

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Вебсистема керування процесами видавництва»

Виконала студентка групи ІПЗ-22-2
КОЗЛИК Софія Олегівна

Керівник роботи:
ФАНТ Микола Олександрович

Рецензент:
ВЛАСЕНКО Олег Васильович

Житомир – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КОЗЛИК Софія Олегівна**

Керівник роботи: **ФАНТ Микола Олександрович**

Тема роботи: **«Вебсистема керування процесами видавництва»**,
затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**

Вихідні дані для роботи: **об'єктом дослідження є процеси керування видавничими проєктами та підготовки книжкових видань у видавництві; предметом дослідження є застосування сучасних методів, засобів і програмних рішень для проєктування та реалізації вебсистеми керування процесами видавництва.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Фант М.О.	02.03.2026	02.03.2026
2	Фант М.О.	02.03.2026	02.03.2026
3	Фант М.О.	02.03.2026	02.03.2026

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновків, джерел інформації, додатків.

Текстова частина викладена на 90 сторінках друкованого тексту. Список використаних джерел містить 32 найменування і займає 3 сторінки. Пояснювальна записка має 6 сторінок додатків. У роботі наведено 78 рисунків. Загальний обсяг роботи – 96 сторінок.

Метою роботи є реалізація вебсистеми керування процесами видавництва, яка має необхідні інструменти для керування проєктами, працівниками, завантаження файлів допоміжних матеріалів та редагування тексту рукописів у редакторі з можливістю відновлення попередніх версій документів.

У роботі було поставлено задачу, проаналізовано аналоги розроблюваної системи, визначено їх переваги та недоліки, , що дозволило сформулювати вимоги до функціоналу вебсистеми та обґрунтувати необхідність її створення. Вибрано та обґрунтовано архітектуру і засоби розробки застосунку. Описано варіанти використання вебсистеми, структуру її бази даних та наведено опис реалізації програмних компонентів. Пояснено процес розгортання вебсистеми із зазначенням важливих аспектів. Розглянуто її інтерфейс та основні сценарії роботи з нею. Наведено результати тестування системи.

У висновках сформульовано підсумки виконаної роботи.

КЛЮЧОВІ СЛОВА: ВИДАВНИЦТВО, СИСТЕМА, КНИГИ, ASP.NET CORE, REACT.

					ІПЗ.КР.Б – 121 – 26 – ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Козлик С.О.			Вебсистема керування процесами видавництва			Літ.	Арк.	Аркушів	
Керівник		Фант М.О.								2	96
Рецензент		Власенко О.В.			Пояснювальна записка			Житомирська політехніка, група ІПЗ-22-2			
Зав. каф.		Вакалюк Т.А.									

ABSTRACT

The explanatory note to the bachelor's final qualification work consists of an introduction, three sections, conclusions, and sources of information.

The main body of the thesis consists of 90 pages of printed text. The reference list contains 32 sources and covers 3 pages. The explanatory note contains 6 pages of appendices. The thesis contains 78 figures. The total volume of the thesis is 96 pages.

The purpose of the work is to implement a web-based publishing processes management system, which has the necessary tools for managing projects, employees, uploading files of supporting materials, and editing manuscript text in the editor with the ability to restore previous versions of documents.

The work defines the project objectives, analyzes existing solutions in the field, and identifies their advantages and disadvantages, which allowed the formulation of requirements for the web system functionality and justified the need for its development. The architecture and development tools of the application were selected and justified. The system use cases, database structure, and implementation of software components are described. The deployment process of the web system is presented, highlighting its key aspects. The system interface and main usage scenarios are also presented. The results of system testing are discussed.

The conclusions formulate the results of the work performed.

KEYWORDS: PUBLISHING, SYSTEM, BOOKS, ASP.NET CORE, REACT.

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ КЕРУВАННЯ ПРОЦЕСАМИ ВИДАВНИЦТВА	10
1.1 Постановка задачі	10
1.2 Аналіз аналогів програмного продукту	13
1.3 Вибір архітектури системи керування процесами видавництва.....	21
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	25
Висновки до першого розділу.....	28
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБСИСТЕМИ КЕРУВАННЯ ПРОЦЕСАМИ ВИДАВНИЦТВА	29
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	29
2.2 Розробка бази даних	38
2.3 Проєктування та реалізація алгоритмів роботи системи	42
2.4 Реалізація вебсистеми керування процесами видавництва	45
Висновки до другого розділу.....	51
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБСИСТЕМОЮ КЕРУВАННЯ ПРОЦЕСАМИ ВИДАВНИЦТВА.....	52
3.1 Порядок встановлення та налаштування параметрів вебсистеми ...	52
3.2 Структура інтерфейсу вебсистеми керування процесами видавництва	55
3.3 Тестування вебсистеми керування процесами видавництва.....	83
Висновки до третього розділу	86
ВИСНОВКИ.....	87
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	88
ДОДАТКИ	91

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – база даних.

API (Application Programming Interface) – прикладний програмний інтерфейс.

CORS (Cross-Origin Resource Sharing) – механізм надання контрольованого доступу до ресурсів вебзастосунку з інших доменів.

JWT (JSON Web Token) – токен для автентифікації та авторизації користувачів.

REST (Representational State Transfer) – архітектурний стиль побудови вебсервісів та програмних інтерфейсів.

SPA (Single Page Application) – односторінковий вебзастосунок.

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Процес створення та видання книг з кожним роком стає дедалі складнішим і багатоступеневим. Над публікацією однієї книги працює велика команда спеціалістів: автори, редактори, коректори, дизайнери, рецензенти і менеджери. Цей процес вимагає чіткої та ефективної координації, контролю над виконанням усіх етапів [1, 2], перевірки їх якості, централізоване зберігання усієї інформації. Часто виконання цих вимог поділяється між розрізненими програмними продуктами, які мають обмежену предметну область, а тому і невелику, порівняно з усіма вимогами видавництва, кількість функціональних можливостей. Хоча дані рішення задовольняють потреби учасників видавничого процесу, але можуть спричинити неочікувані проблеми: втрата даних, ускладнення контролю, зниження ефективності роботи.

Існуючі програмні рішення не повною мірою враховують специфіку видавничого циклу, ролі працівників видавництва, послідовність етапів підготовки рукопису. Зі стрімким зростанням зацікавленості до книжкових видань в останні роки збільшується навантаження на видавництва та їх працівників. Це створює нагальну потребу у розробці спеціалізованої надійної вебсистеми керування процесами видавництва, яка відповідатиме усім стандартам сучасного програмного забезпечення. Дана вебсистема дозволить централізувати зберігання матеріалів, оптимізувати взаємодію між учасниками процесу, синхронізуватиме усі зміни у проектах, зроблені користувачами, тим самим підвищивши надійність, безпеку та ефективність керування проектами видавництва.

Метою кваліфікаційної роботи є розробка вебсистеми керування процесами видавництва, яка враховуватиме специфіку роботи видавництва, надаватиме необхідні інструменти для виконання його задач при підготовці рукописів до публікації і матиме зручний та інтуїтивно зрозумілий інтерфейс.

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

Досягнення поставленої мети потребує виконання наступних **завдань**:

- аналіз предметної області видавничої діяльності, визначення основних складових процесу видання книг;
- визначення основних задач з розробки системи;
- аналіз аналогів вебсистеми;
- проєктування та реалізація структури бази даних;
- розробка сервера вебсистеми;
- реалізація інтерфейсу користувача;
- проведення тестування вебсистеми.

Об'єктом дослідження є процеси керування видавничими проєктами та підготовки книжкових видань у видавництві.

Предметом дослідження є застосування сучасних методів, засобів і програмних рішень для проєктування та реалізації вебсистеми керування процесами видавництва.

Методи дослідження. Для вирішення поставлених завдань у роботі застосовано методи системного аналізу для дослідження предметної області та формування вимог до вебсистеми, принципи об'єктно-орієнтованого проєктування для побудови її архітектури, методи проєктування баз даних для розробки структури зберігання даних, а також сучасні підходи програмної інженерії та проєктування користувацьких інтерфейсів.

Практичне значення отриманих результатів полягає у розробці вебсистеми керування процесами видавництва, яка може бути використана для автоматизації роботи видавництв та редакційних колективів. Вона забезпечує централізоване керування проєктами, їх складовими та взаємодією між учасниками видавничого процесу, що сприяє підвищенню ефективності роботи і зменшенню витрат часу на виконання організаційних операцій. Результати роботи можуть бути використані як основа для подальшого розвитку та впровадження програмного продукту в діяльність реальних видавництв.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

Апробація результатів роботи. За темою кваліфікаційної роботи опубліковано дві тези доповідей у матеріалах науково-практичних конференцій:

1. Козлик С., Фант М. Технологічний стек для системи керування процесами видавництва. Тези доповідей VIII Всеукраїнської науково-технічної конференції «Комп’ютерні технології: інновації, проблеми, рішення». Житомир, 2025. С. 35–36.

2. Козлик С. О., Фант М. О. Гібридна модель контролю доступу у вебсистемі керування процесами видавництва. Інформаційні технології: теорія і практика : матеріали III (IX) Міжнародної інтернет-конференції здобувачів вищої освіти і молодих учених. Дніпро, 2026. С. 517–518.

Структура та обсяг роботи. Випускна кваліфікаційна робота складається зі вступу, 3-х розділів, висновків, списку використаних джерел із 32 найменувань (3 стор.), 1 додатка (6 стор.), 6 таблиць та 78 ілюстрацій по тексту. Повний обсяг роботи становить 96 сторінок, з них 90 сторінок основного тексту.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ВЕБСИСТЕМИ КЕРУВАННЯ ПРОЦЕСАМИ ВИДАВНИЦТВА

1.1 Постановка задачі

Основною задачею випускної кваліфікаційної роботи бакалавра є проєктування та розробка вебсистеми керування процесами видавництва з використанням сучасних технологій та підходів програмування. Розроблена система призначена для автоматизації основних процесів видавничої діяльності та об'єднання інструментів, необхідних для роботи над книгою, в єдиному інформаційному середовищі.

Вебсистема дозволяє оптимізувати процес підготовки видання завдяки використанню вбудованого текстового редактора, підтримці версійності документів, можливості спільної роботи над проєктом, детальному плануванню його етапів та контролю виконання завдань. Користувачі можуть відстежувати прогрес роботи, взаємодіяти між собою і швидко отримувати доступ до необхідної інформації в межах проєкту.

Під час розробки враховувалися особливості організації роботи сучасних видавництв, розподіл обов'язків між різними категоріями працівників та специфіка процесу підготовки книги до публікації. Запропоноване рішення усуває недоліки використання декількох розрізнених програмних продуктів, централізуючи роботу з проєктами, документами, матеріалами та завданнями в межах однієї вебсистеми. Це сприяє підвищенню ефективності роботи працівників, покращенню комунікації між учасниками проєкту та забезпечує більш прозорий контроль над усіма етапами видавничого процесу.

Для конкретизації поставленої мети та полегшення планування процесу розробки було виділено низку взаємопов'язаних підзадач. Першим етапом є проєктування структури вебсистеми та визначення її основних компонентів. На цьому етапі необхідно сформулювати перелік ролей користувачів відповідно до потреб видавництв і особливостей організації їхньої роботи, визначити

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

функціональні можливості кожної ролі та створити матрицю дозволів для контролю доступу до ресурсів системи.

Крім того, планується виконати моделювання системи шляхом розробки сценаріїв використання та побудови концептуальної, логічної й фізичної моделей. Це дозволить формалізувати вимоги до вебсистеми, описати взаємодію її компонентів та отримати цілісне уявлення про реалізацію програмного продукту.

Наступною підзадачею є проєктування структури бази даних, яка забезпечуватиме зберігання інформації про користувачів, видавництва, проєкти, книги, етапи, завдання, матеріали та інші сутності системи. Грамотно спроектована база даних є основою стабільної роботи вебсистеми, оскільки забезпечує цілісність даних, ефективне виконання запитів та можливість подальшого розширення функціональності застосунку.

Другим етапом роботи над вебсистемою є розробка її ядра, тобто серверної частини, яка відповідатиме за реалізацію бізнес-логіки, обробку запитів користувачів та взаємодію з базою даних. На цьому етапі планується реалізація основного функціоналу програми, що забезпечуватиме підтримку всіх ключових бізнес-процесів видавництва.

Першочерговим завданням є реалізація механізмів реєстрації, авторизації та автентифікації користувачів із розподілом прав доступу відповідно до їхніх ролей у системі. Це дозволить забезпечити захист даних та обмежити доступ до функціональності залежно від обов'язків конкретного користувача.

Окремий блок функціональності присвячений керуванню видавництвами та проєктами. Адміністратор видавництва повинен мати можливість створювати та редагувати інформацію про видавництво, запрошувати працівників, формувати проєкти та керувати ними. Для зручності роботи передбачається реалізація механізмів пошуку, фільтрації, сортування та пагінації даних.

Потрібно додати збереження інформації про книги, додаткові матеріали для них із їх завантаженням у хмарне сховище та керуванням ними. Не менш важливим функціоналом є налаштування етапів підготовки рукопису до публікації, перехід між ними, відстеження поточного етапу, оскільки це додасть організованості та гнучкості процесу роботи над текстом рукопису.

Для роботи над проєктом потрібно реалізувати додавання відповідальних за нього працівників та призначення їм завдань, пов'язаних зі зміною тексту рукопису книги. Водночас для забезпечення своєчасного інформування користувачів щодо змін станів проєктів, призначень на них, отримання нових завдань тощо необхідно розробити механізм сповіщень. Також потрібно передбачити коректну обробку запитів і отриманих відповідей системи та її поведінку в разі виникнення помилок у роботі.

Ще одним підзавданням є створення зручного користувацького інтерфейсу. Планується проєктування дизайну, визначення вимог до інтерфейсу, підбір кольорової гами та шрифтів для забезпечення зручності використання вебсистеми. Наступним кроком є розробка інтерфейсу для відображення реалізованих функціональностей серверної частини, які призначені для виконання завдань видавництва. Необхідно забезпечити зрозумілу структуру застосунку та зручну навігацію між сторінками.

Впродовж розробки вебсистеми потрібно проводити її тестування. Для цього слід розробити тестову стратегію та тестові сценарії. За допомогою них виконати функціональне тестування, перевірити вебсистему на адаптивність. У разі виявлення будь-яких помилок чи дефектів їх слід виправити для забезпечення коректної роботи програми.

Результатом виконання описаних завдань є працездатна система керування процесами видавництва у вигляді вебзастосунку, за допомогою якої організовано весь процес видання книги та взаємодію працівників для виконання їхніх завдань. Застосунок складається з бази даних для зберігання

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

інформації системи, клієнтської та серверної частин і програмного інтерфейсу взаємодії між ними.

1.2 Аналіз аналогів програмного продукту

Перед початком розробки вебсистеми було проведено аналіз існуючих програмних продуктів, призначених для керування видавничою діяльністю, контролем над проєктами та автоматизації окремих етапів видавничого процесу. Результати аналізу дозволили визначити необхідні функціональні можливості розроблюваної системи та сформувавши перелік її конкурентних переваг.

Розглянуті аналоги є повністю спеціалізованими видавничими системами або ж частково застосовуються працівниками видавництва для полегшення робочого процесу.

Першим прикладом описаних вище платформ є Editorial Manager (рис. 1.1) від Aries Systems. Вона використовується завантаження, редагування та рецензування рукописів для їх подальшої публікації у наукових виданнях [3].

The screenshot shows the Editorial Manager login interface. At the top, there's a navigation bar with the Elsevier logo and links for Home, Submit a Manuscript, About, and Help. Below this, a large banner reads "Welcome to Editorial Manager® for Elsevier EM Training Journal 1". The main login area includes a form with "Please Enter the Following" instructions, fields for Username and Password, and buttons for Author Login, Reviewer Login, Editor Login, and Publisher Login. There are also links for "Send Login Details", "Register Now", and "Login Help". The footer contains detailed instructions for first-time users, repeat users, authors, and reviewers, as well as a link to "Change your username and/or password".

Рис. 1.1. Платформа Editorial Manager

Серед конкурентних та особливих можливостей застосунку можна виділити функціонал для налаштування редакційних процесів відповідно до

вимог конкретних журналів або видавництв, що додає гнучкості при роботі з рукописами. Також система підтримує призначення відповідальних осіб за окремі етапи редакційного процесу, зокрема рецензентів, редакторів та інших учасників, із можливістю контролю виконання поставлених завдань (рис. 1.2). Варто відзначити управління комунікаціями, реалізоване у програмі: користувачі можуть використовувати попередньо створені шаблони електронних листів для зв'язку з рецензентами чи редакторами; налаштовувати автоматизовані листи і відстежувати власну кореспонденцію для своєчасного реагування.

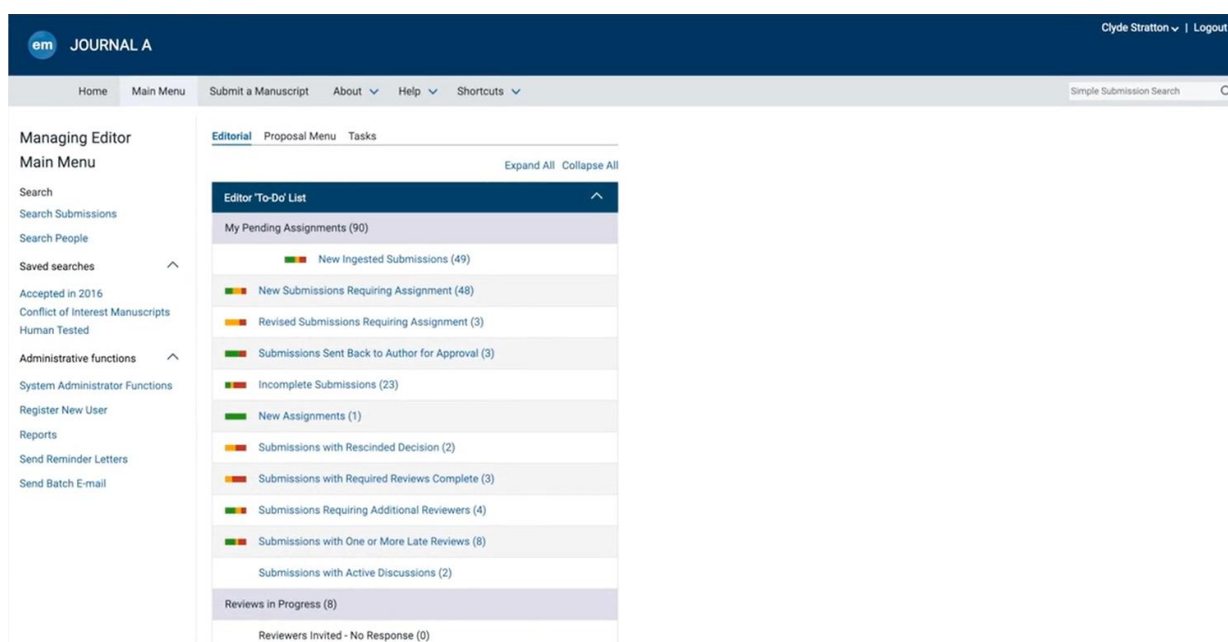


Рис. 1.2. Перегляд завдань в Editorial Manager

Editorial Manager підтримує розподіл користувачів за ролями та перегляд аналітики завдяки генерації звітів про обсяг поданих робіт, ефективність рецензентів, час прийняття рішень тощо. Застосунок має багато інтеграцій, серед яких є ORCID, CrossRef, PubMed та iThenticate для виявлення плагіату та індексації.

Серед недоліків системи потрібно підкреслити її орієнтованість на роботу з науковими журналами та рецензуванням статей та відсутність можливості редагування тексту одразу в системі, у Editorial Manager автори можуть лише завантажити файл рукопису (рис. 1.3). Можна також виокремити високий поріг входження через велику кількість опцій

застосунок, що впливає на час навчання користувачів та ефективність їхньої роботи. Крім того, Editorial Manager має обмежений функціонал: для отримання усіх можливостей для подальшої публікації рукопису необхідно використовувати Prodxion Manager – платформу від Aries Systems для виробництва книг, журналів тощо, комунікації з внутрішніми та зовнішніми зацікавленими сторонами після прийняття та рецензування рукопису.

Order	Item	Description	File Name	Size	Last Modified	Actions	Select
1	*Cover Letter	Cover letter	Cover Letter.docx	26.3 KB	Jun 19, 2023	Download	☐
2	*Declaration of Interest	Declaration of interest	Declaration of interest.docx	13.2 KB	Jun 19, 2023	Download	☐
3	*Manuscript	Manuscript	MANUSCRIPT FILE WITH REFERENCE CHECKING.docx	63.2 KB	Jun 19, 2023	Download	☐
4	Figure	Figure 2	Figure 2.png	18.4 KB	Jun 19, 2023	Download	☐
5	Figure	Figure 1	Figure 1.png	20.3 KB	Jun 19, 2023	Download	☐

Рис. 1.3. Завантаження рукопису в Editorial Manager

knkPublishing – це комплексне програмне рішення, яке складається з декількох програмних інструментів, як-от Production Management Software, Rights and Royalties Management, Editorial Office Software [4] тощо. Ці рішення мають різноманітний функціонал для редакторської роботи, керування проектами, планування повного циклу підготовки рукописів до публікації, відстеження виплат роялті, контроль над авторськими правами.

Як аналог розроблюваної системи розглянемо підсистему Editorial Office Software. Серед її переваг є налаштування всіх ключових складових проєктів (рис. 1.4), їх етапів з відстеженням контрольних точок, налаштування шаблонів розкладів для різних сценаріїв роботи над книгою, створення графіків для їх візуального відображення.

Projects: All ▾ Search New ▾ Delete Process ▾ Project ▾ Production ▾ Open in Excel More options								
No. ↓	Item Type Code	Main Title	Author Name	Status	Project Type Code	Print Run No.	Planned Publication Date	Print Run Planned
P10183	PB	Speech Therapy	Julia Engel	PLANNING	REVISED E...	5	16.02.2021	850
P10182	HC	Geriatric Care	Robert Decker	PRODUCTION	NEW	1	25.03.2021	2,000
P10170	E-PUB	Healthcare Management	Alex Becker	PLANNING	NEW	1		0
P10169	PDF	Healthcare Management	Alex Becker	PLANNING	NEW	1	25.02.2021	0
P10168	DATABASE			IDEA	NEW			0

Рис. 1.4. Перегляд проєктів у Editorial Office Software

Унікальним функціоналом в даному випадку можна вважати керування договорами, адже платформа надає готові шаблони контрактів між авторами та видавництвом, дозволяє їх редагувати чи створювати власні. Крім того, передбачено генерацію аналітичних звітів щодо активності користувачів, обсягів виконаних робіт та часу, витраченого на їх реалізацію (рис. 1.5).

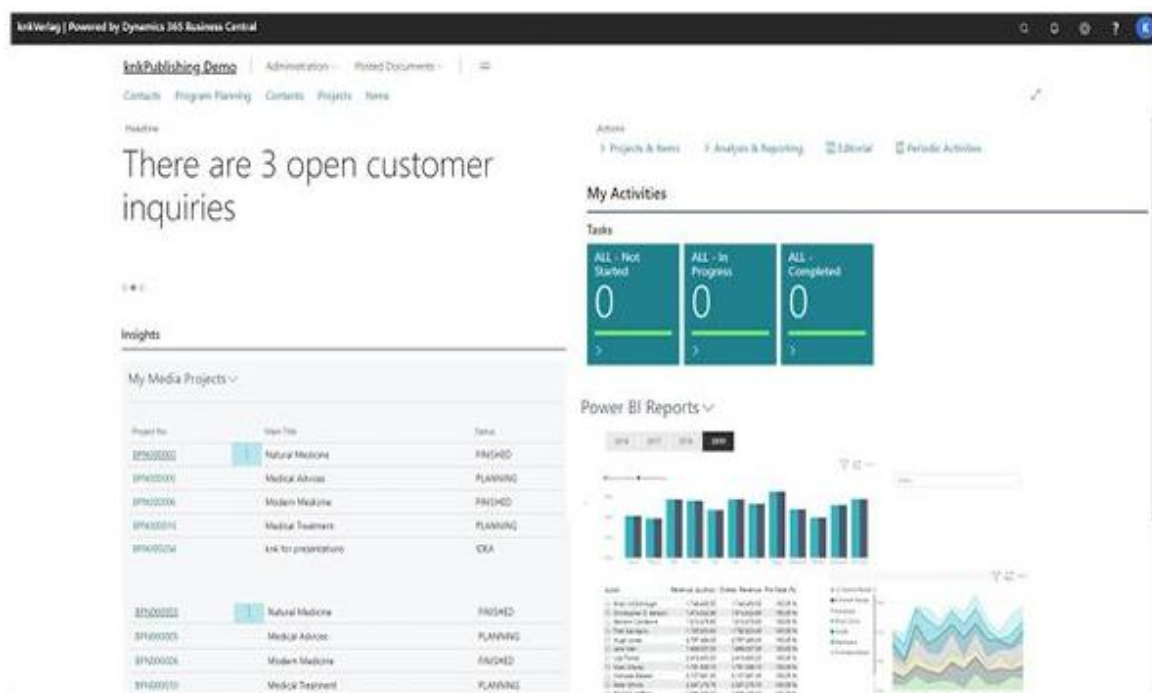


Рис. 1.5. Аналітика видавництва у Editorial Office Software

Одним з найважливіших недоліків Editorial Office Software є відсутність класичного текстового редактора: користувачі редагують текст у сторонніх програмах, після чого завантажують файли з ним у систему, що погіршує відстеження змін та може призвести до втрати даних.

Також, щоб ефективно використовувати систему, необхідні значні час та зусилля для належного вивчення персоналом функціоналу програми, необхідного для роботи у конкретному видавництві. Варто пам'ятати, що

Editorial Office Software є підсистемою комплексного рішення, а тому залежить від інтеграції з іншими модулями knkPublishing, оскільки через розподіл на програмні модулі в підсистемі може не вистачати певного функціоналу, який є частиною іншого рішення. Це значним чином впливає на її впровадження в роботу видавництва.

Scrivener – це застосунок для роботи з рукописами, він надає функціонал для безпосередньої роботи з текстом (рис. 1.6), а також для організації зберігання пов'язаних текстових та графічних матеріалів [5]. Призначений для використання письменниками, сценаристами та іншими творцями контенту. У нашому аналізі розглядається як супутній інструмент для редагування тексту, який може використовуватися видавництвами.

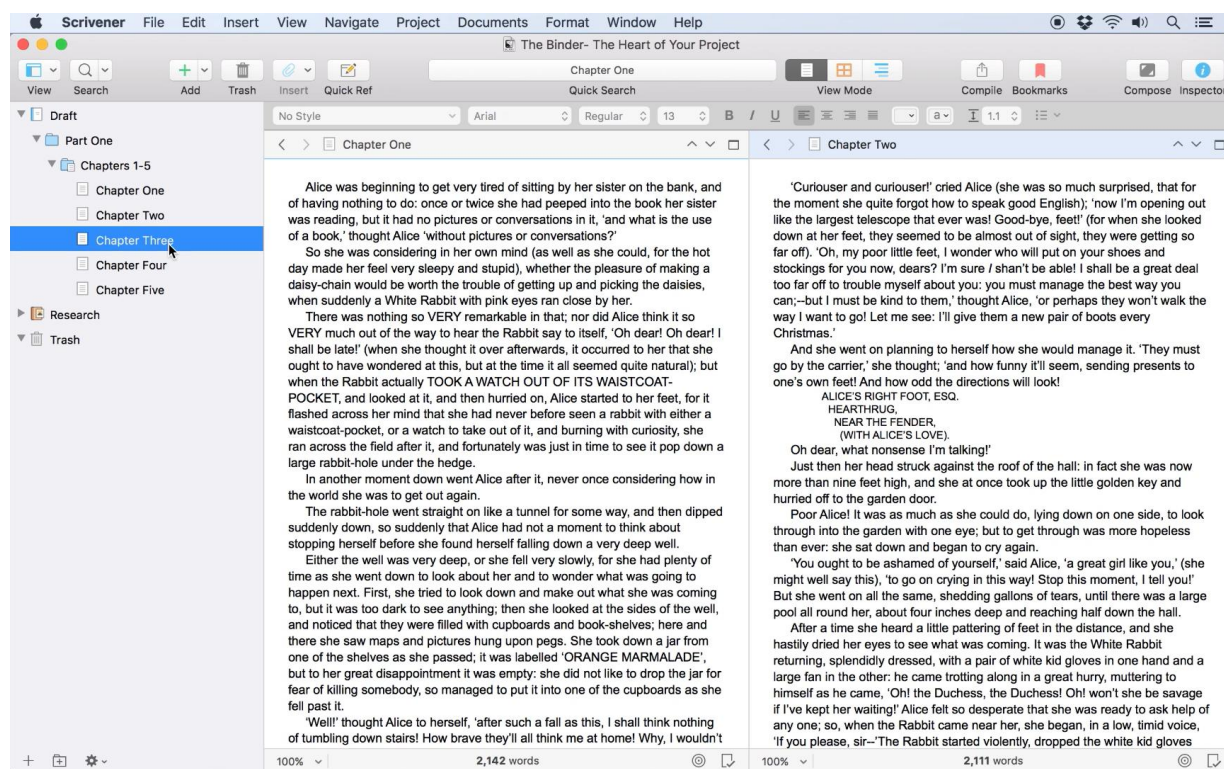


Рис. 1.6. Написання книги у застосунку Scrivener

Scrivener дозволяє організовувати проекти: користувачі можуть створювати проект, який вміщатиме пов'язані текстові документи, які можна розбивати на менші компоненти та реорганізовувати (рис. 1.7). Також є можливість завантаження в програму супутніх матеріалів, пов'язаних з рукописом, наприклад: зображення, нотатки, синопсиси, PDF-файли тощо.

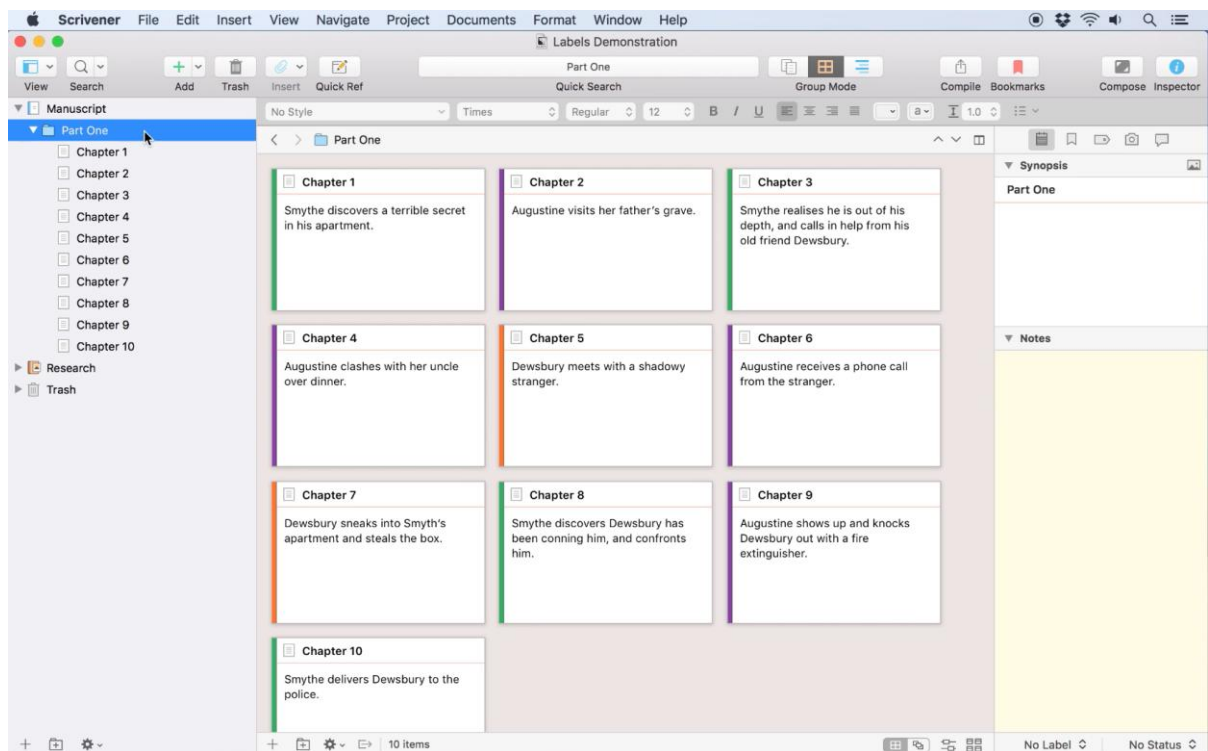


Рис. 1.7. Організація матеріалів у застосунку Scrivener

Оскільки програма орієнтована на письменників та сценаристів, вона пропонує вже реалізовані шаблони для різних типів творів, як-от романи, вірші, сценарії. Зручною є можливість налаштування панелі інструментів, яку користувачі можуть змінювати відповідно до власних потреб.

Недоліком застосунку є відсутність можливості одночасної роботи декількох працівників над одним текстовим проєктом. Як вже зазначалося раніше, Scrivener є інструментом для написання та редагування тексту, тому це лише часткове рішення, яке застосовується працівниками видавництв під час роботи над проєктами.

Reedsy Studio – вебплатформа для створення, редагування та форматування книг [6]. Система орієнтована насамперед на авторів і сценаристів та дозволяє працювати над текстом рукопису, допомагає в організації нотаток книги, побудові персонажів, світу, сюжетних ліній.

Основною функціональністю застосунку є створення та редагування рукописів (рис. 1.8) із підтримкою форматування тексту, структурування їх розділів, імпорту документів з docx файлів, їх експорту у різні формати, зокрема PDF та EPUB.



Рис. 1.8. Робота з текстом книги

Вебплатформа також підтримує спільну роботу над проектами та дозволяє залишати коментарі до тексту. Вона має додаткові інструменти для планування книги, зокрема засоби створення профілів персонажів з їх характеристиками та цілями (рис. 1.9), нотаток, описів світу, символіки та структурування сюжетних ліній. Це спрощує роботу над текстом завдяки організованості допоміжних матеріалів.

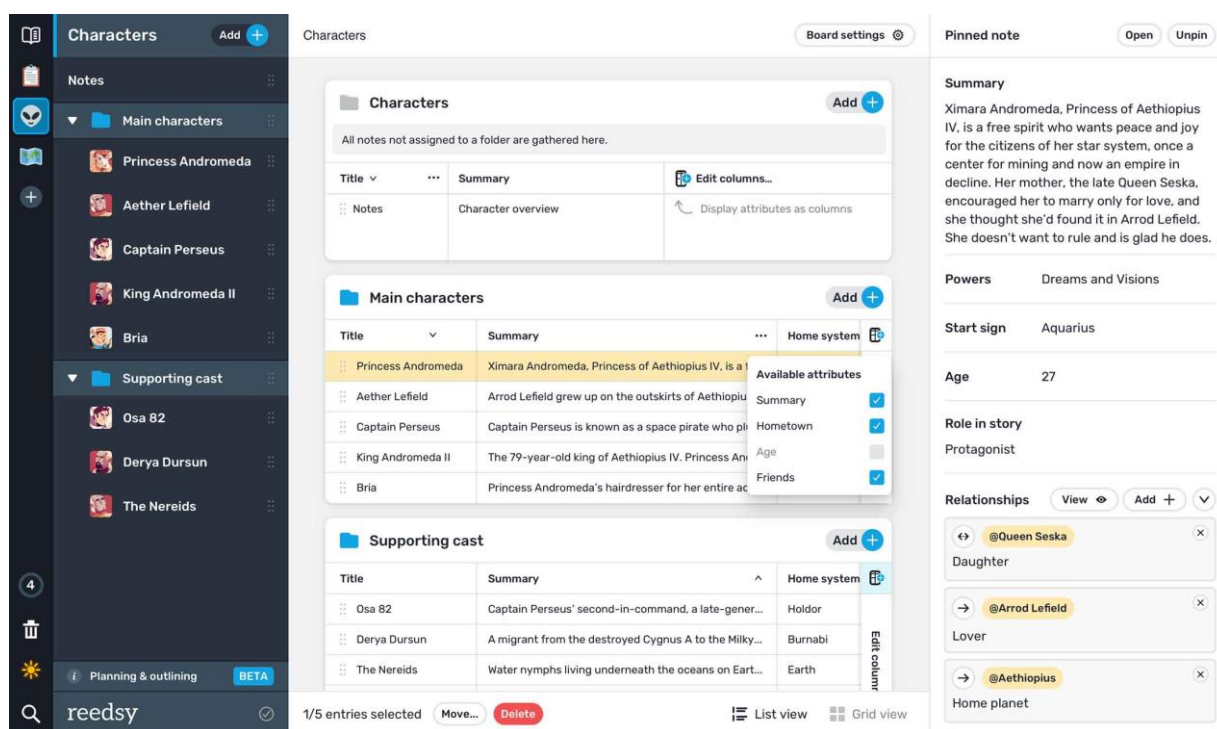


Рис. 1.9. Налаштування характеристик персонажів

Проте Reedsy Studio має ряд обмежень. Вона орієнтована на індивідуальних авторів, а не на повноцінну організацію роботи видавництва. В аналізі вона розглядається як часткове рішення, оскільки не має інструментів для керування ролями користувачів, контролю етапів підготовки видання, надання завдань тощо.

На основі проведеного аналізу аналогів було сформовано порівняльну таблицю їхніх функціональних можливостей (табл. 1.1). Порівняння виконано відповідно до вимог та запланованого функціоналу вебсистеми керування процесами видавництва.

Таблиця 1.1

Функціональні можливості систем

Функціональні можливості	Editorial Manager	Editorial Office Software	Scrivener	Reedsy Studio
Налаштування проекту	+	+	+	+
Додавання матеріалів до рукопису (графічні, текстові)	+	+	+	+
Налаштування етапів підготовки рукопису до публікації	+	+	-	-
Розподіл користувачів відповідно ролям у видавництві	+	+	-	-
Призначення завдань працівникам видавництва та керування ними	+	+	-	-
Модифікація тексту рукопису у текстовому редакторі	-	-	+	+
Генерування звітів про роботу видавництва	+	+	-	-
Отримання сповіщень про стан проекту, завдань	+	+	-	-

Кожна обрана платформа має повну або часткову реалізацію можливостей, які планується втілити в розроблюваній системі. Editorial Manager і Editorial Office Software є спеціалізованими рішеннями, які

використовуються у роботі видавництва. Вони надають унікальні засоби для планування та ведення проєктів, роботи над рукописами, їх редагуванням та рецензуванням. Важливим також є те, що вони підтримують аналітику, яка дозволяє оцінити ефективність та продуктивність видавництва. Scrivener та Reedsy Studio демонструють лише часткові рішення з огляду на розроблювану систему, проте виділення їх позитивних та негативних сторін допоможе під час розробки відповідного функціоналу у майбутньому. Scrivener показав себе як гарний текстовий редактор, який дозволяє зберігати всі допоміжні файли в одному проєкті з рукописом, що полегшує орієнтацію між матеріалами та роботу над текстом. Щодо Reedsy Studio варто зазначити можливість надати доступ до книги для спільної роботи, адже це важливо для модифікації тексту рукопису у видавництві, коли над ним працюються декілька спеціалістів. Щодо іншої функціональності Reedsy Studio є подібним на Scrivener.

Таким чином, було визначено ключові характеристики обраних аналогів, які доцільно враховувати під час розробки вебсистеми. До них належать зручість використання, інтуїтивно зрозумілий дизайн, можливість налаштування видавничих проєктів з додаванням рукописів книг та супровідних матеріалів, планування етапів виконання робіт, призначення завдань працівникам, підтримка роботи з текстом за допомогою вбудованого редактора, а також розмежування прав доступу відповідно до ролей користувачів.

1.3 Вибір архітектури системи керування процесами видавництва

Під час проєктування вебсистеми важливим етапом став вибір її архітектури, адже саме це рішення визначає структуру програмного забезпечення та принципи взаємодії між його компонентами. Від обраної архітектури значною мірою залежать продуктивність, надійність, безпека та зручність подальшого супроводу системи.

Крім того, архітектура впливає на можливість розширення функціональних можливостей застосунку, внесення змін до окремих

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

компонентів без порушення роботи всієї системи та інтеграції із зовнішніми сервісами. Правильно обраний архітектурний підхід також забезпечує масштабованість програмного продукту, що дозволяє ефективно адаптувати його до зростання кількості користувачів, обсягу даних та нових вимог предметної області.

Систему керування процесами видавництва було вирішено побудувати з використанням клієнт-серверної архітектури, в якій передбачається розділення програми на клієнтську і серверну частини. Взаємодія між ними здійснюється через REST API [7], що забезпечує стандартизований обмін даними та дозволяє ефективно організувати роботу вебзастосунку.

Клієнтська частина відповідає за реалізацію інтерфейсу користувача, тобто за взаємодію з ним, відображення даних системи та обробку подій. Вона надсилає запити до сервера, отримує результати їх виконання та відображає їх у зручному та зрозумілому вигляді. Такий підхід дозволяє створити інтерактивний інтерфейс із швидким реагуванням на дії користувача.

Серверна частина відповідає за обробку отриманих запитів, реалізацію бізнес-логіки системи, перевірку прав доступу, взаємодію з базою даних та інтеграцію із зовнішніми сервісами. Саме на сервері виконуються основні операції, пов'язані з керуванням проєктами, працівниками, завданнями, рукописами та іншими сутностями вебсистеми.

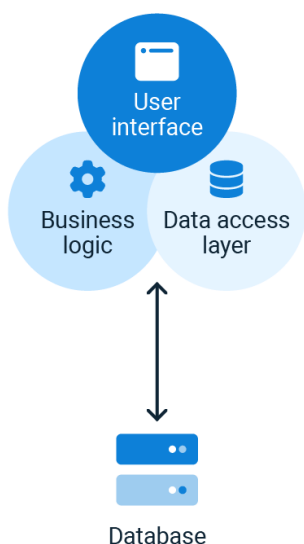
Використання клієнт-серверної архітектури забезпечило чітке розділення відповідальностей між компонентами вебсистеми, спростило її супровід і подальший розвиток. Крім того, такий підхід дозволяє незалежно розробляти та оновлювати клієнтську і серверну частини, а також за потреби змінювати технології їх реалізацій без суттєвого впливу на роботу всієї системи.

Для побудови серверної частини найбільш поширеними є монолітна та мікросервісна архітектури [8, 9] (рис. 1.10). При монолітній архітектурі серверна частина реалізується у вигляді єдиного програмного модуля,

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

компоненти якого розгортаються та функціонують у межах одного застосунку. Взаємодія між окремими частинами системи відбувається безпосередньо всередині застосунку, що спрощує обмін даними та реалізацію бізнес-логіки.

Monolithic Architecture



Microservice Architecture

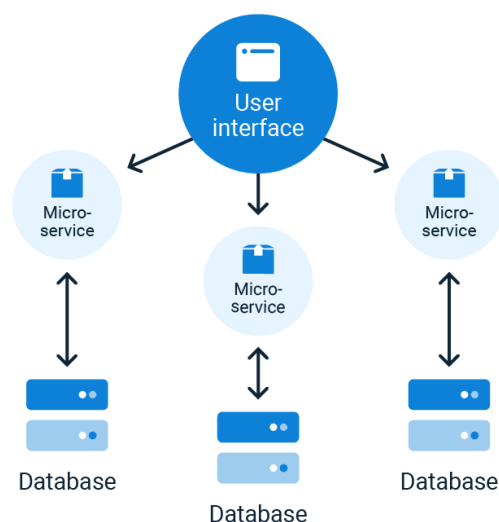


Рис. 1.10. Порівняння монолітної та мікросервісної архітектур

До основних переваг такого підходу належать простота розробки та розгортання, зручність налагодження й тестування, а також відсутність необхідності організовувати взаємодію між окремими сервісами через мережу. Монолітна архітектура є особливо ефективною для невеликих і середніх проєктів, оскільки дозволяє швидко реалізовувати новий функціонал та підтримувати цілісність програмного коду.

Водночас зі збільшенням розміру та складності системи можуть виникати труднощі з її масштабуванням і супроводом. Зміни в окремих компонентах часто потребують повторного розгортання всього застосунку, а велика кодова база може ускладнювати розробку та підтримку. Попри це, для багатьох інформаційних систем монолітна архітектура залишається доцільним рішенням завдяки простоті реалізації та нижчим витратам на розробку й адміністрування.

Мікросервісна архітектура, навпаки, передбачає поділ серверної частини на набір невеликих незалежних сервісів, кожен з яких відповідає за окремий бізнес-процес або функціональну область системи. Основною перевагою мікросервісного підходу є висока гнучкість та масштабованість системи. Окремі сервіси можуть оновлюватися без впливу на роботу інших компонентів, а навантаження може розподілятися лише на ті частини системи, які цього потребують. Крім того, різні сервіси можуть використовувати різні технології та інструменти залежно від поставлених завдань.

Проте мікросервісна архітектура значно ускладнює процеси розробки, тестування та супроводу застосунку, а також потребує істотних ресурсів для розгортання й адміністрування. Тому її використання є найбільш доцільним для великих високонавантажених систем, тоді як для проєктів середнього масштабу вона може призводити до надмірного ускладнення архітектури.

З огляду на все вищевикладене, було прийнято рішення реалізувати серверну частину у вигляді монолітного застосунку з використанням принципів Onion Architecture. Усі компоненти розміщено в межах одного рішення та поділено на логічні рівні, або ж шари:

- Domain – містить усі бізнес-сутності системи, перерахування тощо;
- Application – реалізує бізнес-логіку, варіанти використання системи, доменні інтерфейси;
- Infrastructure – відповідальний за взаємодію із зовнішніми системами, які необхідні попередньому шару для коректної роботи, це можуть бути: бази даних, компоненти для обміну повідомленнями, email провайдери тощо.
- Presentation (API) – реалізує API для взаємодії з клієнтською частиною.

Головне правило даного підходу – залежності спрямовані всередину, тобто зовнішні шари можуть залежати лише від внутрішніх. Ключовим принципом є інверсія залежностей, при якій високорівневі модулі з бізнес-

логікою не залежать від низькорівневих з інфраструктурою, а спираються лише на визначені абстракції. Це забезпечує слабку зв'язність, підвищує тестованість і легку заміну компонентів без впливу на систему загалом.

Клієнтська частина системи реалізована у вигляді односторінкового вебзастосунку (SPA) за допомогою бібліотеки React. Це дозволило мінімізувати кількість перезавантажень сторінки та підвищити швидкість реакції інтерфейсу на дії користувача.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для реалізації вебсистеми керування процесами видавництва було обрано сучасні та ефективні технології (табл. 1.2), які відповідають актуальним тенденціям розробки програмних застосунків та забезпечують створення надійного, масштабованого й зручного у використанні програмного продукту [10]. Вибір технологічного стеку здійснювався з урахуванням функціональних вимог до вебсистеми, особливостей предметної області, а також необхідності забезпечення високої продуктивності, безпеки та можливості подальшого розвитку застосунку.

Таблиця 1.2

Технології, використані для реалізації вебсистеми

Компонент системи	Використана технологія
База даних	PostgreSQL
Серверна частина	Мова програмування C#, фреймворк ASP.NET Core Web API
Клієнтська частина	Бібліотека React, мова програмування TypeScript, CSS-фреймворк Tailwind CSS
Хмарне сховище файлів	Azure Blob Storage
Середовища розробки	Visual Studio, Visual Studio Code

Дані системи зберігаються в СУБД PostgreSQL [11, 12], оскільки вона забезпечує високу надійність, продуктивність та підтримку широкого спектра механізмів для роботи з даними. До її переваг належать підтримка складних

SQL-запитів, транзакцій, обмежень цілісності, тригерів та індексів, що дозволяє ефективно працювати з великими обсягами інформації та гарантувати її узгодженість.

У базі даних реалізовано зберігання всіх основних сутностей вебсистеми, зокрема облікових записів користувачів, видавництв, проєктів, книг, матеріалів, етапів, завдань, сповіщень, звернень користувачів та інших пов'язаних даних. Використання PostgreSQL гарантує надійне зберігання даних та високу швидкість доступу до них, що є критично необхідним для стабільного функціонування вебсистеми керування процесами видавництва.

Для написання серверної частини системи застосовувався ASP.NET Core Web API [13, 14, 15] – сучасний кросплатформний фреймворк, побудований на платформі .NET та орієнтований на розробку вебсервісів і REST API мовою програмування C#. Вибір цієї технології обумовлений її високою продуктивністю, надійністю та широкими можливостями для створення масштабованих вебзастосунків.

Однією з переваг ASP.NET Core є модульна структура, яка дозволяє легко розширювати функціональність застосунку за допомогою middleware-компонентів. Фреймворк також забезпечує ефективну обробку HTTP-запитів [16], підтримує вбудовані механізми автентифікації, авторизації та захисту даних, а також має розвинену систему впровадження залежностей, що спрощує розробку та тестування програмного забезпечення.

Для взаємодії з базою даних використовується Entity Framework Core [17, 18, 19], який забезпечує об'єктно-реляційне відображення даних та спрощує роботу з PostgreSQL. Завдяки цьому зменшується кількість низькорівневого коду для доступу до даних і підвищується зручність супроводу застосунку. Використання ASP.NET Core Web API дозволило реалізувати надійний серверний застосунок із чітким розділенням відповідальностей, якісною реалізацією бізнес-логіки та ефективною взаємодією з базою даних.

Інтерфейс користувача було вирішено реалізувати за допомогою бібліотеки React та мови програмування TypeScript [20, 21, 22, 23]. Поєднання цих технологій забезпечує високу ефективність розробки та підтримки клієнтської частини застосунку.

Однією з ключових переваг TypeScript є статична типізація, яка підвищує надійність коду, зменшує кількість помилок на етапі розробки та покращує його читабельність. React, у свою чергу, базується на компонентному підході, що дозволяє розділяти інтерфейс на незалежні багаторазово використовувані компоненти, спрощуючи масштабування та супровід застосунку. Крім того, використання віртуального DOM оптимізує процес оновлення інтерфейсу шляхом мінімізації змін у реальному DOM, що позитивно впливає на продуктивність вебзастосунку.

Tailwind CSS було обрано для стилізації користувацького інтерфейсу [24]. Даний CSS-фреймворк дозволяє швидко створювати сучасні та адаптивні інтерфейси за рахунок використання набору утилітарних класів, що спрощує процес розробки та забезпечує узгоджений дизайн усіх компонентів системи.

Для зберігання графічних та текстових матеріалів використовується хмарне сховище Azure Blob Storage [25]. Завдяки цьому зменшується навантаження на сервер та основну базу даних застосунку. Дані у Azure Blob Storage зберігаються з реплікацією між дата-центрами, тому навіть у разі збоїв обладнання передані файли будуть збережені чи відновлені. Хмарне сховище підтримує керування правами доступу, використання SAS-токенів, шифрування даних та захищені канали передачі інформації. При такому підході сервер взаємодіє з хмарним сервісом для завантаження та отримання файлів, тоді як посилання на них, дати створення та інші важливі метадані зберігаються у БД. Додатковою перевагою є легка інтеграція Azure Blob Storage із платформою ASP.NET Core завдяки наявності офіційних бібліотек для роботи з файлами.

Visual Studio було обрано для написання серверної частини системи, оскільки воно забезпечує комфортну реалізацію системи завдяки підтримці платформи .NET та наявності багатьох інструментів для розробки, налагодження, тестування програм на мові C#.

Як середовище розробки клієнтської частини застосунку використовувався Visual Studio Code (VS Code) – зручний та швидкий редактор коду для створення застосунків, який легко розширюється та підтримує велику кількість мов програмування та інструментів.

Висновки до першого розділу

У першому розділі було проаналізовано особливості видавничого процесу та основні потреби його учасників, завдяки чому вдалося коректно та докладно сформулювати поставлену задачу. Її розділено на підзадачі, які висвітлюють всі етапи розробки та забезпечують послідовність виконання роботи.

Аналіз існуючих програмних рішень допоміг оцінити стан та різноманітність застосунків, які використовуються у роботі видавництва. У результаті цього було з'ясовано основні характеристики аналогів, визначено їхні сильні та слабкі сторони. Це підтвердило доцільність розробки запланованої системи та дозволило означити і удосконалити функціональні можливості застосунку, що допомогло йому відповідати усім сучасним вимогам та потребам цільових користувачів, а також підвищило його конкурентоспроможність серед аналогічних рішень.

Було обрано клієнт-серверну архітектуру для побудови вебсистеми, серверну частину якої представлятиме моноліт, реалізований з використанням принципів Onion Architecture, що відповідає сучасним стандартам розробки, вимогам проєкту та забезпечує структурованість і зручну підтримку застосунку. Обрані інструментальні засоби та технології дозволили реалізувати весь необхідний функціонал, забезпечивши продуктивність, масштабованість, зручність супроводу та надійну основу для подальшого розвитку програмного продукту.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБСИСТЕМИ КЕРУВАННЯ ПРОЦЕСАМИ ВИДАВНИЦТВА

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Вебсистема керування процесами видавництва повинна відповідати усім наведеним високорівневим вимогам, щоб забезпечити високу якість кінцевого продукту. Вона повинна оптимізувати взаємодію між працівниками видавництва, автоматизувати та покращити процес видання книг.

Визначено наступні бізнес вимоги: ціль проєкту – він створюється з метою спростити керування процесами видавництва книг та створити зручне середовище для організації роботи над рукописами у видавництві; проєкт буде представлено у вигляді вебзастосунку, що містить необхідний контент, пов'язаний з його тематикою та вимогами користувачів (рис. 2.1).

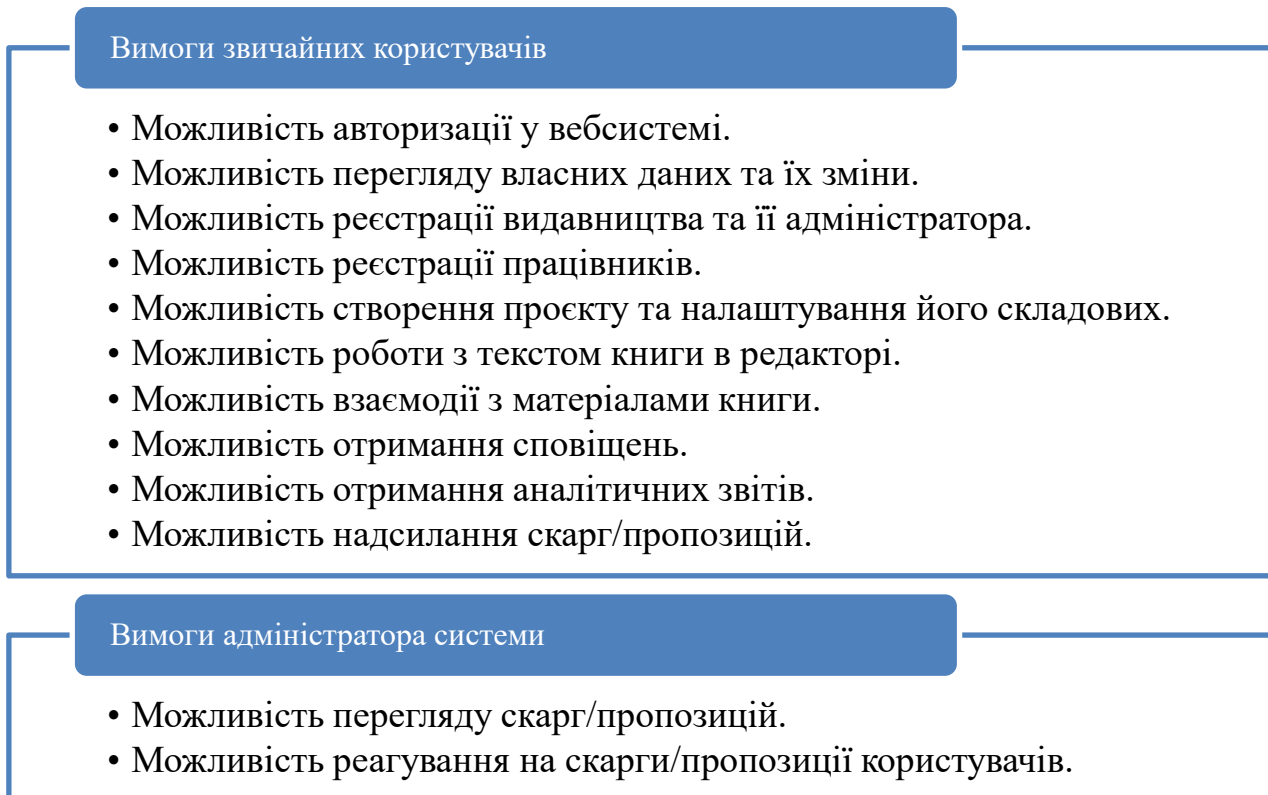


Рис. 2.1. Вимоги користувачів вебсистеми

Об'єктом комп'ютеризації є процес керування видавничою діяльністю, управління його проєктами, яке охоплює планування роботи, виконання завдань і взаємодію з рукописом книги через текстовий редактор.

Функціональні вимоги:

1. У вебсистемі повинна бути можливість реєстрації користувача з присвоєнням йому відповідної ролі (адміністратор видавництва, менеджер, редактор, коректор, художник-ілюстратор, перекладач).

2. Вебсистема повинна зберігати інформацію про видавництво, його проєкти та працівників і надавати користувачу можливості для керування різними даними, залежно від рівня доступу до них.

3. Вебсистема повинна надавати можливості для створення проєктів, налаштування їхніх етапів з дедлайнами, додавання допоміжних матеріалів та модифікацію тексту рукопису у редакторі.

4. У застосунку повинен бути доступ до зручного текстового редактора з необхідними інструментами для редагування книги.

5. Усі користувачі повинні мати особистий кабінет, в якому зберігаються та відображаються їхні дані, а також список проєктів, до яких вони долучені, і їх завдання.

Було визначено та описано нефункціональні вимоги (табл. 2.1), об'єднані за категоріями: сприйняття, надійність, продуктивність, можливість експлуатації.

Таблиця 2.1

Нефункціональні вимоги

Категорія	Вимога	Опис
1.	2.	3.
Сприйняття	Час, необхідний для навчання адміністраторів видавництва	2-3 години
	Час, необхідний для навчання працівників видавництва	1-2 години
	Час відгуку для типових задач	Не більше 20 секунд

продовж. табл. 2.1

1.	2.	3.
	Час відгуку для складних завдань	Не більше 40 секунд
	Зручність інтерфейсу	Інтерфейс повинен бути інтуїтивно зрозумілим та супроводжуватися відповідними підказками
Надійність	Доступність вебсистеми	Час обслуговування вебсистеми, не повинен перевищувати 5% від загального часу роботи
	Середній час безвідмовної роботи	20 робочих днів
	Максимальна норма помилок та дефектів в роботі системи	1 помилка на десять тисяч рядків коду
Продуктивність	Кількість користувачів, які одночасно працюють із вебсистемою	Мінімум 100 користувачів
Можливість експлуатації	Масштабованість	Вебсистема повинна підтримувати збільшення потужності при збільшенні користувачів без суттєвого погіршення ефективності роботи.

Аналіз вимог користувачів дав підстави для визначення варіантів використання вебсистеми керування процесами видавництва. Для цього розроблено та наведено діаграму її використання (рис. А.1).

У даній вебсистемі розрізняють наступних акторів (табл. 2.2): адміністратор системи, адміністратор видавництва, менеджер, редактор, коректор, художник-ілюстратор, перекладач.

Таблиця 2.2

Виявлення акторів	
Актор	Короткий опис
Адміністратор системи	Слідкує за дотриманням правил вебсистеми, підтримує стабільну роботу, реагує на скарги та пропозиції користувачів.
Адміністратор видавництва	Реєструє видавництво та його робітників з відповідними ролями, переглядає список усіх проєктів та працівників, створює проєкти та призначає на них працівників, переглядає аналітику про роботу видавництва.
Менеджер	Надає завдання робітникам видавництва, контролює терміни їх виконання, етапи роботи над проєктом.
Перекладач	Має доступ до оригінального тексту книги, виконує його переклад на обрану мову.
Редактор	Виконує літературне редагування тексту книги, перевіряє її структуру, сюжет, логіку викладу.
Художник-ілюстратор	Переглядає матеріали книги та її текст для натхнення та створення візуальних елементів книги відповідного стилю та атмосфери. Додає створені ілюстрації та обкладинку книги до матеріалів проєкту.
Коректор	Виконує коректуру книги, виправляє орфографічні, пунктуаційні та технічні помилки.

Виявлені варіанти використання системи керування процесами видавництва зведені в табл. 2.3.

Таблиця 2.3

Виявлення варіантів використання

Основний актор	Найменування
1.	2.
Користувач	Авторизація
	Перегляд особистого профілю
	Редагування особистих даних
	Відновлення паролю
Адміністратор видавництва	Реєстрація видавництва
	Реєстрація співробітників
	Перегляд списку співробітників видавництва
	Перегляд списку проєктів
	Створення проєкту
	Призначення працівника на проєкт
	Отримання звітів про роботу видавництва
Адміністратор видавництва, Менеджер	Модифікація проєкту
	Налаштування етапів роботи над книгою
	Надання завдань працівникам видавництва
Менеджер, Редактор, Коректор, Художник-ілюстратор, Перекладач	Перегляд проєкту та його вмісту
	Зміна статусу завдання
	Пошук, фільтрація та сортування завдань
	Робота з текстом книги в редакторі
	Перегляд та відновлення версій тексту книги
	Експорт та імпорт тексту книги
	Перегляд, додавання та редагування матеріалів книги
	Завантаження матеріалів
	Отримання сповіщень про дедлайни завдань, призначення на проєкт тощо
	Надсилання скарг/пропозицій

1.	2.
Адміністратор системи	Перегляд скарг/пропозицій
	Реагування на скарги/пропозиції користувачів

Після визначення та опису основних варіантів використання вебсистеми можна перейти до опису об'єктно-орієнтованої структури вебзастосунку. Як зазначалося в пункті 1.3, для побудови вебсистеми використовувалася клієнт-серверна архітектура, сервер в якій представлений у вигляді моноліту з використанням принципів Onion Architecture.

На рівні Domain (рис. 2.2) представлені усі основні сутності системи, перерахування, моделі, які не залежать від інших шарів. Бізнес-логіка вебсистеми винесена до рівня Application, що дозволяє спростити доменний шар та забезпечити чітке розділення відповідальностей.

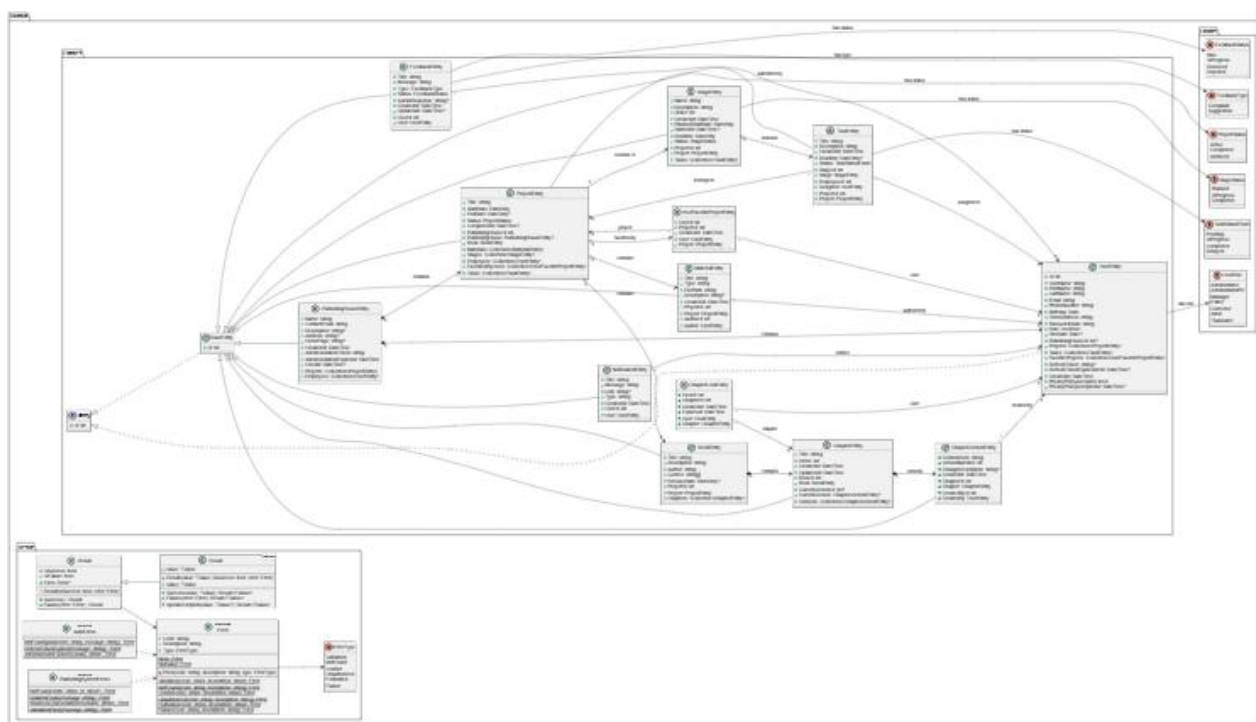


Рис. 2.2. Діаграма класів рівня Domain

Для вебсистеми керування процесами видавництва визначено 14 класів-сутностей та 6 перерахувань (табл. 2.4), які використовуються для зберігання та обробки даних предметної області.

Характеристика сутностей та перерахувань

Назва	Тип	Призначення
1.	2.	3.
BaseEntity	Клас	Батьківський клас для усіх сутностей з ідентифікатором
UserEntity		Відповідає за різні типи користувачів системи і, відповідно, за їхні можливості
PublishingHouseEntity		Представляє видавництво книг, яке має своїх працівників та проєкти
ProjectEntity		Проект, створений адміністратором видавництва, який є контейнером та зберігає інформацію про книгу, матеріали до неї, етапи роботи та призначених працівників
BookEntity		Клас книги з її інформацією
ChapterEntity		Представляє розділ книги з його назвою та порядковим номером
ChapterVersionEntity		Клас для версій контенту розділу
ChapterLockEntity		Використовується для блокування можливості редагування тексту розділу
MaterialEntity		Додатковий матеріал книги, наприклад, зображення, нотатки
StageEntity		Етапи роботи над книгою з відповідними дедлайнами, призначеними працівниками та завданнями
TaskEntity		Завдання, які виконують працівники
UserFavoriteProjectEntity		Обрані проєкти користувача, на які він призначений
NotificationEntity		Клас для представлення сповіщень користувачів
FeedbackEntity		Звернення користувачів до адміністратора системи
UserRole	Перерахування	Визначає тип ролі користувача у системі та у видавництві

1.	2.	3.
ProjectStatus	Перерахування	Визначає статус проекту («активний», «завершений», «заархівований»)
StageStatus		Визначає статус етапу роботи («заплановано», «в процесі», «завершено»)
TaskStatus		Визначає статус виконання завдання працівником («очікування», «у роботі», «завершено», «затримано»)
FeedbackType		Визначає тип звернення: скарга чи пропозиція
FeedbackStatus		Визначає статус опрацювання звернення («нове», «у роботі», «вирішене», «відхилене»)

Окрім сутностей, у доменному шарі використовуються допоміжні класи для представлення результату операції (успіх/помилка), які реалізують патерн Result: класи Result, Result<TValue>, Error, AuthErrors, PublishingSystemErrors та перерахування ErrorType.

Рівень Application відповідає за реалізацію прикладної логіки вебсистеми. У ньому зосереджено обробку запитів, виконання бізнес-операцій та керування потоком даних.

Перш за все варто вказати, що у шарі Application (рис. А.2) визначено набір інтерфейсів, що описують контракти взаємодії з іншими частинами системи, зокрема інфраструктурним рівнем. Це дозволяє реалізувати принцип інверсії залежностей та забезпечити гнучкість архітектури.

Основна бізнес-логіка реалізована у сервісах прикладного рівня, які названі відповідно до сутностей з якими взаємодіють. У них виконується обробка вхідних даних, отримання даних від репозиторіїв у вигляді доменних сутностей та їх перетворення у об'єкти передачі даних (DTO), що використовуються у рівні API. Для обробки результатів виконання операцій використовуються класи реалізованого патерну Result у рівні Domain.

Рівень Infrastructure (рис. А.3) відповідає за реалізацію технічних деталей вебсистеми та взаємодію із зовнішніми ресурсами, такими як база даних, сервіси автентифікації та електронної пошти.

Для взаємодії з базою даних використовуються класи репозиторії, які реалізують інтерфейси, визначені на рівні Application, та інкапсулюють логіку доступу до даних. Вони названі відповідно до сутностей, які отримують з бази даних. Також у цьому шарі є контекст бази даних PublishingDbContext та конфігурації таблиць у БД.

Клас JwtTokenProcessor відповідає за генерацію та обробку JWT-токенів і використовується у AuthService, який забезпечує автентифікацію, авторизацію користувачів та інші операції з їх обліковими записами.

Для відправлення листів та формування їх вмісту створені класи SmtpEmailSender, AzureEmailSender і EmailTemplateProvider. Також на рівні Infrastructure наявні класи EmailSettings, AzureEmailSettings, FrontendConfig та JwtOptions, які використовуються відповідно до патерну Options для керування конфігураційними параметрами вебсистеми.

Клас ExportService використовується для експорту тексту розділу у формат docx, імпорт з нього та з Google Documents. IcalCalendarService формує файл типу ics для імпорту завдання в календар. AzureBlobStorageService використовується для роботи з віддаленим сховищем, а саме для завантаження, отримання та видалення файлів матеріалів. TaskDeadlineReminderJob призначений для фонові перевірки термінів виконання завдань та автоматичного надсилання користувачам нагадувань про наближення дедлайну.

На цьому рівні також є кастомні провайдери, які використовуються для визначення тривалості токенів різного типу та отримання ідентифікатора користувача під час SignalR-з'єднання.

Рівень API (Presentation layer) є зовнішнім шаром вебсистеми, який відповідає за обробку HTTP-запитів, отриманих від клієнтських застосунків, та повернення відповідей. У ньому реалізовані наступні контролери

(рис. А.4): AnalyticsController, AuthController, BookController, CalendarController, ChapterController, ChapterLockController, ExportController, FeedbackController, MaterialController, NotificationController, ProjectController, PublishingHouseController, StageController, TaskController, UserController, UserFavoriteProjectController. Тут розміщені розширення для обробки та повернення проблемних HTTP-відповідей, атрибут для перевірки авторизації користувача, шаблони повідомлень електронної пошти, NotificationHub – hub-клас для надсилення та отримання сповіщень у реальному часі, класи NotificationService і NotificationSender, які відповідають за отримання, видалення, надсилення та формування сповіщень.

2.2 Розробка бази даних

Для зберігання усієї інформації вебсистеми використовується СУБД PostgreSQL. База даних вебзастосунку складається з 21 таблиці, між якими встановлено необхідні зв'язки для забезпечення цілісності та узгодженості даних (рис. 2.3).

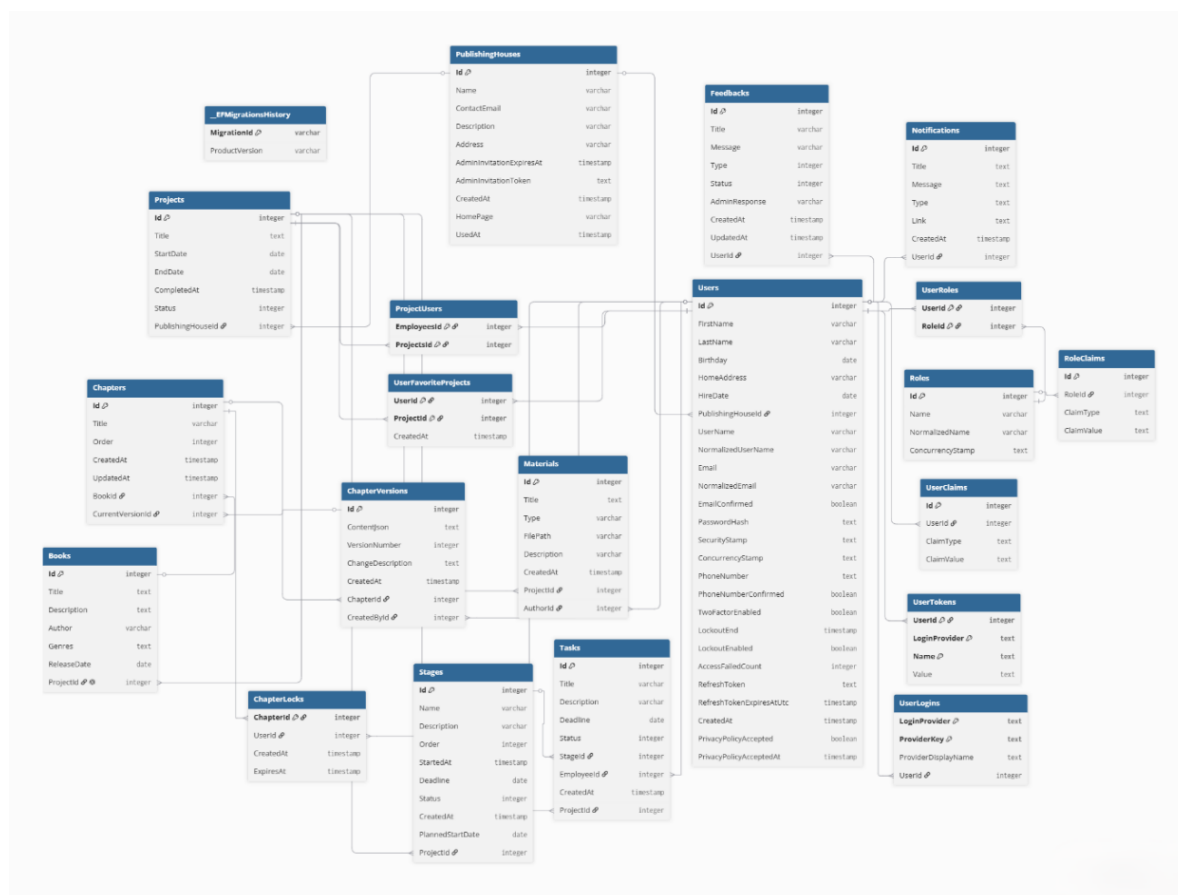


Рис. 2.3. Структурна схема БД

Оскільки для реалізації механізму автентифікації використовується ASP.NET Identity [26], то автоматично створюється набір службових таблиць у базі даних. Вони використовуються для забезпечення функціоналу керування користувачами, ролями та токенами доступу.

Опишемо основні таблиці бази даних.

Таблиця «PublishingHouses» призначена для зберігання даних про видавництва: його назва, електронна пошта для контакту, опис, фізична адреса, посилання на сайт видавництва, дата створення та токен, який використовується лише одного разу для реєстрації адміністратора. Видавництва є основною сутністю вебсистеми, оскільки саме з ними та їх елементами відбуваються всі операції.

Таблиця «Projects» містить дані про проекти видавництва, має дати початку, запланованого та фактичного кінця роботи і статус проекту: активний, завершений чи архівований. Дана таблиця має зовнішній ключ PublishingHouseId, який встановлює зв'язок із таблицею «PublishingHouses». «Projects» містить базову інформацію проєкту, а пов'язані таблиці, такі як Books, Materials та інші, посилаються на неї через зовнішні ключі, формуючи відношення типу «один-до-багатьох» або «один-до-одного». Це дозволяє структурувати дані навколо одного логічного об'єкта та забезпечує їх узгодженість.

Таблиця «Books» зберігає інформацію про книгу, над якою виконується робота в проєкті. Дані про неї включають: назву, анотацію, автора, жанри, запланована дата публікації. Дана таблиця має зовнішній ключ ProjectId, який встановлює зв'язок із таблицею «Projects», формуючи відношення типу «один-до-одного».

Таблиця «Chapters» призначена для зберігання розділів книги, їх назв, порядкових номерів в книгах, дат створення та оновлення. Вона має зовнішній ключ BookId для зв'язку з «Books», відношення між ними є «один-до-багатьох». Також є поле CurrentVersionId для встановлення поточної версії розділу, всі версії якого зберігаються у таблиці «ChapterVersions».

Вона має наступні поля: контент розділу у форматі JSON, номер версії, опис змін та дату створення. На «Chapters» і на «Users» посиляються поля ChapterId і CreatedById відповідно, забезпечуючи зв'язок «один-до-багатьох».

Таблиця «ChapterLocks» використовується для перевірки блокувань розділів, щоб забороняти користувачам редагувати текст, якщо він уже редагується іншим користувачем. Для цього в ній зберігаються ідентифікатори користувача та розділу, час створення блокування та час його закінчення.

Таблиця «Materials» втілює в собі допоміжні матеріали до проєкту, наприклад, нотатки, зображення тощо. Її полями є: назва, дата створення, тип матеріалів (текстовий, графічний), посилання на віддалене сховище, в якому зберігається файл, та опис – що це за файл та як він використовується у проєкті. Ця таблиця має зовнішні ключі ProjectId для зв'язку з «Projects» і AuthorId – для «Users», ці відношення є типу «один-до-багатьох».

Таблиця «Stages» призначена для зберігання даних про етапи роботи над проєктом. Для цього в ній створені наступні поля: назва та короткий опис етапу, в якому вказується які роботи планується виконувати під час тривання цього етапу, порядок етапу (яким він є за номером у списку етапів даного проєкту), дати створення, запланованого та фактичного початку і попередньо визначений термін його закінчення. Також кожен етап має один з визначених статусів: Planned, InProgress та Completed. Таблиця «Stages» має зв'язок з «Projects» через зовнішній ключ ProjectId, відношення між ними є типу «один-до-багатьох».

Таблиця «Tasks» містить дані про завдання користувачів, які призначаються їм на етапах проєктів. Вона містить поля: назву, опис, дату створення, кінцевий термін виконання завдання та статус, який може бути одним з: Pending, InProgress, Completed та Delayed. Таблиця «Tasks» має зв'язок з «Stages» через зовнішній ключ StageId і з таблицею «Users» через EmployeeId, який вказує ідентифікатор користувача-виконавця, обидва відношення є типу «один-до-багатьох». Для спрощення виконання запитів та

оптимізації роботи вебсистеми було додано зовнішній ключ ProjectId для зв'язку з таблицею «Projects».

Таблиця «Users» є однією з найбільших в базі даних, вона містить важливі дані користувачів. Дана таблиця базується на стандартній моделі користувача ASP.NET Identity та містить як типові поля (логін, хешований пароль, email тощо), так і додаткові поля, розроблені відповідно до потреб вебсистеми: прізвище та ім'я користувача, день народження, домашня адреса, дата найму, видавництво, у якому працює користувач (якщо це не адміністратор системи), токен для відновлення доступу та поле для фіксації факту згоди користувача з політикою конфіденційності. Тому таблиця має зовнішній ключ PublishingHouseId, щоб утворити зв'язок із таблицею «PublishingHouses».

Таблиця «Notifications» відповідає за зберігання сповіщень, які отримує користувач при роботі з вебсистемою. Вона містить назву та вміст сповіщення, його тип, посилання на ресурс, дату створення та зовнішній ключ UserId для зв'язку з таблицею «Users».

Між таблицями «Projects» і «Users» існує два відношення типу «багато-до-багатьох». Оскільки на один проєкт може бути призначено багато працівників, і у одного працівника може бути декілька проєктів, для реалізації даного зв'язку створено таблицю «ProjectUsers», яка має комбінований первинний ключ, який складається з ідентифікаторів відповідних таблиць. Інша таблиця «UserFavoriteProjects» створена для зберігання обраних проєктів користувачів, на які вони призначені. Вона має аналогічний первинний ключ для забезпечення унікальності комбінації ідентифікаторів.

Для зберігання звернень користувачів призначена таблиця «Feedbacks». Вона має наступні поля: заголовок та повідомлення звернення, його тип, статус і автор, дати створення та оновлення, а також відповідь адміністратора вебсистеми після опрацювання. Для зберігання авторства має зв'язок з таблицею «Users» через зовнішній ключ UserId.

Серед згенерованих таблиць ASP.NET Identity варто звернути увагу на «Roles» і «UserRoles», оскільки вони зберігають усі ролі вебсистеми та зв'язки користувачів з ними відповідно.

2.3 Проєктування та реалізація алгоритмів роботи системи

На етапі розробки вебсистеми було спроектовано та реалізовано алгоритми, що забезпечують виконання основних бізнес-процесів видавництва. До них належать механізми реєстрації та авторизації користувачів, керування видавництвами та проєктами, робота з книгами, матеріалами, завданнями, сповіщеннями та іншими компонентами системи. Розглянемо детальніше алгоритми реалізації деяких ключових функціональних можливостей вебзастосунку.

Для роботи у вебсистемі користувачі повинні бути зареєстровані та мати відповідні права доступу. Особливістю розробленої системи є централізований підхід до створення облікових записів працівників видавництва. Самостійно зареєструватися у вебсистемі може лише адміністратор під час створення нового видавництва. Після успішної реєстрації він отримує можливість налаштовувати робоче середовище та керувати складом працівників. Їх він додає через відповідний функціонал системи, вказуючи основні дані та роль працівника. Такий підхід дозволяє контролювати процес створення облікових записів та запобігати несанкціонованому доступу до даних зареєстрованих видавництв.

Для початку розглянемо процес реєстрації видавництва та його адміністратора (рис. А.5). Неавторизований користувач заповнює форму створення видавництва, вказуючи необхідну інформацію про організацію та контактну електронну адресу. Після надсилання форми сервер виконує перевірку введених даних на коректність та відсутність конфліктів із вже існуючими записами. Якщо перевірка пройшла успішно, інформація про видавництво зберігається у базі даних, а на вказану електронну адресу надсилається лист із подальшими інструкціями.

У повідомленні міститься посилання на сторінку реєстрації адміністратора видавництва. Для ідентифікації створеного видавництва та забезпечення захищеного завершення процесу реєстрації використовується унікальний токен, який генерується вебсистемою автоматично. Після переходу за посиланням користувач заповнює форму адміністратора, вказуючи особисті дані та пароль для входу в систему.

Після надсилання форми сервер повторно перевіряє коректність отриманих даних, створює обліковий запис адміністратора та пов'язує його зі створеним видавництвом. У разі успішного завершення операції користувач отримує можливість авторизуватися у вебсистемі та розпочати її налаштування. Для запобігання накопиченню неактивних записів реалізовано механізм автоматичного очищення даних: якщо протягом 24 годин після створення видавництва його адміністратора не було зареєстровано, запис про видавництво автоматично видаляється із системи.

Далі опишемо процес реєстрації працівників видавництва, зображений на відповідній діаграмі (рис. 2.4). Вони можуть мати наступні ролі у вебсистемі: Manager, Editor, Corrector, Artist, Translator. Для створення їхніх акаунтів адміністратору потрібно відкрити сторінку «Працівники», обрати варіант «Зареєструвати працівника» та коректно заповнити форму. Під час заповнення форми необхідно вказати основні дані користувача, його електронну адресу та роль у системі. Після надсилання форми вебсистема перевіряє введені дані на коректність і відповідність встановленим правилам валідації.

Якщо перевірка завершується успішно, інформація про нового працівника зберігається у базі даних, а для нього створюється обліковий запис у неактивному стані. Одночасно на вказану електронну адресу надсилається повідомлення про створення акаунту із посиланням для завершення реєстрації та встановлення власного пароля. Для підтвердження особи користувача та захисту від несанкціонованого доступу до облікового

запису використовується унікальний токен, який генерується сервером та додається до посилання.

Після переходу за отриманим посиланням працівник задає пароль до свого облікового запису. Система перевіряє дійсність токена, коректність введеного пароля та активує акаунт користувача. Після завершення цього процесу працівник може увійти в свій обліковий запис.

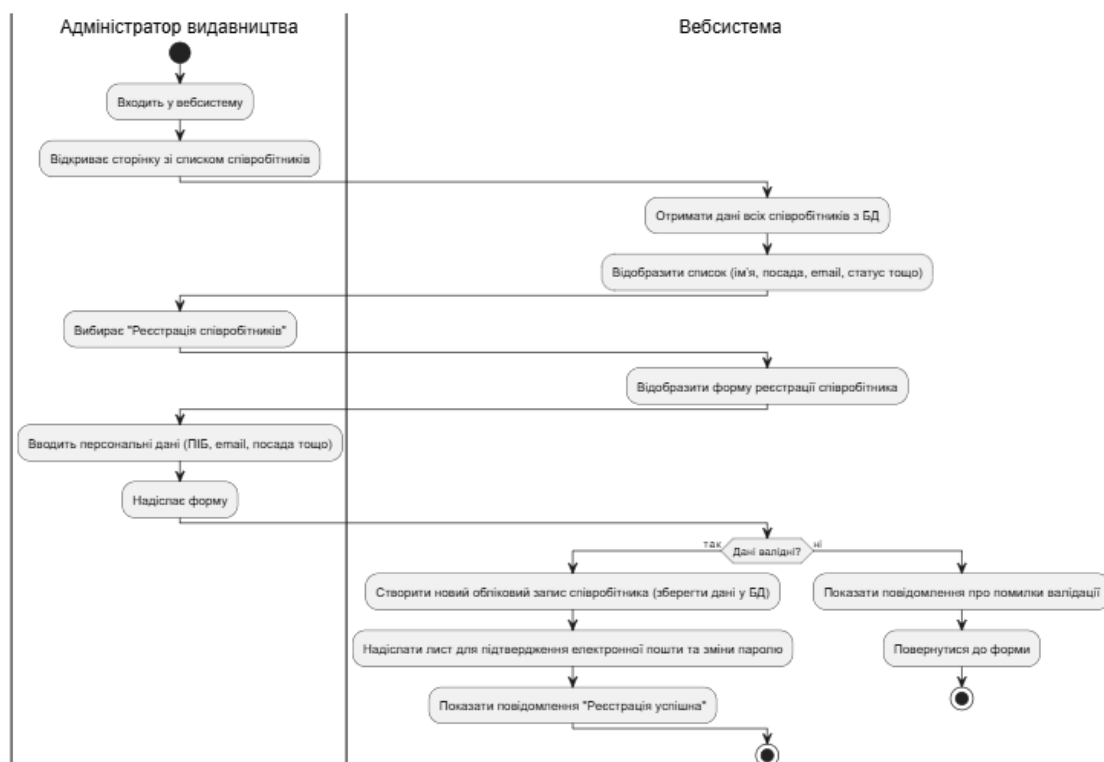


Рис. 2.4. Діаграма активності для реєстрації працівників видавництва

Після успішної авторизації у вебсистемі користувач отримує доступ до функціональних можливостей, визначених його роллю та наданими правами. Такий підхід забезпечує розмежування повноважень між різними категоріями користувачів і дозволяє обмежити доступ до функцій та даних відповідно до їхніх посадових обов'язків.

Опишемо процес імпорту вмісту розділу книги з Google Docs (рис. 2.5). Користувач вибирає варіант «Імпортувати з Google Docs». Клієнтська частина використовує авторизацію за допомогою Google, отримує accessToken, після чого відкриває вікно для вибору файлів. Далі користувач може обрати необхідний документ серед запропонованих та підтвердити операцію. Сервер вебсистеми отримує запит на імпорт, звертається до Google

Drive API для отримання вмісту документа у вигляді HTML та при успішній операції передає його клієнту. Він очищає його та вставляє в текстовий редактор. Користувач зберігає розділ.

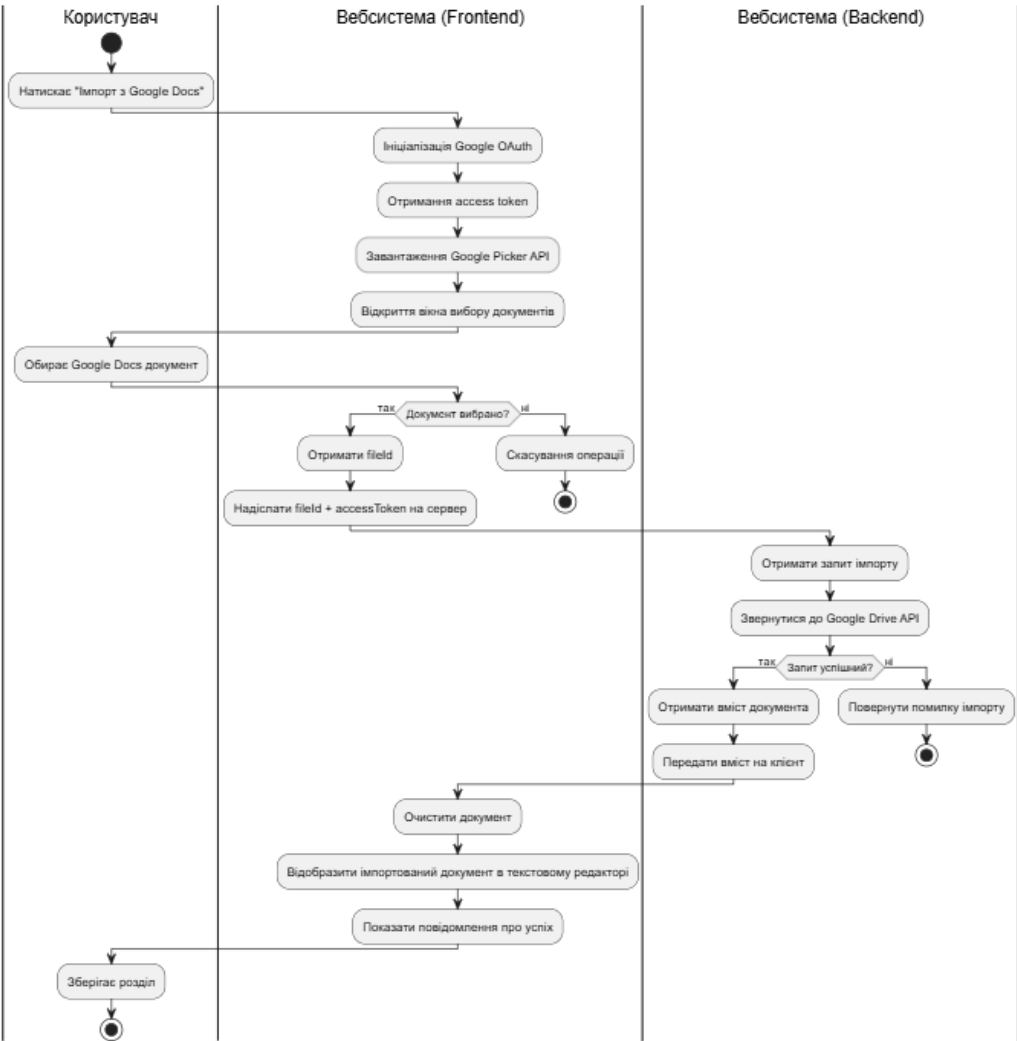


Рис. 2.5. Діаграма активності для імпорту документа з Google Docs

2.4 Реалізація вебсистеми керування процесами видавництва

Для розмежування доступу до ресурсів системи було вирішено впровадити RBAC модель контролю доступу у поєднанні з контекстними перевірками та перевірками належності ресурсу користувачеві [27]. Для реалізації RBAC частини було визначено основні ролі системи та правила їх присвоєння у ній.

Для доступу до API ендпоінтів застосовувався атрибут [Authorize(Roles = "...")], в якому зазначався перелік ролей, для яких доступна дана можливість. Тим не менш, для повноцінного захисту від

несанкціонованого доступу необхідно впровадити додаткові перевірки. З цією метою було створено сервіс AccessService, який містить набір методів для формування предикатів доступу до сутностей системи, які використовуються у сервісах системи та передаються у методи репозиторіїв для їх застосування у запитах до бази даних.

Для більшості сутностей розроблено метод, який повертає вираз типу Expression<Func<TEntity, bool>>, призначений для використання під час формування LINQ-запитів у репозиторіях [28]. Такий підхід дозволяє відділити бізнес-логіку від логіки отримання даних, виконувати перевірки доступу до них та фільтрацію даних безпосередньо на рівні БД.

Приклад реалізації предиката доступу для проєктів:

```
public Expression<Func<ProjectEntity, bool>>
AddProjectAccessFilter()
{
    var user = currentUser;
    Expression<Func<ProjectEntity, bool>> accessPredicate =
        user.Role switch
        {
            nameof(UserRole.Administrator) =>
                p => true,

            nameof(UserRole.AdministratorPH) =>
                p => p.PublishingHouseId ==
                    user.PublishingHouseId,

            _ => p => p.Employees.Any(e => e.Id == user.UserId)
        };

    return accessPredicate;
}
```

Приклад його використання у сервісі ProjectService:

```
Expression<Func<ProjectEntity, bool>> predicate =
accessService.AddProjectAccessFilter();
var project = await projectRepository.GetByIdAsync(id, predicate);
```

Таким чином, якщо користувач не має доступу до відповідного проєкту, запис не буде отримано з бази даних, а у відповідь повернеться повідомлення про відсутність такого проєкту, що також допомагає приховати недоступні записи.

Описаний підхід має низку переваг: централізація логіки перевірки доступу; повторне використання предикатів у різних частинах системи; оптимізація запитів до бази даних; зменшення навантаження на пам'ять застосунку.

Одним з ключових компонентів вебсистеми є текстовий редактор, призначений для роботи з розділами книги. Він забезпечує зручну роботу з контентом розділів, форматування тексту, версійність документів та блокування редагування.

На клієнтській стороні для реалізації редактора було використано бібліотеку TipTap, яка побудована на основі ProseMirror та дозволяє власноруч налаштовувати необхідну функціональність завдяки застосуванню комбінації різноманітних розширень [29]. Завдяки цьому було втілено підтримку заголовків, списків, цитат, стилізації тексту та інших елементів форматування.

Лістинг налаштування редактора:

```
const editor = useEditor({
  editable: isEditable,
  extensions: [
    StarterKit,
    Placeholder.configure({
      placeholder,
    }),
    Highlight,
    TextAlign.configure({
      types: ['heading', 'paragraph'],
    }),
    HorizontalRule,
    Subscript,
    Superscript,
    Typography,
    CharacterCount,
  ],
  content: parseEditorContent(content),
  onUpdate: ({ editor }) => {
    onChange(JSON.stringify(editor.getJSON()));
  },
  immediatelyRender: false,
});
```

TipTap редактор підтримує зберігання розділів у двох форматах: HTML та JSON. На серверній частині контент зберігається саме в останньому форматі, що дозволяє фіксувати структуру тексту для його подальшого отримання та відображення у редакторі.

Важливою частиною в реалізації текстового редактора є механізм зберігання, перегляду та відновлення попередніх версій розділу. При кожному новому збереженні тексту через редактор автоматично створюється нова версія стану документа, посилання на яку додається до запису розділу у БД. Кожна версія містить текст розділу, номер версії, дату створення та інформацію про користувача, який виконав зміни. Це дозволяє отримувати поточну версію документа, відстежувати історію його редагування, а у випадку виникнення помилок чи здійснення небажаних змін відновити попередній вміст розділу.

Список усіх версій виводиться в бічній панелі, для вибору доступні перегляд версії та її відновлення. При перегляді вмісту отримується JSON версії та передається у редактор для відображення у режимі «Лише для читання».

У компоненті ChapterEditor, який являє собою простір редактора з усіма його налаштуваннями, відслідковується отримання запиту на перегляд версії, тобто чи є редактор у режимі «Читання»:

```
const [previewContent, setPreviewContent] = useState<string | null>(null);
const isPreviewMode = previewContent !== null;
```

Ця змінна передається у компонент TiptapEditor і впливає на його конфігурацію.

Лістинг використання компоненту TiptapEditor:

```
<TiptapEditor
  chapterId={selectedChapterId}
  content={editorContentValue}
  onChange={handleContentChange}
  placeholder={`Почніть редагувати розділ
"${chapterQuery.data?.title}"...`}
  isReadOnly={isPreviewMode}
  onLockStateChange={setIsLockedByOtherUser}
/>
```

Крім вхідного параметра `isReadOnly` на режим редактора ще впливає змінна `isLockEditable`, яка призначена для перевірки наявності блокування:

```
const isEditable = Boolean(isLockEditable && !isReadOnly);
```

Для відновлення при натисканні на відповідну кнопку на сервер передається ідентифікатор версії, який у сервісі призначається у поле для поточного стану розділу і зберігається в базі даних.

Лістинг коду відновлення попереднього стану документа:

```
public async Task<Result> RestoreVersionAsync(int chapterId, int
versionId)
{
    Expression<Func<ChapterEntity, bool>> predicate =
    accessService.AddChapterAccessFilter();
    var chapter = await chapterRepository.GetByIdAsync(chapterId,
    predicate);
    if (chapter == null)
    {
        return Result.Failure(PublishingSystemErrors.NotFound(
        "Chapter", chapterId));
    }

    var versionExists = await
    chapterVersionRepository.ExistsAsync(versionId);
    if (!versionExists)
    {
        return Result.Failure(PublishingSystemErrors.NotFound(
        "ChapterVersion", versionId));
    }

    chapter.CurrentVersionId = versionId;
    chapter.UpdatedAt = DateTime.UtcNow;
    await chapterRepository.UpdateAsync(chapter);
    return Result.Success();
}
```

Іншим, не менш важливим доповненням до функціоналу редактора є механізм блокування редагування розділів. Він необхідний для запобігання виникненню конфліктів змін та втрати даних під час одночасної роботи декількох користувачів із одним документом.

Після відкриття тексту у редакторі система перевіряє стан розділу, тобто чи не використовується він вже іншим працівником видавництва. Якщо розділ заблоковано, то виводиться спеціальне повідомлення про це, а його контент стає доступним лише для читання.

Для зберігання інформації про блокування використовується окрема сутність ChapterLockEntity. Сервіс для роботи з нею має два методи: AcquireLockAsync та ReleaseLockAsync. Саме перший метод відповідає за встановлення блокування, його продовження та повернення у відповідь на запит іншого користувача.

На клієнтській стороні було розроблено власний хук для отримання стану блокування розділу useChapterLock. Після відкриття сторінки редагування хук автоматично надсилає запит на сервер для встановлення блокування або отримання даних про нього. Через кожний визначений період часу запит повторюється для поновлення інформації про поточний стан розділу. У середині хука використовується useEffect, який виконує запит на блокування під час монтування компонента та звільнення ресурсу після виходу користувача з редактора.

Застосунок має інтеграцію з Google Docs. Крім імпорту чи експорту в звичайний файл .docx формату, у користувачів є можливість обрати імпорт необхідного документу зі свого акаунту в Google документах. Для цього здійснюється авторизація користувача за допомогою OAuth 2.0 із використанням хука useGoogleLogin з бібліотеки @react-oauth/google. Після авторизації застосунок отримує токен доступу до API Google.

Для вибору документа відкривається інтерфейс вибору файлів Google Drive та Google Picker API [30, 31]. Вибір файлу здійснюється через компонент DocsView з фільтрацією за відповідним MIME-типом (application/vnd.google-apps.document).

Лістинг налаштування вибору файлу:

```
const picker = new window.google.picker.PickerBuilder()
  .addView(
    new window.google.picker.DocsView(
      window.google.picker.ViewId.DOCS)
    .setMimeTypes('application/vnd.google-apps.document'),
  )
  .setOAuthToken(accessToken)
  .setDeveloperKey(apiKey)
  .setCallback((data: GooglePickerSelection) => {
    if (data.action === window.google.picker.Action.PICKED) {
```

```

        const file = data.docs?.[0];
        if (!file?.id) {
            toast.error('Не вдалося отримати файл Google Docs');
            return;
        }

        void handlePick({
            fileId: file.id,
            accessToken,
        });
    }
})
.build();

```

picker.setVisible(true);

Серверній частині передається ідентифікатор вибраного файлу разом з токеном доступу. Він виконує запит до Google Drive API для отримання вмісту документа, після чого передає відповідь клієнту.

Висновки до другого розділу

У даному розділі було проаналізовано усі вимоги користувачів, нефункціональні і функціональні вимоги та з їх урахуванням визначено варіанти використання вебсистеми. Встановлено необхідність у реалізації семи ролей, розділення яких зумовлене бізнес-вимогами.

Було спроектовано об'єктно-орієнтовану структури застосунку з детальним поясненням усіх об'єктів та зв'язків між ними, які демонструються на відповідних діаграмах.

Реалізована база даних містить 21 таблицю. У ній наявні зв'язки типів «один-до-одного», «один-до-багатьох» і «багато-до-багатьох».

Описано деякі алгоритми роботи з вебсистемою, продемонстровано їх на діаграмах. Деталізовано опис реалізації окремих функцій та наведено програмний код. Серед них є механізм авторизації, конфігурація текстового редактора, версійність та блокування розділів, імпорт з Google Docs.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБСИСТЕМОЮ КЕРУВАННЯ ПРОЦЕСАМИ ВИДАВНИЦТВА

3.1 Порядок встановлення та налаштування параметрів вебсистеми

Розроблена вебсистема розгортається з використанням хмарних сервісів Microsoft Azure та платформи Vercel. Завдяки цьому вебзастосунок доступний для користувачів через мережу Інтернет з будь-яких пристроїв та різних браузерів.

Архітектура розгортання вебсистеми (рис. 3.1) складається з наступних компонентів: Azure Database for PostgreSQL Flexible Server для бази даних, Azure App Service – серверної частини, Vercel – клієнтської, Azure Blob Storage використовується як файлове сховище, а Email Communication Service у складі Azure Communication Services для відправлення електронних листів.

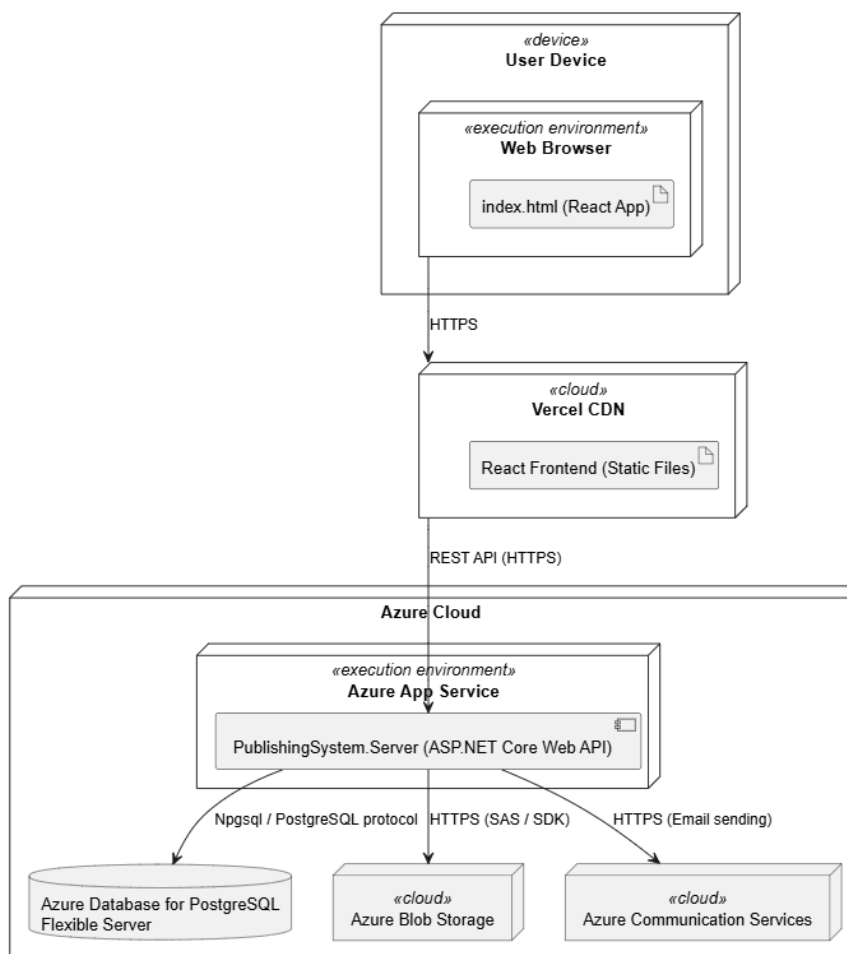


Рис. 3.1. Діаграма розгортання вебсистеми

Перед розгортанням вебсистеми необхідно створити та налаштувати відповідні ресурси у хмарному середовищі Microsoft Azure. Це включає розгортання бази даних, серверної частини, файлового сховища та сервісів для надсилання електронних повідомлень, а також конфігурацію взаємодії між ними.

Коректне налаштування хмарної інфраструктури є важливим етапом, оскільки забезпечує стабільну роботу вебзастосунку, безпечний обмін даними між компонентами системи та доступність сервісу для кінцевих користувачів. Додатково виконуються параметри підключення клієнтської та серверної частин, налаштування змінних середовища та прав доступу до ресурсів.

При створенні та налаштуванні Azure App Service необхідно додати змінні середовища, які використовуються серверною частиною вебсистеми. Для під'єднання до бази даних відповідна змінна створюється на вкладці «Connection strings» із зазначенням типу PostgreSQL.

Важливо зазначити, що під час виконання застосунку рядки підключення доступні як змінні середовища з відповідними префіксами залежно від їх типу. Тому для PostgreSQL рядок підключення з назвою «DefaultConnection» під час виконання буде доступний як «POSTGRESQLCONNSTR_DefaultConnection». Після публікації застосунку обов'язковим є виконання міграції бази даних для створення необхідної структури таблиць у розгорнутій PostgreSQL базі даних.

Додатково у вкладці «App settings» необхідно налаштувати параметри автентифікації та генерації JWT-токенів, а також додати рядки підключення до Azure Blob Storage і Azure Communication Services. Це забезпечує коректну роботу механізмів зберігання файлів, надсилання електронних повідомлень та авторизації користувачів у вебсистемі.

Клієнтська частина вебсистеми розгортається на платформі Vercel, що забезпечує швидку доставку статичних ресурсів і стабільну роботу користувацького інтерфейсу. Після розгортання необхідно виконати її

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

конфігурацію шляхом задання основних параметрів середовища виконання. Зокрема, визначаються адреса API серверної частини, отримана після налаштування Azure App Service, ідентифікатор клієнта для автентифікації через Google, API-ключ для використання Google Picker API, а також змінна, що містить шлях до SignalR Hub на бекенді. Ці параметри забезпечують коректну взаємодію клієнтської частини із зовнішніми сервісами та серверною логікою застосунку.

Завершальним етапом налаштування архітектурної взаємодії є конфігурація політики CORS на стороні сервера. Оскільки після розгортання клієнтська та серверна частини вебсистеми часто функціонують на різних доменах, піддоменах або портах, сучасні веббраузери за замовчуванням блокуватимуть крос-доменні запити відповідно до політики єдиного походження. Належне налаштування заголовків CORS дозволяє серверу чітко визначити перелік довірених джерел, яким дозволено надсилати запити та отримувати ресурси. Це забезпечує не лише стабільний обмін даними між фронтом і бекендом, а й захищає вебсистему від несанкціонованих міжсайтових запитів із сторонніх зловмисних ресурсів.

Після публікації всіх компонентів вебсистеми вона стає доступною користувачам через веббраузер за відповідною адресою, отриманою при розгортанні клієнтської частини. Використання хмарних сервісів для розгортання забезпечує стабільну роботу застосунку та його доступність з різних пристроїв без необхідності встановлення додаткового програмного забезпечення.

Застосування хмарної інфраструктури також дозволяє автоматично масштабувати обчислювальні ресурси залежно від навантаження, що є особливо важливим при збільшенні кількості користувачів або обсягу оброблюваних даних. Крім того, хмарні платформи забезпечують підвищений рівень безпеки завдяки вбудованим механізмам контролю доступу, автентифікації та моніторингу, що позитивно впливає на надійність роботи вебсистеми.

3.2 Структура інтерфейсу вебсистеми керування процесами видавництва

При переході на сайт користувач побачить форму для входу в особистий акаунт, оскільки весь функціонал системи доступний лише авторизованим користувачам. Розглянемо процес створення у системі нового видавництва та налаштування його компонентів адміністратором, оскільки він має найбільше можливостей у своєму видавництві. Для реєстрації видавництва необхідно перейти на відповідну сторінку (рис. 3.2), посилання на неї розташоване на сторінці входу біля форми.

РЕЄСТРАЦІЯ ВИДАВНИЦТВА

Створити видавництво

Заповніть основні дані, щоб зареєструвати нове видавництво в системі.

Назва видавництва

Vivat

Контактний email

ipz222_kso@student.ztu.edu.ua

Адреса

проспект Миру, 21

Вебсайт

https://localhost:5173/vivat

Опис

Коротко опишіть видавництво

Видавництво успішно зареєстровано. Будь ласка, перейдіть на електронну пошту для продовження реєстрації адміністратора видавництва.

[Повернутися до входу](#) [Зареєструвати видавництво](#)

Рис. 3.2. Реєстрація видавництва

Усі поля реєстраційної форми повинні бути коректно заповнені користувачем перед надсиланням даних на сервер. Під час введення інформації система виконує перевірку обов'язкових полів, формату електронної адреси, мінімальної довжини пароля, відповідності підтвердження пароля та інших встановлених правил валідації. У разі виявлення помилок реєстрація не виконується, а користувачу

відображаються зрозумілі повідомлення про помилки безпосередньо на формі (рис. 3.3).

The screenshot shows a web form titled "РЕЄСТРАЦІЯ ВИДАВНИЦТВА" (Publisher Registration) with the subtitle "Створити видавництво" (Create Publisher). Below the subtitle is a instruction: "Заповніть основні дані, щоб зареєструвати нове видавництво в системі." (Fill in the main data to register a new publisher in the system). The form contains several input fields: "Назва видавництва" (Publisher Name) with the value "Vivat", "Контактний email" (Contact Email) with the value "ipz222_kso@student.ztu.edu.ua", "Адреса" (Address) with the value "проспект Миру, 21", and "Вебсайт" (Website) with the value "localhost:5173/vivat". There is also a large text area for "Опис" (Description) with the placeholder "Коротко опишіть видавництво". At the bottom of the form, there is a red error message box that says "Вкажіть коректний URL сайту." (Specify a correct website URL). Below the form are two buttons: "Повернутися до входу" (Return to login) and "Зареєструвати видавництво" (Register publisher).

Рис. 3.3. Виведення помилок при некоректно введених даних

Після заповнення та відправлення форми, з’явиться повідомлення про успішну реєстрацію та прохання переглянути лист, який прийшов на вказану при надсиланні електронну адресу, та слідувати вказівкам для продовження реєстрації (рис. 3.4).

The screenshot shows an email interface. At the top, it says "Запрошення адміністратора" (Administrator Invitation) with "External" and "Вийдіть" (Logout) buttons. Below this is the email header: "DoNotReply <DoNotReply@Oc4d9b5d-5729-4f78-80fb-7d35270c759b.eu2.azurecomm.net>" and a "кому мені" (to whom) dropdown. The main content of the email is titled "Запрошення до Publishing System" (Invitation to Publishing System). It starts with "Привіт," (Hello,) and says "Вас запросили стати адміністратором видавничої системи для Vivat." (You have been invited to become an administrator of the publishing system for Vivat.). It then says "Щоб прийняти запрошення та активувати свій обліковий запис, натисніть кнопку нижче та встановіть пароль." (To accept the invitation and activate your account, click the button below and set a password.). There is a purple button labeled "Прийняти запрошення" (Accept invitation). Below the button, it says "Це посилання для запрошення дійсне протягом 24 годин. Якщо ви не очікували цього електронного листа, можете сміливо його ігнорувати." (This link for the invitation is valid for 24 hours. If you did not expect this email, you can safely ignore it.). At the bottom, it says "З повагою, Команда Publishing System" (With regards, Publishing System Team). The footer of the email says "© 2026 Publishing System. All rights reserved."

Рис. 3.4. Лист про реєстрацію видавництва

У листі буде посилання на сторінку реєстрації адміністратора видавництва (рис. 3.5). Для цього використовуватиметься електронна пошта видавництва. Слід зауважити, що таким чином може реєструватися лише адміністратор, для інших працівників видавництва облікову записи створюються власноруч ним, покроковий процес цього буде показано далі.

РЕЄСТРАЦІЯ АДМІНІСТРАТОРА

Створити акаунт адміністратора видавництва

Заповніть необхідні дані, відкривши посилання з отриманого листа.

Ім'я користувача

Ім'я

Прізвище

Телефон

Дата народження

Домашня адреса

Пароль

Дата найму

☒ Я погоджуюся з [Політикою конфіденційності](#).

[Повернутися до входу](#)

Зареєструвати адміністратора

Рис. 3.5. Реєстрація адміністратора видавництва

При реєстрації користувач повинен підтвердити свою згоду з політикою конфіденційності шляхом встановлення відповідної позначки у формі. Для ознайомлення з умовами обробки та зберігання персональних даних у формі передбачено посилання на окрему сторінку з текстом політики конфіденційності (рис. 3.6). Відсутність підтвердження згоди унеможливилює завершення процедури реєстрації, що забезпечує дотримання вимог щодо інформування користувачів про порядок використання їхніх персональних даних.

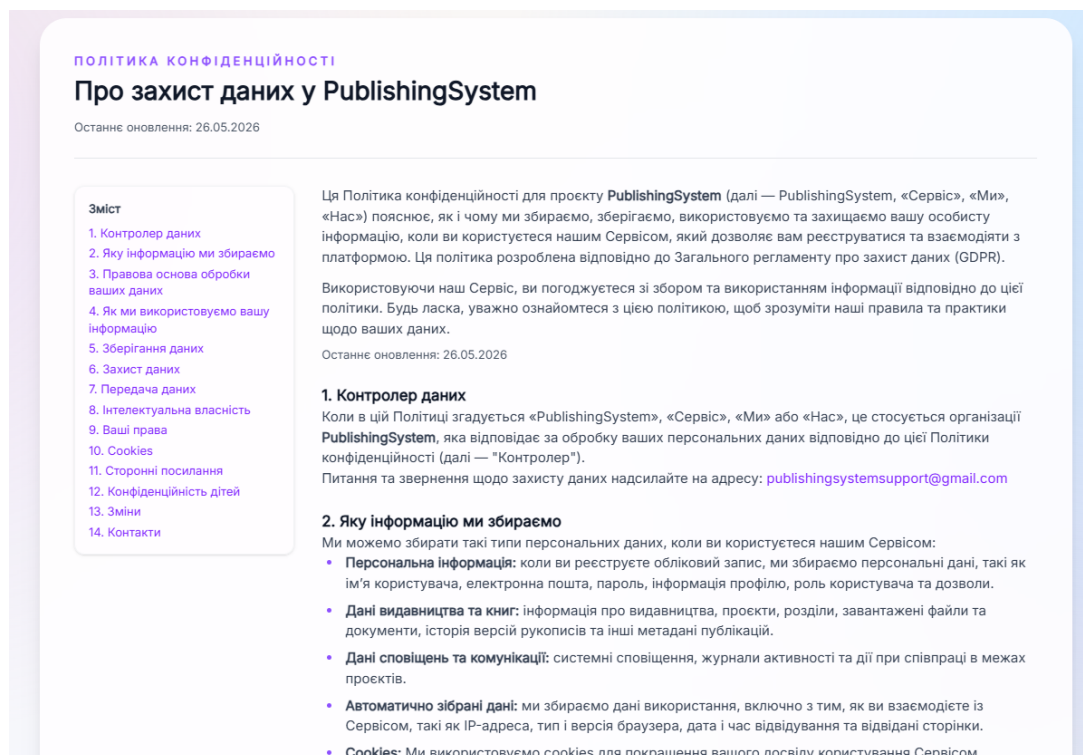


Рис. 3.6. Сторінка політики конфіденційності

Після створення акаунту користувач може повернутися на сторінку входу (рис. 3.7). Для авторизації необхідно ввести унікальне ім'я користувача та пароль, вказані під час реєстрації.

Рис. 3.7. Вхід в особистий акаунт

Після успішної автентифікації вебсистема перевіряє права доступу користувача та перенаправляє його до кабінету видавництва (рис. 3.8). У межах кабінету користувачеві надається доступ до основних функціональних можливостей системи відповідно до його ролі.

Рис. 3.8. Сторінка кабінету видавництва

Адміністратор може відредагувати дані видавництва. Для цього необхідно відкрити форму редагування, яка відображається у вигляді модального вікна та містить поля для внесення змін до інформації про видавництво (рис. 3.9).

Рис. 3.9. Редагування даних видавництва

Після внесення коригувань адміністратор зберігає оновлені дані, які одразу застосовуються в системі. У цей момент користувач отримує системне сповіщення про успішне виконання операції. Для прикладу було змінено опис видавництва (рис. 3.10).

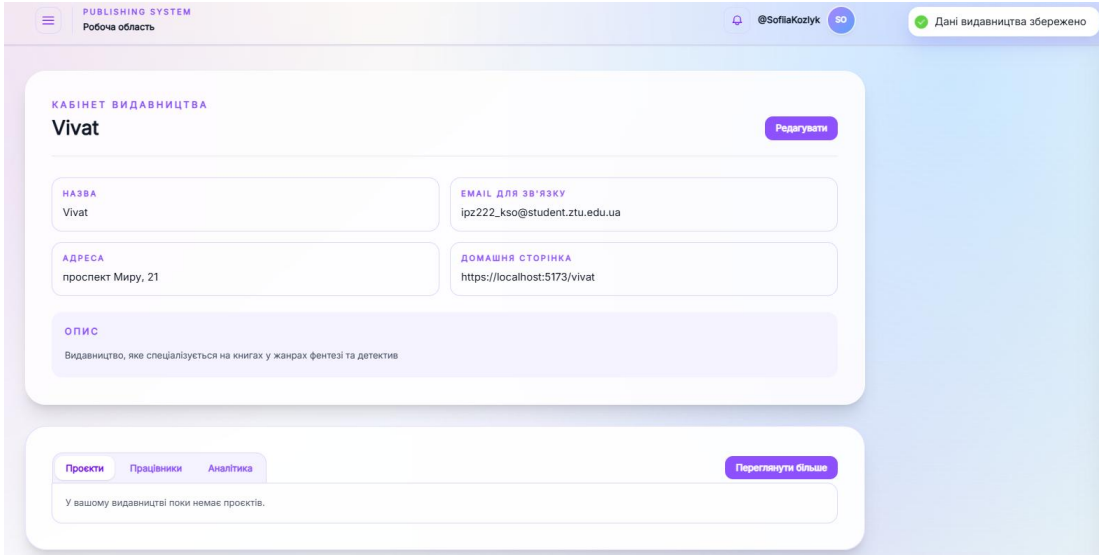


Рис. 3.10. Змінені дані

Під основною інформацією видавництва розміщується блок з даними про проекти та працівників видавництва. При зміні опції автоматично змінюється контент блоку. Тобто при виборі вкладки «Працівники» у блоці відобразиться певна кількість карток з короткою інформацією про кожного працівника (рис. 3.11).

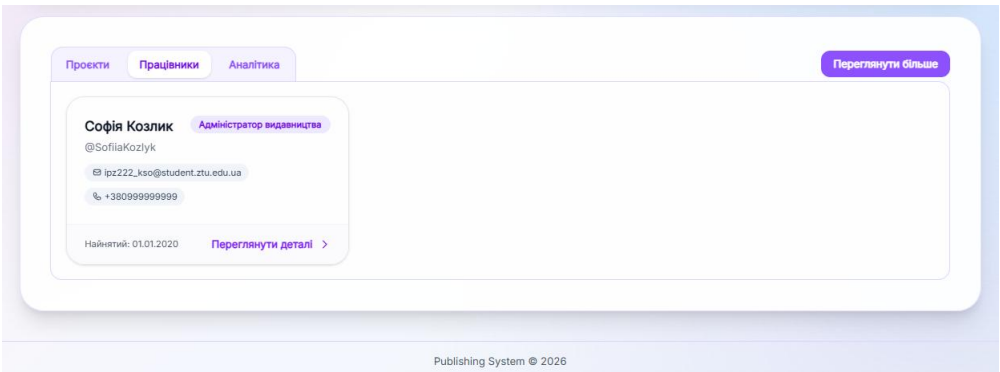


Рис. 3.11. Вкладка «Працівники» на сторінці кабінету видавництва

При натисканні на кнопку «Переглянути більше» вебзастосунок перенаправить користувача на сторінку з картками усіх працівників видавництва. До неї можна також перейти за допомогою посилань на бічні панелі (рис. 3.12).

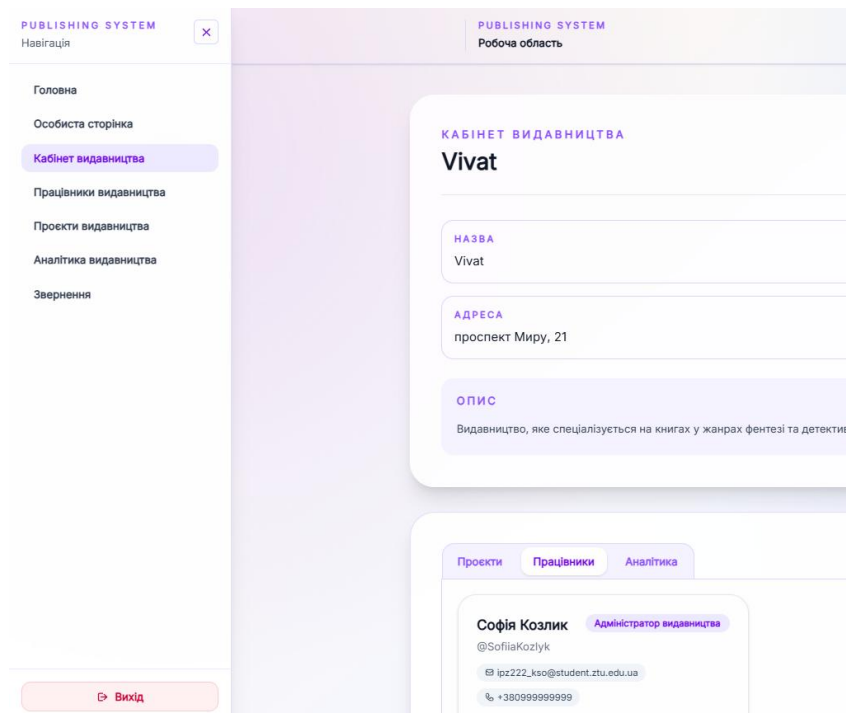


Рис. 3.12. Бічна панель з посиланнями

На сторінці працівників видавництва (рис. 3.13) налаштувати відображення даних відповідно до своїх потреб. Зокрема, передбачено вибір кількості карток, що відображаються на одній сторінці. Окрім цього, на сторінці реалізовано функціонал додавання нового працівника

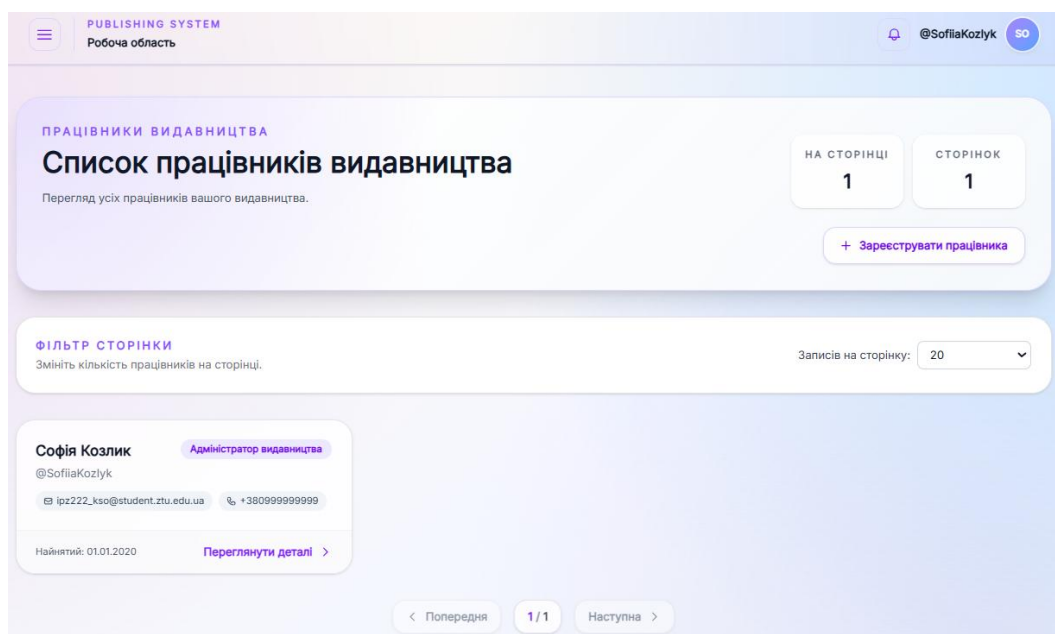


Рис. 3.13. Сторінка з усіма працівниками

Для реєстрації нового працівника необхідно натиснути на відповідну кнопку, після чого відкриється модальне вікно з формою. Додамо до нашого видавництва нового користувача з роллю «Менеджер» (рис. 3.14).

Рис. 3.14. Реєстрація нового працівника

Після заповнення усіх необхідних даних та їхньої валідації, створюється акаунт працівника, а на його електронну адресу надсилається повідомлення (рис. 3.15) про реєстрацію у системі, яке містить інформацію щодо подальших дій та посилання для активації облікового запису.

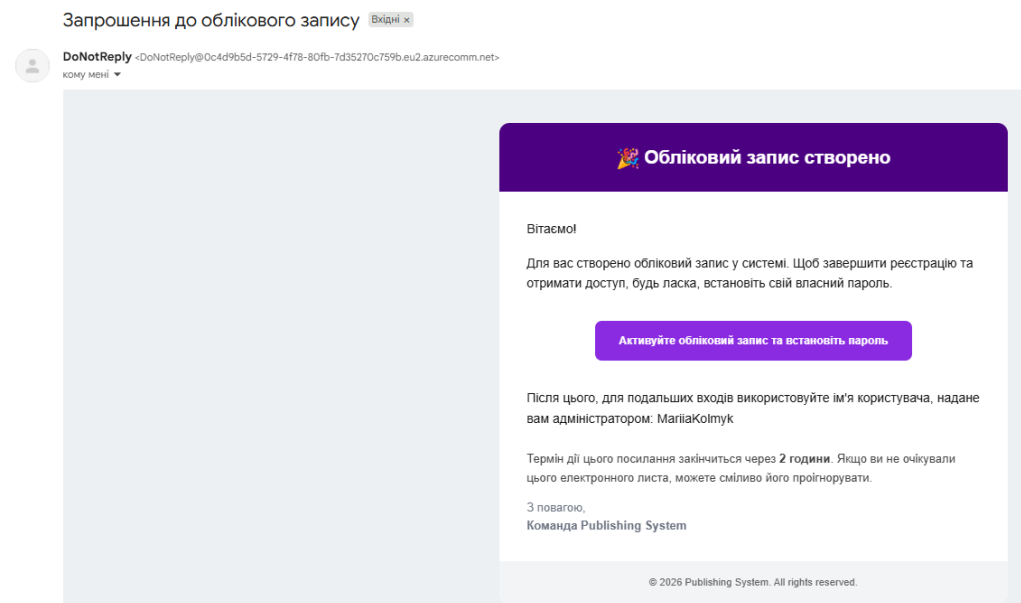


Рис. 3.15. Лист про реєстрацію працівника

Після переходу за отриманим посиланням працівник потрапляє на сторінку завершення реєстрації, де створює та підтверджує пароль для свого облікового запису (рис. 3.16). Після успішного встановлення пароля

обліковий запис активується, що надає користувачу можливість авторизуватися у вебсистемі та виконувати завдання відповідно до своєї ролі.

Рис. 3.16. Встановлення паролю для активації акаунту

Повернімося до акаунту адміністратора та кабінету його видавництва. Аналогічним чином, після натискання кнопки «Переглянути більше» з вибраною опцією «Проекти», система перенаправить користувача на сторінку з картками усіх проєктів видавництва (рис. 3.17). Для демонстрації цієї сторінки заздалегідь створено проєкт. Там також є можливість вибору кількості виведених елементів та створення нового проєкту.

Рис. 3.17. Сторінка з усіма проєктами

Форма для додавання проєкту (рис. 3.18) має спрощену структуру, оскільки проєкт виконує роль контейнерної сутності для пов'язаних

компонентів, таких як книга, етапи, завдання та додаткові матеріали. Основна інформація обмежується базовими параметрами, необхідними для створення та подальшої організації робочого процесу в межах проєкту.

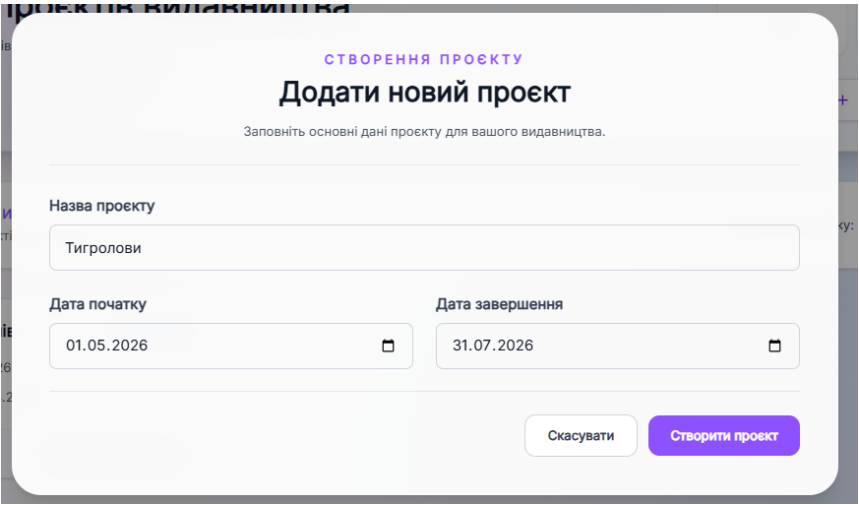


Рис. 3.18. Створення проєкту

Основні налаштування розміщені на сторінці конкретного проєкту (рис. 3.19). Кожен компонент має власну мету при роботі над проєктом, тому рекомендується коректно налаштовувати їх для отримання позитивного досвіду використання доступних інструментів. Кожен проєкт можна відредагувати, змінити його статус, додати до вибраних або видалити з усім його вмістом.

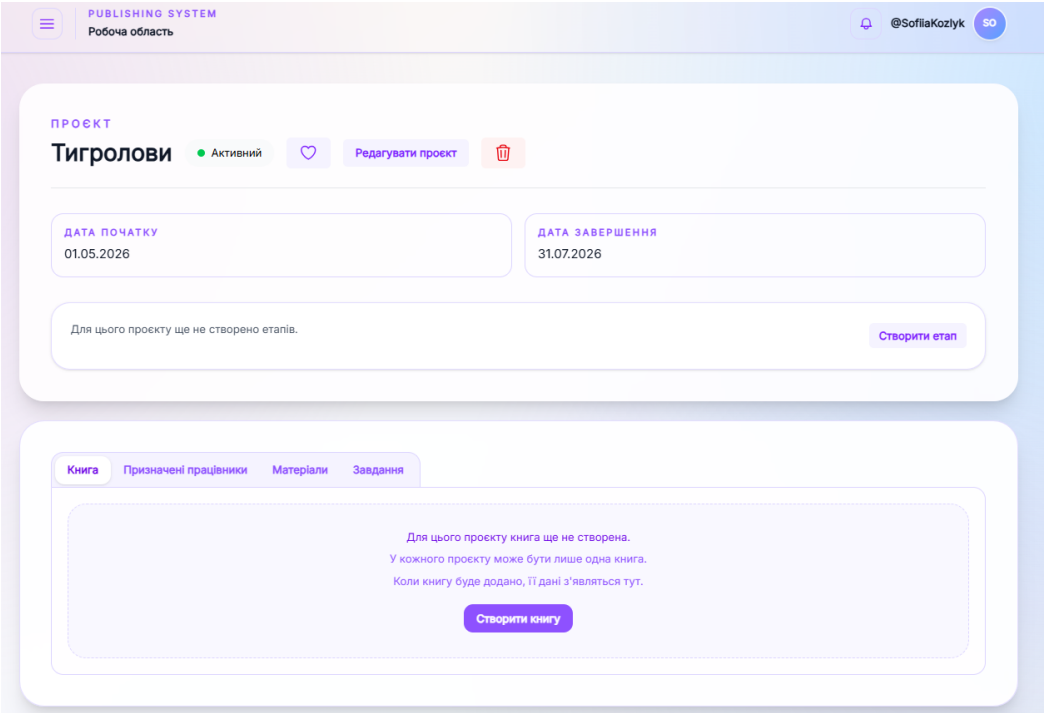


Рис. 3.19. Сторінка нового проєкту

Для початку пропонується додати книгу (рис. 3.20), оскільки вона є основним об’єктом взаємодії на проєкті. Потрібно ввести назву, автора, опис книги, її жанри, заплановану дату публікації. У разі некоректного заповнення полів форми на неї виводитимуться підказки для виправлення.

КНИГА ПРОЄКТУ

Створити книгу

Для кожного проєкту може бути лише одна книга. Заповніть основні дані.

Назва

Тигролови

Автор

Іван Багряний

Дата релізу

03.08.2026

Жанри

пригодницький роман, драма

Вкажіть жанри через кому.

Опис

Іван Багряний — автор романів «Сад Гетсиманський» і «Тигролови». В'язень сталінських таборів, емігрант й один з найгостріших публіцистів XX століття.

Григорій Многогрішний, головний герой роману, тікає з потяга, який везе його в заслання до Сибіру, й тим самим викликає виклик радянській тоталітарній системі.

Скасувати Створити книгу

Рис. 3.20. Створення книги

Після успішного створення інформація про книгу відображається на сторінці проєкту (рис. 3.21). При потребі інформацію про неї можна відредагувати.

КНИГА ПРОЄКТУ

Тигролови

Автор: Іван Багряний

пригодницький роман драма

ОПИС

Іван Багряний — автор романів «Сад Гетсиманський» і «Тигролови». В'язень сталінських таборів, емігрант й один з найгостріших публіцистів XX століття. Григорій Многогрішний, головний герой роману, тікає з потяга, який везе його в заслання до Сибіру, й тим самим викликає виклик радянській тоталітарній системі. Його невпинно переслідують кат НКВС. Блукаючи тайгою, хлопець зустрічає українську родину, яка дає йому притулок і допомагає віднайти душевний спокій та кохання. Але чи зможе він залишитися в безпеці? Чи дозволить тоталітарна машина йому жити вільно? Це захопливий пригодницький роман, який дотепер залишається одним з найкращих текстів в українській літературі.

ДАТА РЕЛІЗУ

03.08.2026

КІЛЬКІСТЬ РОЗДІЛІВ

0

Відкрити текст книги Редагувати книгу

Рис. 3.21. Додана книга проєкту

До заповнення книги текстом повернімося пізніше, наступним пропонується налаштувати етапи проєкту (рис. 3.22). Кожен етап має свою назву, опис, порядковий номер, дату початку та закінчення, які вказуються при створенні.

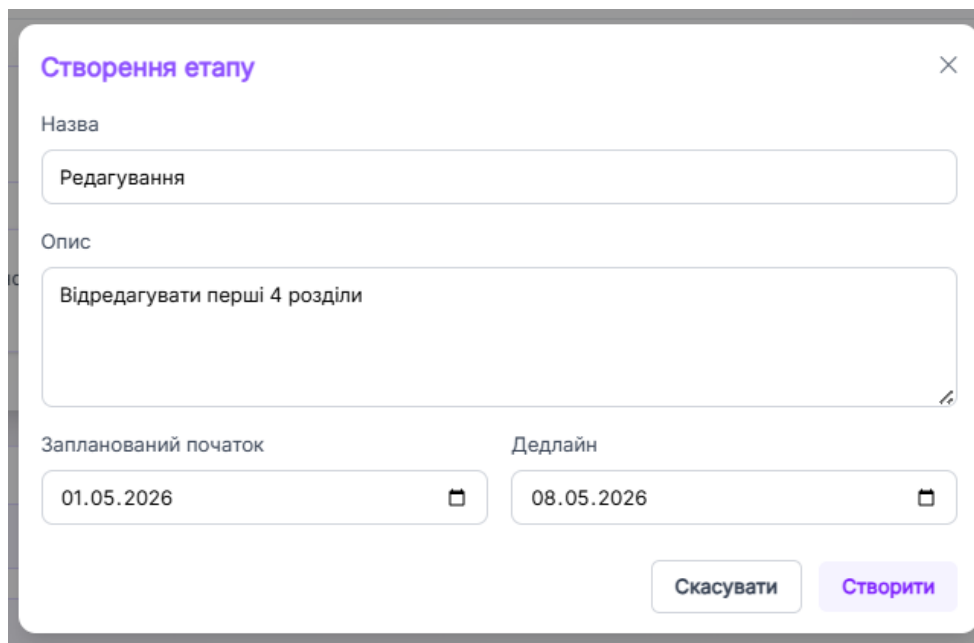


Рис. 3.22. Створення етапу

Усі етапи виконання проєкту можуть переглядатися у вигляді індикатора прогресу (рис. 3.23). Заповнений фіолетовий круг позначає завершений етап, коло з фіолетовим контуром – активний, а сірий круг – етап, який ще не розпочато. Така візуальна індикація дозволяє швидко оцінити стан виконання робіт у межах проєкту.

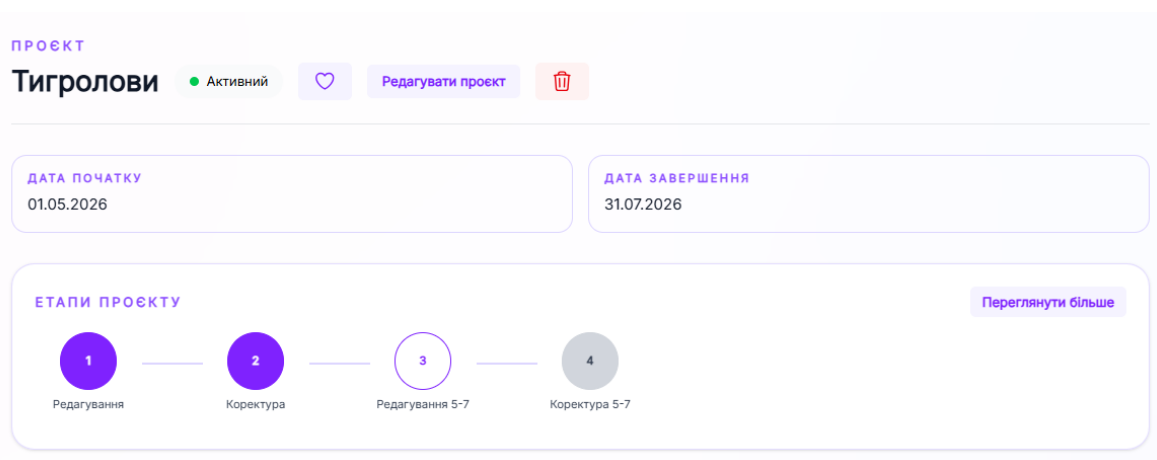


Рис. 3.23. Індикатор прогресу зі створеними етапами

Також передбачено можливість перегляду етапів у вигляді списку в модальному вікні (рис. 3.24), через яке можна створити новий етап або перейти до деталей конкретного.



Рис. 3.24. Модальне вікно з усіма етапами проекту

Деталі окремого етапу (рис. 3.25) відкриваються після натискання на відповідний круг або при виборі у модальному вікні. Користувач має можливість редагувати або видалити обраний етап.

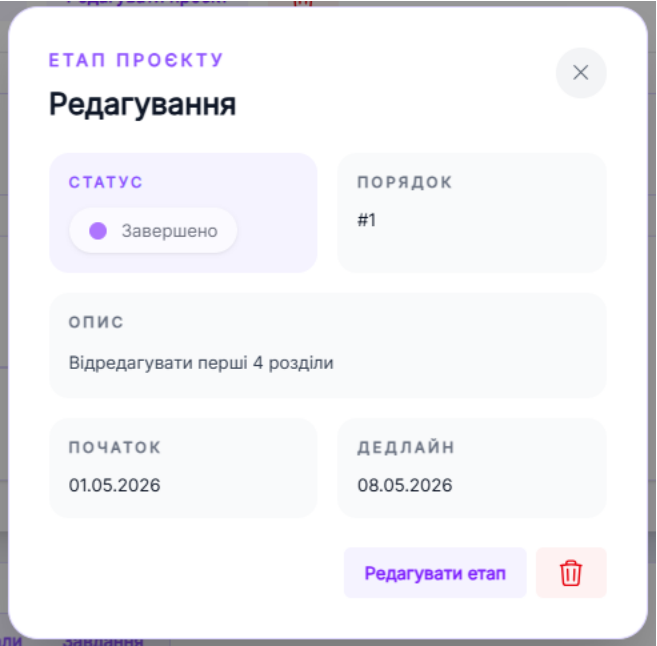


Рис. 3.25. Деталі етапу

Статус етапу можна змінити при натисканні на нього (рис. 3.26). Після цього з’являється випадаючий список з можливими переходами.

ЕТАП ПРОЄКТУ

Редагування 5-7

СТАТУС

☐ В процесі

☒ Завершено

Рис. 3.26. Зміна статусу етапу

Якщо етап було додано помилково або у ньому немає потреби, його можна видалити (рис. 3.27). Перед виконанням операції система відображає вікно з попередженням для підтвердження дії, що запобігає випадковому видаленню даних.

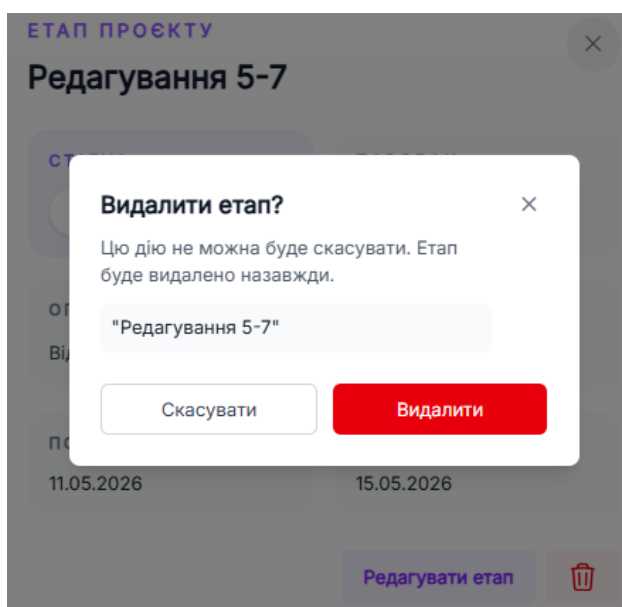


Рис. 3.27. Попередження про видалення етапу

Серед інших налаштувань проєкту важливим є призначення на нього працівників. На відповідній вкладці розміщений список учасників з можливістю їх приєднання та видалення (рис. 3.28 – 3.29).

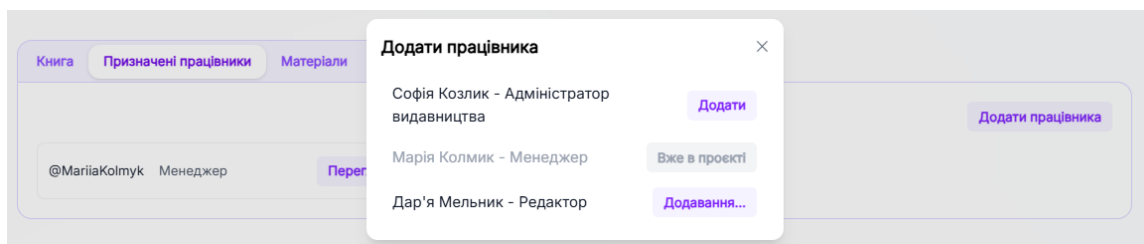


Рис. 3.28. Приєднання працівників

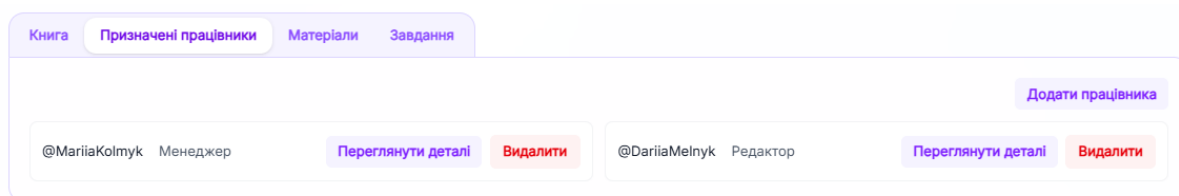


Рис. 3.29. Перегляд усіх працівників на проєкті

Після призначення працівників та створення етапів можна перейти до надання завдань у межах проєкту. На відповідній вкладці відображається список усіх завдань (рис. 3.30), згрупованих за етапами, які можна розкривати чи приховувати. Усі етапи мають зручну шкалу прогресу, а завдання – візуалізацію дедлайнів, яка привертає увагу у разі їх наближення або протермінування.

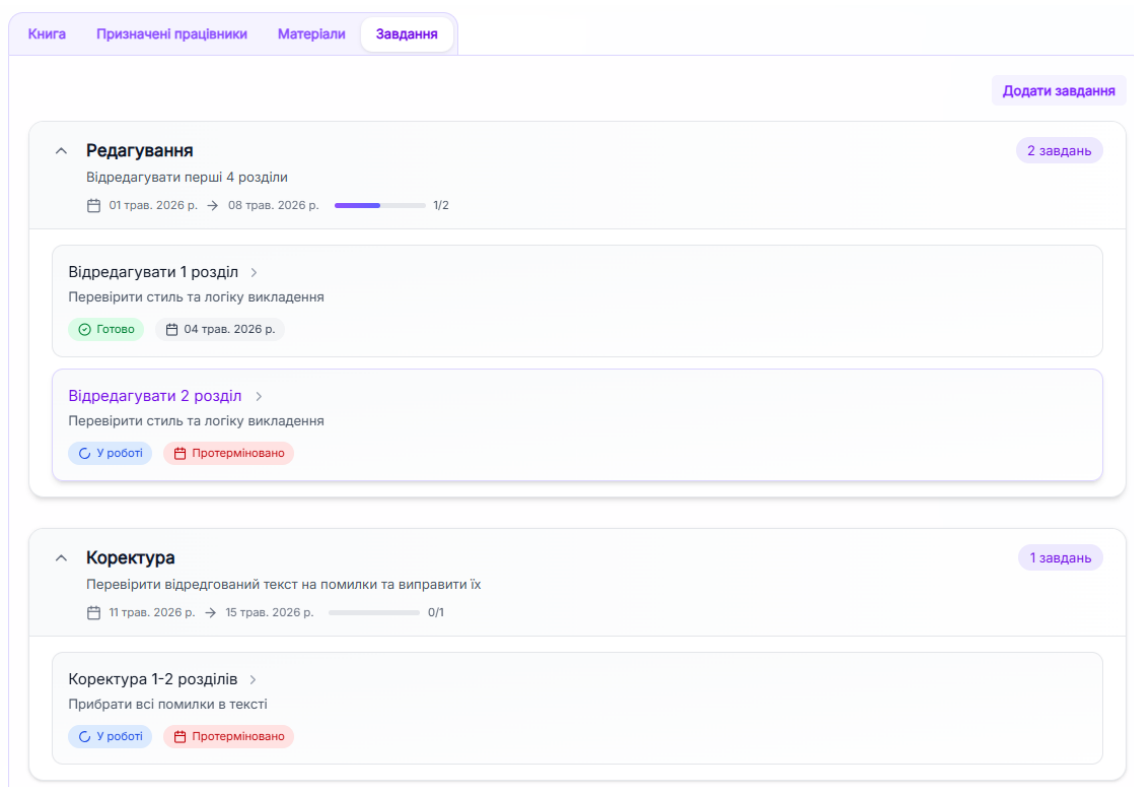


Рис. 3.30. Усі завдання проєкту

На вкладці є кнопка для показу форми додавання нового завдання (рис. 3.31). Для цього необхідно ввести його назву, опис, дедлайн, вибрати етап, якого воно стосується, та працівника, який повинен буде його виконати.

Рис. 3.31. Додавання нового завдання

При кліку по завданню відкривається модальне вікно з його деталями (рис. 3.32). Там можна відредагувати його після натискання на відповідну кнопку і змінити його статус аналогічно до зміни статусу проекту та етапу. За потреби його можна видалити. Також є можливість завантажити іс файл для імпорту завдання в календар.

Рис. 3.32. Деталі завдання

Іншою важливою складовою проєкту є допоміжні матеріали. Вони представлені на вкладці «Матеріали» (рис. 3.33). Ними можуть бути зображення, текстові та pdf файли. При їх створенні необхідно вказати назву, опис та завантажити потрібний файл.

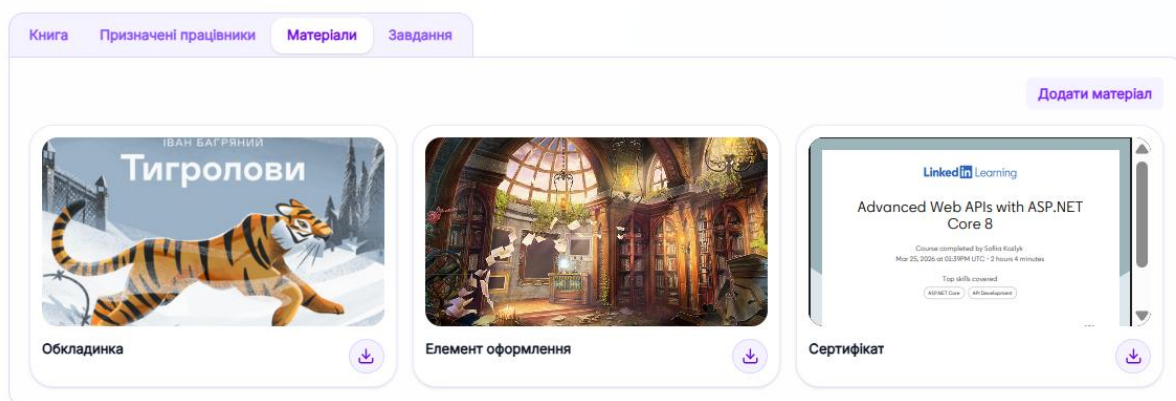


Рис. 3.33. Усі матеріали проєкту

На модальному екрані, яке відкривається при натисканні на матеріал, представлена інформація про нього (рис. 3.34). Відображається його вміст, назва, опис, тип та дата завантаження. Для нього доступні наступні дії: редагування, завантаження файлу та видалення.

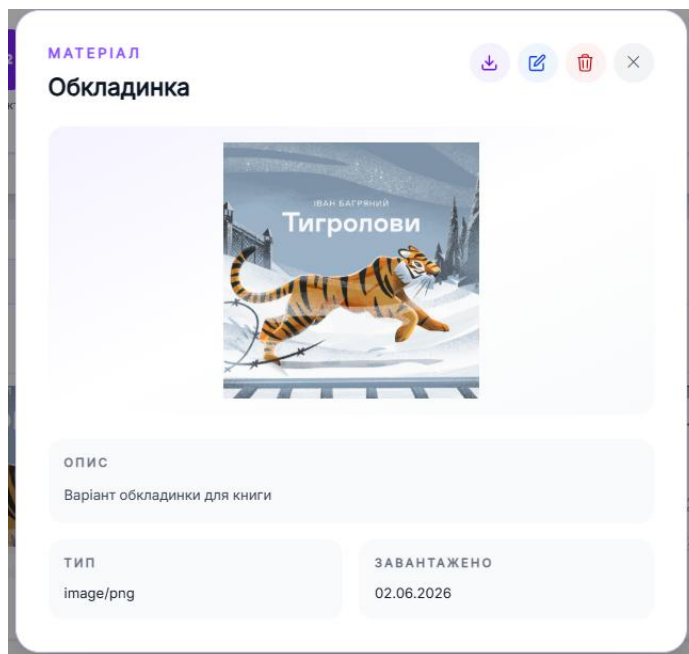


Рис. 3.34. Деталі матеріалу

Повернімося до роботи з книгою проєкту, а точніше з її вмістом. Для цього необхідно натиснути на кнопку «Відкрити текст книги». Вебсистема перенаправить користувача на сторінку з редактором розділів (рис. 3.35). У

його просторі є сам редактор та бічна панель з переліком розділів книги і кнопками для їх створення та приховування.

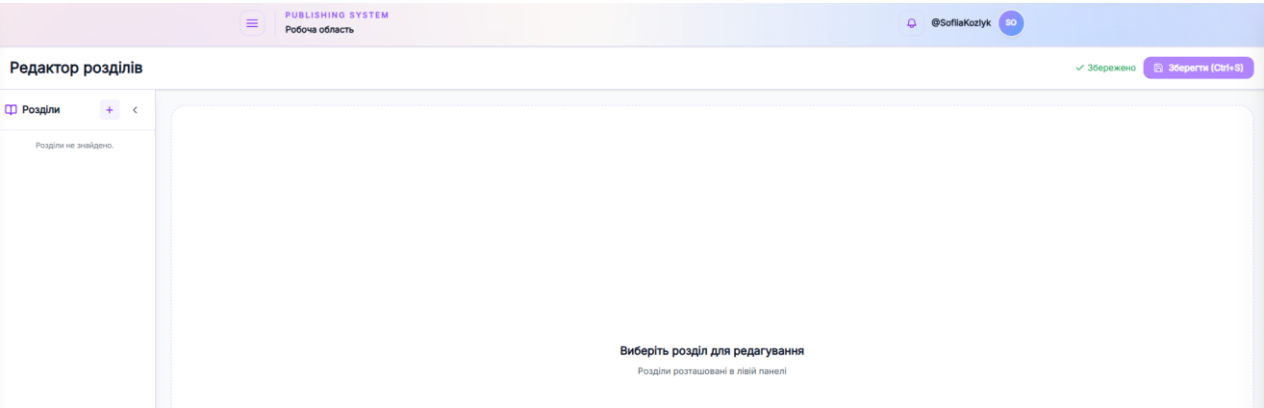


Рис. 3.35. Порожній редактор розділів

Для початку роботи необхідно створити новий розділ з його назвою. Її можна за потреби змінити. Щоб редагувати вміст, потрібно натиснути на вибраний розділ у переліку і його контент відкриється у редакторі. Оскільки це новий розділ, у ньому нічого не введено. Щоб не переносити текст власноруч, можна імпортувати його з docx або документу з Google Docs. Для імпорту з docx файлу необхідно вибрати відповідну іконку серед інструментів редактора та у вікні, що відкриється, потрібний файл (рис. 3.36 – 3.37).

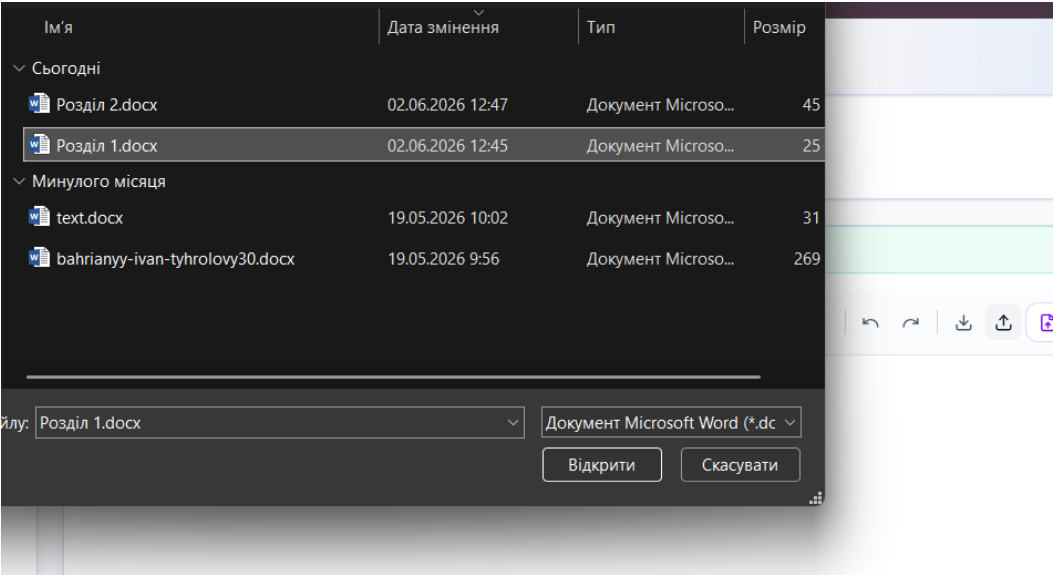


Рис. 3.36. Імпорт тексту розділу з docx файлу

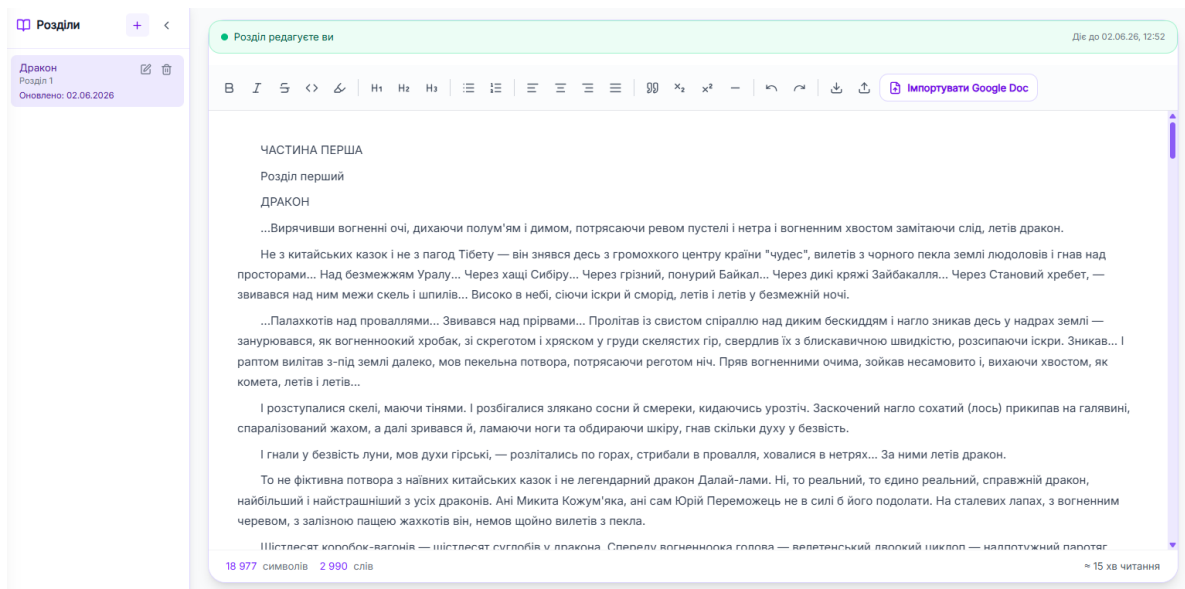


Рис. 3.37. Результат імпорту з docx файлу

Для імпорту з Google Docs (рис. 3.38 – 3.39) необхідно вибрати відповідну кнопку. Після цього відкриється вікно для авторизації за допомогою Google. При успішному вході з'явиться наступне вікно, в якому буде доступний вибір документа для імпорту.

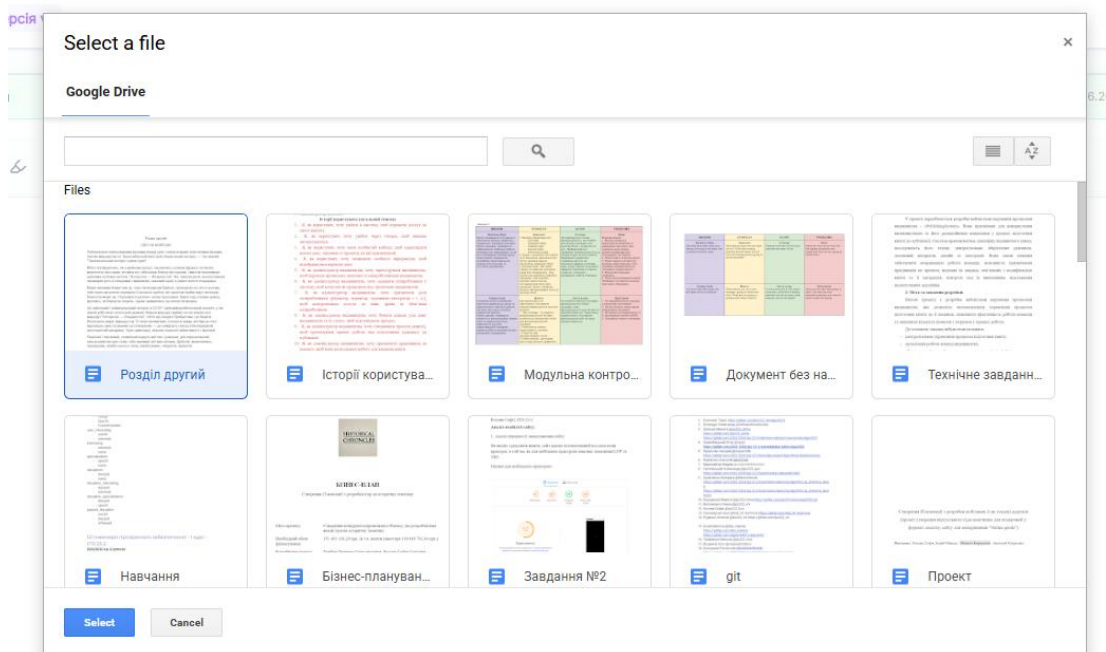
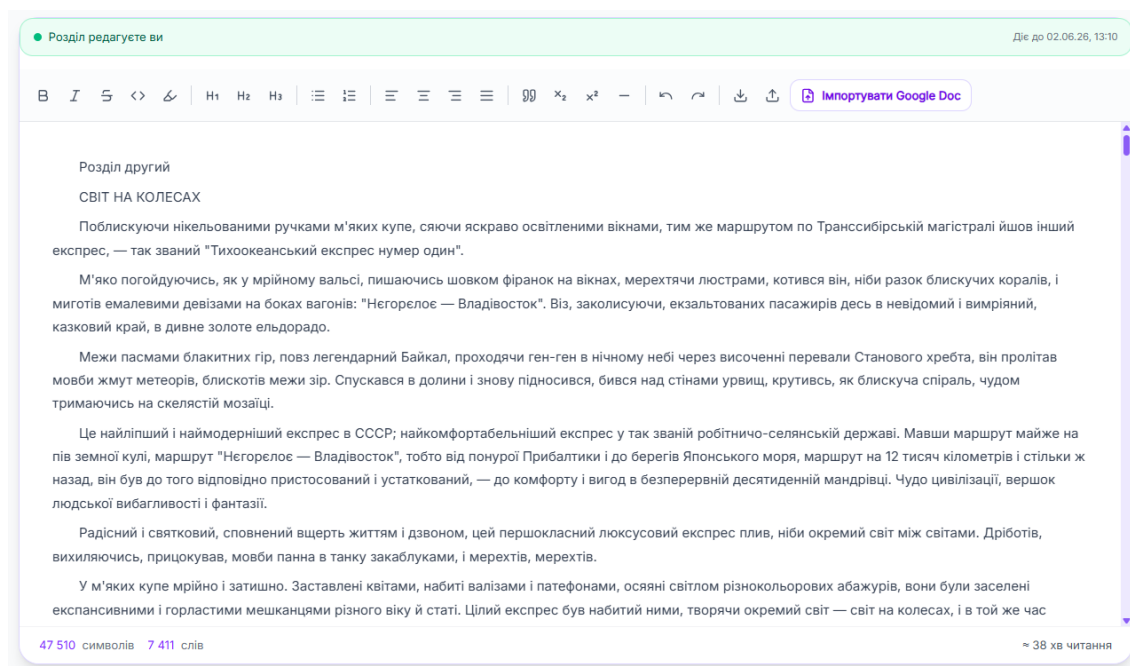
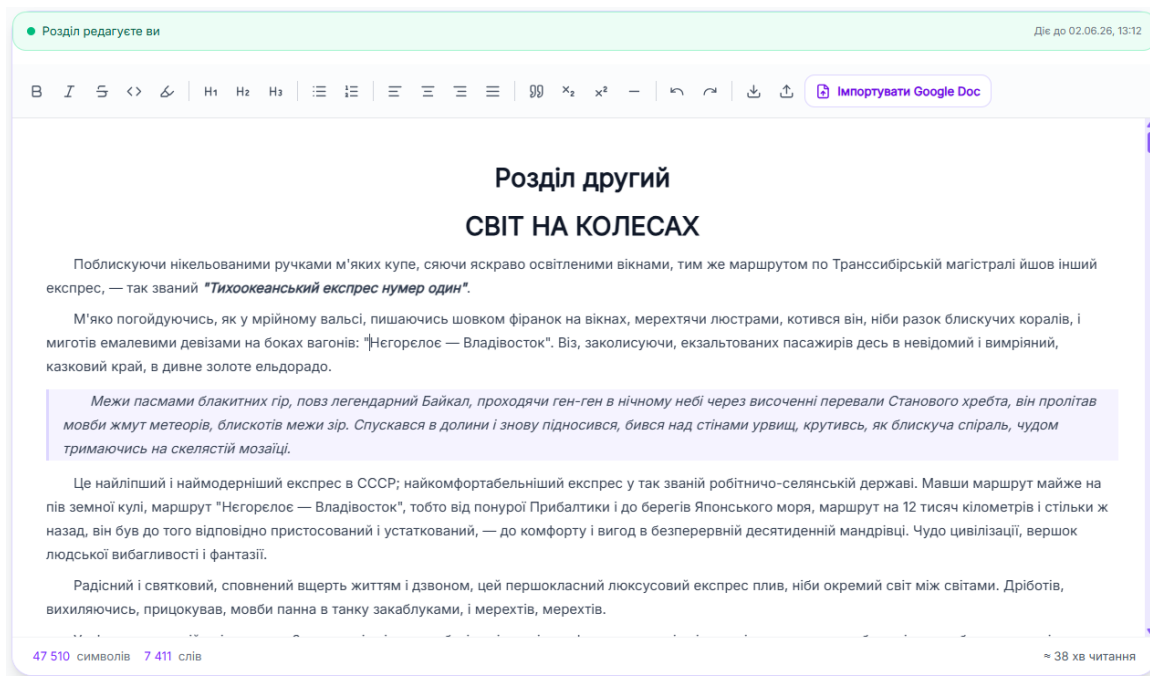


Рис. 3.38. Імпорт тексту розділу з Google Docs



Текстовий редактор підтримує різні можливості форматування тексту, зокрема зміну стилю, створення списків, додавання цитат та інші інструменти редагування (рис. 3.40). Це забезпечує зручне оформлення та структурування вмісту документа.



поза межами вебсистеми. Також у межах редактора передбачено функціонал видалення розділів книги для коригування її структури.

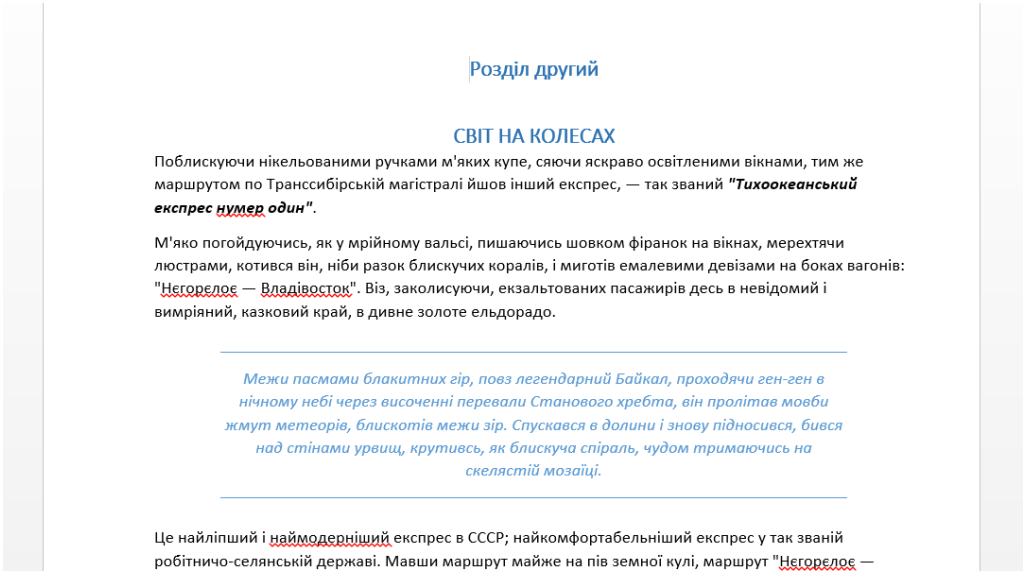


Рис. 3.41. Вміст експортованого docx файлу

Історія всіх редагувань тексту розділу (рис. 3.42) відображається у правій панелі редактора у вигляді версій документа, які автоматично створюються системою при кожному збереженні змін. Такий механізм дозволяє зберігати повну історію модифікацій вмісту, забезпечуючи контроль за процесом редагування та можливість відстеження внесених правок.

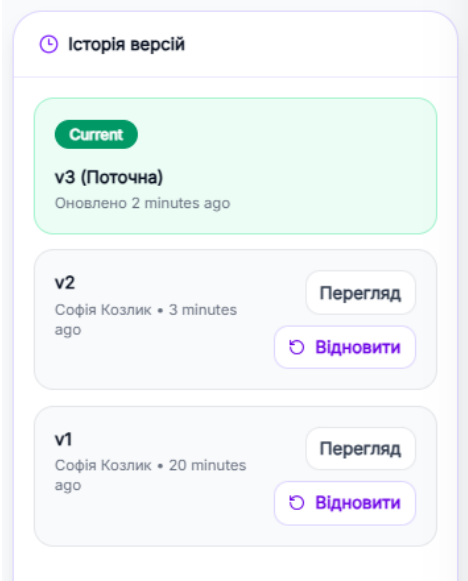


Рис. 3.42. Історія змін розділу

Користувач має можливість переглядати кожну версію в режимі «Читання» (рис. 3.43), що дозволяє ознайомитися з попереднім станом тексту

без внесення змін. У разі необхідності передбачено функціонал відновлення обраної версії, що дає змогу повернути документ до попереднього стану та скасувати небажані редагування.

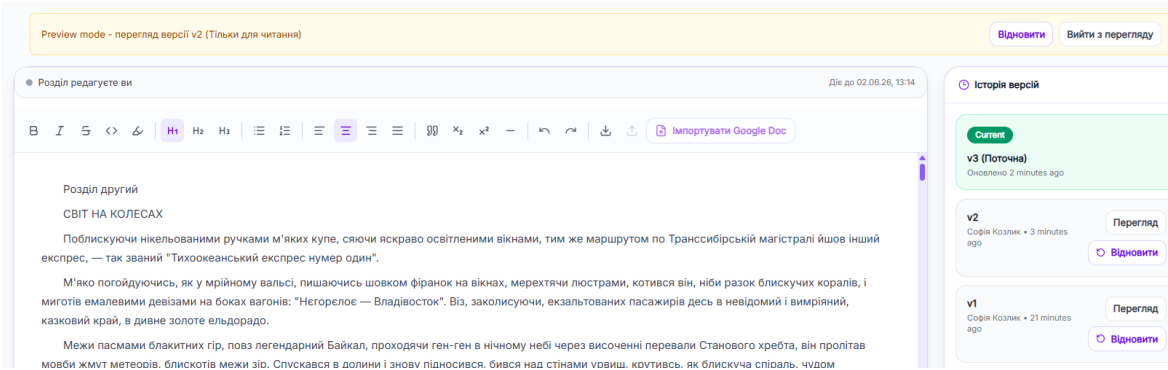


Рис. 3.43. Перегляд попередньої версії

При редагуванні розділу для інших користувачів він блокується, тобто стає доступним лише для читання, щоб запобігти перезапису та втраті даних. Відповідні повідомлення відображаються у верхній частині редактора. Якщо розділ дозволено редагувати, відображається «Розділ редагуєте ви», в протилежному випадку – «Розділ редагується користувачем...» з вказаним іменем користувача (рис. 3.44).

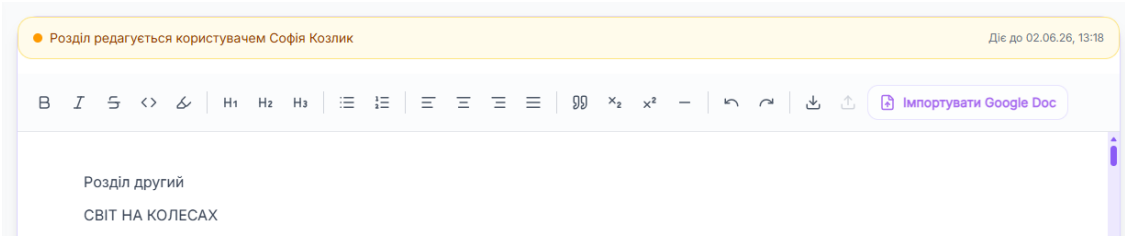


Рис. 3.44. Повідомлення про редагування розділу

Для адміністратора видавництва є можливість перегляду його статистичних даних (рис. 3.45). На сторінці аналітики відображаються необхідні метрики для аналізу роботи видавництва, як-от: кількість працівників, проєктів, діаграми з оглядом статусів проєктів, найактивніших працівників, даних завдань.

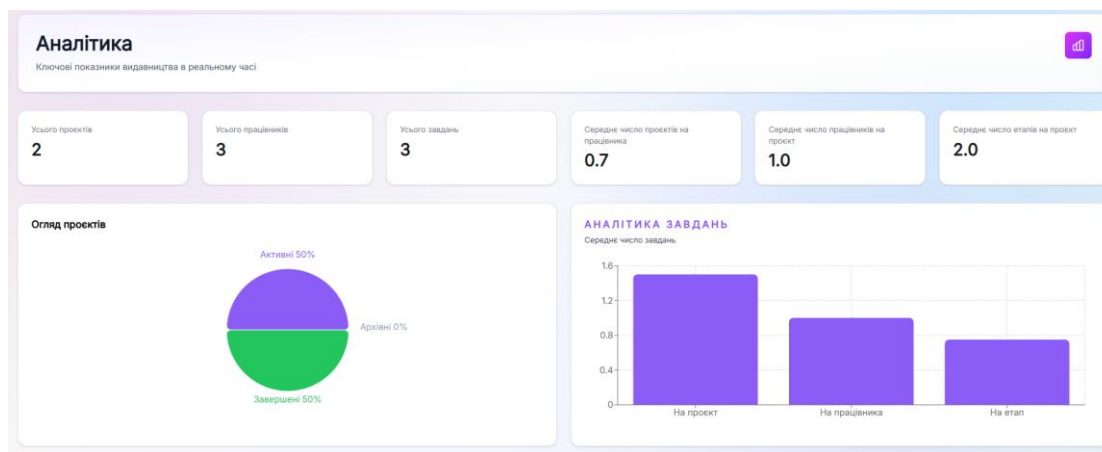


Рис. 3.45. Частина сторінки аналітики видавництва

Перейдемо до особистої сторінки зареєстрованого раніше працівника. На ній користувач може переглянути інформацію про себе, змінити її, змінити пароль та отримати доступ до вибраних проєктів (рис. 3.46).

КАБІНЕТ ПРАЦІВНИКА
Особиста сторінка

ІМ'Я КОРИСТУВАЧА MariiaKolmyk	РОЛЬ Менеджер
ІМ'Я Марія	ПРИЗВИЩЕ Колмик
EMAIL [redacted]	ТЕЛЕФОН +380999999999
ДАТА НАРОДЖЕННЯ 2001-01-01	ДАТА НАЙМУ 02.01.2020
ДОМАШНЯ АДРЕСА вулиця Київська, 1	

Змінити особисту інформацію Змінити пароль Мої завдання Мої проєкти

ОСОБИСТІ ПРОЄКТИ

У вас поки немає обраних проєктів.

Рис. 3.46. Особиста сторінка користувача

До сторінки проєктів користувача (рис. 3.47) можна перейти за допомогою посилання на бічній панелі або на особистій сторінці. Вона схожа на сторінку з проєктами видавництва, оскільки там розміщені картки проєктів, на які призначений працівник, та компонент вибору розміру сторінки. На свої проєкти користувач може перейти та виконувати на них дії, які дозволені йому відповідно до його ролі. Також користувач може додати проєкт у вибрані або прибрати з них.

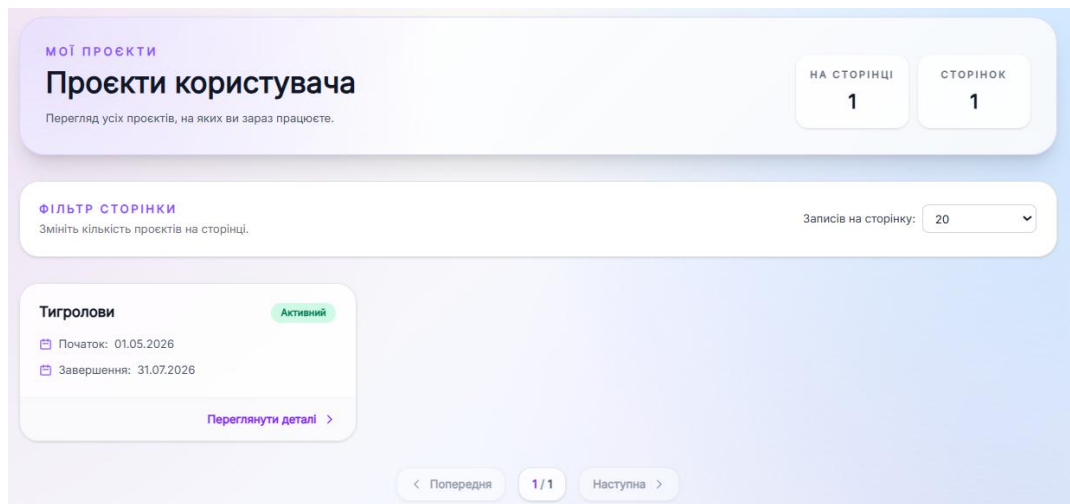


Рис. 3.47. Сторінка з проєктами, на які призначений користувач

На сторінці завдань користувача (рис. 3.48) відображаються всі його завдання, згруповані за проєктами. Така структура подання даних забезпечує зручну навігацію та дозволяє швидко орієнтуватися у великій кількості завдань.

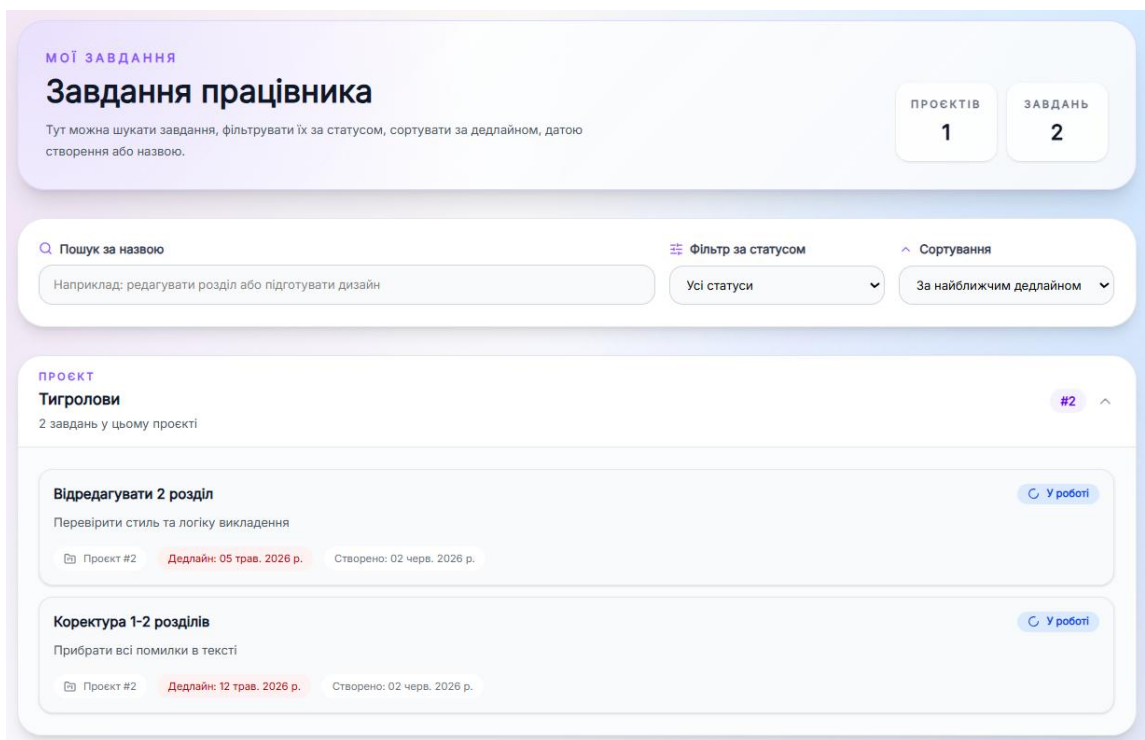


Рис. 3.48. Сторінка із завданнями користувача

Користувач може фільтрувати завдання за статусом, сортувати за назвою, датою створення та найближчим дедлайном, а також виконувати пошук за назвою (рис. 3.49). Це підвищує ефективність роботи з ними та спрощує їх контроль.

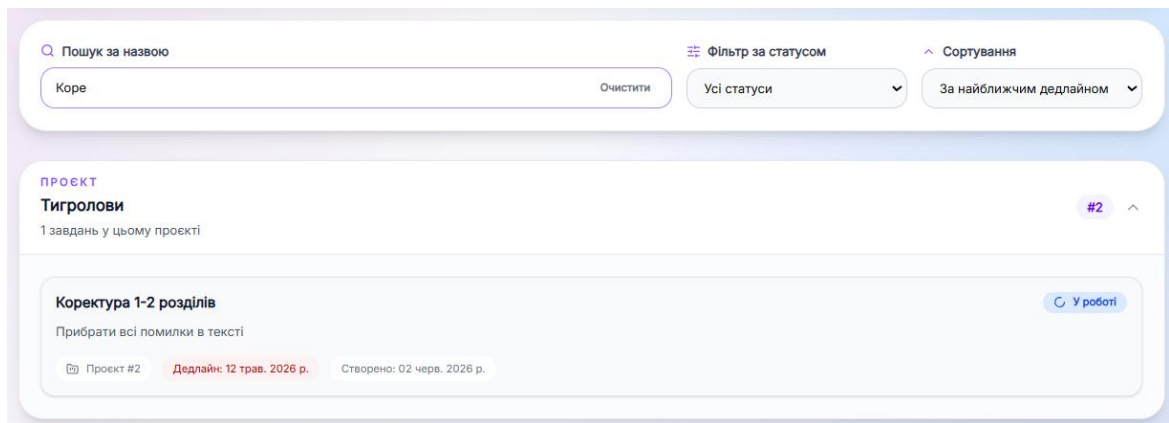


Рис. 3.49. Пошук завдання

Під час роботи у вебсистемі користувач отримує сповіщення, які відображаються у відповідній секції (рис. 3.50). За потреби можна очистити список сповіщень для підтримки актуальності відображуваних даних.

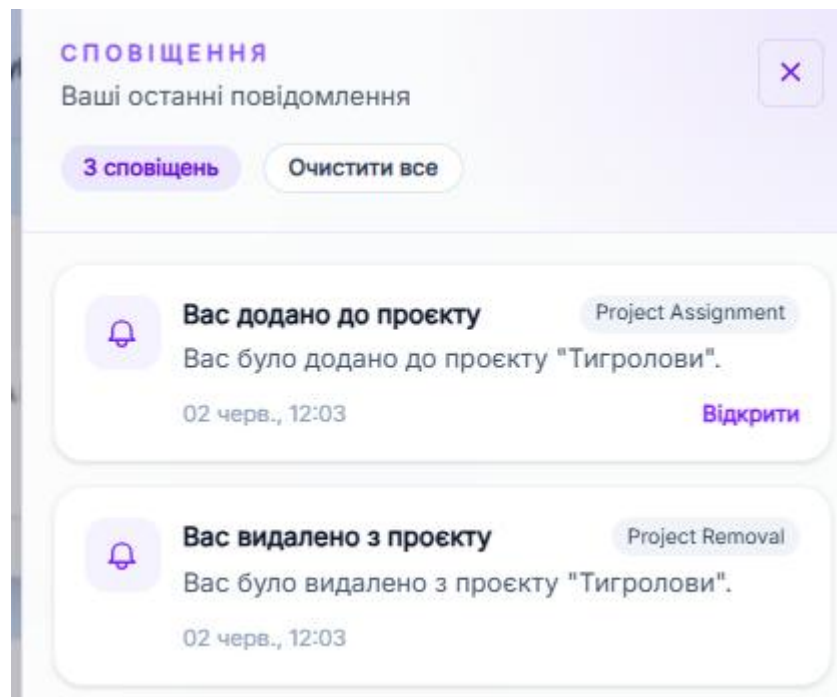


Рис. 3.50. Бічна панель зі сповіщеннями

Якщо користувач втратив свій пароль, він може скористатися функціоналом відновлення доступу, який реалізовано у вебсистемі. Для цього на спеціальній сторінці (рис. 3.51) необхідно вказати електронну адресу, прив'язану до облікового запису.

ВІДНОВЛЕННЯ ДОСТУПУ

Відновлення пароля

Вкажіть email-адресу, і ми надішлемо інструкцію для скидання пароля.

Електронна пошта

Лист для відновлення пароля надіслано.
Перевірте електронну пошту.

Надіслати лист

Пригадали пароль? [Повернутися до входу](#)

Рис. 3.51. Сторінка для вказання електронної адреси

Після надсилання запиту система виконує перевірку коректності введеної адреси та наявності відповідного користувача в базі даних. У разі успішної валідації на зазначену електронну адресу надсилається лист (рис. 3.52) з інструкціями для відновлення доступу та посиланням на сторінку для цього.

Скидання пароля
Вхідні ✕

DoNotReply <DoNotReply@0c4d9b5d-5729-4f78-80fb-7d35270c759b.eu2.azurecomm.net>
 кому мені ▼

Скидання пароля

Привіт 🙌

Ми отримали запит на скидання вашого пароля. Натисніть кнопку нижче, щоб створити новий.

Скинути пароль

Термін дії цього посилання закінчиться через **15 хвилин**. Якщо ви не надсилали запит на скидання пароля, можете сміливо ігнорувати цей електронний лист.

З повагою,
Команда Publishing System

Рис. 3.52. Лист з інструкціями для відновлення паролю

Після переходу на сторінку встановлення нового пароля користувач повинен створити його та підтвердити шляхом повторного введення

(рис. 3.53). Після успішного підтвердження новий пароль зберігається в системі та використовується для подальшої автентифікації користувача.

Рис. 3.53. Сторінка для відновлення паролю

Звичайний користувач має сторінку «Звернення» (рис. 3.54), на якій відображаються усіх його відгуки щодо роботи вебсистеми. Ця сторінка призначена для зворотного зв'язку та написання скарг або пропозицій адміністратору вебсистеми.

Рис. 3.54. Усі відгуки користувача

Усі скарги і пропозиції користувачів може переглядати адміністратор застосунку і реагувати на них, змінюючи їх статус та даючи відповідь на них

(рис. 3.55). Для зручності реалізовано функціонал для пошуку звернень за назвою, їх фільтрації за статусом і типом та сортування за назвою або датою створення.

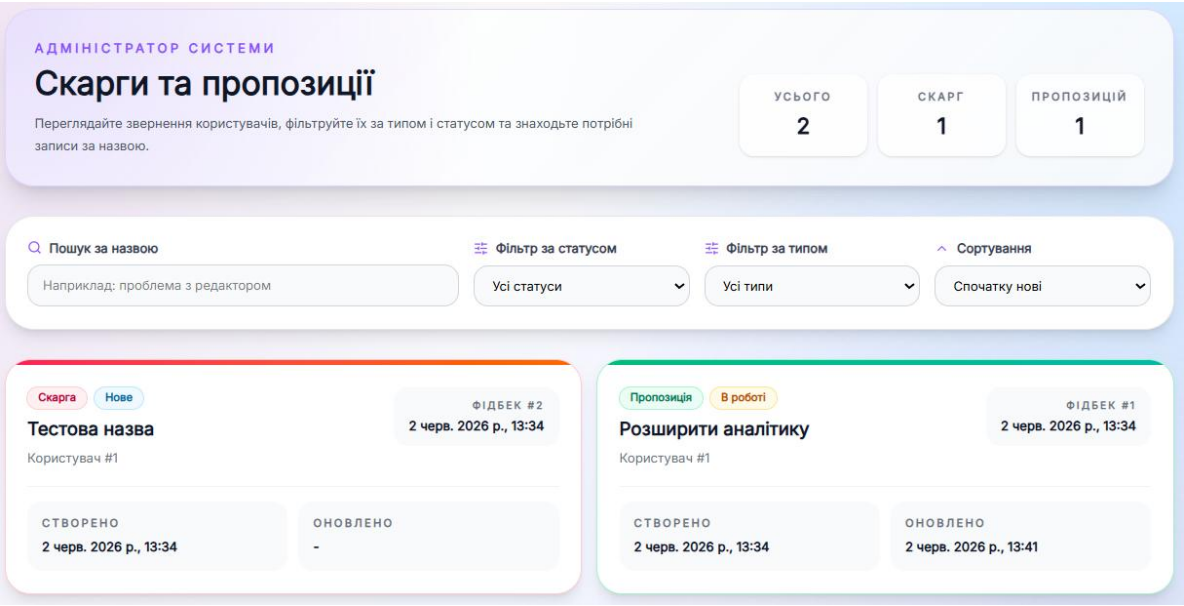


Рис. 3.55. Відгуки усіх користувачів

Деталі звернення можна переглядати у модальному вікні (рис. 3.56), яке з’являється після натискання на вибрану картку. Там відображаються текст звернення та відповідь на нього після його опрацювання адміністратором.

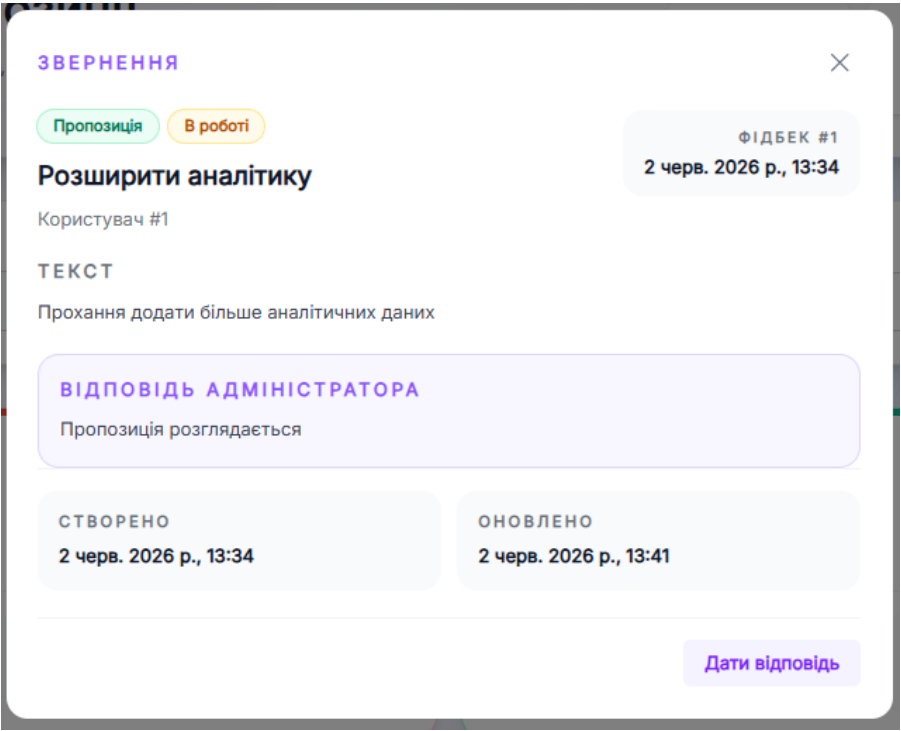


Рис. 3.56. Деталі конкретного звернення

3.3 Тестування вебсистеми керування процесами видавництва

Після реалізації всієї запланованої функціональності відповідно до вимог користувачів вебсистеми було проведено її тестування з метою перевірки коректності роботи, відповідності встановленим вимогам, виявлення можливих помилок та їх подальшого усунення. Тестування дозволило оцінити стабільність функціонування системи та правильність взаємодії між її основними компонентами.

Протягом процесу розробки були створені та виконані модульні тести, спрямовані на перевірку окремих компонентів вебсистеми, зокрема сервісів бізнес-логіки та механізмів доступу до даних [32]. Це забезпечило підвищення надійності програмного коду та дозволило виявити потенційні помилки на ранніх етапах розробки.

Основним видом тестування вебсистеми стало ручне функціональне тестування, під час якого перевірялися ключові сценарії використання відповідно до визначених вимог. Такий підхід дозволив оцінити коректність роботи системи з точки зору кінцевого користувача та виявити можливі недоліки у реалізації окремих функцій.

Ключовими аспектами, що потребували особливої уваги під час тестування були: авторизація та контроль доступу до даних, створення етапів проєкту, завантаження файлів матеріалів, робота з текстовим редактором з коректним керуванням розділами книги, версійність документів та блокування розділів; пошук, фільтрація і сортування завдань користувача та звернень зі сторони адміністратора вебсистеми. За результатами проведеного тестування встановлено, що вебзастосунок функціонує коректно та відповідає всім визначеним вимогам і очікуванням користувачів.

Тестування інтерфейсу вебсистеми було спрямоване на перевірку коректності відображення сторінок застосунку та функціонування їхніх елементів. У процесі тестування оцінювалася стабільність роботи користувацького інтерфейсу та відповідність його поведінки очікуваним сценаріям використання. Зокрема, було перевірено функціонування

навігаційного меню, форм введення даних, а також коректність відображення складних елементів, таких як індикатор прогресу для етапів виконання проєктів. Окрему увагу приділено роботі інструментів текстового редактора, а саме форматуванню тексту та взаємодії з елементами керування.

Також було виконано перевірку адаптивності інтерфейсу вебсистеми. Для цього використовувалися інструменти розробника браузера Google Chrome, що дозволяють емулювати роботу застосунку на різних типах пристроїв та розмірах екранів. У межах тестування було обрано два основні типи пристроїв: ноутбук та планшет (iPad Air).

Особлива увага приділялася правильному відображенню навігаційного меню, списків, модальних вікон та елементів текстового редактора. На екрані ноутбука інтерфейс відображається у повному розширенні з повною структурою елементів сторінки (рис. 3.57).

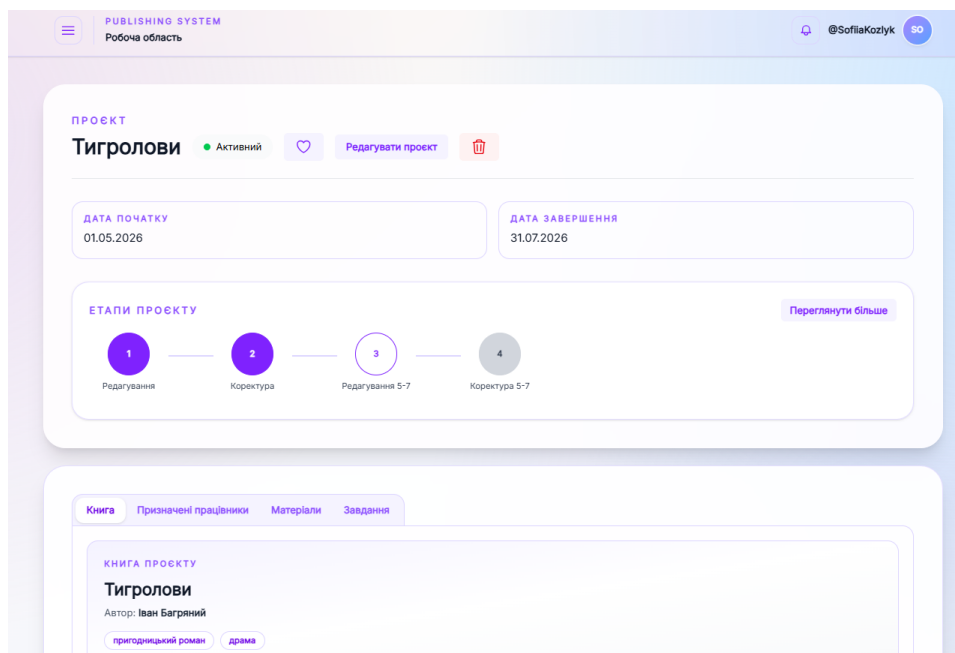


Рис. 3.57. Відображення сторінки проєкту на екрані ноутбука

На планшеті перевірено адаптивність інтерфейсу та коректне масштабування елементів, що забезпечує зручність використання вебсистеми на пристроях з меншими екранами (рис. 3.58). Результати тестування підтвердили, що інтерфейс коректно адаптується до різних розмірів екранів та дозволяє комфортно виконувати роботу користувача.

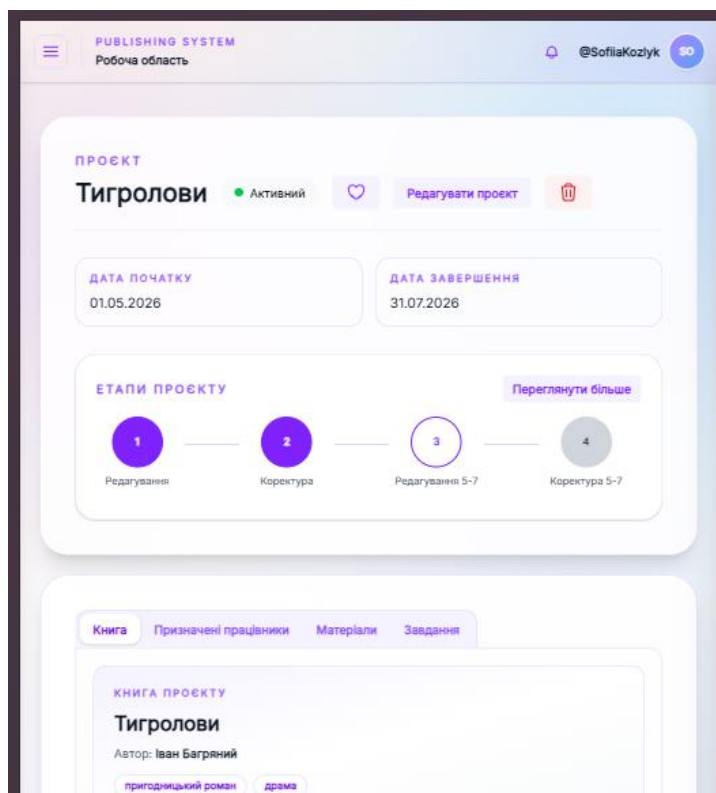


Рис. 3.58. Відображення сторінки проекту на екрані планшета

Важливим етапом тестування вебсистеми стала перевірка її безпеки, під час якої оцінювалися механізми автентифікації та авторизації користувачів, а також контроль доступу до захищених ресурсів. Тестування проводилося із використанням облікових записів із різними ролями та рівнями доступу, що дозволило перевірити коректність реалізації рольової моделі та перевірок належності ресурсу користувачу. Тестування охоплювало як серверну, так і клієнтську частини застосунку, що забезпечило комплексну оцінку захищеності вебсистеми.

Окремо було проведено тестування зручності користування, метою якого була оцінка простоти взаємодії користувачів із вебсистемою та зрозумілості її інтерфейсу. У ході аналізу було встановлено, що система має логічну та структуровану навігацію, інтуїтивно зрозумілі елементи користувацького інтерфейсу та передбачувану поведінку елементів керування. Це дозволяє користувачам швидко опановувати функціонал системи та ефективно виконувати необхідні операції, що в цілому забезпечує позитивний користувацький досвід.

Висновки до третього розділу

У даному розділі було детально описано процес розгортання вебсистеми. Надані пояснення до усіх важливих моментів конфігурації використовуваних сервісів. Представлено діаграму розгортання вебсистеми з усіма компонентами та взаємодіями між ними.

Наступною наводилася покрокова інструкція використання вебсистеми. Розглянуто весь її функціонал – від реєстрації користувача до адміністрування застосунку. Для кращого розуміння використовувалися облікові записи декількох користувачів з різними правами у вебсистемі. Особлива увага зверталася у тих місцях, де користувачі можуть найчастіше помилятися, наприклад, при заповненні форм.

Наведено опис видів тестування, які застосовувалися, та їх результати. Встановлено, що розроблений застосунок коректно виконує всі функції та відповідає визначеним вимогам. Усі виявлені недоліки були незначними та усунутими в процесі розробки.

Таким чином, вебсистема є готовою до практичного використання та подальшого впровадження у роботу видавництв.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						86
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи є реалізована вебсистема керування процесами видавництва, яка відповідає усім визначеним та описаним вимогам.

Було проведено аналіз предметної області, поставлено задачі, які необхідно втілити для досягнення запланованої мети. Знайдено аналоги вебсистеми, серед яких для аналізу вибрано 4: Editorial Manager, Editorial Office Software, Scrivener і Reedsy Studio. Було визначено та обґрунтовано вибір архітектури вебсистеми. Запропоновано використання клієнт-серверної архітектури, в якій сервер представлений у вигляді моноліту, побудованого за принципами Onion Architecture, а клієнт – SPA застосунку. Для їх реалізації вибрано мови програмування C# і TypeScript, фреймворк ASP.NET Core, бібліотеку React, хмарний сервіс Azure.

Визначено варіанти використання вебсистеми після аналізу всіх вимог та спроектовано її об'єктно-орієнтовану структуру з детальним описом усіх розроблених компонентів. Побудовано базу даних, яка містить 21 таблицю та відповідає усім потребам вебсистеми для забезпечення коректного зберігання даних, швидкого виконання запитів та легкого масштабування.

Змодельовано алгоритми роботи вебсистеми з їх текстовим описом та представленням у вигляді діаграм дій. Описана її реалізація з наведенням фрагментів програмного коду гібридної моделі контролю доступу, конфігурації текстового редактора, відновлення версій розділу та їх блокування при редагуванні іншим користувачем, імпорту з Google Docs.

Наведено діаграму розгортання системи, детально пояснено її алгоритм та важливі аспекти. Розглянуто функціонал вебсистеми для різних типів користувачів, а саме: неавторизований користувач, адміністратор видавництва, його працівник і загальний адміністратор. Проведено тестування вебсистеми, для якого використовувалися наступні його види: функціональне, інтерфейсу користувача, безпеки, зручності користування.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						87
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Власюк А. І., Белзецький Р. С. Основи редагування, коректури та верстки технічних текстів : навч. посіб. Вінниця : ВНТУ, 2015. 96 с.
2. Гузенко С. В. Теорія і практика редагування : навч. посіб. Миколаїв : МНУ ім. В. О. Сухомлинського, 2019. 220 с.
3. Editorial Manager - Aries Systems Corporation. Aries Systems Corporation. URL: <https://www.ariessys.com/solutions/editorial-manager/> (дата звернення: 02.01.2026).
4. Editorial Office Software. knkPublishing. URL: <https://www.knkpublishingsoftware.com/editorial-office/> (дата звернення: 05.01.2026).
5. Scrivener – Novel & Book Writing Software For Writers. Literature & Latte. URL: <https://www.literatureandlatte.com/scrivener/overview> (дата звернення: 06.01.2026).
6. Reedsy Studio: Plan, Draft, Edit, and Format your Book. Reedsy Studio. URL: <https://reedsy.com/studio> (дата звернення: 07.01.2026).
7. What is a REST API? Examples, Uses, and Challenges. Postman Blog. URL: <https://blog.postman.com/rest-api-examples/> (дата звернення: 20.01.2026).
8. Monolithic vs Microservices - Difference Between Software Development Architectures. Amazon Web Services, Inc. URL: <https://aws.amazon.com/compare/the-difference-between-monolithic-and-microservices-architecture/> (дата звернення: 14.01.2026).
9. Powell P., Smalley I. Monolithic vs. Microservices Architecture. IBM. URL: <https://www.ibm.com/think/topics/monolithic-vs-microservices> (дата звернення: 14.01.2026).
10. Козлик С., Фант М. Технологічний стек для системи керування процесами видавництва. Тези доповідей VIII Всеукраїнської науково-технічної конференції «Комп'ютерні технології: інновації, проблеми, рішення». Житомир, 2025. С. 35–36.

11. PostgreSQL. PostgreSQL. URL: <https://www.postgresql.org/> (дата звернення: 27.01.2026).

12. Simon M. Getting Started with SQL and Databases: Managing and Manipulating Data with SQL. Apress, 2023. 390 с.

13. C# 9 and .NET 5 – Modern Cross-Platform Development: Build intelligent apps, websites, and services with Blazor, ASP.NET Core, and Entity Framework Core using Visual Studio Code, 5th Edition. Packt Publishing, 2020. 822 с.

14. Overview - A tour of C#. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview> (дата звернення: 05.01.2026).

15. Overview of ASP.NET Core. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/aspnet/core/overview?view=aspnetcore-10.0> (дата звернення: 20.01.2026).

16. Overview of HTTP - HTTP | MDN. MDN Web Docs. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview> (дата звернення: 20.01.2026).

17. Gorman B. Practical Entity Framework Core 10: Database Access for Enterprise Applications. 3-тє вид. Apress, 2025. 558 с.

18. Overview of Entity Framework Core - EF Core. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/ef/core/> (дата звернення: 29.01.2026).

19. Use a database with minimal API, Entity Framework Core, and ASP.NET Core. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/training/modules/persist-data-ef-core/> (дата звернення: 19.01.2026).

20. Безверхий О., Куценко О. Ефективність застосування бібліотеки React. Інформаційні технології та суспільство. 2022. № 2 (4). С. 13–19.

21. Goldberg J. Learning TypeScript: Enhance Your Web Development Skills Using Type-Safe JavaScript. O'Reilly Media, 2022. 318 с.

22. Rippon C. Learn React with TypeScript: A Beginner's Guide to Reactive Web Development with React 18 and TypeScript, 2nd Edition. Packt Publishing, 2023. 474 с.

23. Schwarzmüller M. React Key Concepts: Consolidate Your Knowledge of React's Core Features. Packt Publishing, 2022. 590 с.

24. Tailwind CSS. Tailwind CSS. URL: <https://tailwindcss.com/> (дата звернення: 02.04.2026).

25. Azure Blob Storage. Microsoft Azure. URL: <https://azure.microsoft.com/en-us/products/storage/blobs> (дата звернення: 04.05.2026).

26. Introduction to ASP.NET Identity. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity> (дата звернення: 12.01.2026).

27. Козлик С. О., Фант М. О. Гібридна модель контролю доступу у вебсистемі керування процесами видавництва. Інформаційні технології: теорія і практика : матеріали III (IX) Міжнародної інтернет-конференції здобувачів вищої освіти і молодих учених. Дніпро, 2026. С. 517–518.

28. Ingebrigtsen E. Metaprogramming in C#: Automate Your .NET Development and Simplify Overcomplicated Code. Packt Publishing, 2023. 352 с.

29. Tiptap Docs. Tiptap - Dev Toolkit Editor Suite. URL: <https://tiptap.dev/docs> (дата звернення: 04.05.2026).

30. Google Drive API. Google for Developers. URL: <https://developers.google.com/workspace/drive/api/reference/rest/v3> (дата звернення: 18.05.2026).

31. Overview of web apps. Google for Developers. URL: <https://developers.google.com/workspace/drive/picker/guides/overview> (дата звернення: 18.05.2026).

32. Oshero R. The Art of Unit Testing: With Examples in C#. 2-ге вид. Manning Publications Co. LLC, 2013. 292 с.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						91
Змн.	Арк.	№ докум.	Підпис	Дата		

Діаграми вебсистеми

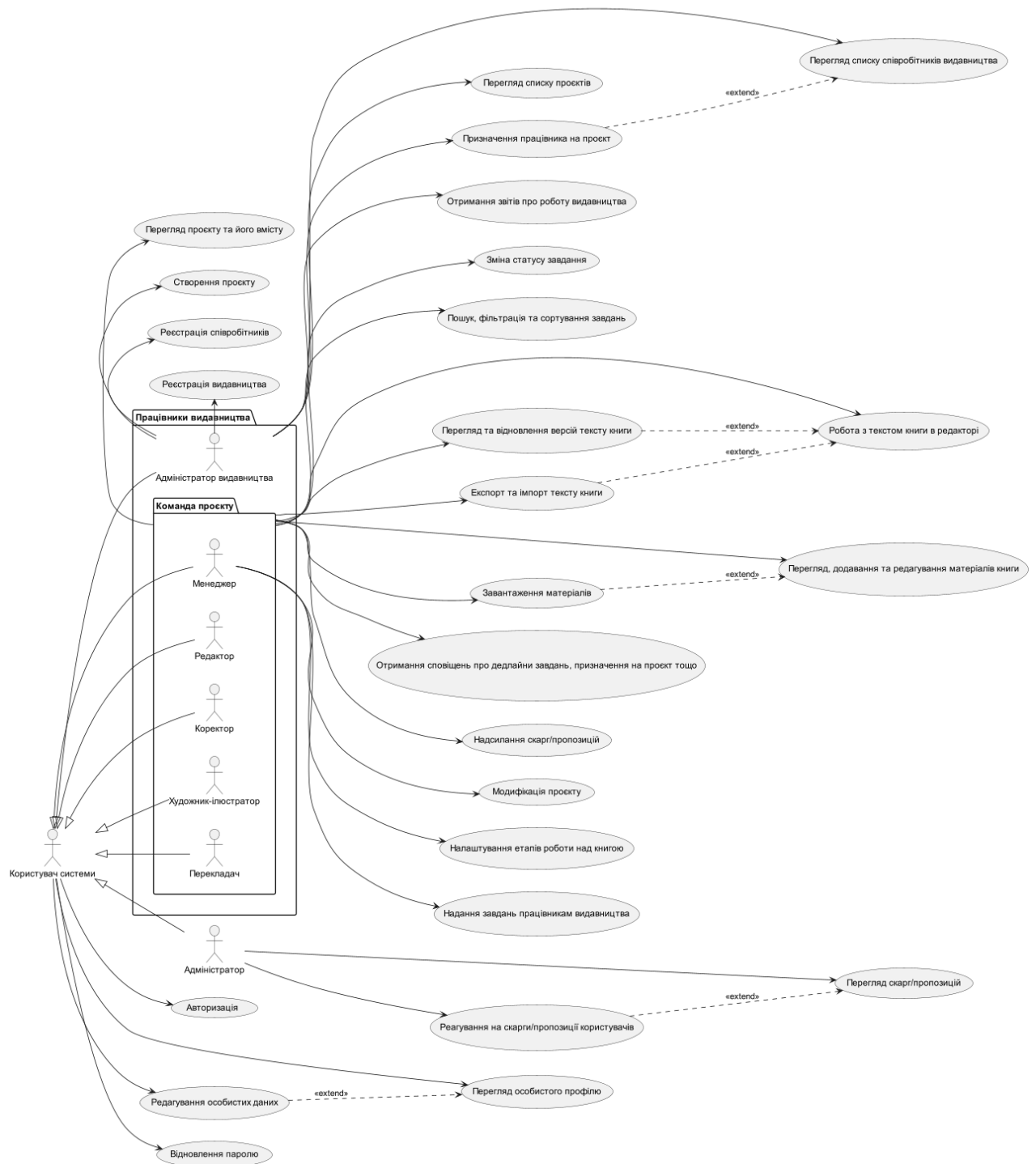


Рис. А.1. Діаграма використання вебсистеми

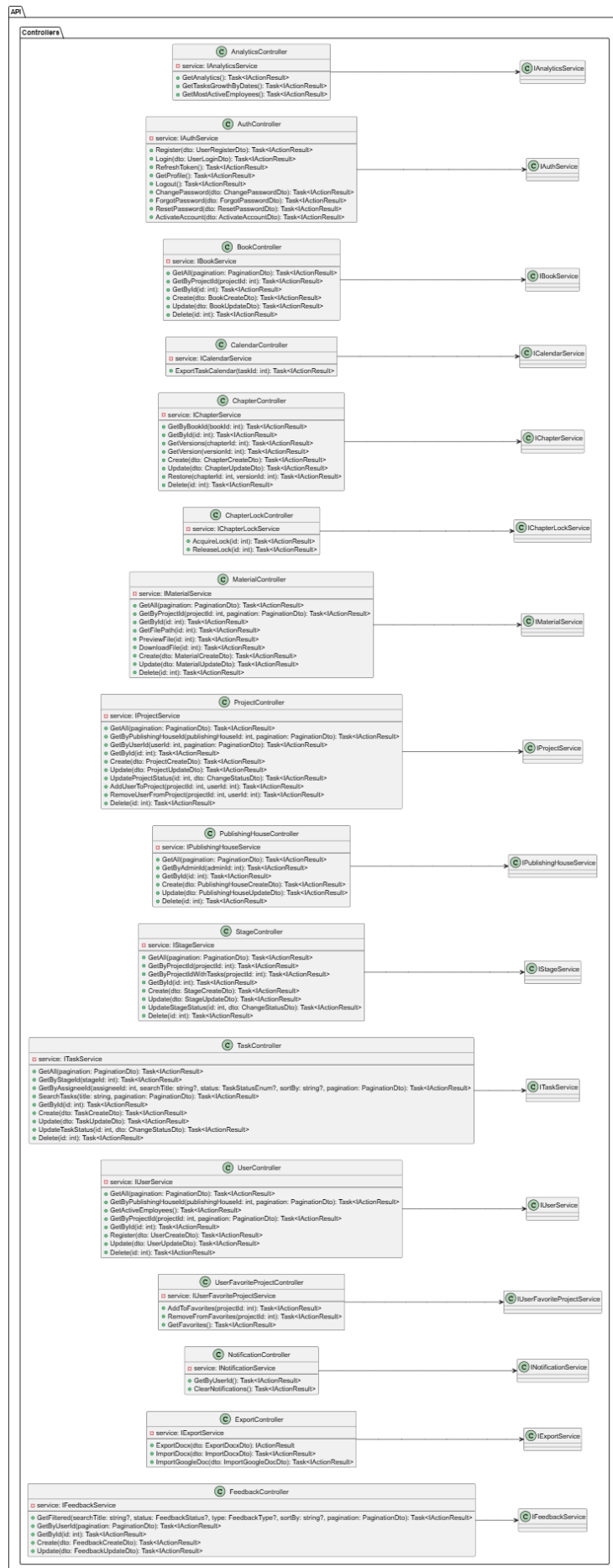


Рис. А.4. Діаграма класів рівня API

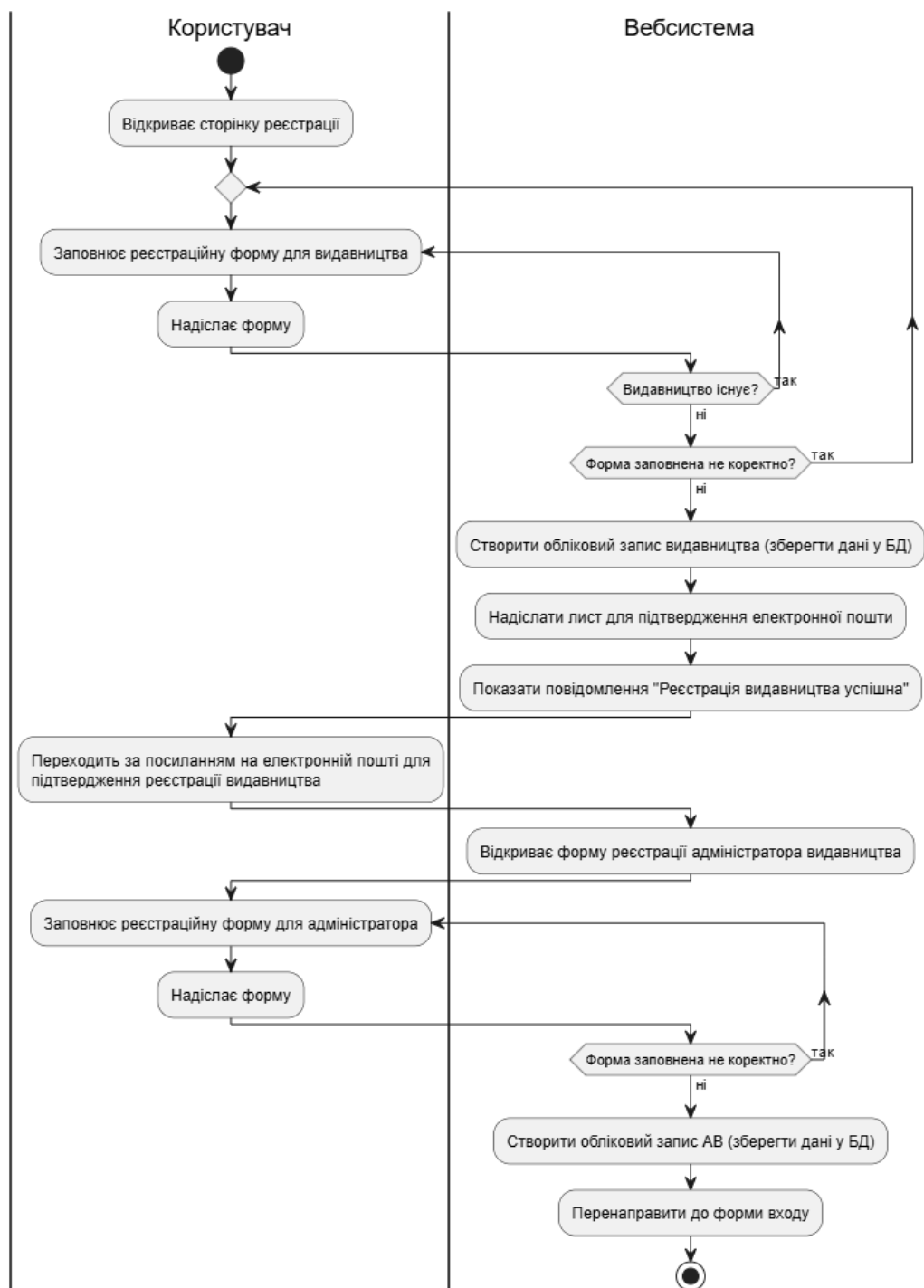


Рис. А.5. Діаграма активності для реєстрації видавництва та його адміністратора