання;
- сценарій вихідної кореспонденції – підготовка проекту документа, погодження з кількома підрозділами, підписання керівництвом, реєстрація вихідного номеру, передача на відправлення [10].

Система має бути відкритою до інтеграції з іншими державними інформаційними системами, такими як реєстри юридичних осіб, електронний суд, системи кадрового обліку тощо. Тому у функціональну модель включені спеціалізовані інтерфейсні шлюзи API, які дозволяють обмінюватися структурованими XML/JSON повідомленнями із зовнішніми платформами.

Усі функціональні блоки працюють з урахуванням вимог безпеки інформації. Має підтримуватись авторизація, аудит, логування дій користувачів, резервне копіювання. Система проєктується з дотриманням законодавства України у сфері електронного документообігу, а також Закону «Про електронні документи та електронний документообіг», «Про захист інформації в інформаційно-телекомунікаційних системах» та нормативів ДСТУ.

2.4. Розробка схеми бази даних розподіленої програмної системи

Під час розробки системи слід спроєктувати таку базу даних, яка зможе забезпечити не лише збереження документів, а й ефективну їх обробку, пошук, контроль стану виконання та взаємодію з іншими компонентами системи. База

даних повинна відповідати вимогам масштабованості, стресостійкості та цілісності, адже система функціонує у розподіленому середовищі, де кожен структурний підрозділ працює з єдиним інформаційним простором, але може мати локальні вузли обробки. На рисунку 2.6 зображено схему взаємодії компонентів бази даних.

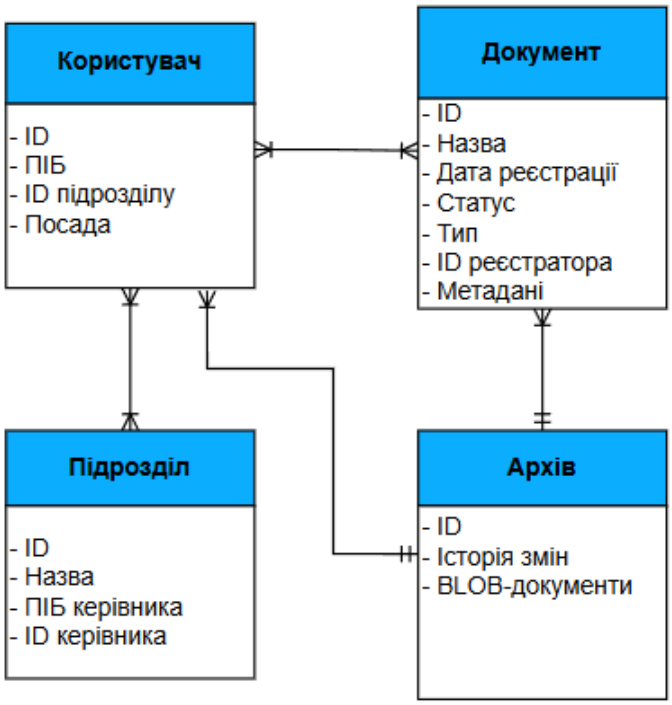

Рисунок 2.6 – Схема бази даних системи обліку документів

Структура бази даних побудована на реляційній моделі з поділом на логічні області, відповідальні за реєстрацію, зберігання, маршрутизацію та контроль документів. Ключовою сутністю є документ, який містить не лише основні реквізити, такі як дата, номер, тип, автор, а й метадані, що характеризують його рух, статуси виконання, пов’язані доручення та внутрішні позначки. Крім того, кожен документ асоціюється з об’єктами, які представляють підрозділи-виконавці, користувачів, що мають доступ до документа, та відповідні файли, які можуть зберігатися як у файловій системі, так і безпосередньо у базі даних у вигляді BLOB. Підтримка маршрутизації документів реалізована через окрему логіку зберігання етапів проходження.

Кожен маршрут описується як набір дій з вказаними ролями, що мають здійснювати обробку, а також часовими межами виконання. Це дозволяє реалізувати контроль виконавської дисципліни та фіксацію історії руху документа. Умови маршрутизації можуть бути динамічними, залежними від типу документа або ініціатора. Система також передбачає логіку зберігання користувачів і ролей із можливістю гнучкого налаштування прав доступу. Для цього створюється окрема модель авторизації, яка враховує як загальні політики доступу, так і специфічні обмеження на рівні документів або дій. Інтеграція із зовнішніми сервісами фіксується у відповідних журналах, що дозволяє вести аудит усіх операцій.

Особливістю схеми є підтримка розподіленого зберігання даних. Залежно від підрозділу або географічного розташування, деякі таблиці можуть бути розміщені на окремих вузлах із періодичною синхронізацією через механізми реплікації або кластеризації. Такий підхід дозволяє зменшити навантаження на центральний сервер і забезпечити роботу навіть у випадках тимчасового розриву з’єднання між вузлами. Важливим компонентом є блок для зберігання службової інформації, включно з журналами подій, логами помилок, діями користувачів та станами інтеграційних процесів. Це не лише сприяє стабільності системи, а й забезпечує можливості подальшого аналізу продуктивності, виявлення аномалій або оптимізації роботи.

2.5. Розробка блок-схем алгоритмів програмних модулів

Процес створення документа починається з ініціалізації нового запису, введення користувачем необхідних реквізитів та вибору типу документа. Після перевірки обов’язкових полів система присвоює унікальний номер та надсилає документ до бази даних для подальших операцій. На рисунку 2.7 зображено алгоритм створення та збереження документа.

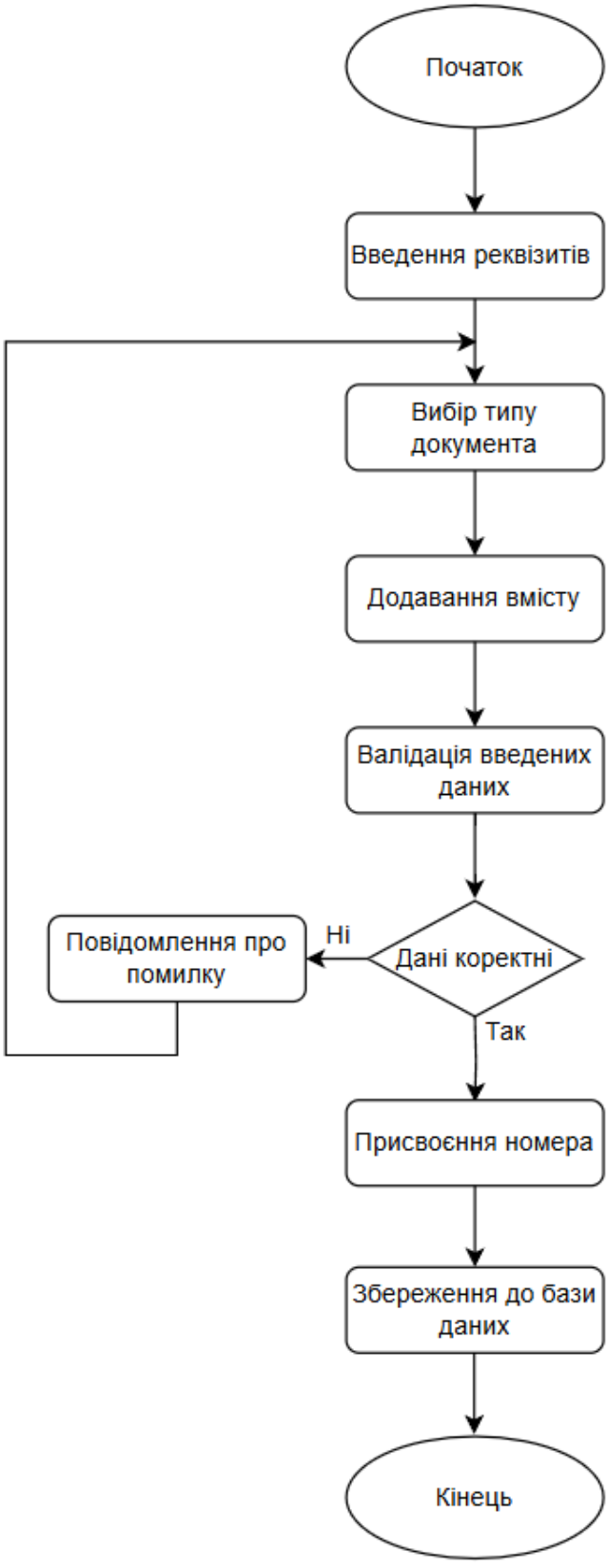

Рисунок 2.7 – Блок-схема алгоритму створення та збереження документа

На основі типу документа, обраного відправника або отримувача, система ініціює запуск попередньо заданого маршруту. Маршрут може мати у собі декілька отримувачів. У такому випадку надсилання документа відбуватиметься послідовно з повторенням алгоритму для кожного отримувача. На рисунку 2.8 зображено алгоритм маршрутизації (погодження) документу.

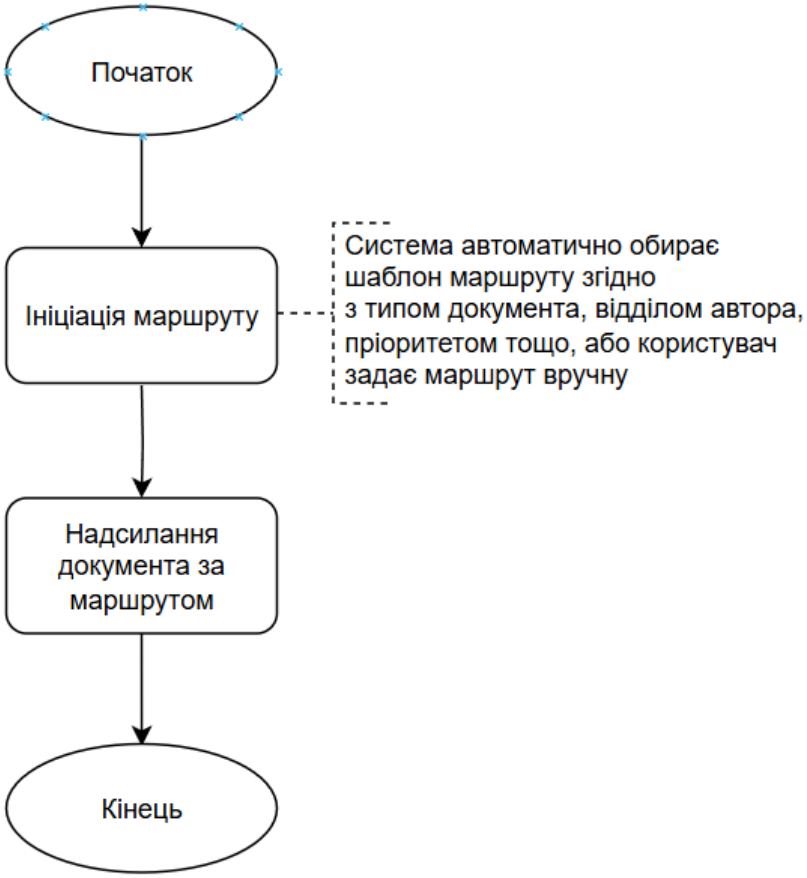

Рисунок 2.8 – Блок-схема алгоритму маршрутизації

Після завершення погодження документ переходить до стадії підписання. Відповідальний користувач ініціює середовище електронного пілпису, після чого відбувається процес накладання КЕП. На рисунку 2.9 зображено блок-схему алгоритму накладання електронного підпису (КЕП) на документ.

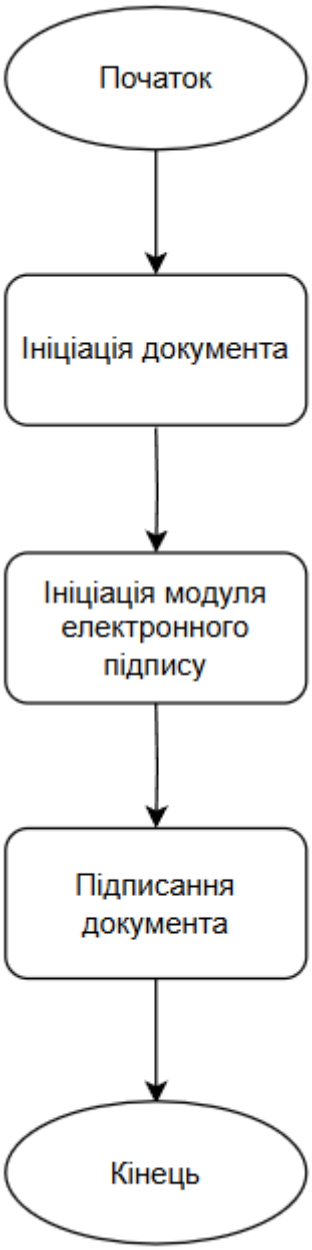

Рисунок 2.9 – Блок-схема алгоритму накладання електронного підпису

Підписаний документ надходить до системи контролю виконання. Система фіксує термін виконання доручення, виконавців та періодично перевіряє статус документів. Якщо доручення виконано, документ надсилається до архіву. У разі вичерпання часу на виконання, виконавцю надсилається нагадування, а документ надходить до архіву. На рисунку 2.10 зображено схему роботи модуля контролю виконання.

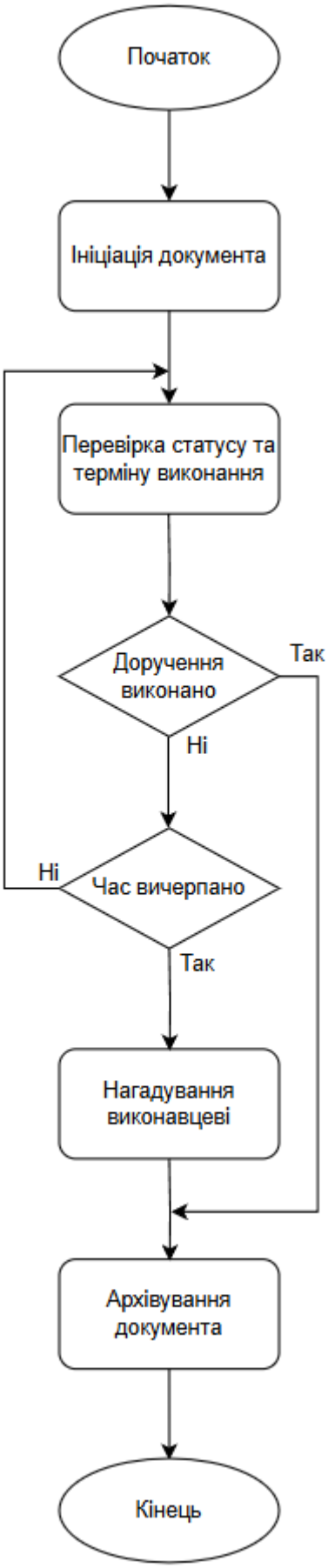

Рисунок 2.10 – Блок-схема алгоритму контролю виконання

2.6. Розробка моделі візуального інтерфейсу компонентів розподіленої програмної системи

Враховуючи специфіку розподіленої системи, інтерфейс користувача повинен бути адаптованим до роботи в різних середовищах. Модель візуального інтерфейсу розроблялася з дотриманням принципів юзабіліті, доступності, інформаційної логіки та адаптивності. В її основі закладена модульна структура, яка забезпечує можливість відображення лише тих компонентів, що є необхідними для користувача.

На рисунку 2.11 представлено головну сторінку користувацького інтерфейсу розподіленої системи обліку документів. Вона містить панель навігації, розміщену зліва, з піктограмами доступу до основних функціональних модулів. Центральна частина вікна відображає актуальні завдання, повідомлення та статуси обробки документів.

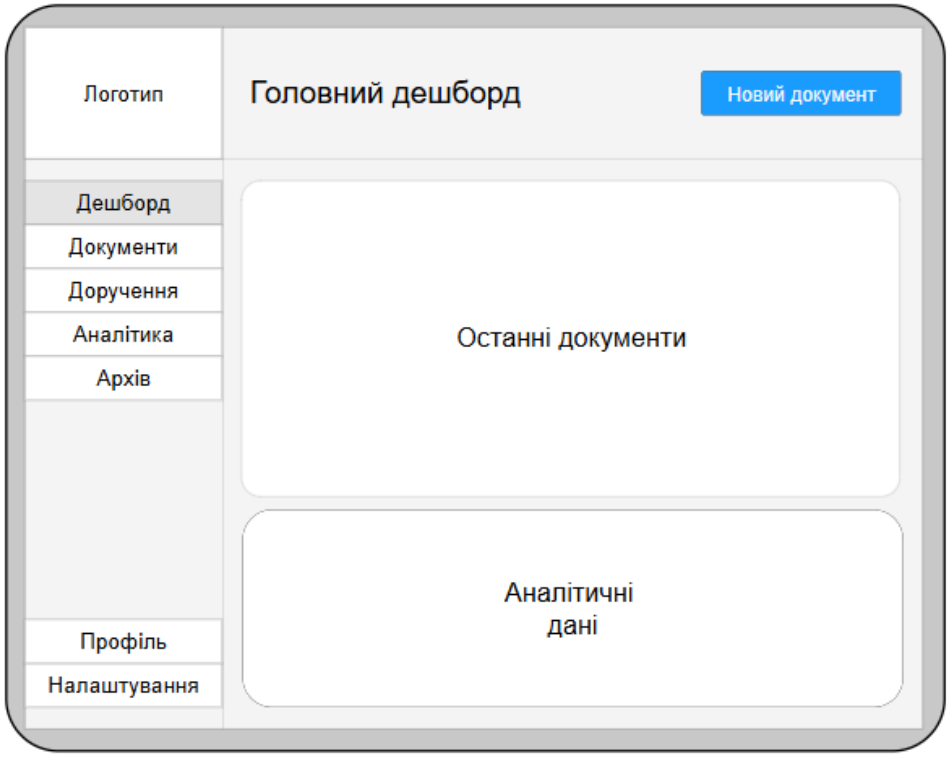

Рисунок 2.11 – Головна сторінка системи обліку документів

Далі у меню росташований пункт «Документи». У ньому знаходяться інструменти для перегляду та редагування наявних у базі даних активних документів. Також, наявний інструмент пошуку необхідних документів з гнучкими фільтрами пошуку та асинхронним виведенням знайдених документів у таблицю в центрі вікна. На рисунку 2.12 зображено пункт керування документами розподіленої системи обліку документів.

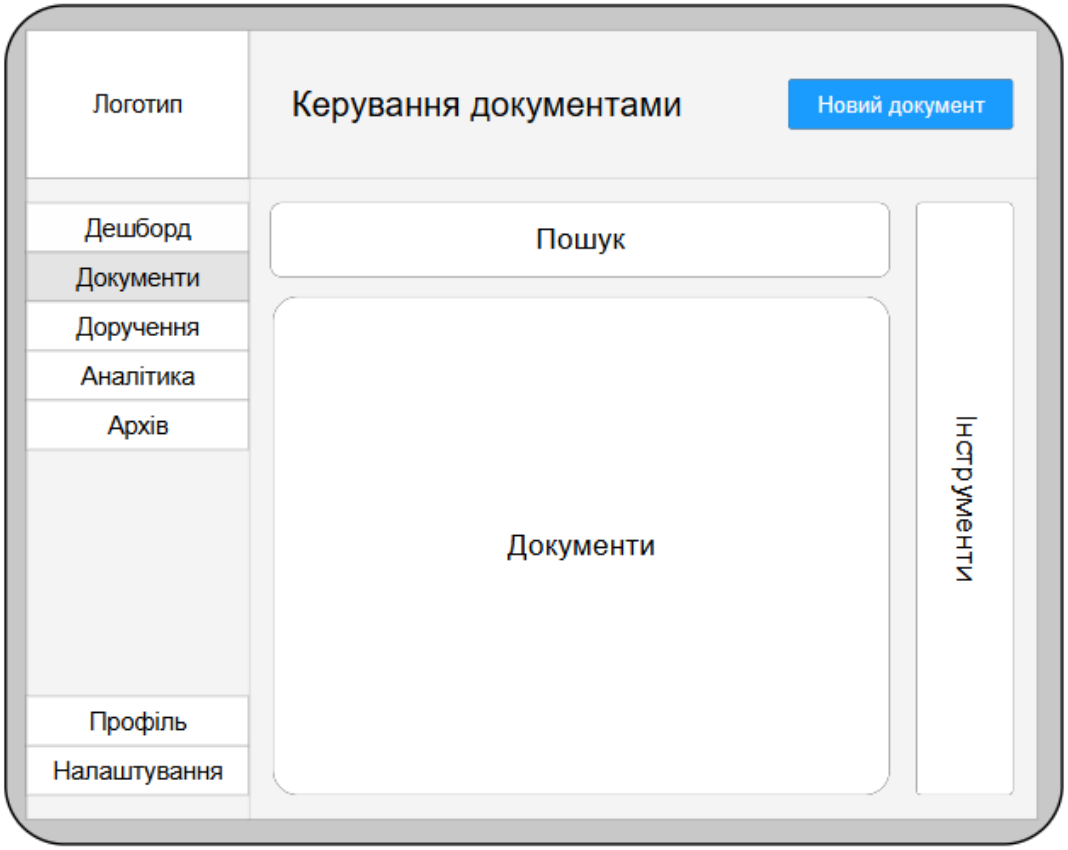

Рисунок 2.12 – Сторінка «Керування документами» системи обліку документів

Наступним пунктом меню є «Доручення». Тут росташовані інструменти перегляду та керування дорученнями, а також є розділення на активні та протерміновані доручення. Натиснувши на відповідну піктограму, користувач отримає панель з обраним типом доручень та зможе за необхідності редагувати будь-яке з них. На рисунку 2.13 зображено сторінку «Керування дорученнями» системи обліку документів.

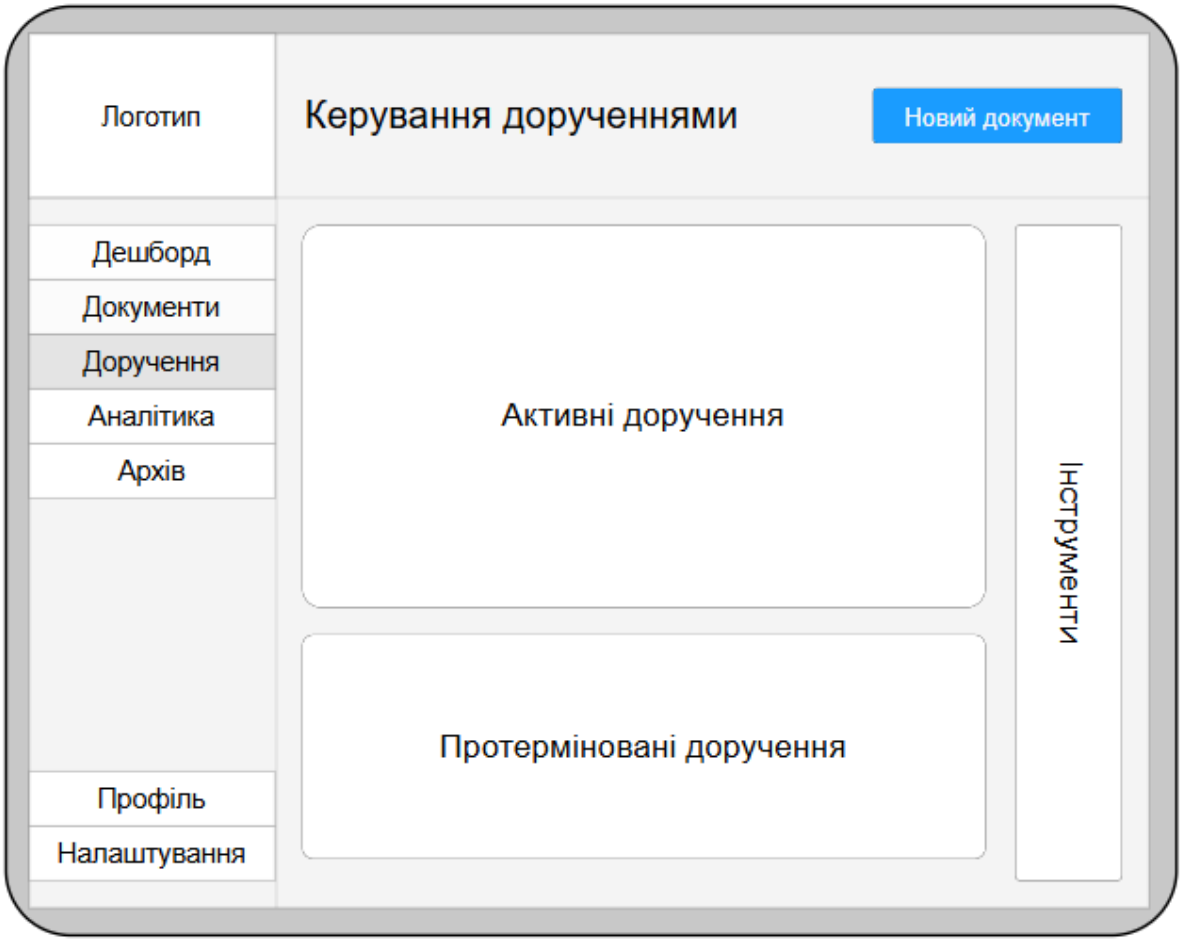

Рисунок 2.13 - Сторінка «Керування дорученнями» системи обліку документів

У пункті «Аналітика» знаходятся усі аналітичні дані системи: про надходження документів, про виконання доручень, про загальну навантаженість системи та можливість створення звітів. Доступність аналітичної інформації залежить від ролі користувача у системі та може бути редагована адміністратором у будь-який момент. Пункт навантаженості системи містить у собі:

- Поточну та середню затримки відповідей системи;
- Серверний аптайм;
- Графік навантаження;
- Графік стабільності з’єднання.

На рисунку 2.14 зображено сторінку «Аналітика» системи обліку документів.

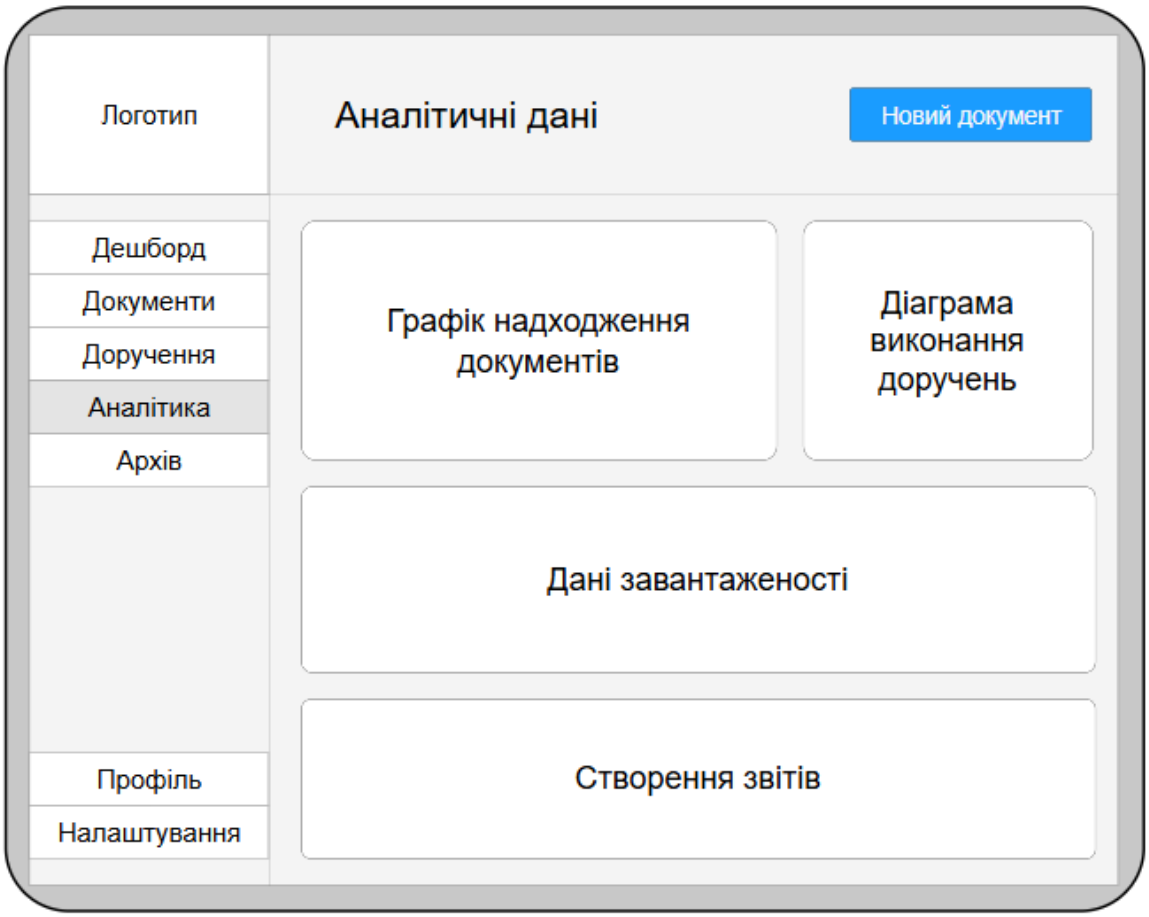

Рисунок 2.14 - Сторінка «Аналітика» системи обліку документів

Останнім пунктом меню є «Архів». Його вмістом можуть бути як вже завершені документи, так і створені файли звітів. При натисканні на відповідний пункт, користувач перейде до інтерактивного вікна взаємодії з необхідними йому даними. У пункті «Останні архівовані документи» користувач має змогу переглянути або редагувати завершені доручення, якщо його роль це дозволяє. У пункті «Останні створені звіти» користувач має змогу переглянути створені іншими користувачами звіти системи. За наявності відповідних прав, користувач може переглядати, роздруковувати, видаляти або надсилати необхідні звіти. На рисунку 2.15 зображено сторінку «Архів» системи обліку документів.

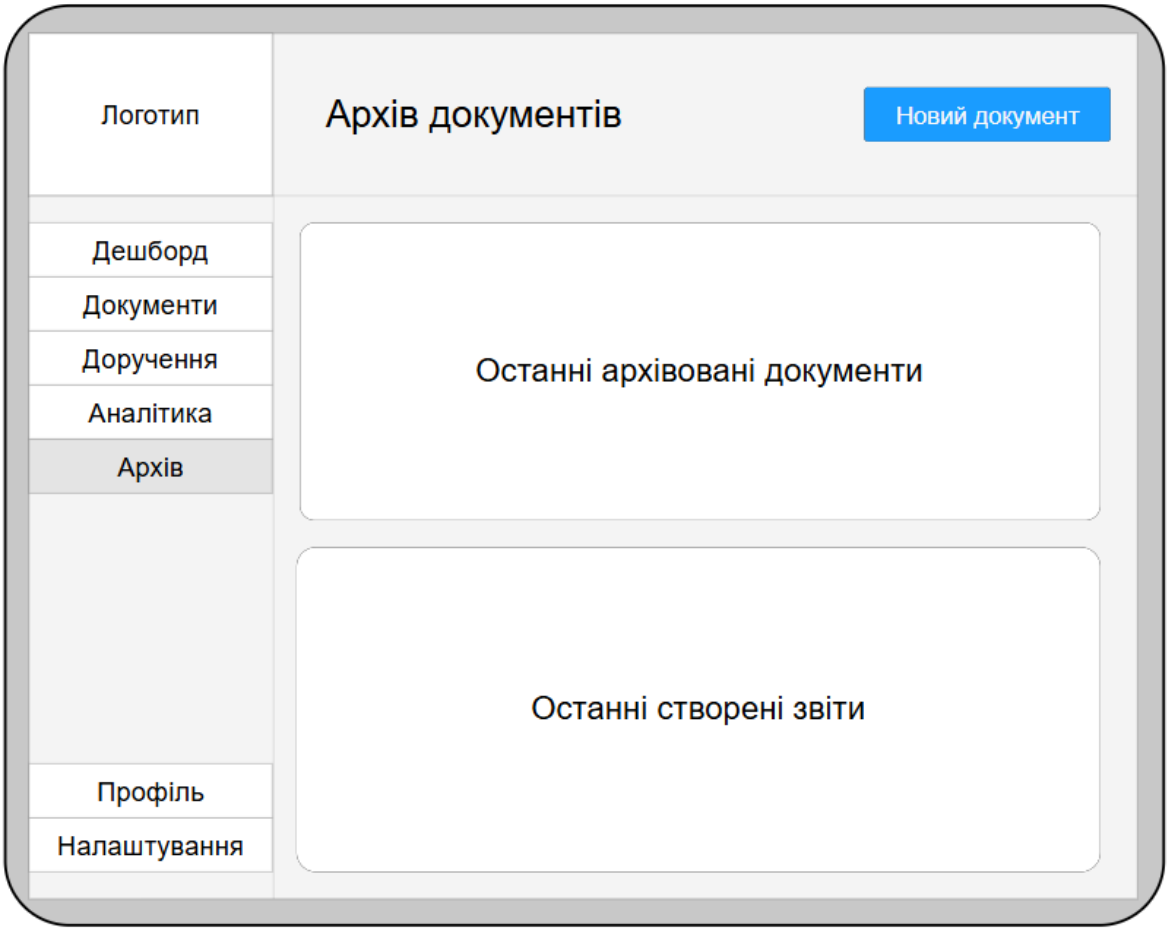

Рисунок 2.15 – Сторінка «Архів» системи обліку документів

Інтерфейс передбачає динамічне завантаження компонентів, що мінімізує час відгуку при роботі з системою та дозволяє підтримувати масштабованість. Додатково реалізовано адаптацію до темної та світлої теми оформлення, підтримку локалізації системи та мобільних пристроїв.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ РОЗПОДІЛЕНОЇ СИСТЕМИ ОБЛІКУ ДОКУМЕНТІВ АДМІНІСТРАЦІЇ МІСЬКОГО РАЙОНУ

3.1. Розробка бази даних сервера програмної системи

У ході розробки програмної системи особлива увага була приділена проєктуванню логічної структури бази даних та реалізації її фізичної моделі. Правильно побудована база даних є фундаментом стабільної роботи інформаційної системи, оскільки вона визначає, як саме будуть організовані взаємозв’язки між документами, користувачами, процесами маршрутизації, контролем виконання, архівуванням та іншими модулями системи. В якості серверного середовища було обрано фреймворк Django, який забезпечує зручний механізм для створення моделей бази даних з використанням засобів ORM (Object-Relational Mapping) [11]. Це дозволило реалізувати всю структуру бази даних у вигляді Python-класів, кожен з яких відповідає окремій таблиці. Django автоматично генерує SQL-запити для створення, зміни чи видалення таблиць, що значно прискорює процес розробки та уніфікує код сервера.

Було реалізовано кілька логічних блоків, кожен з яких виконує власну роль у функціонуванні системи документообігу. У системі реалізовано окрему модель для зберігання інформації про користувачів, яка містить дані про ім’я, прізвище, електронну пошту, пароль, роль у системі (співробітник, керівник, адміністратор), а також рівень доступу до функціональних модулів. Також передбачено поле активності, яке дозволяє вмикати або блокувати облікові записи. Центральною сутністю системи є модель документів. Вона містить такі атрибути, як назва документа, дата створення, автор, тип документа (вхідний, вихідний, внутрішній), статус (чернетка, на розгляді, затверджено, відхилено), прикріплений файл, а також опис або коментарі. Модель зв’язана з

користувачами через зовнішній ключ. Модель маршрутизації містить інформацію про послідовність проходження документа між різними користувачами. Реалізовано збереження списку учасників процесу погодження та поточний статус маршруту. Додатково передбачено поля для термінів виконання та пріоритету. Для забезпечення своєчасного виконання завдань до документів додано модель, що дозволяє слідкувати за термінами виконання, фіксувати дату прийняття, дату завершення, а також причини затримки. Передбачено логіку ручного або автоматичного надсилання сповіщень при наближенні дедлайнів. Завершені документи переносяться в архівну таблицю, що зберігає всю історію змін, копію основної інформації про документ, а також дату архівації. Передбачено додаткове шифрування документів для підвищення безпеки при збереженні. Для забезпечення прозорості всі дії користувачів логуються. У таблиці зберігається інформація про тип дії, користувача, що її виконав, а також дату і час події. В окремій таблиці накопичуються агреговані дані для побудови звітів (кількість документів за типами, навантаження на підрозділи, середній час погодження, рівень дотримання строків). Ці дані використовуються у візуальних дашбордах на клієнтській частині системи.

Моделі реалізовано у файлі models.py всередині Django-додатку. Загальна структура створених моделей має такий вигляд:

```
class Document(models.Model):
    title = models.CharField(max_length=255)
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    type = models.CharField(choices = DOC_TYPES, max_length = 50)
    status = models.CharField(choices=STATUS_CHOICES, default='draft')
    file = models.FileField(upload_to='documents/')
    description = models.TextField(blank=True)
```

Після створення моделей застосовуються міграції, які автоматично створюють відповідні таблиці в базі даних [12].

Для забезпечення обміну даними між клієнтською та серверною частинами системи реалізовано набір RESTful API за допомогою Django REST Framework. Це дозволяє клієнтському застосунку виконувати HTTP-запити до серверної частини та працювати з JSON-даними. Було створено такі кінцеві точки:

/api/documents;
/api/routes;
/api/users;
/api/logs;
/api/reports.

Оскільки документообіг працює з конфіденційною інформацією, у структурі бази даних реалізовано розмежування прав доступу на основі ролей, зберігання даних користувачів у зашифрованому вигляді й логування змін усіх дій користувачів.

3.2. Розробка програмних модулів розподіленої системи обліку документів

Для забезпечення функціональності системи обліку документів адміністрації міського району було реалізовано ряд окремих програмних модулів, кожен з яких виконує окрему роль у загальній архітектурі. Завдяки модульному підходу вдалося забезпечити розподілення логіки, масштабованість, тестованість та зручність у подальшому супроводі системи.

Розробка проводилася з використанням сучасного технологічного стеку:

- Back-end – мова програмування Python, фреймворк Django та бібліотека Django REST Framework для побудови REST API.
- Front-end – бібліотека React для побудови односторінкового додатку.

- База даних – SQLite3.

- Передача даних – JSON через HTTPS-з’єднання.

Процес автентифікації реалізований через сучасні технології, які забезпечують надійний захист персональних даних. Паролі користувачів зберігаються у базі даних лише в хешованому вигляді. Це дозволяє забезпечити високу стійкість до атак типу «brute-force» або отримання доступу до відкритих паролів у разі витоку даних. Для реалізації сесійного керування в системі впроваджено механізм токенізації з використанням стандарту JSON Web Token. Такий підхід дозволяє передавати інформацію про авторизованого користувача між клієнтом і сервером у зашифрованому вигляді, забезпечуючи як безпечність з'єднання, так і швидкість взаємодії.

```
class LoginView(APIView):
    def post(self, request):
        username = request.data.get("username")
        password = request.data.get("password")
        user = authenticate(username=username, password=password)
        if user:
            refresh = RefreshToken.for_user(user)
            return Response({
                "refresh": str(refresh),
                "access": str(refresh.access_token),
                "username": user.username,
                "role": user.profile.role})
        return Response({"error": "Invalid credentials"}, status=400)
```

Модуль створення та обробки документів забезпечує реалізацію повного життєвого циклу документа в межах організаційного процесу. Користувач має змогу створювати нові документи, заповнюючи необхідні реквізити (тип документа, його заголовок, опис та супровідна інформація). Передбачена можливість додавання файлів у вигляді вкладень, що дозволяє зберігати

пов’язаний контент у єдиному інформаційному просторі. Важливою складовою модуля є можливість визначення маршруту погодження документа, в якому вказуються відповідальні особи та порядок проходження.

Під час створення документа ініціатор має можливість вказати послідовність осіб або груп, які повинні ознайомитись з документом або схвалити його. Модуль маршрутизації, у свою чергу, керується цими параметрами для автоматичного визначення наступного етапу обробки. Кожен учасник процесу отримує повідомлення про нове завдання, а його дії фіксуються в системі з точними часовими мітками. Це дозволяє відстежувати реальний стан виконання кожного документа та гарантує прозорість процедур. Завершення маршруту означає або повне погодження документа, або його відхилення. У разі відхилення ініціатор отримує повідомлення з коментарем, що дозволяє виправити документ або надати додаткові роз’яснення. Після остаточного погодження система передає документ до архіву, надсилає на підпис чи готує його до виконання.

```
class ApprovalRoute(models.Model):
    document=models.ForeignKey(Document,on_delete=models.CASCADE)
    approver = models.ForeignKey(User, on_delete=models.CASCADE)
    order = models.IntegerField()
    approved = models.BooleanField(default=False)
    approved_at = models.DateTimeField(null=True, blank=True)
```

За зберігання завершених документів та ведення повної хронології змін від повідає модуль архівації. Після того, як документ пройшов усі етапи погодження, розгляду, виконання та підписання, він автоматично або за ініціативою користувача переноситься до архіву. У цьому стані він недоступний для подальших змін, але може бути переглянутий користувачами, які мають відповідний рівень доступу. Архівні копії документів зберігаються в зашифрованому вигляді, що гарантує їх захист від несанкціонованого доступу та забезпечує дотримання вимог інформаційної безпеки. Модуль також

передбачає можливість створення резервних копій і їх експорт у форматах, придатних для довготривалого зберігання або передачі між системами. Паралельно з архівацією реалізовано детальний механізм логування дій користувачів. Кожна операція супроводжується автоматичним записом у журнал змін. У журналі фіксується тип дії, точний час її виконання, ідентифікатор користувача, який її здійснив, а також коментарі. Це дозволяє не лише відтворити повну історію роботи з документом, а й провести аудит у разі виникнення суперечливих ситуацій.

```
class DocumentCreateView(generics.CreateAPIView):
    queryset = Document.objects.all()
    serializer_class = DocumentSerializer
    permission_classes = [permissions.IsAuthenticated]

    def perform_create(self, serializer):
        serializer.save(
            created_by=self.request.user,
            status='draft' )
```

За звітність та моніторинг відповідає модуль візуалізації аналітики та генерації звітів. Він надає керівництву та адміністративним працівникам наочної інформації про стан документообігу в системі та ефективність виконання внутрішніх процесів. Для реалізації інтерфейсної частини модуля було використано React, а для побудови візуальних графіків і діаграм бібліотеку recharts, яка забезпечує зручні та гнучкі засоби інтерактивної візуалізації даних. На головному дешборді відображаються ключові показники у вигляді графіків та інфографіки. Окремий розділ присвячено аналізу навантаження на користувачів, що дозволяє оцінити рівномірність розподілу обов’язків і вчасно виявляти перевантаження. Окрім динамічної аналітики в режимі реального часу, модуль підтримує функціональність формування звітів у популярних форматах CSV та PDF. Це дозволяє зберігати звіти, друкувати їх

або передавати іншим посадовим особам для подальшого аналізу. Користувач може самостійно налаштовувати параметри звіту, обираючи необхідні періоди, типи документів чи відповідальних осіб.

```
class ChangeLog(models.Model):
    document=models.ForeignKey(Document,on_delete=models.CASCADE)
    action = models.CharField(max_length=100)
    performed_by=models.ForeignKey(User,  on_delete=models.SET_NULL,
null=True)
    timestamp = models.DateTimeField(auto_now_add=True)
    comment = models.TextField(null=True, blank=True)
```

Адміністративний модуль забезпечує повний контроль за правами користувачів, системними налаштуваннями та аудитом дій. Доступ до нього мають виключно користувачі-адміністратори, що гарантує безпеку та цілісність конфігурації всієї системи. Основна функціональність модуля реалізована на бекенді за допомогою Django, а клієнтська частина через React із захищеним маршрутом доступу та перевіркою прав користувача. Адміністратор має змогу переглядати повний перелік зареєстрованих у системі користувачів, включаючи їхні ролі, статус активності та інші метадані. Реалізовано інструменти керування правами доступу. Можна змінювати ролі, блокувати облікові записи або видаляти користувачів, які більше не мають повноважень працювати з системою. Створення нових облікових записів також виконується через адміністративний інтерфейс, де передбачено валідацію введених даних та автоматичне призначення ролі відповідно до організаційної структури. Також, адміністративний модуль дозволяє налаштовувати глобальні параметри роботи системи. Це включає терміни виконання задач за замовчуванням, політику архівації завершених документів, правила автоматичного видалення або шифрування, мову інтерфейсу, часові зони тощо. Таким чином забезпечується гнучкість системи під потреби конкретної

адміністративної одиниці. На рисунку 3.1 зображено панель адміністрування фреймворку Django [13].

\

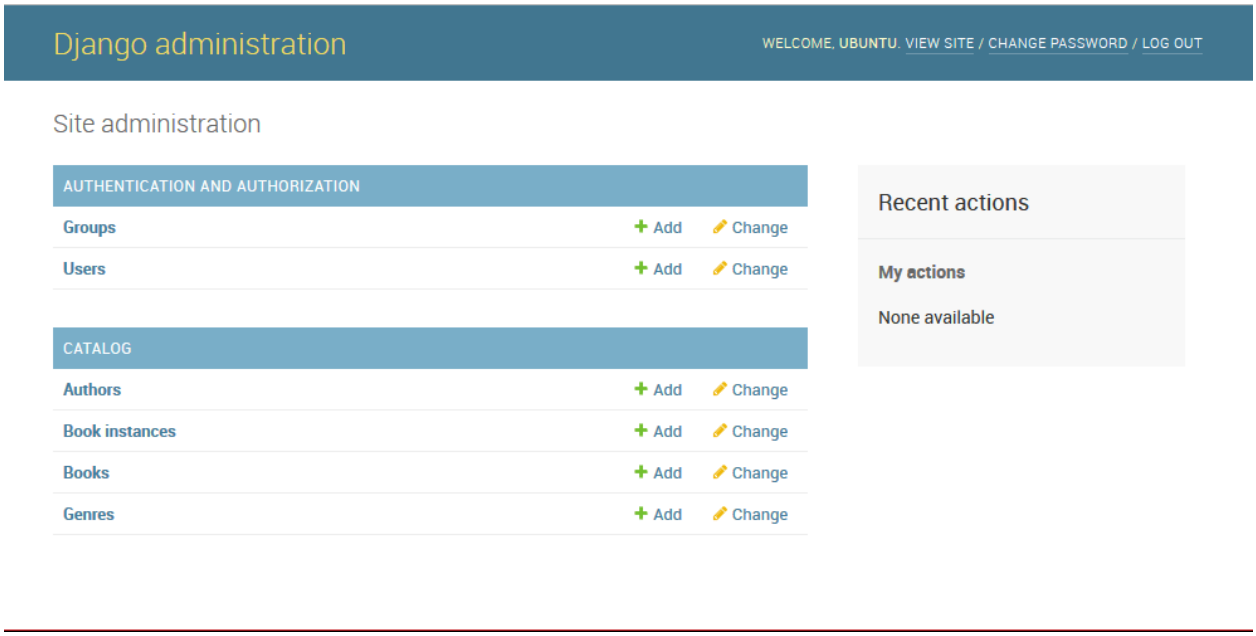

Рисунок 3.1 – Панель адміністрування Django REST Framework

Окрему увагу приділено реалізації функціоналу перегляду журналів подій. Усі ключові дії користувачів (входи в систему, зміна ролей, редагування документів, погодження або видалення) логуються та відображаються в зручному вигляді з фільтрами за датою, користувачем або дією. Це дозволяє адміністраторам вчасно реагувати на підозрілі активності та дотримуватись вимог інформаційної безпеки. Модуль спрощує технічне адміністрування системи та є основою її стабільного функціонування.

3.3. Розробка візуального інтерфейсу розподіленої системи обліку документів

У ході розробки було створено комплексний візуальний інтерфейс користувача для розподіленої системи обліку документів, який забезпечує зручну навігацію, відображення аналітики, роботу з документами, завданнями

та профільними налаштуваннями. Основною частиною інтерфейсу є головний дешборд, який служить точкою входу до системи після авторизації. Дешборд поєднує у собі таблицю з останніми документами, що дозволяє швидко переглядати найактуальніші записи та аналітичну панель, яка візуалізує статистичні дані за допомогою інтерактивних графіків (стовпчикових та кругових діаграм). Ця панель дає змогу оцінити динаміку надходження документів та їхній розподіл за статусами. На рисунку 3.2 зображено скріншот інтерфейсу дешборду.

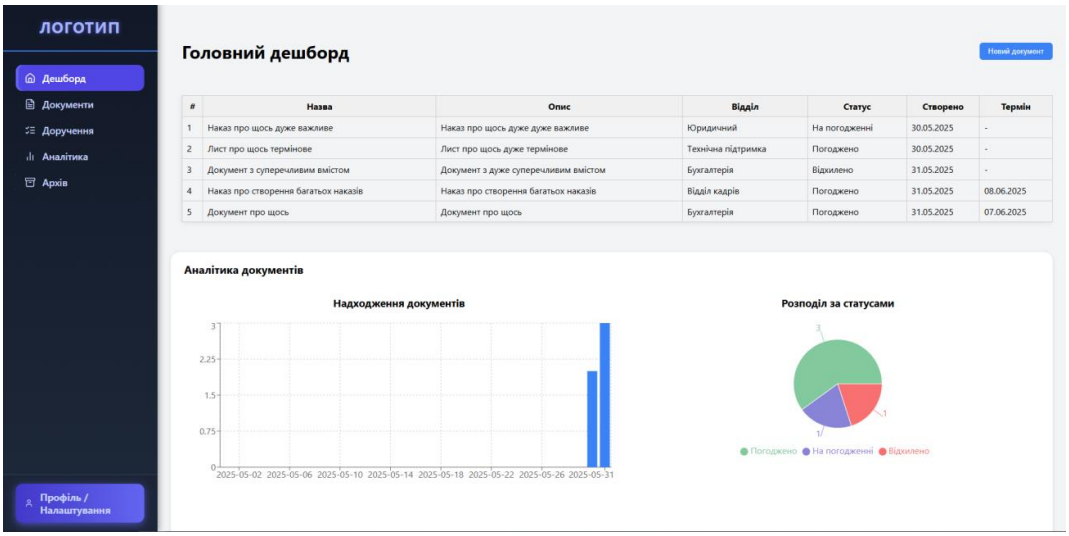

Рисунок 3.2 – Інтерфейс головної сторінки системи обліку документів

Для роботи з документами розроблено окрему сторінку, яка забезпечує повний функціонал керування створення, редагування та асинхронний пошук документів. Інтерфейс сторінки інтуїтивний і орієнтований на швидке виконання операцій користувачем. Пошук здійснюється асинхронно за допомогою бази даних та технологій паралельного обчислення. Документи відображаються у вигляді таблиці, поряд розташовані інструменти керування документами (додати, редагувати, роздрукувати, надіслати, архівувати та видалити). На рисунку 3.3 зображено скріншот інтерфейсу сторінки керування документами розподіленої системи обліку документів адміністрації міського району.

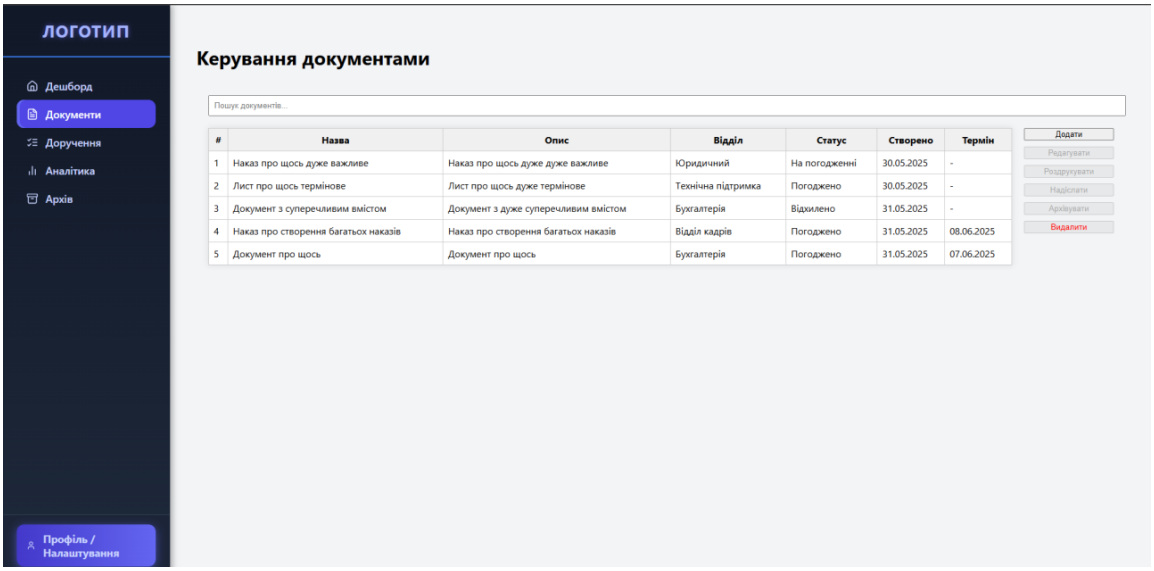

Рисунок 3.3 – Інтерфейс сторінки керування документами системи обліку документів

Створення нового документа здійснюється у модальному вікні браузера. На рисунку 3.4 зображено скріншот модального вікна створення нового документа.

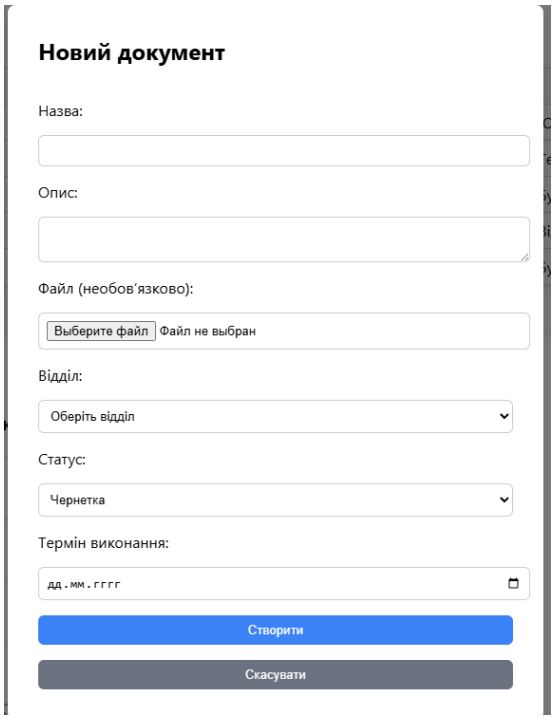

Рисунок 3.4 – Інтерфейс модального вікна створення документа

Сторінка аналітики дозволяє адміністраторам моніторити системні ресурси, збирати інформацію про кількість надходжень документів та створювати звіти з документообігу. На рисунку 3.5 зображено скріншот сторінки аналітики за звітності системи обліку документів.

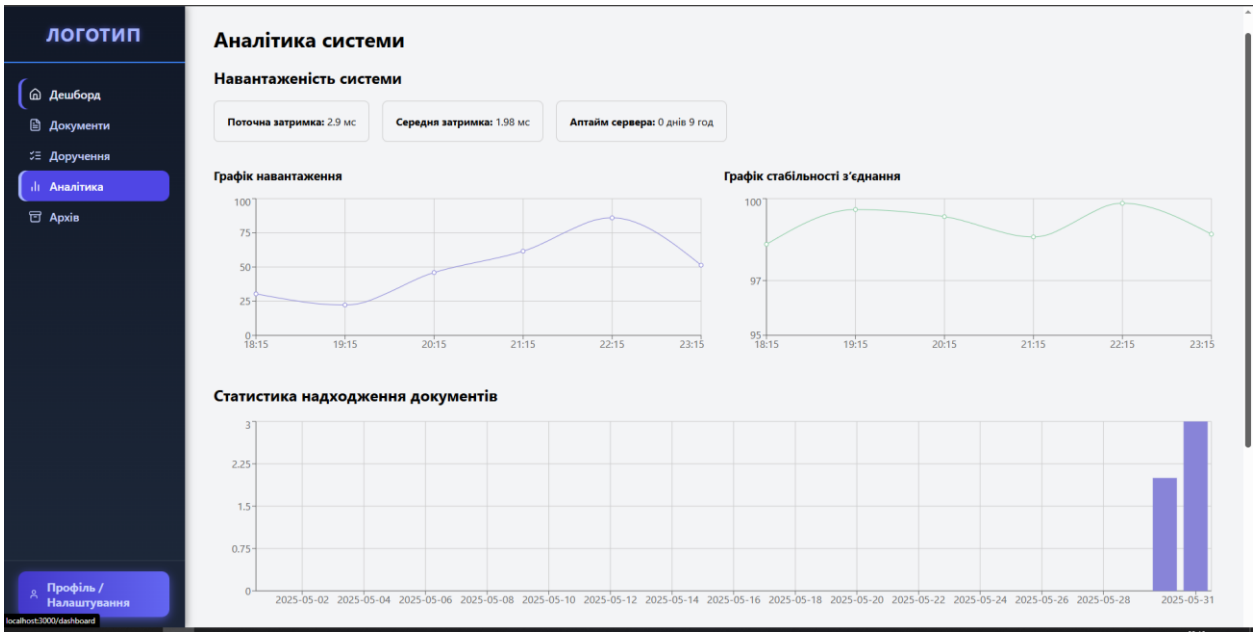

Рисунок 3.4 – Інтерфейс сторінки аналітики та звітності системи

Загальна структура інтерфейсу базується на адаптивній верстці з використанням CSS Flexbox і Grid, що забезпечує коректне відображення на різних пристроях та зручність користування. Навігаційне меню виконано у вигляді бокової панелі (sidebar), що дозволяє швидко переміщатися між розділами системи.

3.4. Оцінка ступеня завантаженості сервера обліку документів адміністрації міського району

З метою забезпечення стабільної роботи серверної частини системи обліку документів адміністрації міського району було реалізовано механізм моніторингу продуктивності, оцінку навантаження на центральний процесор

(CPU) та інтенсивність звернень до сервера. Цей моніторинг дозволяє своєчасно виявляти пікові навантаження, потенційні вузькі місця в архітектурі та оцінити масштабованість системи. Моніторинг реалізовано у вигляді фонової задачі, яка запускається при старті сервера. Вона щохвилини фіксує поточне завантаження CPU за допомогою бібліотеки psutil, зберігаючи ці дані в пам'яті програми. Зібрана інформація доступна через спеціальні API-ендпоінти у форматі JSON, що дозволяє візуалізувати її у вигляді графіків на клієнтській частині. Система підраховує кількість HTTP-запитів до сервера за хвилину, що дозволяє співвідносити динаміку навантаження з активністю користувачів. Для реалізації моніторингу використовувались такі інструменти:

- psutil – вимірювання поточного завантаження CPU;
- django middleware – підрахунок запитів і фіксації затримок;
- recharts – візуалізація статистики на фронтенді.

Для оцінки поведінки системи в умовах високого навантаження було використано інструмент Locust [14] – популярний фреймворк для написання навантажувальних тестів мовою Python. З його допомогою були створені сценарії, які імітують паралельну роботу десятків віртуальних користувачів, що здійснюють автентифікацію, запити до списку документів, а також перегляд окремих сторінок. На рисунку 3.5 зображено скріншот інтерфейсу інструмента Locust.

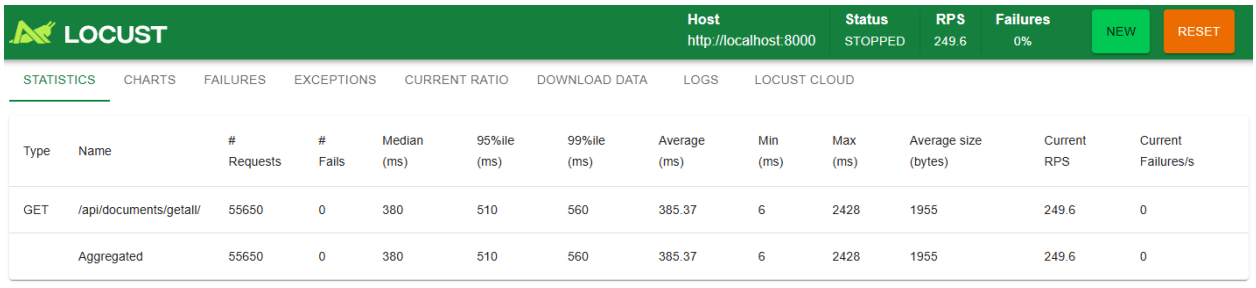

Рисунок 3.5 – Інтерфейс інструмента Locust python

У ході тестування було симульовано навантаження кількістю 100 користувачів, які одночасно надсилають запити на отримання усіх документів

з бази даних, що є найвищим з можливих навантажень на сервер. Результати моделювання знаходяться у пункті меню «Аналітика». На рисунку 3.6 зображено стан системи без надмірного навантаження.

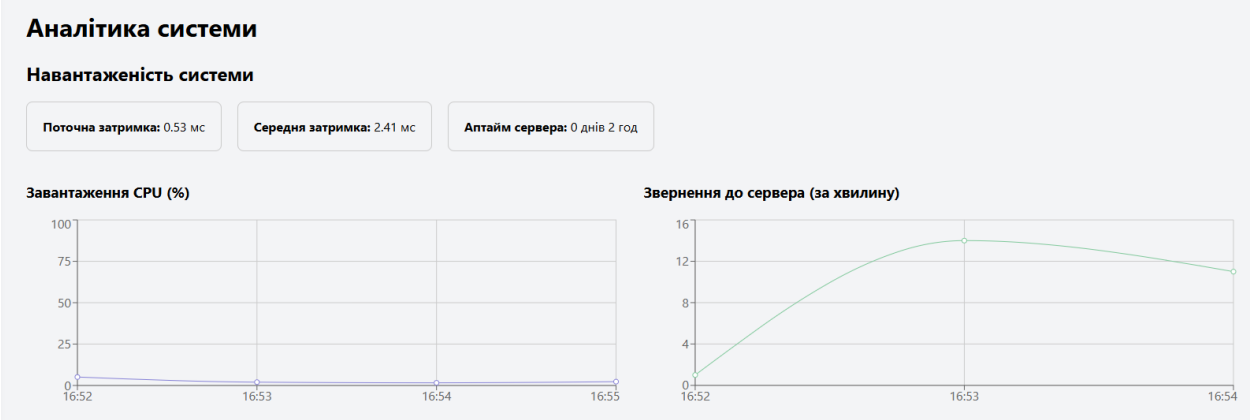

Рисунок 3.6 – Навантаженість системи у стані помірного навантаження

Тут відображено графіки аналітичних даних сервера, а саме:

- поточна на середня затримка відповіді сервера;
- завантаження центрального процесора сервера;
- кількість запитів до сервера за хвилину;
- серверний аптайм.

Щоб навантажити сервер, у locust необхідно вказати рівень навантаження та кількість користувачів (рис. 3.7)

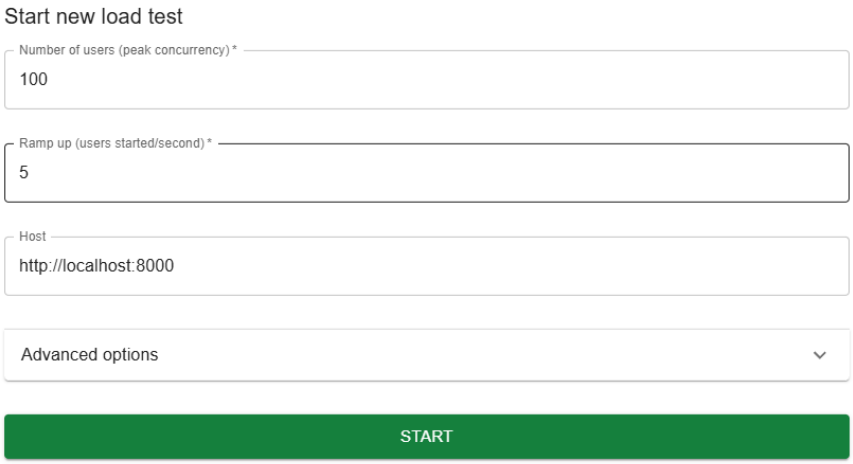

Рисунок 3.6 – Вікно налаштувань стрес-тесту Locust

Після налаштування та запуску стрес-тесту необхідно зачекати деякий час для того, щоб сервер опрацював вказану кількість запитів та зберіг статистику, яку можна буде відобразити на клієнті. По завершені тесту треба перейти до пункту «Аналітика» та звернути увагу на графіки та затримку відповіді. На рисунку 3.7 зображено результати стрес-тесту сервера з емуляцією 100 користувачів (приблизно 250 запитів за секунду).

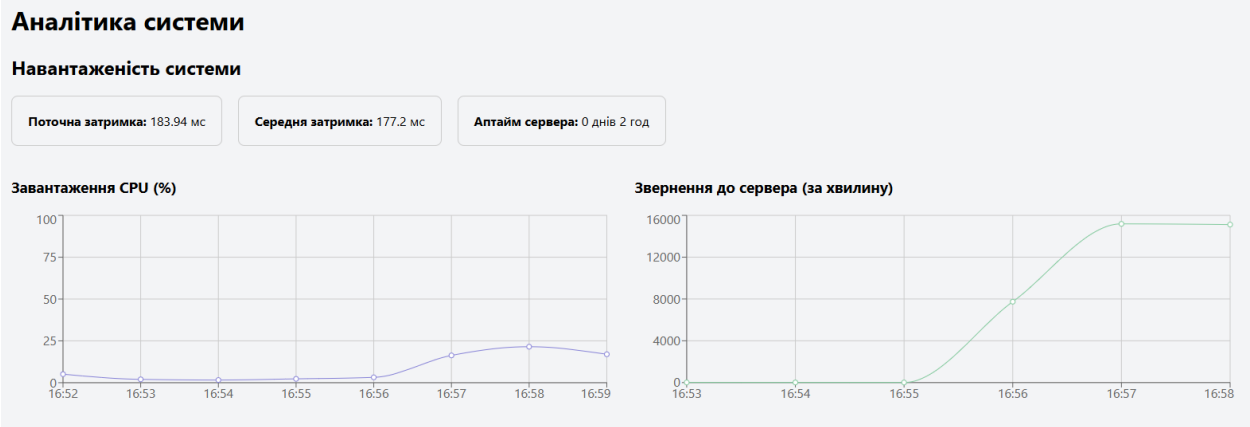

Рисунок 3.7 – Навантаженість системи у стані надмірного навантаження

Судячи з графіків, пікове навантаження на центральний процесор становить 24%, що приблизно дорівнює 950 MHz. Затримка відповіді приблизно дорівнює 177 мілісекунд, що є майже непомітною затримкою та не створює незручностей під час роботи [15].

3.5. Розробка програмних засобів адміністрування доступу до системи обліку документів

Для реалізації безпечного доступу до системи було використано вбудовані механізми фреймворку Django. Django – це високорівневий фреймворк для розробки веб-додатків на Python, який спочатку орієнтується

на безпеку, масштабованість та ефективну розробку. Він надає повноцінну систему аутентифікації та авторизації. Основні можливості включають:

- реєстрація та вхід користувачів через вбудовану модель User, яка підтримує збереження логіна, пароля, email, імені тощо;
- збереження паролів у зашифрованому вигляді, що унеможливлює зворотне відновлення пароля навіть у разі витоку бази даних;
- механізм груп та прав дозволяє гнучко визначати ролі користувачів та надавати їм доступ лише до відповідних частин функціоналу;
- декоратори доступу дозволяють обмежити доступ до певних представлень (views) лише для авторизованих або уповноважених користувачів.

Django має вбудовану адміністративну панель, яка дозволяє адмініструвати користувачів, призначати їм права, а також переглядати й редагувати будь-які моделі, зареєстровані в системі. без потреби створювати окремий інтерфейс. На рисунку 3.8 зображено скріншот інтерфейсу панелі адміністрування фреймворку Django.

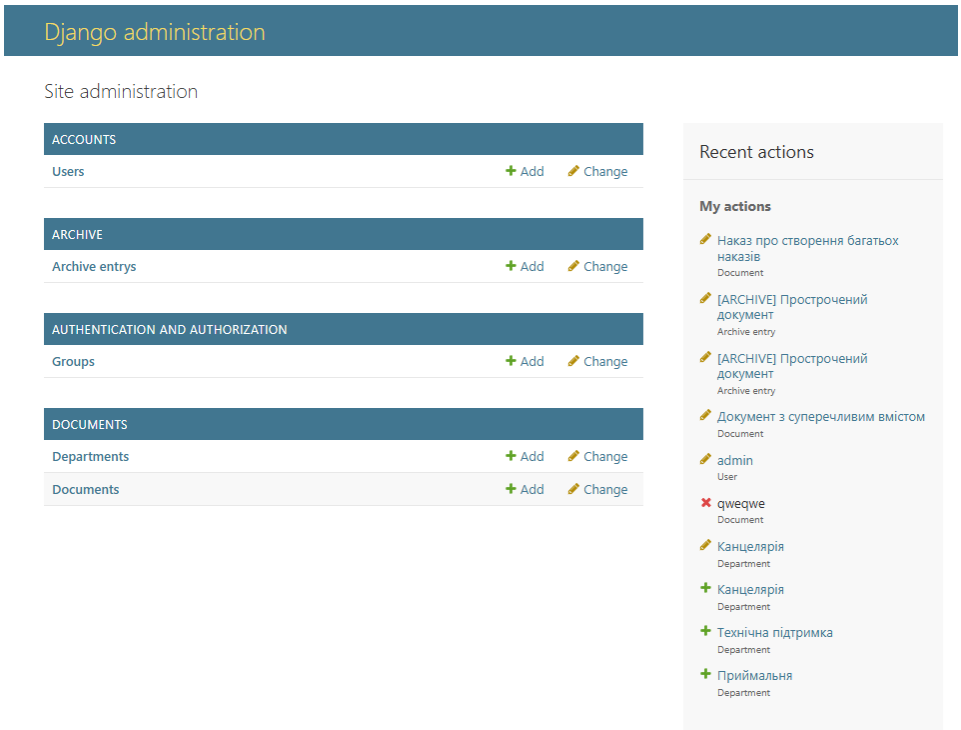

Рисунок 3.8 – Панель адміністрування Django

Ця панель автоматично генерується на основі моделей і є зручним інструментом керування даними. Усі користувацькі облікові дані зберігаються в базі даних через модель django.contrib.auth.models.User. Паролі не зберігаються у відкритому вигляді, бо при створенні або зміні пароля використовується метод set_password, який виконує хешування. Результат хешування включає:

- алгоритм pbkdf2_sha256;
- сіль (випадкова послідовність байтів);
- хешоване значення.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено прототип автоматизованої розподіленої системи обліку документів, що враховує специфіку організаційної структури адміністрації міського району та відповідає сучасним вимогам до масштабованості, безпеки, зручності та стійкості до навантажень.

У першому розділі проведено аналіз проблеми перевантаження серверів у розподілених інформаційних системах, вивчено особливості функціонування існуючих систем документообігу, зокрема СЕДО Megapolis.DocNet та АСКОД. Було визначено їх переваги й недоліки, що дозволило обґрунтувати доцільність розробки нової системи.

У другому розділі виконано проєктування математичних, функціональних та структурних моделей програмної системи. Побудовано DFD-діаграми, ER-модель, визначено принципи маршрутизації документів, сформовано вимоги до бази даних та розроблено концепцію модульної архітектури системи з можливістю масштабування.

У третьому розділі реалізовано практичну частину роботи. Розроблено веб-застосунок з використанням фреймворку Django для серверної частини та React для клієнтського інтерфейсу. Забезпечено обробку документів, автентифікацію користувачів, аналітику запитів та графічне відображення навантаження. Особливу увагу приділено питанням безпеки, зокрема захищеному зберіганню даних, шифруванню паролів та розмежуванню прав доступу.

Було також впроваджено систему моніторингу продуктивності сервера, запис та візуалізацію статистики використання CPU, аналіз часу відгуку, оцінку кількості запитів за хвилину. Для перевірки стресостійкості системи використано інструмент навантажувального тестування Locust, що дозволить виявити критичні точки і покращити ефективність системи.

Загалом, у дипломній роботі було розроблено комплексну систему, здатну ефективно обслуговувати документообіг адміністрації з урахуванням високих вимог до надійності, безпеки та масштабованості. Результати дослідження можуть бути використані як основа для впровадження повноцінної системи у реальному середовищі, а також для подальшого розвитку в напрямі розширення функціоналу та інтеграції з державними інформаційними ресурсами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Розподілені інформаційні системи : навч. посібник / А.Є. Коваленко. – Київ: НТУУ "КПІ", 2011. – 256 с.
2. Комп’ютерні інтелектуальні системи та мережі. Матеріали XVIII Всеукраїнської науково практичної WEB конференції аспірантів, студентів та молодих вчених (25-27 березня 2025 р.). – Кривий Ріг: Криворізький національний університет, 2025. – 432 с
3. Документно-інформаційні комунікації в умовах глобалізації: матеріали III Всеукраїн. наук.-практ. конф., м. Полтава, 22 листопада 2018 р. / редкол.: І.Г. Передерій, О.Є. Гомотюк та ін. Полтава: ПолтНТУ, 2108. 314 с.
4. Інформаційно-документаційне забезпечення розвитку соціокомунікаційного простору культури в Україні : монографія / Вікторія Добровольська. – Київ : НАКККіМ, 2020. – 352 с.
5. MEGAPOLIS.DocNet – електронна система документообігу [Електронний ресурс] – Режим доступу: https://inbase.com.ua/vsi-produkty/megapolis/ – Назва з екрана.
6. АСКОД – автоматизована система керування документами [Електронний ресурс] – Режим доступу: https://askod.online/index.ua.html – Назва з екрана.
7. Теорія систем масового обслуговування: навч. посібник / А. Л. Литвинов ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2018. – 141 с.
8. Чим відрізняється Frontend та Backend розробка [Електронний ресурс]? – Режим доступу: https://frontend.lviv.ua/chym-vidriznyayetsya-frontend-ta-backend-rozrobka-shho-obraty – Назва з екрана.
9. Про затвердження Порядку роботи з електронними документами у діловодстві та їх підготовки до передавання на архівне зберігання. – Введ. 11-11-2014. – К. : «Міністерство юстиції України»

10. Види документообігу [Електронний ресурс] – Режим доступу: https://inbase.com.ua/vydy-dokumentoobigu-detalnyj-gajd/ – Назва з екрана.
11. What is an ORM [Електронний ресурс] – Режим доступу: https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/ – Назва з екрана.
12. Django моделі [Електронний ресурс] – Режим доступу: https://tutorial.djangogirls.org/uk/django_models/ – Назва з екрана.
13. Django admin site [Електронний ресурс] – Режим доступу: https://docs.djangoproject.com/en/5.2/ref/contrib/admin/ – Назва з екрана.
14. Locust: An open source load testing tool [Електронний ресурс] – Режим доступу: https://locust.io/ – Назва з екрана.
15. Latency: What Is It and Why Does It Matter [Електронний ресурс] – Режим доступу: https://www.indusface.com/learning/what-is-latency/ – Назва з екрана

ДОДАТОК А

А.1. Посібнік користувача

А.1.1. Загальна інформація

Розроблена система обліку документів призначена для автоматизації процесів створення, реєстрації, погодження, маршрутизації, зберігання та архівування службових документів у межах міської адміністрації. Система реалізована як веб-додаток із сучасним інтерфейсом та доступом через браузер.

А.1.2. Вхід до системи

1. Перейдіть за адресою веб-інтерфейсу системи;
2. У вікні авторизації введіть свій логін і пароль;
3. Натисніть "Увійти";
4. Після входу буде доступний персоналізований кабінет користувача залежно від вашої ролі в системі.

А.1.3. Головна панель (Дешборд)

На головному екрані відображаються:

- список останніх створених документів;
- поточна аналітика (навантаження сервера, кількість запитів, активність користувачів);
- кнопка для створення нового документа;
- доступ до звітів та модулів статистики.

А.1.4. Створення нового документа

1. У розділі “Новий документ” натисніть кнопку "Створити";
2. Заповніть обов’язкові поля: назва, тип, виконавець, дата;
3. За потреби додайте вкладення (файл);
4. Натисніть “Зберегти”. Документ буде зареєстровано та відправлено по маршруту;

А.1.5. Маршрутизація документа

Після створення, документ автоматично направляється за вказаним маршрутом (відповідно до посади, відділу чи задачі). Кожен користувач бачить документи, які чекають на його погодження чи виконання.

А.1.6. Архів документів

Документи, що завершили свій життєвий цикл, автоматично переносяться до архіву. У розділі “Архів” доступні функції:

- пошук за реквізитами;
- фільтрація за роками, типами, статусами;
- експорт документа в PDF;
- перегляд історії змін.

А.1.7. Аналітика

У розділі “Аналітика” доступна інформація про:

- затримки у виконанні документів;
- статистика навантаження на сервер (CPU, кількість запитів);
- динаміка надходження документів;

- розподіл статусів (погоджено, очікує, відхилено).

А.2. Програмний код

А.2.1. Головна сторінка

```
return (
    <Router>
     <div className="app-container">
      <aside className="sidebar">
       <div>
        <div className="sidebar-header">Логотип</div>
        <nav className="sidebar-nav">
         <ul className="nav-list">
          <li>
           <NavLink
             to="/dashboard"
             className={({ isActive }) => isActive ? 'nav-item active' : 'nav-item'}
           >
            <Home size={20} /> Дешборд
           </NavLink>
          </li>
          <li>
           <NavLink
             to="/documents"
             className={({ isActive }) => isActive ? 'nav-item active' : 'nav-item'}
           >
            <FileText size={20} /> Документи
```

```
            </NavLink>
          </li>
          <li>
            <NavLink
              to="/tasks"
              className={({ isActive }) => isActive ? 'nav-item active' : 'nav-item'}
            >
              <ListChecks size={20} /> Доручення
            </NavLink>
          </li>
          <li>
            <NavLink
              to="/analytics"
              className={({ isActive }) => isActive ? 'nav-item active' : 'nav-item'}
            >
              <BarChart2 size={20} /> Аналітика
            </NavLink>
          </li>
          <li>
            <NavLink
              to="/archive"
              className={({ isActive }) => isActive ? 'nav-item active' : 'nav-item'}
            >
              <Archive size={20} /> Архів
            </NavLink>
          </li>
```

```
            </ul>
          </nav>
        </div>
        <div className="sidebar-footer">
          <div className="profile-button" onClick={() => setIsProfileOpen(true)}>
            <User size={20} /> Профіль / Налаштування
          </div>
        </div>
      </aside>

      <main className="main-content">
        <Routes>
          <Route path="/dashboard" element={<Dashboard />} />
          <Route path="/documents" element={<DocumentsPage />} />
          <Route path="/tasks" element={<TasksPage />} />
          <Route path="/analytics" element={<AnalyticsForm />} />
          <Route path="/archive" element={<ArchivePage />} />
          <Route path="*" element={<Dashboard />} />
        </Routes>
      </main>

      {isProfileOpen && <ProfileModal onClose={() => setIsProfileOpen(false)}
/>}
    </div>
  </Router>
 );
}
```

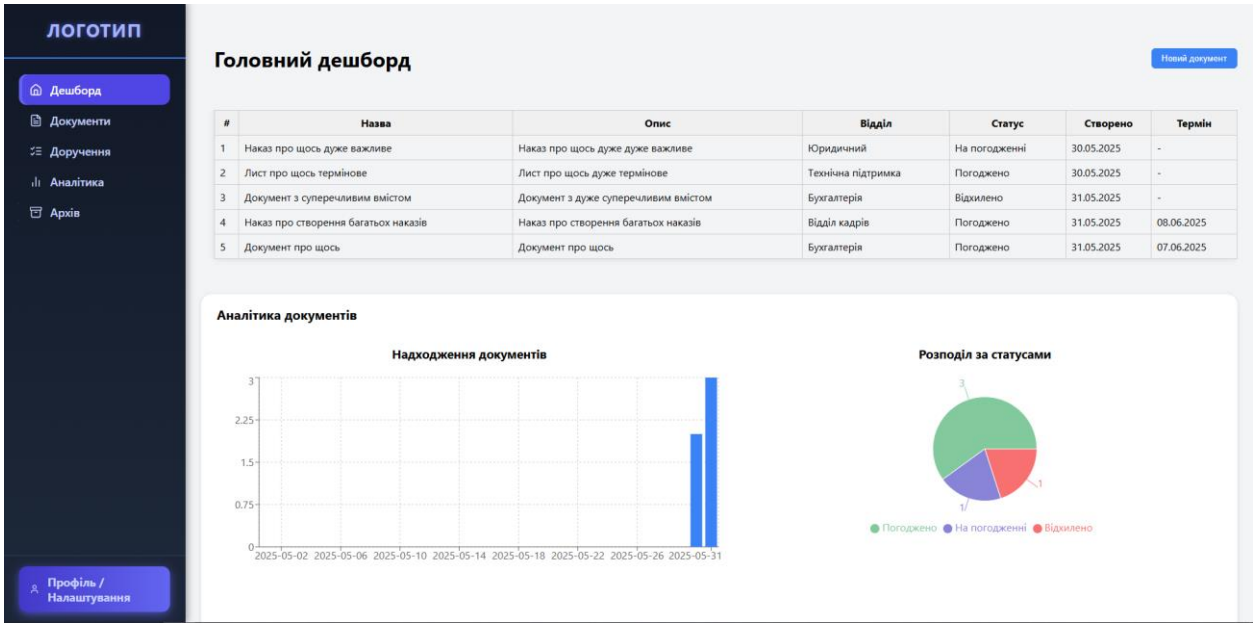

Рисунок А.2.1 – Інтерфейс головної сторінки системи обліку документів

А.2.2. Сторінка керування документами

```
return (
  <div style={{ padding: 20 }}>
    <input
      type="text"
      placeholder="Пошук документів..."
      value={searchQuery}
      onChange={e => setSearchQuery(e.target.value)}
      style={{ width: '100%', padding: '8px', marginBottom: 20 }}
    />

    <div style={{ display: 'flex' }}>
      <div style={{ flex: 1, marginRight: 20 }}>
        {loading ? (
          <p>Завантаження...</p>
        ) : (
```

```
                <table style={{ width: '100%', borderCollapse: 'collapse', boxShadow: '0 0 10px rgba(0,0,0,0.1)' }}>
                    <thead style={{ backgroundColor: '#f0f0f0' }}>
                    <tr>
                        <th style={thStyle}>#</th>
                        <th style={thStyle}>Назва</th>
                        <th style={thStyle}>Опис</th>
                        <th style={thStyle}>Відділ</th>
                        <th style={thStyle}>Статус</th>
                        <th style={thStyle}>Створено</th>
                        <th style={thStyle}>Термін</th>
                    </tr>
                    </thead>
                    <tbody>
                    {filteredDocs.length === 0 ? (
                        <tr>
                            <td colSpan="7" style={{ padding: 10, textAlign: 'center' }}>
                                Документи не знайдено
                            </td>
                        </tr>
                    ) : (
                        filteredDocs.map((doc, idx) => (
                            <tr
                                key={doc.id}
                                onClick={() => setSelectedDocId(doc.id)}
                                style={{
                                    backgroundColor: doc.id === selectedDocId ? '#d0eaff' : 'white',
                                    cursor: 'pointer',
```

```
                    }}
                >
                    <td style={tdStyle}>{idx + 1}</td>
                    <td style={tdStyle}>{doc.title}</td>
                    <td style={tdStyle}>{doc.description}</td>
                    <td style={tdStyle}>{doc.department?.name || '-'}</td>
                    <td style={tdStyle}>{doc.status_display || doc.status}</td>
                    <td style={tdStyle}>{new Date(doc.created_at).toLocaleDateString()}</td>
                    <td style={tdStyle}>
                      {doc.deadline ? new Date(doc.deadline).toLocaleDateString() : '-'}
                    </td>
                  </tr>
                ))
              )}
              </tbody>
            </table>
          )}
        </div>

        <div style={{ width: 150, display: 'flex', flexDirection: 'column', gap: 10 }}>
          <button onClick={handleAdd}>Додати</button>
          <button onClick={() => alert(`Редагування документа id=${selectedDocId}`)} disabled={!selectedDocId}>
            Редагувати
          </button>
```

```
            <button onClick={() => alert(`Друк документа id=${selectedDocId}`)} disabled={!selectedDocId}>
              Роздрукувати
            </button>
            <button onClick={() => alert(`Надіслати документ id=${selectedDocId}`)} disabled={!selectedDocId}>
              Надіслати
            </button>
            <button onClick={() => alert(`Архівація документа id=${selectedDocId}`)} disabled={!selectedDocId}>
              Архівувати
            </button>
            <button
              onClick={() => {
                if (!selectedDocId) return alert('Оберіть документ для видалення');
                if (window.confirm('Ви впевнені, що хочете видалити документ?')) {
                  alert(`Видалити документ id=${selectedDocId}`);
                }
              }}
              disabled={!selectedDocId}
              style={{ color: 'red' }}
            >
              Видалити
            </button>
          </div>
        </div>
        {showForm && <NewDocumentForm onClose={handleCloseForm} />}
```

```
        </div>
    );
}
```

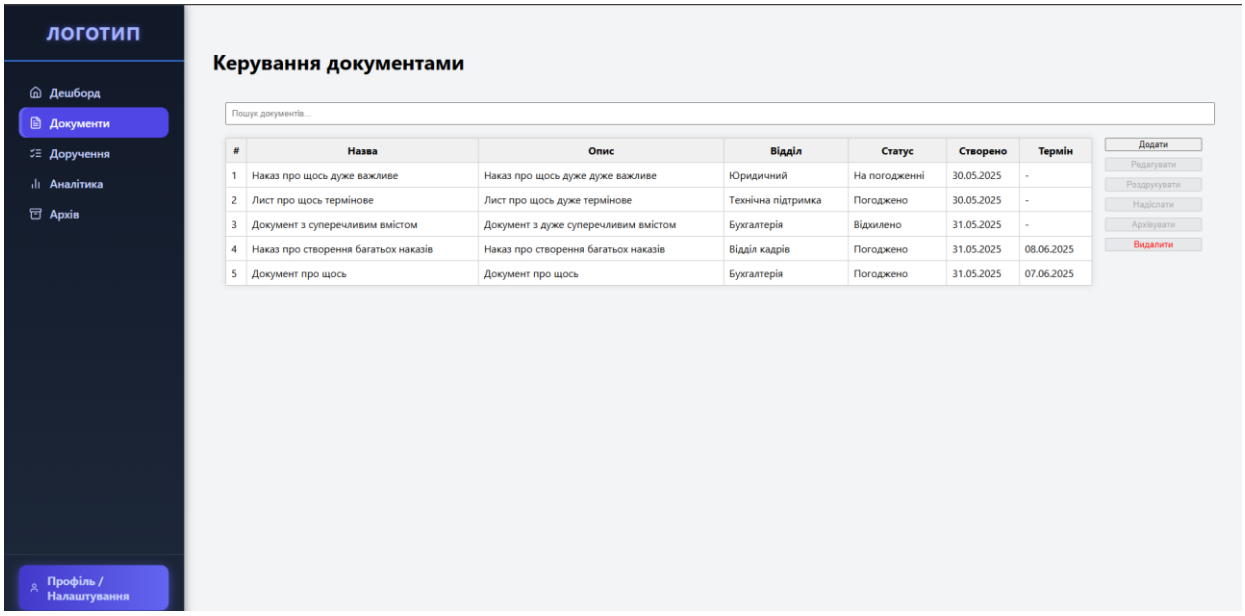

Рисунок А.2.2 – Інтерфейс сторінки керування документами системи обліку документів

А.2.3. Сторінка аналітики та звітності

```
const STATUS_COLORS = {
    'Погоджено': '#82ca9d',
    'На погодженні': '#8884d8',
    'Відхилено': '#f87171'
};
function AnalyticsPage() {
    const [latency, setLatency] = useState({ current: 0, average: 0 });
    const [uptime, setUptime] = useState('');
    const [loadData, setLoadData] = useState([]);
    const [stabilityData, setStabilityData] = useState([]);
    const [docStats, setDocStats] = useState([]);
```

```
  const [statusDistribution, setStatusDistribution] = useState([]);
  useEffect(() => {
    fetchSystemStatus();
    fetchDocStats();
    fetchStatusDistribution();
  }, []);
  const fetchSystemStatus = async () => {
    const res = await fetch('http://localhost:8000/api/analytics/system-status/');
    const data = await res.json();
    setLatency({ current: data.latency.current, average: data.latency.average });
    setUptime(data.uptime);
    setLoadData(data.cpu_chart); // тепер тут CPU
    setStabilityData(data.request_chart); // тепер тут HTTP запити
  };
  const fetchDocStats = async () => {
    const res = await fetch('http://localhost:8000/api/documents/documents-daily/');
    const data = await res.json();
    setDocStats(data);
  };

  const fetchStatusDistribution = async () => {
    const res = await fetch('http://localhost:8000/api/documents/document-status-distribution/');
    const data = await res.json();
    console.log("Status distribution:", data);
    setStatusDistribution(data);
  };
  const downloadPDF = () => {
```

```
      window.open('http://localhost:8000/api/documents/reports/generate/',
'_blank');
  };
  return (
    <div style={{ padding: 20 }}>
      <h1>Аналітика системи</h1>
      <section style={{ marginBottom: 40 }}>
        <h2>Навантаженість системи</h2>
        <div style={{ display: 'flex', gap: 20, marginBottom: 20, flexWrap: 'wrap'
}}>
          <div style={{ padding: 20, border: '1px solid #ccc', borderRadius: 8 }}>
            <strong>Поточна затримка:</strong> {latency.current} мс
          </div>
          <div style={{ padding: 20, border: '1px solid #ccc', borderRadius: 8 }}>
            <strong>Середня затримка:</strong> {latency.average} мс
          </div>
          <div style={{ padding: 20, border: '1px solid #ccc', borderRadius: 8 }}>
            <strong>Аптайм сервера:</strong> {uptime}
          </div>
        </div>
        <div style={{ display: 'flex', flexWrap: 'wrap', gap: 30 }}>
          <div style={{ flex: 1, minWidth: 300 }}>
            <h3>Завантаження CPU (%)</h3>
            <ResponsiveContainer width="100%" height={250}>
              <LineChart data={loadData}>
                <CartesianGrid stroke="#ccc" />
                <XAxis dataKey="time" />
                <YAxis domain={[0, 100]} />
                <Tooltip />
```

```
                        <Line type="monotone" dataKey="cpu" stroke="#8884d8" />
                    </LineChart>
                </ResponsiveContainer>
            </div>
            <div style={{ flex: 1, minWidth: 300 }}>
                <h3>Звернення до сервера (за хвилину)</h3>
                <ResponsiveContainer width="100%" height={250}>
                    <LineChart data={stabilityData}>
                        <CartesianGrid stroke="#ccc" />
                        <XAxis dataKey="time" />
                        <YAxis />
                        <Tooltip />
                        <Line type="monotone" dataKey="requests" stroke="#82ca9d"
/>
                    </LineChart>
                </ResponsiveContainer>
            </div>
        </div>
    </section>
    <section style={{ marginBottom: 40 }}>
        <h2>Статистика надходження документів</h2>
        <ResponsiveContainer width="100%" height={300}>
            <BarChart data={docStats}>
                <CartesianGrid stroke="#ccc" />
                <XAxis dataKey="date" />
                <YAxis />
                <Tooltip />
                <Bar dataKey="count" fill="#8884d8" />
            </BarChart>
```

```
    </ResponsiveContainer>
</section>
<section style={{ marginBottom: 40 }}>
    <h2>Виконання доручень</h2>
    <ResponsiveContainer width="100%" height={300}>
      <PieChart>
        <Pie
          data={statusDistribution}
          dataKey="count"
          nameKey="label"
          cx="50%"
          cy="50%"
          outerRadius={100}
          label
        >
          {statusDistribution.map((entry, index) => (
            <Cell
              key={index}
              fill={STATUS_COLORS[entry.label] || '#ccc'}
            />
          ))}
        </Pie>
        <Legend />
        <Tooltip />
      </PieChart>

    </ResponsiveContainer>
</section>
<section>
```

```
        <h2>Спеціальний звіт</h2>
        <button onClick={downloadPDF}>Завантажити PDF-звіт</button>
      </section>
    </div>
);
```

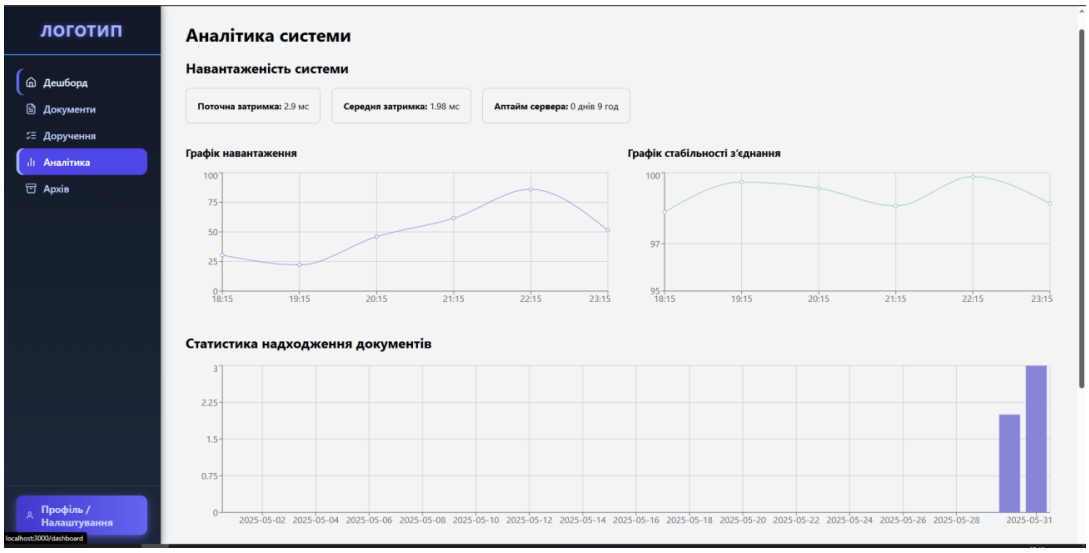

Рисунок А.2.3 – Інтерфейс сторінки аналітики та звітності системи