

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:

«Сервіс з надання послуг електронної бібліотеки»

Виконав студент групи ІПЗ-20-1
РЯБОВА Єлизавета Вадимівна

Керівник роботи:
ВЛАСЕНКО Олег Васильович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2024

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
«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **РЯБОВА Єлизавета Вадимівна**

Керівник роботи: **ВЛАСЕНКО Олег Васильович**

Тема роботи: **«Сервіс з надання послуг електронної бібліотеки»,**

затверджена Наказом закладу вищої освіти від «23» лютого 2024 р., №74/с

Вихідні дані для роботи: **об'єктом дослідження є процеси, пов'язані з управлінням та наданням доступу до бібліотечних ресурсів через електронні платформи, а предметом дослідження є особливості організації та функціонування електронних бібліотечних систем, аналіз їхніх можливостей щодо забезпечення ефективного доступу користувачів до інформаційних ресурсів.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Власенко О.В.	27.02.2024р.	27.02.2024р.
2	Власенко О.В.	27.02.2024р.	27.02.2024р.
3	Власенко О.В.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з веб- і мобільного додатку сервісу та пояснювальної записки. Пояснювальна записка до випускної роботи містить текстову частину, викладену на 78 сторінках друкованого тексту. Список використаних джерел містить 26 найменувань і займає 3 сторінки. У роботі наведено 57 рисунків та 17 таблиць. Також у роботі є 24 сторінок додатків. Загальний обсяг роботи – 102 сторінок.

Метою роботи є створення сервісу, який надасть можливість зручно купувати, згруповувати та переглядати книги користувачам мобільного додатку та забезпечить ефективне керування доступними даними адміністраторам у веб-додатку.

Для реалізації сервісу було обрано клієнт-серверну архітектуру.

Використані технології – ASP.NET, React та React Native.

Сервіс пропонує зручний інтерфейс, що забезпечує легкість у пошуку та виборі книг, організації їхнього читання.

КЛЮЧОВІ СЛОВА: ВЕБ-ДОДАТОК, МОБІЛЬНИЙ ДОДАТОК, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, REACT NATIVE, REACT, ASP.NET.

					ІІЗ.КР.Б-121-24-ІІЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Сервіс з надання послуг електронної бібліотеки. Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Рябова С.В.					3	102
Керівник		Власенко О.В.				Житомирська політехніка, група ІІЗ-20-2		
Н. контр.		Єфремов Ю.М.						
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The Bachelor's qualification work consists of a web application, a mobile application service, and an explanatory note. The explanatory note for the thesis contains the textual part, presented in 78 pages of printed text. The list of references includes 26 titles and occupies 3 pages. The work contains 57 figures and 17 tables. There are also 24 pages of appendices in the work. The total volume of the work is 102 pages.

The thesis aims to develop a service facilitating easy book purchase, categorization, and browsing for users via the mobile application. Simultaneously, it ensures effective data management for administrators through the web application.

The implementation utilizes a client-server architecture. Key technologies employed include ASP.NET for the web application, React for the frontend development, and React Native for the mobile application.

The service provides a user-friendly interface that simplifies book searching, selection, and reading organization.

KEYWORDS: WEB APPLICATION, MOBILE APPLICATION, CLIENT-SERVER ARCHITECTURE, REACT NATIVE, REACT, ASP.NET.

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1 АНАЛІЗ НАПРЯМІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СЕРВІСУ	9
1.1 Постановка задачі	9
1.2 Аналіз аналогів програмного продукту	11
1.3 Вибір архітектури сервісу	19
1.4 Обґрунтування вибору засобів реалізації та вимоги до апаратного забезпечення	21
Висновки до першого розділу.....	23
РОЗДІЛ 2 ПРОЄКТУВАННЯ СЕРВІСУ	25
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи	25
2.2 Розробка бази даних системи.....	33
2.3 Проєктування та реалізація алгоритмів системи	39
2.4 Реалізація сервісу	43
Висновки до другого розділу.....	50
РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ ДЛЯ ОРГАНІЗАЦІЇ ТА ВИКОНАННЯ ЗАВДАНЬ	51
3.1 Порядок встановлення та налаштування параметрів сервісу.....	51
3.2 Структура інтерфейсу сервісу	52
3.3 Тестування сервісу	69
Висновки до третього розділу	73
ВИСНОВКИ.....	74
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	75
ДОДАТКИ.....	78

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – інтерфейс застосунків

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

CRUD – основні операції, які використовуються для управління даними
(створення, читання, оновлення, видалення)

ORM – об'єктно-реляційне відображення (Object-relational mapping)

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

У сучасному інформаційному суспільстві інформаційні технології набувають все більшого значення, ставши невід'ємною складовою життя. Однією з ключових сфер, де інформаційні технології відіграють провідну роль, є управління знаннями та забезпечення доступу до інформації. Ця тенденція особливо помітна в контексті електронних бібліотек. Електронні бібліотеки дозволяють ефективно керувати значними обсягами інформації, забезпечуючи швидкий та зручний доступ до різноманітних джерел знань. Крім того, такі сервіси активно використовуються книгарнями для реалізації не лише паперових, а й електронних книг. Особливого поширення набули мобільні додатки для придбання та читання електронних книг.

Метою даної роботи є розробка сервісу електронної бібліотеки, що включатиме веб- та мобільний додатки, які забезпечуватимуть зручний доступ до бібліотечних ресурсів, ефективне управління контентом та покращений користувацький досвід. Основним результатом має стати створення інтегрованої системи, яка підтримуватиме основні функції електронної бібліотеки. Ця розробка забезпечить користувачам безперешкодний доступ до літературних надбань у зручному цифровому форматі.

Для досягнення поставленої мети було визначено такі **завдання**:

1. Провести аналіз потреб користувачів та існуючих рішень для електронних бібліотек та виявити їхні переваги і недоліки.
2. Розробити архітектуру системи, яка включатиме веб- та мобільний додатки.
3. Реалізувати бекенд-систему з використанням технології ASP.NET та SQL Server.
4. Розробити фронтенд-частину веб-додатку на основі React.
5. Створити мобільний додаток, використовуючи React Native.
6. Провести ретельне тестування системи для забезпечення її коректної роботи.

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

Об'єктом дослідження є процеси та явища, пов'язані з управлінням та наданням доступу до бібліотечних ресурсів через електронні платформи.

Предметом дослідження є вивчення особливостей організації та функціонування електронних бібліотечних систем, аналіз їхніх можливостей щодо забезпечення ефективного доступу користувачів до інформаційних ресурсів. Розглядаються питання управління електронними колекціями, створення та підтримки користувацьких облікових записів, розробки зручних інтерфейсів для пошуку та перегляду матеріалів.

Розроблений сервіс електронної бібліотеки буде корисним для магазинів, які потребують ефективного управління та доступу користувачів до наданих товарів у вигляді електронних книг. Впровадження даного сервісу дозволить підвищити продуктивність роботи з ресурсами, забезпечити зручний та оперативний доступ до необхідної інформації, а також покращити загальний рівень обслуговування та задоволеності клієнтів.

РОЗДІЛ 1 АНАЛІЗ НАПРЯМІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СЕРВІСУ

1.1 Постановка задачі

Назва системи. Сервіс з надання послуг електронної бібліотеки.

Призначення системи. Система призначена для створення та підтримки ефективного сервісу електронної бібліотеки, який надає користувачам зручний та швидкий доступ до літературних ресурсів у цифровому форматі.

Мета. Впровадження сервісу електронної бібліотеки дозволить забезпечити безпроблемний доступ до різноманітних літературних ресурсів, спростить процес пошуку та отримання електронних книг, створить можливість для користувачів користуватись зручними інструментами пошуку та сортування матеріалів. Крім того, сервіс дозволить створювати особисті бібліотеки, відзначати прочитані твори та обмінюватися враженнями й рекомендаціями.

Вхідні дані. Інформація про літературні ресурси, дані користувачів.

У процесі реалізації проекту потрібно виконати такі завдання:

1. Провести аналіз потреб користувачів та існуючих рішень для електронних бібліотек та виявити їхні переваги і недоліки.

У рамках цього необхідно провести дослідження ринку електронних бібліотек, зокрема вивчити популярні сервіси, такі як YakaBoo та Google Play Книги. Важливо визначити основні потреби користувачів, їхні очікування та проблеми, з якими вони стикаються при використанні таких сервісів. Аналіз існуючих рішень дозволить виявити сильні та слабкі сторони конкурентних продуктів, а також зрозуміти, які функціональні можливості користувачі вважають найважливішими, що дозволить сформулювати чіткі вимоги до нової системи, орієнтовані на покращення користувацького досвіду.

2. Розробити архітектуру системи, яка включатиме веб- та мобільний додатки.

Створення архітектури системи є ключовим етапом, що визначає

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

структуру та взаємодію всіх компонентів програмного комплексу. Архітектура повинна бути розроблена таким чином, щоб забезпечити ефективну взаємодію між серверною частиною, веб-додатком і мобільним додатками.

3. Реалізувати бекенд-систему з використанням технології ASP.NET та SQL Server.

Наступним завданням є створення серверної частини системи, яка буде обробляти запити від користувачів та управляти даними. Використання ASP.NET дозволить створити надійний та масштабований бекенд, який легко інтегрується з базою даних SQL Server. Важливо забезпечити ефективну роботу з даними, розробити API для взаємодії з фронтенд-додатками та реалізувати механізми аутентифікації та авторизації користувачів.

4. Розробити фронтенд-частину веб-додатку на основі React.

Фронтенд веб-додатку має бути розроблений з урахуванням вимог до зручності та інтуїтивності інтерфейсу. Використання бібліотеки React дозволить створити динамічний та швидкий інтерфейс, що забезпечить користувачам зручний доступ до всіх функцій системи. Веб-додаток має включати функції пошуку, сортування, перегляду та завантаження літературних ресурсів.

5. Створити мобільний додаток, використовуючи React Native.

Для забезпечення доступу до електронної бібліотеки з мобільних пристроїв необхідно розробити додаток на основі React Native, що дозволить створити кросплатформений додаток, який працюватиме як на iOS, так і на Android.

6. Провести ретельне тестування системи для забезпечення її коректної роботи.

Завершальним етапом є тестування всіх компонентів системи для виявлення та усунення можливих помилок. Необхідно провести функціональне тестування для перевірки коректності роботи всіх функцій

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

1.2 Аналіз аналогів програмного продукту

Важливим етапом у процесі підготовки до розробки програмного продукту є аналіз доступних на ринку аналогів. Завдяки отриманій інформації, ми матимемо можливість побачити, що саме очікують користувачі від мобільного додатку-бібліотеки. Основною метою аналізу є вивчення функціональних можливостей, які пропонують ці продукти, а також визначення їхніх переваг та недоліків. Такий підхід дозволить не тільки краще зрозуміти поточний стан ринку, але й виявити прогалини та можливості для вдосконалення власної розробки.

Google Play Книги

Google Play Книги [1] - це мобільний додаток, що дозволяє користувачам отримати доступ до різноманітних електронних та аудіокниг. Він пропонує зручний інтерфейс для швидкого пошуку та читання книг, а також надає ряд функцій для організації та взаємодії з контентом (рис. 1.1).

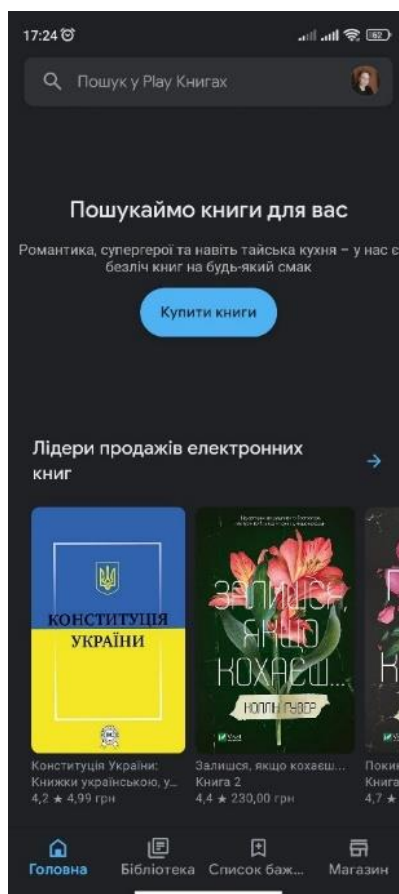


Рисунок 1.1 – Головна сторінка Google Play Книги

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

Користувачі можуть легко знаходити книги за різними категоріями та встановлювати фільтри для точного вибору матеріалів. Сторінка книги (рис. 1.2) містить усю важливу інформацію, дизайн мінімалістичний і зрозумілий. Перегляд книги зручний, у повноекранному режимі для читання і з переглядом сторінок окремо для налаштування вигляду (рис. 1.3).

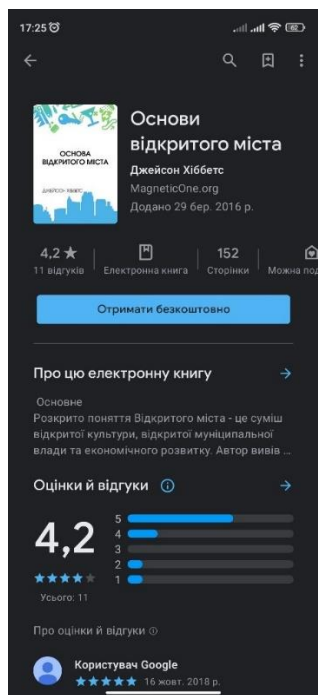


Рисунок 1.2 – Сторінка книги Google Play Книги



Рисунок 1.3 – Інтерфейс для читання книги Google Play Книги

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Переваги:

- Зручний інтерфейс;
- Рекомендації на основі прочитаних книг;
- Можливість додавати книги у список бажань, який можна сортувати за кількома категоріями, та створювати власні полиці;
- Наявність аудіокниг;
- Наявність оцінки на основі відгуків;
- Детальні налаштування додатку;
- Зручний інтерфейс для читання;
- Завантаження книг для перегляду офлайн;
- Доступна синхронізація на кількох пристроях через обліковий запис.

Недоліки:

- Відсутність сортування результатів пошуку.

Yakaboo

Мобільний додаток відомої інтернет-книгарні Yakaboo [2] надає можливість переглядати куплені книжки прямо в додатку. На відміну від решти розглянутих аналогів, Yakaboo пропонує лише один спосіб входу – через номер телефону. Інтерфейс додатку зрозумілий (рис. 1.4), є поділ на категорії (рис. 1.5), проте відсутні фільтрація і сортування. Користувач може переглянути прочитані та завантажені книги, а також нотатки і відгуки, у бібліотеці. Крім того, при наявності промокоду, можна отримати доступ до колекцій, які пропонують окремі компанії, що відрізняє цей додаток від решти (рис. 1.6).

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

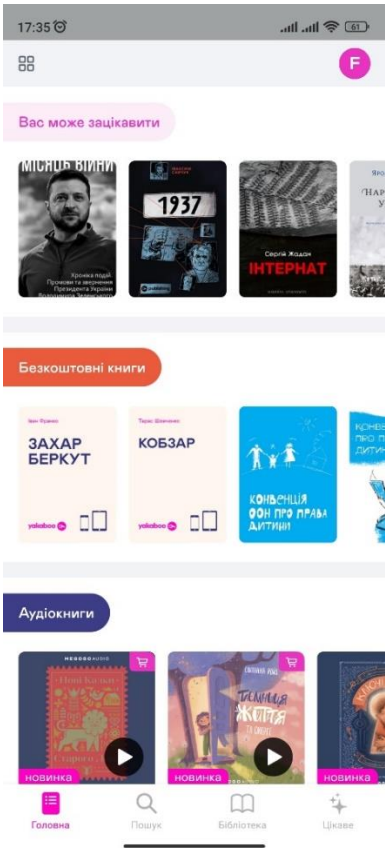


Рисунок 1.4 – Головна сторінка Yakaboo



Рисунок 1.5 – Сторінка категорій Yakaboo

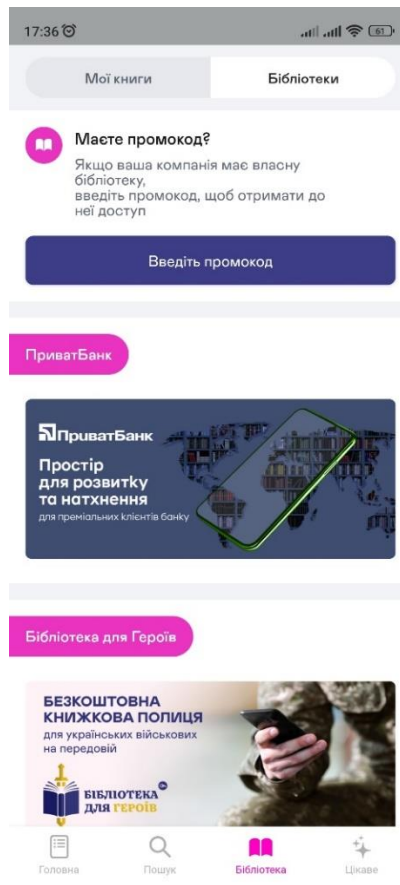


Рисунок 1.6 – Бібліотеки Yakaboo

Переваги:

- Зрозумілий інтерфейс;
- Можливість додавати книги у список бажань;
- Зручний інтерфейс для читання;
- Можливість оцінити книгу та залишити відгук;
- Зручний інтерфейс для читання;
- Завантаження книг для перегляду офлайн;
- Доступна синхронізація на кількох пристроях через обліковий запис.

Недоліки:

- Відсутність рекомендацій на основі прочитаних книг;
- Відсутність сортування;
- Відсутність можливості створити колекцію;

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

– Відсутність персональних рекомендацій.

Букнет

Наступний аналог – це онлайн-платформа для самвидаву «Букнет», а саме їхній мобільний додаток [3]. Більшість книг викладені користувачами, але також є твори, які входять до шкільної програми. Для читання підписка не потрібна, проте за потреби автори мають можливість створити передплату чи продавати свої книги. Інтерфейс додатку простий і зрозумілий (рис. 1.7).

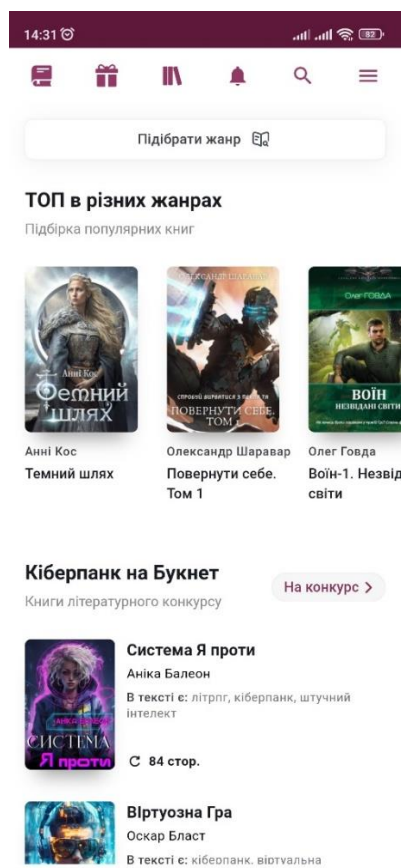


Рисунок 1.7 – Головна сторінка Букнет

Окрім меню зверху, є також бургер-меню з більшим вибором дій (рис. 1.8). Користувач може лишати коментарі, додавати книги у вже створені колекції, такі як «Хочу прочитати» чи «Сподобалось», або створити власні, особисті чи публічні. Перегляд тексту книги простий, є можливість прослухати автоматично створене аудіо, налаштувати шрифт, відступи тощо.

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

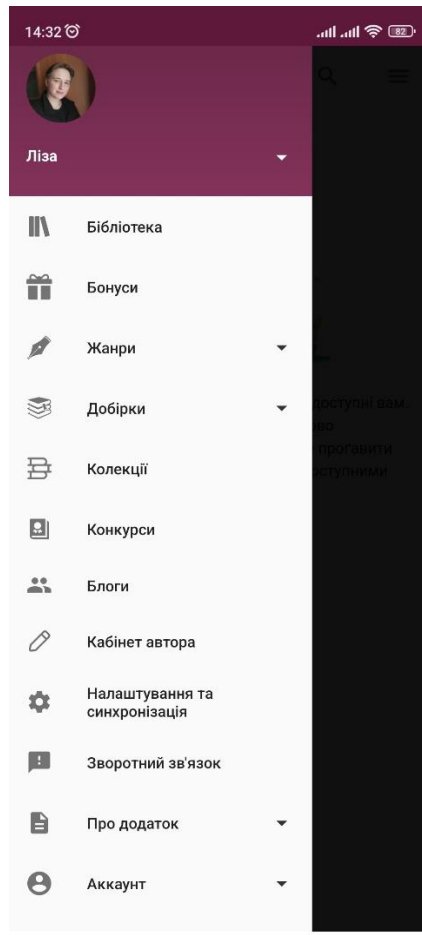


Рисунок 1.8 – Меню Букнет

Переваги:

- Простий інтерфейс;
- Рекомендації на основі прочитаних книг;
- Можливість створювати добірки;
- Зручний інтерфейс для читання;
- Можливість викладати та монетизувати власні твори;
- Доступна синхронізація на кількох пристроях через обліковий запис.

Недоліки:

- Більшість книг – самвидав, що зменшує цільову аудиторію додатку;
- Відсутність аудіокниг;
- Відсутність сортування;

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

– Відсутність можливості оцінити книгу.

Враховуючи проведений аналіз, складено порівняльну таблицю для розглянутих додатків (табл.1.1). Кращими з 3-х додатків є Google Play Книги та Yakaboo, тому при розробці будуть враховуватись їхні кращі сторони з метою реалізації якісного програмного продукту.

Таблиця 1.1

Порівняння аналогів

Критерії	Google Play Книги	Yakaboo	Букнет
Зручність інтерфейсу	5/5	5/5	4/5
Наявність аудіокниг	+	+	-
Фільтрація і сортування	Наявна при перегляді жанрів у каталозі, але відсутня для результатів пошуку	Фільтрація присутня лише для деяких категорій у каталозі	Наявне сортування за популярністю чи за оновленням, фільтрація відсутня
Організація каталогу	Поділ на жанри, рекомендації поділені на популярні, спеціальні пропозиції, новинки, лідери продажів серед книг	Поділ на жанри, рекомендації поділені на категорії «Вас може зацікавити», «Безкоштовні книги» та «Аудіокниги» на головній сторінці	Поділ на жанри, рекомендації популярних книг, новинок тощо на головній сторінці
Оцінювання і рецензування	+	+	+/-, відсутня можливість залишити оцінку
Збереження матеріалів	Наявна можливість створення добірок, є такі добірки як список бажань, переглянуті книги,	«Також читаю зараз» і список бажань, окремо винесені збережені цитати і написані відгуки	Велика кількість автоматично створених добірок, є можливість створити власні

	заплановано		
Завантаження матеріалів	+	+	+
Створення колекцій	+	-	+
Персональні рекомендації	-	-	-
Налаштування перегляду тексту	+	+	+
Синхронізація між кількома пристроями	+	+	+
Способи реєстрації	Лише акаунт Google	Лише за номером телефону	Акаунти Google, Facebook або Букнет
Особливості	Можливість детальної фільтрації бібліотеки користувача	Можливість мати доступ до корпоративної бібліотеки за промокодом	Можливість викладати власні твори

1.3 Вибір архітектури сервісу

Вибір правильної архітектури додатку є ключовим етапом розробки, оскільки це визначає його подальшу масштабованість та підтримку. Правильна архітектура сприяє легкості обслуговування та управління кодом, полегшує додавання нового функціоналу і реагування на зміни. Вона також впливає на продуктивність, безпеку, тестування та можливість інтеграції з іншими системами. Обрана методологія повинна враховувати розмір та складність проекту, а також враховувати технічні вимоги та стратегії розвитку. Все це робить вибір архітектури важливим аспектом успіху проекту та його здатності адаптуватися до змін та зростання [4].

Наш програмний продукт складатиметься з трьох основних частин – веб-додаток для адміністрування, мобільний додаток для клієнтів-користувачів системи та API.

Загалом, сервіс може бути реалізований із застосуванням одним із 3-х видів архітектури, які найбільше підходять до його потреб [5]:

- монолітна архітектура - всі компоненти програми (веб-додаток, мобільний додаток та API) об'єднані в один блок. Це робить її простою у розробці та розгортанні, завдяки чіткій структурі та централізованому управлінню. Керування та оновлення коду полегшується завдяки його локалізації в одному місці. Ця архітектура добре підходить для невеликих проектів з чітко визначеними та стабільними вимогами [6];
- мікросервісна архітектура - програмний комплекс розбивається на незалежні, слабо зв'язані мікросервіси, кожен з яких виконує певну вузькоспеціалізовану функцію. Це забезпечує високу масштабованість, адже горизонтальне масштабування окремих мікросервісів дозволяє гнучко реагувати на зростання навантаження. Внесення змін до одного мікросервісу не впливає на інші, роблячи цю архітектуру гнучкою та стійкою до збоїв [7];
- клієнт-серверна архітектура - це підхід до розробки програмного забезпечення, за якого ПЗ поділене на два основних компоненти: клієнт (веб-додаток або мобільний додаток) та сервер, який обробляє запити клієнта та надсилає відповіді. Ця архітектура проста та зрозуміла, що робить її легкою для розробки. Масштабування серверної частини для обробки зростаючого навантаження не складає проблем. Чіткий розподіл функціоналу між клієнтом та сервером полегшує розробку та тестування [8].

У нашому випадку важливо враховувати гнучкість під час розробки, легкість масштабування, надійність системи і можливість використовувати різні технології.

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

Монолітна архітектура, зазвичай, характеризується єдиною кодовою базою, де всі компоненти тісно взаємодіють один з одним. Хоча така архітектура може бути простою у впровадженні на початкових етапах проекту, вона має суттєві обмеження. Основна проблема полягає у відсутності гнучкості та труднощах з масштабуванням. Зміни в одному модулі часто вимагають переробки інших модулів, що ускладнює підтримку та розвиток системи. До того ж, масштабування моноліту зазвичай означає масштабування всієї системи, що є неефективним з точки зору ресурсів.

З іншого боку, мікросервісна архітектура пропонує розділення додатка на незалежні сервіси, кожен з яких відповідає за конкретну функціональність. Це дозволяє досягти високої гнучкості, легкості масштабування та використання різних технологій для різних сервісів. Однак, впровадження мікросервісної архітектури має свої виклики. Вона вимагає значних ресурсів для управління та координації окремих сервісів, розробка стає складнішою через необхідність налагодження міжсервісної взаємодії, а також виникає потреба у забезпеченні високої надійності кожного сервісу. Для невеликого проекту або системи з обмеженим функціоналом ці витрати можуть бути непропорційно великими.

Враховуючи ці аспекти, найкращим рішенням буде використання клієнт-серверної архітектури. Вона забезпечує розділення функціональності між клієнтом та сервером, що дозволяє досягти прийнятного рівня гнучкості та масштабованості без зайвих ускладнень.

1.4 Обґрунтування вибору засобів реалізації та вимоги до апаратного забезпечення

Під час вибору інструментальних засобів взято до уваги отриману в ході проведеного аналізу інформацію та наш досвід роботи з доступними технологіями. Тому було вирішено використовувати ASP.NET для реалізації

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

серверної частини, React та React Native для клієнтської частини і СУБД SQL Server.

ASP.NET [9] – потужний фреймворк .NET, який забезпечує високу продуктивність, масштабованість та інтеграцію з іншими продуктами Microsoft, що полегшує розробку та управління додатком. Обидва компоненти клієнтської частини сервісу – веб-додаток та мобільний додаток – використовуватимуть одне API, розроблене на ASP.NET. Це рішення спрощує розробку та підтримку, адже API потребує лише одноразового написання та тестування.

React [10] значно полегшує розробку веб-додатка завдяки компонентній архітектурі, що дає змогу створювати динамічні інтерфейси, та широкий підтримці спільноти розробників. Разом з ASP.NET, React створює потужний стек для розробки веб-додатків.

React Native [11] – це фреймворк для розробки мобільних додатків, який використовує JavaScript для створення нативних додатків для iOS та Android. Оскільки React Native використовує одну мову програмування для розробки додатків для обох платформ, це полегшує процес розробки та підтримки. Крім того, використання React Native дозволяє використовувати багато переваг, що притаманні нативним додаткам, такі як висока швидкодія, доступ до апаратних можливостей пристроїв та можливість використання платформо-специфічних функцій.

Вибір SQL Server [12] в якості системи управління базами даних для електронної бібліотеки обумовлений кількома факторами. SQL Server відомий своєю високою надійністю і стабільністю та пропонує різноманітні варіанти оптимізації для підвищення продуктивності, такі як оптимізація запитів, індексація та кешування, що дозволяє забезпечити швидкий доступ до даних, навіть при роботі з великим обсягом інформації. SQL Server має глибоку інтеграцію з іншими продуктами Microsoft, що полегшує розробку та управління додатками. Крім того, він забезпечує високий рівень безпеки даних, що є критичним для захисту конфіденційної інформації, зокрема в

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

електронній бібліотеці.

Вибір ASP.NET, React, React Native та SQL Server для розробки сервісу електронної бібліотеки ґрунтується на ретельному аналізі та нашому досвіді роботи з доступними технологіями. Цей стек технологій гарантує високу продуктивність, масштабованість, безпеку, зручність розробки та підтримки.

Мінімальні системні вимоги до пристрою користувача:

- 512 Mb пам'яті;
- 128 Mb вільного дискового простору;
- Процесор з тактовою частотою не нижче 1GHz;
- Операційна система Android або IOS.

Мінімальні системні вимоги до сервера:

- 128 Gb пам'яті;
- 1 Tb вільного дискового простору;
- Мінімум 8 Gb оперативної пам'яті;
- Процесор з тактовою частотою не нижче 2,5GHz;
- Операційна система Windows Server або Linux.

Висновки до першого розділу

Перший розділ роботи описує проведений комплексний аналіз для визначення стратегії розробки та впровадження електронної бібліотеки. В рамках цього розділу здійснено постановку задачі та аналіз потреб користувачів, що допомогло направити подальшу роботу у потрібний напрямок. Порівняння існуючих на ринку аналогів дало уявлення про те, чого очікуватимуть користувачі від програмного продукту. Також, вивчено можливі варіанти і зроблено вибір архітектури та інструментальних засобів, які найбільше підходять для реалізації сервісу.

У результаті виконаних аналітичних досліджень були визначені ключові напрямки подальшого розвитку проекту. Основним завданням є ефективна реалізація обраної архітектури та забезпечення функціонування сервісу.

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Важливими аспектами, на які варто звернути увагу в подальшому, є реалізація функціональності відповідно до потреб користувачів та забезпечення стабільної роботи сервісу.

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

РОЗДІЛ 2 ПРОЄКТУВАННЯ СЕРВІСУ

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

У цьому пункті описано фундаментальні аспекти розроблюваної системи, ґрунтуючись на результатах аналізу вимог. Для створення оптимальної системи, необхідно чітко визначити вимоги до розроблюваного сервісу.

Функціональні вимоги

Функціональні вимоги описують те, що система повинна робити, як вона повинна реагувати на дії користувачів та які функції вона має надавати.

У системі передбачено три типи користувачів:

- гість – має можливість увійти в акаунт або створити новий, щоб продовжити роботу як авторизований користувач;
- авторизований користувач – має доступ до базових функціональних можливостей системи;
- адміністратор – володіє правами, необхідними для управління системою.

Вимоги кожного з них відрізняються, тому будуть описано окремо.

Гість:

- авторизація;
- реєстрація.

Авторизований користувач:

- зміна особистої інформації;
- видалення акаунту;
- можливість вийти з акаунту;
- пошук книг та його сортування за назвою та ціною;
- перегляд списку категорій та книг, які до них належать;
- перегляд конкретної книги;

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

- читання книги в додатку;
- додавання книг у добірки
- перегляд списку добірок;
- перегляд конкретної добірки;
- створення та керування добірками;
- написання та видалення коментарів;
- зберігання входу в акаунт на 1 день;
- збереження налаштувань інтерфейсу читання;
- можливість прокрутки сторінок та перегляду номеру поточної сторінки й загальної кількості сторінок;
- збереження сторінки, яка була переглянута останньою.

Адміністратор:

- можливість вийти з акаунту;
- перегляд усіх книг;
- перегляд списку авторів і категорій, включаючи ті, до яких не належить жодна книга;
- пошук книг та його сортування;
- додавання та видалення книг, редагування загальної інформації та заміна файлів;
- додавання та керування авторами і категоріями;
- перегляд статистики;
- додавання нового адміністратора.

Нефункціональні вимоги

На відміну від функціональних вимог, які описують те, що система повинна робити, нефункціональні вимоги визначають як вона повинна це робити. Ці вимоги стосуються якості, ефективності та зручності використання системи, а також інших характеристик, які впливають на її загальну цінність для користувачів.

1. Продуктивність:

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

- 1.1. Система повинна забезпечувати швидке завантаження сторінок та книг.
- 1.2. Пошук книг повинен бути швидким та ефективним.
2. Надійність:
 - 2.1. Система повинна бути доступна для користувачів у цілодобовому режимі.
3. Безпека:
 - 3.1. Доступ до системи повинен бути захищений надійними методами автентифікації.
 - 3.2. Дані мають передаватися через захищені канали зв'язку.
4. Масштабованість:
 - 4.1. Система повинна мати можливість працювати із значною кількістю користувачів та книг.

Вимоги до часових характеристик реакції системи на дії користувачів

Вимоги до часових характеристик визначають час, який система повинна витратити на виконання певних дій у відповідь на дії користувачів. Ці вимоги важливі для забезпечення швидкої та чутливої роботи системи, що робить її зручною та приємною для користувачів.

1. Час відгуку:
 - 1.1. Натискання кнопки:
 - 1.1.1. Мобільний додаток: менше 1 секунди.
 - 1.1.2. Веб-додаток: менше 2 секунд.
 - 1.2. Перехід між сторінками:
 - 1.2.1. Мобільний додаток: менше 2 секунд.
 - 1.2.2. Веб-додаток: менше 3 секунд.
 - 1.3. Пошук книг:
 - 1.3.1. Мобільний додаток: менше 1 секунди.
 - 1.3.2. Веб-додаток: менше 2 секунд.

2. Час завантаження:

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

2.1. Книги:

2.1.1. Мобільний додаток: до 10 секунд

2.2. Зображень:

2.2.1. Мобільний додаток: менше 3 секунд.

2.2.2. Веб-додаток: менше 3 секунд.

3. Читання:

3.1. Перегортання сторінок: Плавне та без затримок.

Варіанти використання системи

Варіанти використання - це описи послідовності дій, які користувач виконує для досягнення певної мети за допомогою системи (рис. 2.1). На їхній основі будується логічна структура сервісу, що дає уявлення, які компоненти потрібні, як вони взаємодіють між собою та які дані їм потрібні. Це робить проектування більш чітким, структурованим та послідовним. Варіанти використання також слугують тестовими сценаріями, на основі яких можна перевірити, чи правильно працює сервіс.

Детальніше розглянемо можливості системи на діаграмі варіантів використання [13]:

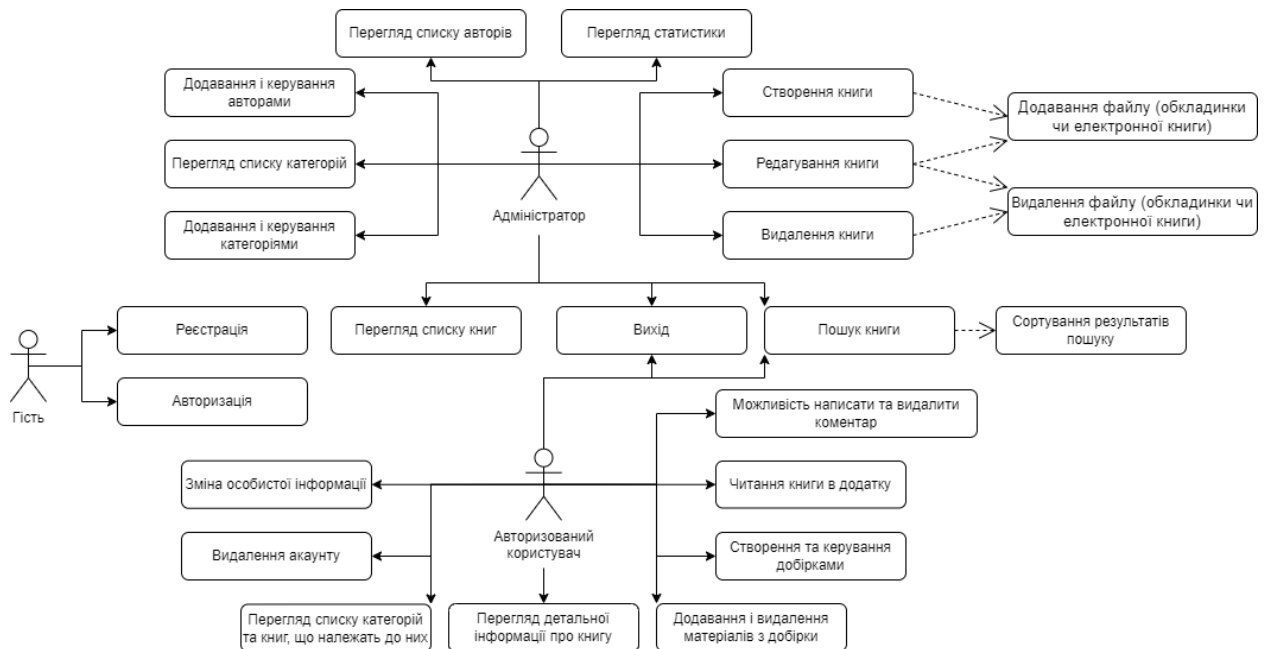


Рисунок 2.1 - Діаграма варіантів використання

Опис прецедентів діаграми варіантів використання (рис. 2.1) наведено в табл. 2.1.

Таблиця 2.1

Опис прецедентів

Основні актори	Короткий опис
Реєстрація	
Гість	Гість може створити новий акаунт і продовжити роботу з мобільним додатком як авторизований користувач.
Авторизація	
Гість	Гість може увійти в існуючий акаунт і продовжити роботу з мобільним додатком як авторизований користувач або з вебдодатком як адміністратор.
Вихід	
Авторизований користувач, адміністратор	Авторизований користувач та адміністратор можуть вийти зі свого акаунту.
Перегляд списку книг	
Адміністратор	Адміністратор може переглянути загальний список усіх книг.
Пошук книги	
Авторизований користувач, адміністратор	Користувач може ввести ключове слово або словосполучення для того, щоб знайти потрібну книгу.
Сортування результатів пошуку	
Авторизований користувач, адміністратор	Користувач може обрати спосіб сортування результатів пошуку.
Перегляд списку категорій	
Адміністратор	Користувач може переглянути усі створення категорії, включаючи ті, до яких не належить жодна книга.
Перегляд списку категорій та книг, що належать до них	
Авторизований користувач	Користувач може переглянути усі створення категорії, до яких належить хоча б одна книга. Якщо він обере одну з категорій, йому буде показано список книг, що належать до неї.

Додавання і керування категоріями	
Адміністратор	Користувач може додати нову або редагувати чи видалити існуючу категорію.
Перегляд списку авторів	
Адміністратор	Користувач може переглянути список авторів.
Додавання і керування авторами	
Адміністратор	Користувач може додати нового або редагувати чи видалити існуючого автора.
Перегляд статистики	
Адміністратор	Користувач може переглянути статистику використання сервісу.
Створення книги	
Адміністратор	Користувач може додати нову книгу, що також включає в себе додавання файлу-обкладинки та електронної книги.
Редагування книги	
Адміністратор	Користувач може редагувати книгу – змінити інформацію про неї або один з файлів. Включає в себе додавання та видалення файлів.
Видалення книги	
Адміністратор	Користувач може видалити книгу, що також видалить пов'язані з нею файли.
Додавання файлу (обкладинки чи електронної книги)	
Адміністратор	Система додає файл, зображення чи документ, як частину створення книги.
Видалення файлу (обкладинки чи електронної книги)	
Адміністратор	Система видаляє файл, зображення чи документ, як частину редагування чи видалення книги.
Зміна особистої інформації	
Авторизований користувач	Користувач може змінити свою інформацію – ім'я та прізвище.
Видалення акаунту	
Авторизований користувач	Користувач може видалити свій акаунт.

Можливість написати та видалити коментар	
Авторизований користувач	Користувач може написати свої враження про книгу та, за потреби, видалити свій коментар.
Читання книги в додатку	
Авторизований користувач	Користувач може читати електронну книгу в додатку. В інтерфейсі читання можна налаштувати перегляд по сторінках та горизонтальну чи вертикальну прокрутку. Також, є індикатор прогресу, на якому можна змінити сторінку, із зазначенням поточної сторінки та загальної кількості сторінок. Остання сторінка і обрані налаштування зберігаються.
Створення та керування добірками	
Авторизований користувач	Користувач може створювати та видаляти добірки, змінювати їхнє ім'я.
Перегляд детальної інформації про книгу	
Авторизований користувач	Користувач може побачити детальний опис книги, перейшовши на сторінку книги. Там він має можливість додати книгу до добірки, лишити коментар чи почати читати.
Додавання і видалення матеріалів з добірки	
Авторизований користувач	Користувач може додавати та видаляти книги з добірок.

З описом вимог та варіантів використання, маємо чітке уявлення про те, що має робити система, для кого вона призначена та які дані їй потрібні. Ця інформація є фундаментом для побудови діаграми класів, яка візуалізує структуру системи з точки зору об'єктно-орієнтованого програмування.

Усі елементи поєднані між собою за допомогою різних видів зв'язків:

- Асоціація – довільний зв'язок, може бути напрямленою, тоді стрілка вказує напрямок взаємодії від одного об'єкта до іншого;
- Наслідування – відношення між більш загальним (батьківським) і менш загальним (дочірнім) класом. Підклас успадковує атрибути та методи батьківського класу, при цьому може доповнювати їх новими функціональностями;

- Агрегація – відношення між класами, яке можна описати як «частина-ціле»;
- Композиція – вид агрегації, при якому частини мають такий самий час життя, як і ціле. Якщо ціле зникає, то його частини також припиняють існування.

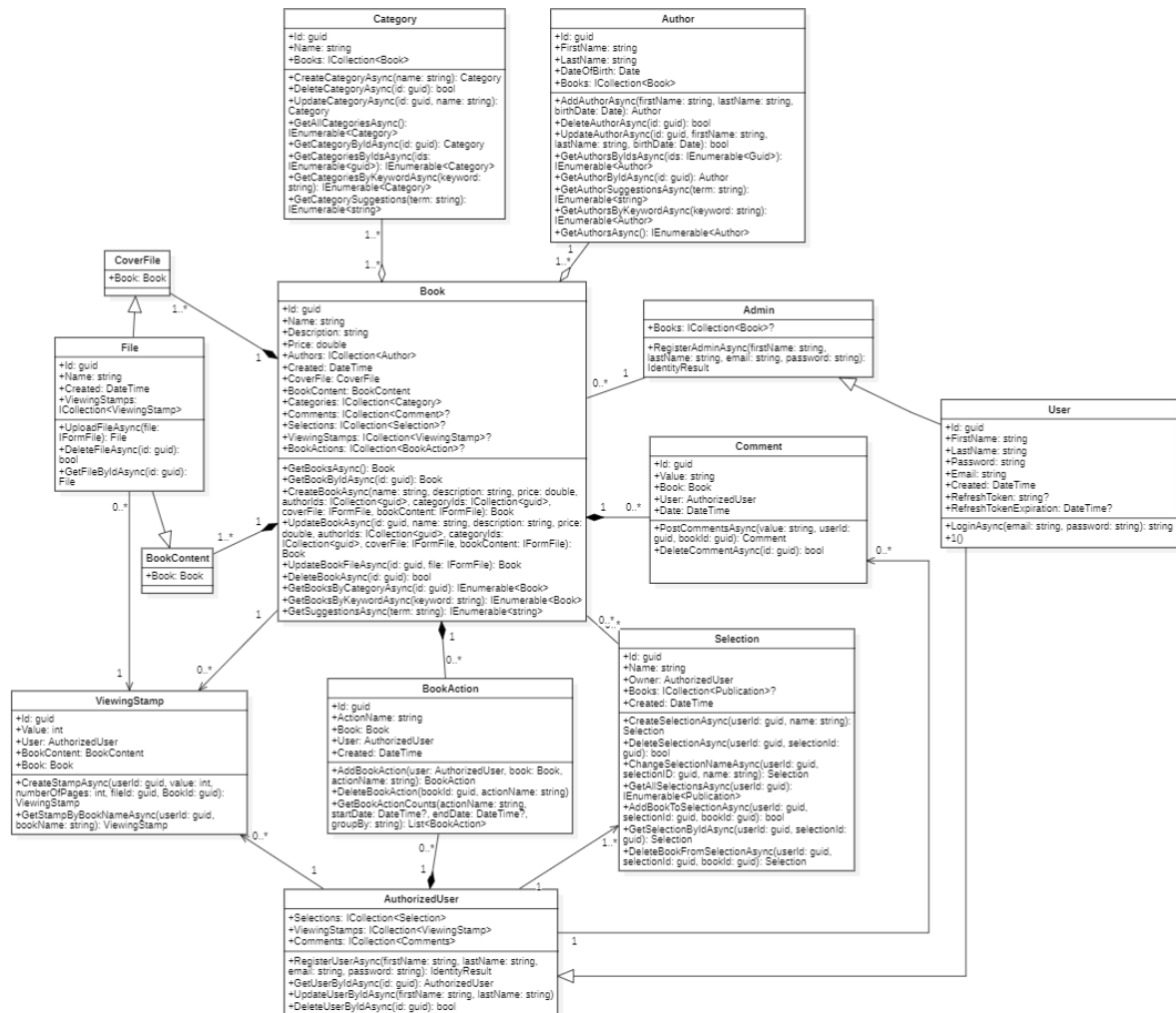


Рисунок 2.2 – Діаграма класів

Для забезпечення оптимальної роботи системи були визначені такі класи:

1. Book – інформація про електронну книгу. Пов'язаний з класами BookContent, CoverFile, Author, Category, Comment, Selection, ViewingStamps та BookAction.

2. File – описує зображення чи текстовий файл, прикріплені до книги. Пов'язаний асоціативним зв'язком з ViewingStamp, наслідується класами BookContent та CoverFile.
3. BookContent – описує текстовий файл, саму книгу, наслідує клас File.
4. CoverFile – описує зображення-обкладинку книги, наслідує клас File.
5. ViewingStamp – позначає сторінку, де зупинився користувач при перегляді файлу, пов'язаний з BookContent, Book і AuthorizedUser.
6. Author – позначає автора, пов'язаний агрегацією з Book.
7. Category – позначає жанр, пов'язаний агрегацією з Book.
8. Comment – описує коментар, пов'язаний з Book і AuthorizedUser.
9. Selection – зображує добірки, в які авторизовані користувачі можуть зберігати публікації, пов'язаний з AuthorizedUser і Book.
10. User – загальний клас, який описує користувача, наслідується класами Admin та AuthorizedUser.
11. AuthorizedUser – авторизований користувач, доповнює клас User, пов'язаний з Comment, Selection та ViewingStamp.
12. Admin – адміністратор, доповнює клас User, пов'язаний з Book.
13. BookAction – описує дії з книгами, які будуть використані для створення статистики, пов'язаний з Book та AuthorizedUser.

2.2 Розробка бази даних системи

Для збереження інформації сервісу використовується SQL Server. Це реляційна СУБД, розроблена компанією Microsoft, яка використовується для збереження, управління та обробки великих обсягів даних, забезпечуючи високу надійність, продуктивність та масштабованість.

БД складається з таких таблиць:

1. AspNetRoles – таблиця, створена фреймворком Identity для роботи з ролями.

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

2. AspNetUserRoles – створена фреймворком Identity, відповідає за збереження інформації щодо ролей користувачів.
3. AspNetUsers – створена фреймворком Identity, зберігає дані користувачів. Розширена користувацькою таблицею User.
4. AuthorBook – таблиця для реалізації зв'язку багато-до-багатьох між книжками та авторами.
5. Authors – таблиця для збереження даних авторів.
6. BookActions – таблиця для користувацьких дій над книжками, зберігає дату кожного додавання у добірку, коментаря та читання для оформлення статистики.
7. BookCategory – таблиця для реалізації зв'язку багато-до-багатьох між книжками та категоріями.
8. BookContents – таблиця для інформації про файл електронної книги.
9. Books – загальна інформація про книгу.
10. BookSelection – таблиця для реалізації зв'язку багато-до-багатьох між книжками та добірками.
11. Categories – інформація про категорії.
12. Comments – таблиця для зберігання коментарів.
13. CoverFiles – таблиця для збереження інформації про файл-обкладинку.
14. Selections – інформація про добірки.
15. ViewingStamps – таблиця для збереження інформації про сторінку, на якій зупинився користувач, коли закрив книгу.

Опис кожної таблиці БД наведений у табл. 2.2 – 2.16.

Таблиця 2.2

Опис таблиці «AspNetRoles»

Поле	Тип	Призначення
Id	uniqueidentifier	Ідентифікатор
Name	nvarchar(256)	Назва ролі

ConcurrencyStamp	nvarchar(MAX)	Значення, яке відповідає за поточну версію рядка, використовується для запобігання конфліктам при оновленні даних
------------------	---------------	---

Таблиця 2.3

Опис таблиці «AspNetUserRoles»

Поле	Тип	Призначення
UserId	uniqueidentifier	Ідентифікатор користувача
RoleId	uniqueidentifier	Ідентифікатор ролі

Таблиця 2.4

Опис таблиці «AspNetUsers»

Поле	Тип	Призначення
Id	uniqueidentifier	Ідентифікатор
FirstName	nvarchar(MAX)	Ім'я користувача
LastName	nvarchar(MAX)	Прізвище
Created	datetime2(7)	Дата створення акаунту
Discriminator	nvarchar(21)	Поле на позначення класу, який використовується для даного користувача – Admin чи AuthorizedUser
UserName	nvarchar(256)	Ім'я користувача
Email	nvarchar(256)	Електронна пошта
PasswordHash	nvarchar(MAX)	Хеш паролю
SecurityStamp	nvarchar(MAX)	Спеціальний токен, який змінюється при виконанні певних операцій з обліковим записом користувача
ConcurrencyStamp	nvarchar(MAX)	Значення, яке відповідає за поточну версію рядка, використовується для

		запобігання конфліктам при оновленні даних
RefreshToken	nvarchar(MAX)	Токен для оновлення JWT токена
RefreshTokenExpiration	datetime2(7)	Дата закінчення терміну дії токена

Таблиця 2.5

Опис таблиці «AuthorBook»

Поле	Тип	Призначення
AuthorsId	uniqueidentifier	Ідентифікатор автора
BooksId	uniqueidentifier	Ідентифікатор книги

Таблиця 2.6

Опис таблиці «Authors»

Поле	Тип	Призначення
Id	uniqueidentifier	Ідентифікатор
FirstName	nvarchar(450)	Ім'я
LastName	nvarchar(450)	Прізвище
DateOfBirth	date	Дата народження

Таблиця 2.7

Опис таблиці «BookActions»

Поле	Тип	Призначення
Id	uniqueidentifier	Ідентифікатор
ActionName	nvarchar(MAX)	Назва дії
BookId	uniqueidentifier	Ідентифікатор книги
UserId	uniqueidentifier	Ідентифікатор користувача
Created	datetime2(7)	Дата виконання дії

Таблиця 2.8

Опис таблиці «BookCategory»

Поле	Тип	Призначення
BooksId	uniqueidentifier	Ідентифікатор книги
CategoriesId	uniqueidentifier	Ідентифікатор категорії

Таблиця 2.9

Опис таблиці «BookContents»

Поле	Тип	Призначення
Id	uniqueidentifier	Ідентифікатор
BookId	uniqueidentifier	Ідентифікатор книги
Name	nvarchar(MAX)	Ім'я файлу
FileType	nvarchar(MAX)	Тип файлу, в даному випадку буде «text»
Created	datetime2(7)	Час створення запису про файл

Таблиця 2.10

Опис таблиці «Books»

Поле	Тип	Призначення
Id	uniqueidentifier	Ідентифікатор
Name	nvarchar(MAX)	Назва
Description	nvarchar(MAX)	Опис
Created	datetime2(7)	Дата створення запису
Price	float	Ціна

Таблиця 2.11

Опис таблиці «BookSelection»

Поле	Тип	Призначення
BooksId	uniqueidentifier	Ідентифікатор книги
SelectionsId	uniqueidentifier	Ідентифікатор добірки

Таблиця 2.12

Опис таблиці «Categories»

Поле	Тип	Призначення
Id	uniqueidentifier	Ідентифікатор
Name	nvarchar(MAX)	Назва

Таблиця 2.13

Опис таблиці «Comments»

Поле	Тип	Призначення
Id	uniqueidentifier	Ідентифікатор
Value	nvarchar(MAX)	Текст коментаря
BookId	uniqueidentifier	Ідентифікатор книги
UserId	uniqueidentifier	Ідентифікатор користувача
Date	datetime2(7)	Дата написання

Таблиця 2.14

Опис таблиці «CoverFiles»

Поле	Тип	Призначення
Id	uniqueidentifier	Ідентифікатор
BookId	uniqueidentifier	Ідентифікатор книги
Name	nvarchar(MAX)	Ім'я файлу
FileType	nvarchar(MAX)	Тип файлу, в даному випадку буде «image»
Created	datetime2(7)	Час створення запису про файл

Таблиця 2.15

Опис таблиці «Selections»

Поле	Тип	Призначення
Id	uniqueidentifier	Ідентифікатор
Name	nvarchar(MAX)	Назва добірки
OwnerId	uniqueidentifier	Ідентифікатор користувача
Created	datetime2(7)	Дата створення добірки

Опис таблиці «ViewingStamps»

Поле	Тип	Призначення
Id	uniqueidentifier	Ідентифікатор
Value	int	Номер поточної сторінки
BookId	uniqueidentifier	Ідентифікатор книги
UserId	uniqueidentifier	Ідентифікатор користувача
BookContentId	uniqueidentifier	Ідентифікатор файлу електронної книги
NumberOfPages	int	Загальна кількість сторінок книги

У такому вигляді зберігаються всі дані сервісу.

2.3 Проєктування та реалізація алгоритмів системи

Для моделювання взаємодії з розроблюваною системою використано діаграми активності, які надають можливість візуалізувати ключові дії користувачів [14]. Цей метод дозволяє не лише систематизувати та узагальнити основні етапи взаємодії, а й забезпечує глибше розуміння роботи системи в контексті конкретних сценаріїв використання.

Діаграма активності включає ряд елементів, таких як:

- початковий вузол (круг) позначає початок виконання дії;
- контроль потоку (стрілка) використовується для визначення взаємодії між різними об'єктами і вказує напрямок потоку виконання;
- дія (заокруглений прямокутник) представляє конкретну роботу чи активність, яка виконується в рамках системи;
- розгалуження (ромб з підписом-умовою) вказує на точку прийняття рішення в залежності від виконання заданої умови;
- об'єднання (ромб) позначає об'єднання розгалужених шляхів. Цей елемент з'єднує різні варіанти виконання дій в єдиний потік;

- паралельність (риска, з якої виходять кілька стрілок) вживається для позначення кількох паралельних процесів;
- кінець паралельності (риска, до якої йдуть кілька стрілок) – завершення паралельно виконуваних процесів та об'єднання їх в єдиний потік виконання;
- кінцевий вузол (круг у колі) означає закінчення виконання процесу та виходження з діаграми активності.

У процесі моделювання вирішено продемонструвати процес пошуку та перегляду книги.

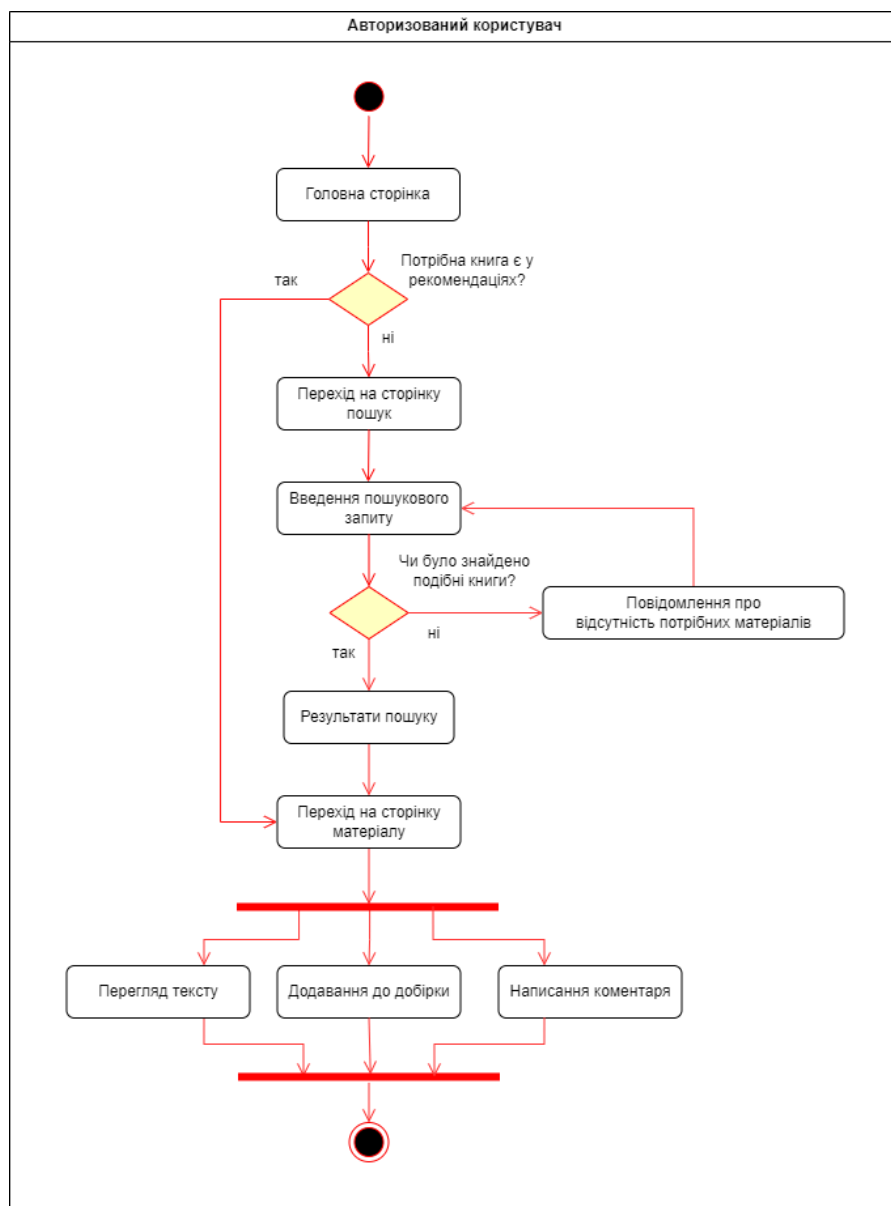


Рисунок 2.3 – Діаграма активності для пошуку та перегляду книг

На рис. 2.3 показано процес пошуку та перегляду книг. Авторизований користувач може вибрати потрібний матеріал зі списку рекомендацій або, якщо його немає, перейти на сторінку пошук та ввести запит у пошуковий рядок. Якщо результати відсутні, відвідувач може повторити пошук. У разі успішного знаходження результатів користувач переходить на сторінку матеріалу, де він має можливість виконати кілька дій. Зокрема, він може почати читати книгу, додати її до добірки для подальшого використання або залишати коментарі. Після виконання обраних дій процес завершується, а користувач може продовжити інші взаємодії з платформою. Цей процес дозволяє забезпечити зручність та багатофункціональність для користувачів під час взаємодії з електронною бібліотекою.

Діаграми послідовності використовуються для моделювання та візуалізації послідовності взаємодії між об'єктами в системі в хронологічному порядку, що дозволяє виявити можливі проблеми та оптимізувати процес [15]. Вони грають важливу роль у проектуванні, розробці та документації систем, а також можуть служити для розробки тестових сценаріїв.

Для моделювання процесів у вигляді діаграми послідовності застосовуються такі елементи:

- Лінія життя (вертикальна пунктирна лінія, що йде від кожного об'єкта) – представляє тривалість існування об'єкта впродовж сценарію;
- Період активації (прямокутник) – період, коли об'єкт виконує якусь операцію;
- Повідомлення (синхронне: суцільна стрілка із зафарбованим кінцем, також передбачає відповідь у вигляді пунктирної стрілки, асинхронне: суцільна стрілка із загостреними кінцями, не має відповіді) – взаємодія між двома об'єктами;
- Фрейми або фрагменти – використовуються для групування повідомлень, наприклад, у разі розгалуження, паралельне виконання дій, цикл тощо.

На представленій діаграмі послідовності ілюструється взаємодія класів під час процесу створення нової книги (рис. 2.4). Спочатку адміністратор отримує список існуючих авторів та категорій. Обравши їх та заповнивши потрібну інформацію, він викликає метод для створення нової книги. У ньому здійснюється перевірка, чи всі зазначені автори і категорії існують. Потім відбувається обробка двох отриманих файлів, обкладинки і електронної книги. Після збереження файлів, створюється нова книга і об'єкт повертається адміністратору.

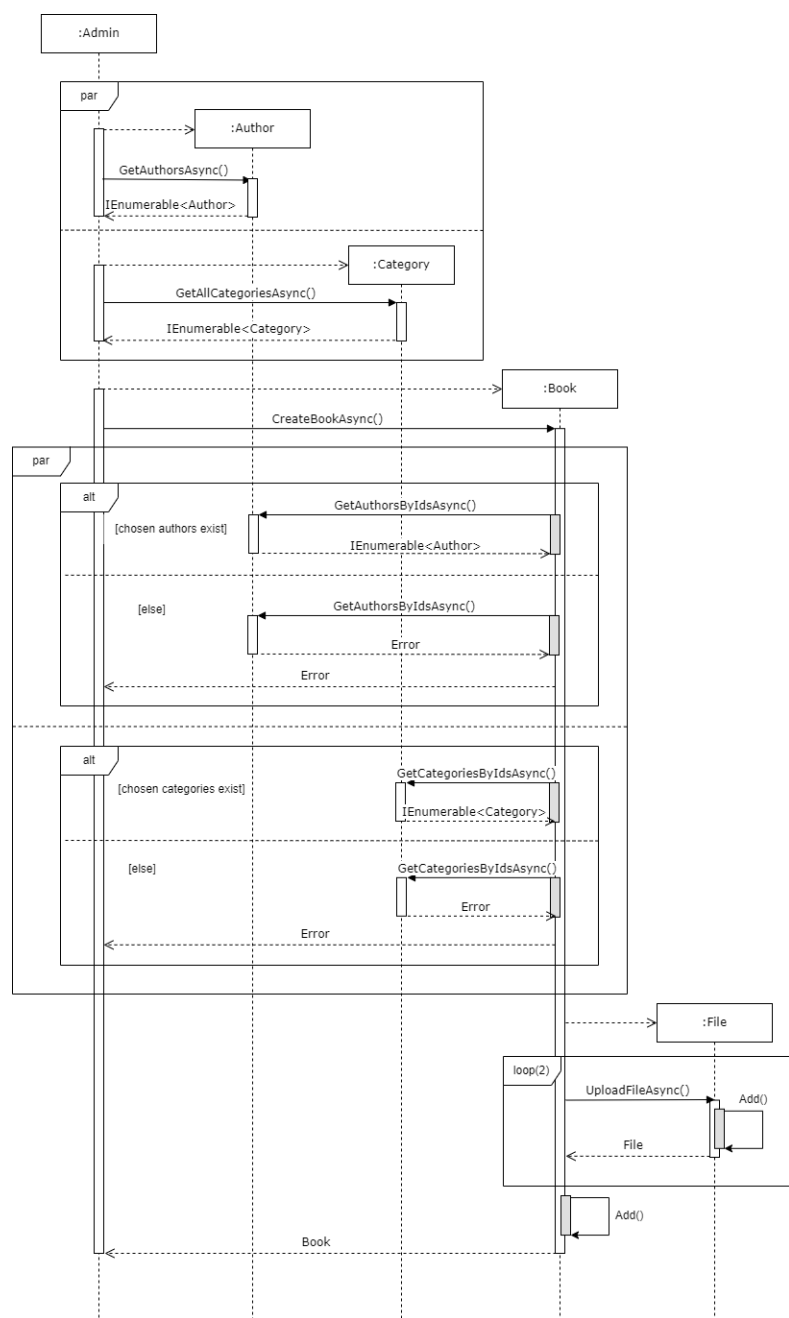


Рисунок 2.4 - Діаграма послідовності додавання книги адміністратором

Описані діаграми надали чітке уявлення про те, як мають відбуватися певні процеси сервісу, що значно покращило розуміння його функціоналу.

2.4 Реалізація сервісу

Розробка програмного забезпечення включає кілька ключових етапів, одним з яких є реалізація сервісу. Цей етап охоплює практичне втілення ідей, концепцій та технічних вимог у конкретний програмний код, що формує основу функціональності сервісу.

У даному пункті буде детально розглянуто процес реалізації сервісу, включаючи опис архітектури, основних модулів та компонентів, а також ключових алгоритмів і методів, використаних при розробці. Основна мета цього розділу полягає в тому, щоб надати чітке уявлення про те, як реалізовані різні частини сервісу та як вони взаємодіють між собою для забезпечення належної роботи системи.

Реалізація сервісу включає такі аспекти:

1. Основні модулі та їх функціональність: Детальний опис основних модулів системи, їх призначення та реалізація.
2. Ключові алгоритми: Пояснення алгоритмів, що використовуються для виконання основних функцій сервісу.
3. Інтерфейси та взаємодія компонентів: Опис інтерфейсів між різними компонентами системи та методів їх взаємодії.
4. Використання сторонніх бібліотек та фреймворків: Перелік сторонніх інструментів, використаних при реалізації сервісу, та пояснення їхньої ролі.

Увесь функціонал реалізовано у вигляді контролерів та сервісів для розділення логіки. Загалом їх 10, проте розглянемо лише декілька з них.

Контролер авторизації AuthController відповідає за реєстрацію, авторизацію та оновлення токenu. У сервісі, пов'язаному з цим контролером, відбу-

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

вається генерація токена доступу та поновлення, описана сама логіка створення нового токена доступу, реєстрація та вхід в акаунт. Детальніше розглянемо процес авторизації.

Для усіх дій з користувачами використовується Identity Framework [16]. Він забезпечує інтеграцію з різними механізмами аутентифікації та авторизації, включаючи стандартні схеми безпеки та можливість кастомізації. Використання цього фреймворку спрощує розробку та підтримку системи авторизації, забезпечуючи високий рівень безпеки та ефективності управління правами доступу користувачів.

За вхід у акаунт відповідає метод LoginAsync. У ньому за допомогою наданих даних знаходиться користувач, перевіряється його пароль та ролі, після чого створюються два токени: токен доступу (access token) і токен оновлення (refresh token). Токен доступу використовується для автентифікації користувача при кожному запиті до сервера, в той час як токен оновлення використовується для оновлення токена доступу без необхідності повторної авторизації.

Процес створення токена відбувається так:

- створення тверджень (claims) з інформацією про користувача;
- створення ключа для підпису за допомогою секретного ключа, описаного в конфігурації;
- створення підпису;
- створення JWT (JSON Web Token) за допомогою тверджень, терміну дії, підпису;
- перевірка існування токена відновлення – за потреби, він генерується і зберігається у БД;
- повернення результату – генерується строковий представник токена доступу і повертається об'єкт Token.

```
public async Task<Token> CreateJWTTokenAsync(User user, List<string>
roles)
{
    var claims = new List<Claim>
```

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

```

{
    new Claim(ClaimTypes.NameIdentifier, user.Id.ToString()),
    new Claim(ClaimTypes.Email, user.Email),
    new Claim(ClaimTypes.Name, user.UserName)
};

foreach (var role in roles)
{
    claims.Add(new Claim(ClaimTypes.Role, role));
}

var key = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Jwt:Key"]))
);
var credentials = new SigningCredentials(key,
SecurityAlgorithms.HmacSha256);

var token = new JwtSecurityToken(
    issuer: _configuration["Jwt:Issuer"],
    audience: _configuration["Jwt:Audience"],
    claims: claims,
    expires: DateTime.Now.AddHours(1),
    signingCredentials: credentials);

if (user.RefreshToken == null || user.RefreshTokenExpiration <=
DateTime.Now)
{
    var refreshTokenValue = CreateRefreshToken();
    user.RefreshToken = refreshTokenValue;
    user.RefreshTokenExpiration = DateTime.Now.AddDays(1);
    await _userManager.UpdateAsync(user);
    await _context.SaveChangesAsync();
}

var accessToken = new
JwtSecurityTokenHandler().WriteToken(token);

return new Token { AccessToken = accessToken, RefreshToken =
user.RefreshToken };
}

```

Коли закінчується термін дії токена доступу, відбувається створення нового без залучення користувача. Це може відбуватися як під час активної сесії користувача, так і при вході у додаток, якщо токен відновлення ще дійсний. За це відповідає метод RefreshTokenAsync, який працює таким чином:

- отримання інформації користувача із застарілого токена;
- знаходження користувача;

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

- перевірка валідності токена оновлення;
- отримання ролей користувача;
- створення нового JWT токена;
- повернення результатів.

```
public async Task<Token> RefreshTokenAsync(Token token)
{
    var principal = GetPrincipalFromExpiredToken(token.AccessToken);
    var user = await
_userManager.FindByNameAsync(principal.Identity.Name);
    if (user == null || user.RefreshToken != token.RefreshToken ||
user.RefreshTokenExpiration <= DateTime.Now)
    {
        throw new SecurityTokenException("Invalid refresh token");
    }

    var roles = await _userManager.GetRolesAsync(user);
    return await CreateJWTTokenAsync(user, roles.ToList());
}
```

Контролер BooksController відповідає за дії з книгами. Доступ до нього мають лише авторизовані користувача, на що вказує атрибут [Authorize(AuthenticationSchemes = "Bearer")]. Розглянемо додавання нової книги та пошук з ключовим словом/фразою.

Доступ до методу PostBook мають лише адміністратори. Спочатку метод перевіряє, чи існують усі вказані автори та категорії. Після цього, створюється об'єкт книги і відбувається завантаження файлів. Якщо завантаження успішне, книга зберігається у БД та повертається на клієнтську частину.

```
public async Task<ResponseBookDTO> CreateBookAsync(RequestBookDTO dto)
{
    var authors = await _context.Authors.Where(a =>
dto.AuthorIds.Contains(a.Id)).ToListAsync();
    if (authors.Count != dto.AuthorIds.Count)
    {
        throw new ArgumentException("Some authors not found");
    }

    var categories = await _context.Categories.Where(c =>
dto.CategoryIds.Contains(c.Id)).ToListAsync();
    if (categories.Count != dto.CategoryIds.Count)
    {
        throw new ArgumentException("Some categories not found");
    }
}
```

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

```

var book = _mapper.Map<Book>(dto);
book.Authors = authors;
book.Categories = categories;
book.Created = DateTime.Now;

var newFile = await _fileService.UploadFileAsync(dto.CoverFile);
book.CoverFile = _mapper.Map<CoverFile>(newFile);
newFile = await _fileService.UploadFileAsync(dto.BookContent);
book.BookContent = _mapper.Map<BookContent>(newFile);

await _context.Books.AddAsync(book);
await _context.SaveChangesAsync();

return _mapper.Map<ResponseBookDTO>(book);
}

```

Метод GetBooksByKeywordAsync (додаток А) відповідає за пошук книг за ключовим словом і працює таким чином:

- перевірка ключового слова;
- розбиття ключових слів;
- пошук точних відповідей;
- пошук часткових відповідей;
- об'єднання результатів, мапінг і їх повернення.

З мобільного додатку найважливішим є інтерфейс читання книги. Він реалізований за допомогою react-native-pdf [17] і підтримує такі дії – горизонтальна чи вертикальна прокрутка сторінок, поділ на сторінки, перехід на іншу сторінку за допомогою індикатора прогресу, збереження поточної сторінки у кеш та БД (додаток Б). Тут реалізовано користувацьку подію на жест назад – з'являється вікно з підтвердженням дії, після якого номер поточної сторінки зберігається в БД:

```

const backHandler = BackHandler.addEventListener(
  "hardwareBackPress",
  () => {
    Alert.alert(
      "Вихід з перегляду книги",
      "Ви впевнені, що хочете повернутися на попередню сторінку?",
      [
        {
          text: "Відміна",
          onPress: () => null,
          style: "cancel",
        },
      ],
    );
  },
);

```

```

        {
            text: "Вийти",
            onPress: async () => {
                await saveProgressToDB(currentPage, numberOfPages, id,
bookId);
                navigation.goBack();
            },
        },
    ],
    { cancelable: false }
);
return true;
}
);

```

Найважливішою частиною веб-додатку є форма додавання/редагування книги (додаток В). Вона використовує різні бібліотеки, такі як Formik [18] для управління станом форми та валідацією, Yup [19] для визначення схем валідації для додавання нової та редагування існуючої, та Material UI [20] для побудови інтерфейсу користувача.

При завантаженні сторінки відбувається ініціалізація станів – надсилається запит на отримання авторів і категорій. Після заповнення полів і проходження валідації, викликається функція обробки подання форми. Вона перевіряє, чи редагується існуюча книга або створюється нова, формує відповідний запит та відправляє його на сервер.

У процесі розробки було використано такі фреймворки і бібліотеки:

- Identity Framework використовується для управління аутентифікацією та авторизацією користувачів у додатках ASP.NET. Він забезпечує функціональність для реєстрації, входу в систему, управління ролями та правами доступу.
- Entity Framework [21] є ORM (Object-Relational Mapping) фреймворком для роботи з базами даних у .NET. Він дозволяє розробникам працювати з базою даних, використовуючи об'єктно-орієнтований підхід.

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

- AutoMapper [22] використовується для автоматичного мапінгу об'єктів між різними класами, що зменшує обсяг коду, необхідного для ручного копіювання властивостей.
- Recharts [23] є бібліотекою для побудови графіків та діаграм у React додатках. Вона дозволяє створювати інтерактивні та привабливі візуалізації даних.
- react-toastify [24] використовується для відображення повідомлень про успішні або помилкові дії користувачів у вигляді "тостів" (спливаючих повідомлень).
- Material UI є бібліотекою компонентів для React, яка реалізує принципи дизайну Material Design від Google. Вона забезпечує готові до використання компоненти для створення сучасних і привабливих інтерфейсів користувача.
- Yup – це бібліотека для валідації схем об'єктів у JavaScript. Вона використовується для валідації форм у веб-додатках, забезпечуючи простий і гнучкий спосіб визначення правил валідації для різних типів даних.
- Formik – це бібліотека для роботи з формами у React. Вона допомагає легко керувати станом форми, обробляти події введення, виконувати валідацію та обробляти подання форми.
- React Native Paper [25] є бібліотекою компонентів для React Native, яка реалізує принципи дизайну Material Design. Вона забезпечує компоненти для створення естетичних і функціональних мобільних додатків.
- react-native-pdf використовується для відображення PDF-документів у мобільних додатках на базі React Native. Вона дозволяє інтегрувати перегляд PDF-файлів безпосередньо у додаток.
- @react-native-async-storage/async-storage [26] є бібліотекою для зберігання даних у локальному сховищі на мобільних пристроях. Вона дозволяє зберігати ключ-значення дані асинхронно.

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

Висновки до другого розділу

У другому розділі розглянуто ключові аспекти проектування та реалізації програмного комплексу. Наведено загальну структуру програмного комплексу, яка включає опис архітектури системи, її компоненти та їх взаємодію для забезпечення цілісності роботи додатку.

Описано інформаційне забезпечення системи, зокрема структуру БД, їх взаємозв'язки та способи зберігання та доступу до інформації. Також описано алгоритми взаємодії окремих модулів системи.

Крім того, складено опис об'єктної структури системи, представлений у вигляді діаграми класів, що відображає об'єктну модель системи та взаємозв'язки між класами. Розглянуто алгоритми роботи основних методів класів, які виконують ключові функції системи.

Також наведено приклади програмного коду, що реалізує основні модулі системи, з поясненням логіки їх роботи та функціональних можливостей.

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ ДЛЯ ОРГАНІЗАЦІЇ ТА ВИКОНАННЯ ЗАВДАНЬ

3.1 Порядок встановлення та налаштування параметрів сервісу

Цей сервіс складається з трьох основних частин: дві клієнтські програми - веб-додаток і мобільний додаток, а також серверна частина. Клієнтські додатки призначені для взаємодії з користувачем і надають інтерфейс для взаємодії з основними функціями сервісу. Веб-додаток доступний через веб-браузер, тоді як мобільний додаток - через мобільний пристрій.

На діаграмі розгортання можна побачити, що клієнтські додатки запускаються на пристроях користувачів і пов'язані протоколом TCP/IP з серверною частиною, яка включає в себе бекенд, БД і зберігання файлів, які знаходяться на одному сервері. Це організаційне рішення може забезпечувати простоту управління та доступ до спільних ресурсів (наприклад, БД) з боку сервера.

Для ілюстрації елементів сервісу було розроблено діаграму розгортання, (рис. 3.1). На даній діаграмі представлено фізичну архітектуру системи, що включає в себе різні компоненти та їх взаємодію. Розглянемо основні складові системи:

1. Мобільний пристрій користувача представлений на діаграмі як вузол з локальним сховищем та мобільним додатком. Здійснює взаємодію з сервером мобільного додатку через мережу TCP/IP.
2. Комп'ютер адміністратора, обладнаний браузером, який має доступ до веб-сайту через мережу TCP/IP.
3. API працює на платформі ASP.NET Core 8. Містить контролери, такі як AuthControllers.cs, BooksController.cs, а також моделі, наприклад, AuthorizedUser.cs, Selection.cs, Book.cs, CoverFile.cs, BookContent.cs, Comment.cs. Взаємодіє з браузером адміністратора та мобільним додатком.

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

- 4. БД працює на платформі SQL Server 2022 Express. Забезпечує зберігання та управління даними, знаходиться на тому ж сервері, що і API.
- 5. Сховище для зберігання файлів, пов’язаних з книгами, знаходиться на тому ж сервері, що і API.

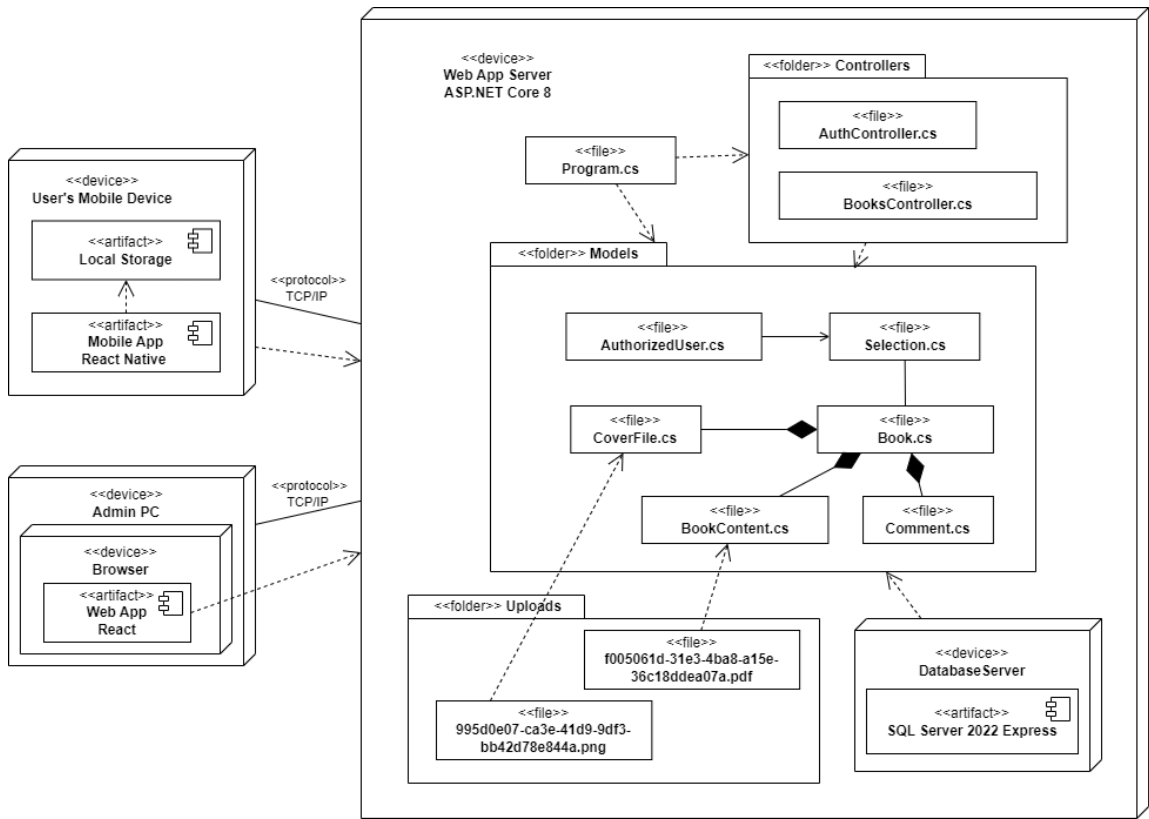


Рисунок 3.1 – Діаграма розгортання

Ця діаграма розгортання надає повну картину фізичного розташування компонентів системи та їх взаємодії через мережеві протоколи, що є важливим для розуміння та управління інфраструктурою системи.

3.2 Структура інтерфейсу сервісу

Інтерфейс сервісу забезпечує взаємодію користувачів з системою. У даному випадку є два інтерфейси – веб та мобільний. У цьому розділі ми детально розглянемо їхні основні складові, їх взаємодію та роль у забезпеченні користувацького досвіду.

Спочатку розглянемо мобільний додаток, який використовується звичайними користувачами для перегляду книг. При вході в додаток, сервіс направляє користувача на сторінку авторизації (рис. 3.2).

На сторінці авторизації користувач має можливість ввести свої дані для входу в систему. Це включає в себе поле для введення електронної пошти та поле для введення пароля. Після введення вірних даних і натискання кнопки "Увійти", система перевіряє їх на відповідність і, у разі успішної авторизації, відкриває головний екран додатку.

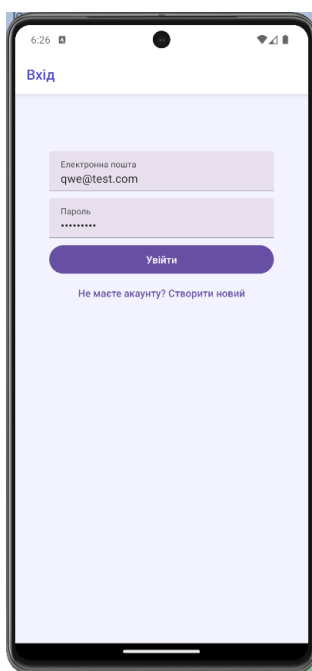


Рисунок 3.2 – Сторінка входу

На сторінці входу також є можливість перейти на сторінку реєстрації (див. рис. 3.3). Ця опція дозволяє новим користувачам створити обліковий запис в системі. На сторінці реєстрації користувачеві потрібно ввести обов'язкові дані, такі як ім'я, прізвище, електронну пошту, пароль та його підтвердження. Після заповнення всіх необхідних полів і натискання кнопки "Зареєструватися", система проводить перевірку введених даних. У разі успішної реєстрації користувач входу в систему під створеним акаунтом.

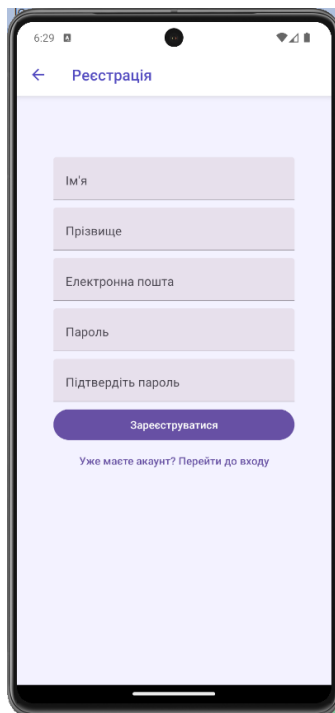


Рисунок 3.3 – Сторінка реєстрації

Якщо користувач успішно увійшов у свій акаунт або створив новий, він потрапляє на головну сторінку додатку (рис. 3.4). На цій сторінці він зустрічає новинки та книги різних категорій, які були обрані випадковим чином. Головна сторінка додатку призначена для того, щоб користувачі могли сформувати представлення про те, який контент надає система.



Рисунок 3.4 – Сторінка реєстрації

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

На сторінці пошуку є можливість знайти книги за ключовим словом (рис. 3.5). При переході на цю сторінку відображаються декілька випадкових книг, рядок для введення ключового слова або фрази для пошуку, а також випадаючий список з вибором опцій сортування.

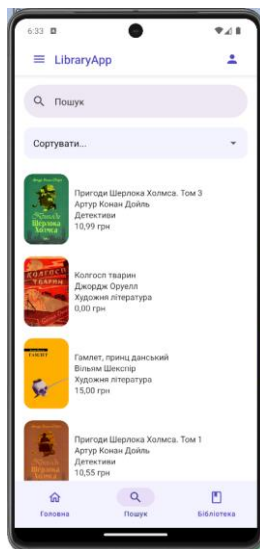


Рисунок 3.5 – Сторінка пошуку

Коли користувач починає вводити пошуковий запит, з'являється список з варіантами назв книг, які можуть бути результатом (рис. 3.6). Цей функціонал, що називається автодоповненням або підказкою пошуку, допомагає швидше знаходити потрібні книги, показуючи можливі варіанти. Користувач може обирати один із запропонованих варіантів або продовжувати введення свого запиту.

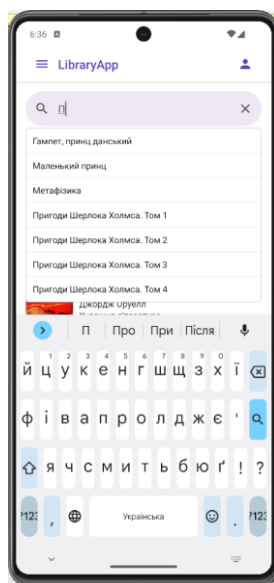


Рисунок 3.6 – Введення запиту

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

Якщо користувач натисне на клавішу «Enter» або вибере один з варіантів з автодоповнення, здійсниться пошук. На сторінці буде показано спочатку точні співпадиння, потім часткові (рис. 3.7).

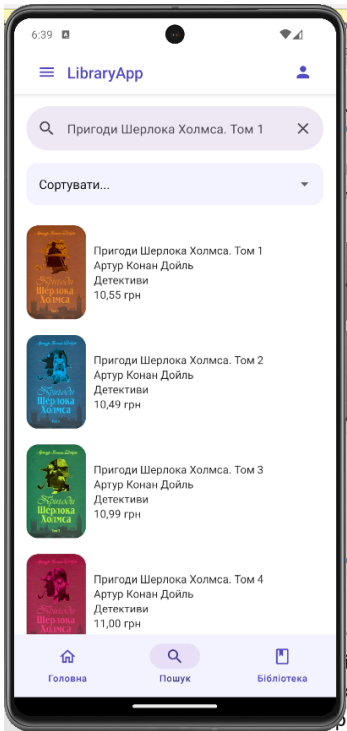


Рисунок 3.7 – Результати пошуку

Користувач може сортувати результати пошуку за назвою або ціною, вибравши відповідний пункт з випадаючого списку (рис. 3.8).

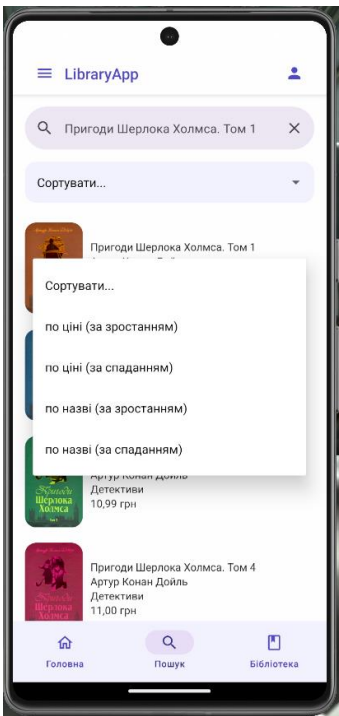


Рисунок 3.8 – Варіанти сортування

Вибравши один із критеріїв сортування, список результатів відразу відсортується (рис. 3.9).

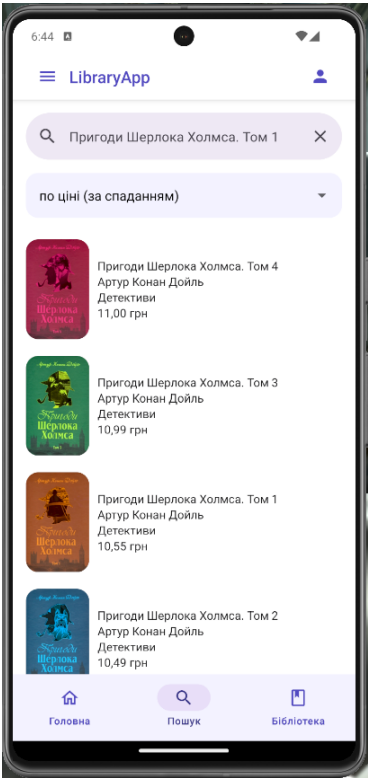


Рисунок 3.9 – Результати сортування

У бібліотеці користувач може переглянути свої добірки (рис. 3.10). Деякі з них створені автоматично, такі як «Обрані», «Читаю», «Прочитано».

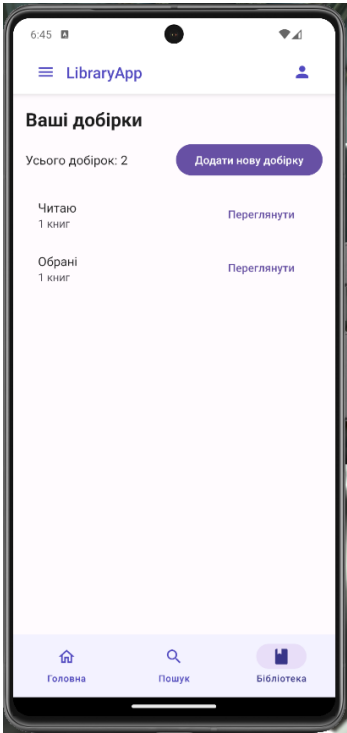


Рисунок 3.10 – Бібліотека

За бажанням користувач може створити нову добірку, натиснувши на кнопку «Додати нову добірку». Після цього відкриється форма, де він має ввести назву, яка буде ідентифікувати добірку (рис. 3.11). Після введення назви і підтвердження створення добірки, вона з'явиться в бібліотеці користувача разом з іншими.

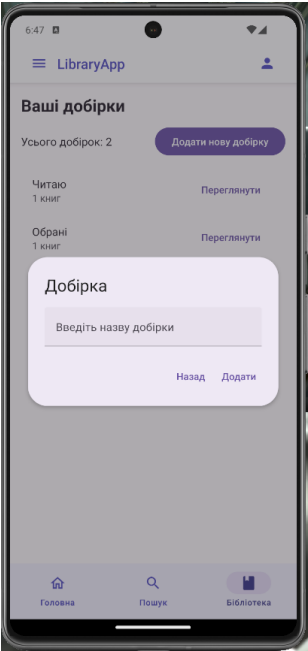


Рисунок 3.11 – Форма створення добірки

Користувач може видаляти створені ним добірки, а також змінювати їх назву (рис. 3.12). У всіх добірках є можливість видаляти книги.

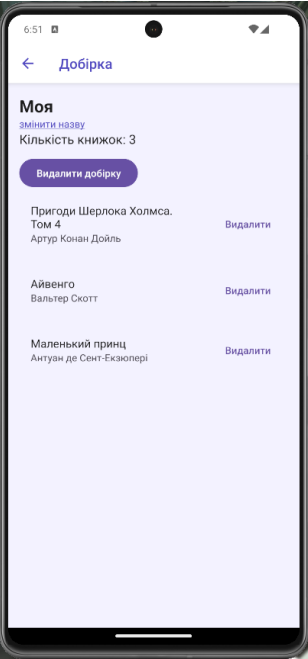


Рисунок 3.12 – Сторінка перегляду добірки

З головної сторінки додатку можна відкрити меню, в якому розміщені різні категорії (рис. 3.13). Це меню знаходиться у верхньому лівому кутку екрану.

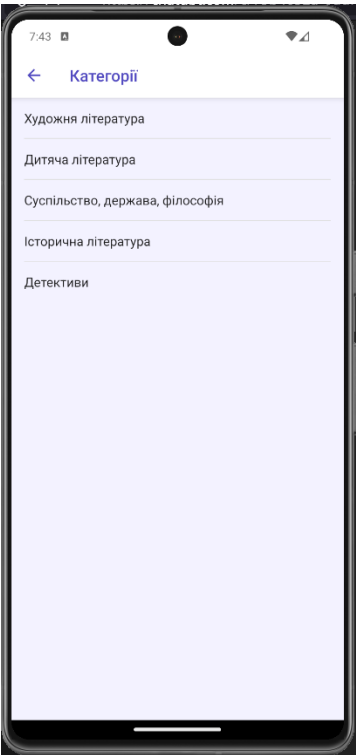


Рисунок 3.13 – Сторінка категорій

Обравши одну з категорій, користувач переходить на список цієї категорії, де він може побачити книги, що відносяться до неї (рис.3.14).

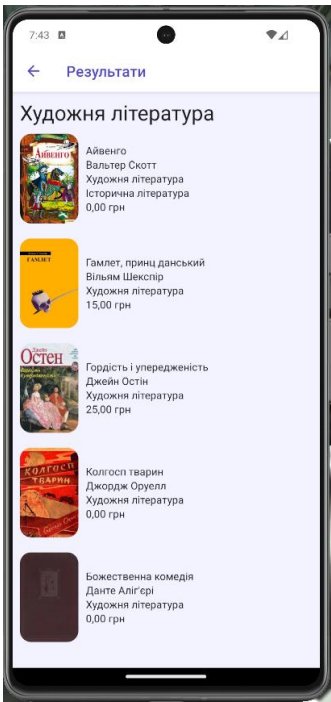


Рисунок 3.14 – Сторінка категорії

На усіх сторінках, де показано списки книг, користувач може перейти на сторінку з детальною інформацією про кожну книгу. На цій сторінці він зможе здійснити ряд дій, таких як додавання книги до своєї добірки (рис. 3.16), початок читання або написання коментаря (рис. 3.15).

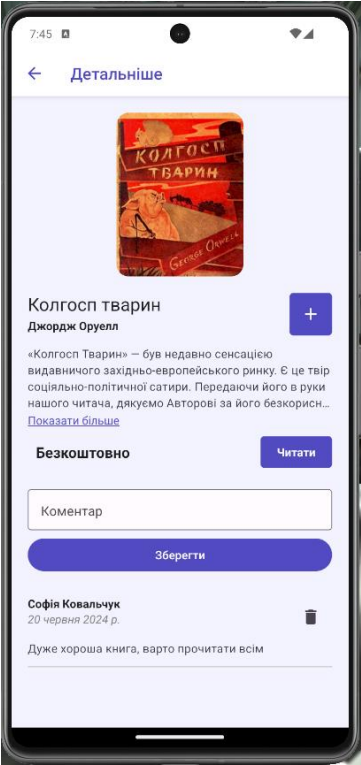


Рисунок 3.15 – Сторінка книги



Рисунок 3.16 – Вікно додавання книги до добірки

Інтерфейс читання (див. рис. 3.17) відкривається після натискання на кнопку «Читати» або «Купити». У цьому інтерфейсі користувач перегортає сторінки за допомогою жестів або індикатора прогресу, розташованого у нижній частині екрану, який дозволяє швидко переходити на інші сторінки.

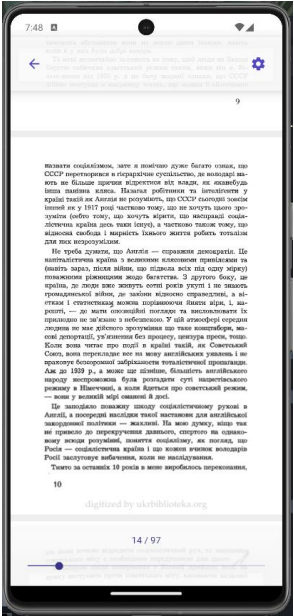


Рисунок 3.17 – Інтерфейс читання

У правому верхньому кутку інтерфейсу знаходиться значок налаштувань (рис. 3.18). Натискання на цей значок відкриває меню налаштувань, де користувач може змінювати параметри перегляду книги, що включають зміна напрямку прокрутки та поділ на сторінки.

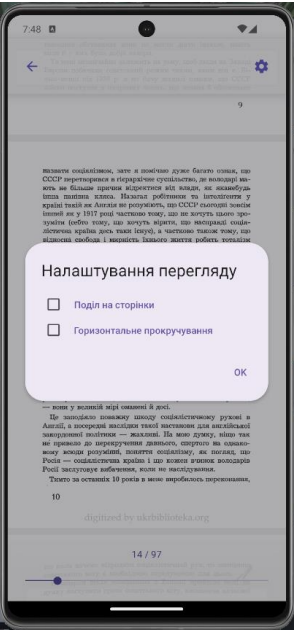


Рисунок 3.18 – Параметри перегляду

Значок профілю розташований на головній сторінці у верхньому правому кутку. Натискання на цей значок перенаправляє користувача на сторінку свого профілю (рис. 3.19). Тут він має можливість виконати декілька дій, таких як редагування своєї інформації (рис. 3.20), видалення акаунту або вихід з нього.

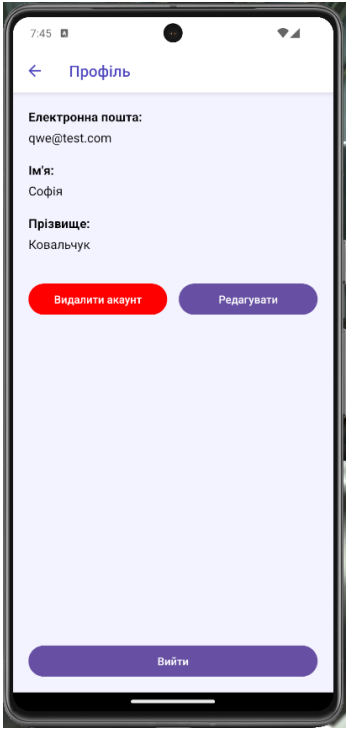


Рисунок 3.19 – Профіль

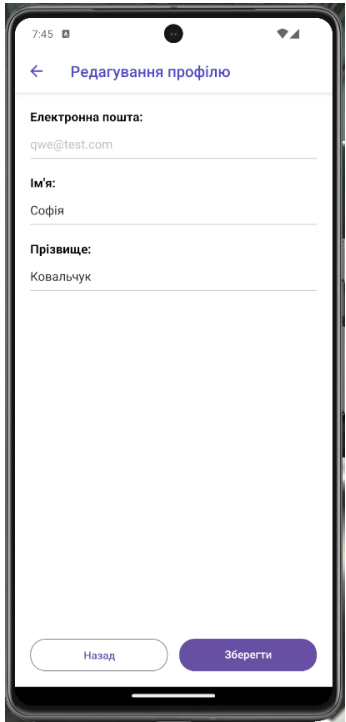


Рисунок 3.20 – Редагування профілю

Тепер розглянемо веб-додаток, який призначений для керування контентом. При запуску додатку сервіс направляє нас на форму авторизації. На формі авторизації адміністратор повинен ввести свої облікові дані, такі як ім'я користувача та пароль, для входу до системи (рис. 3.21).

Вхід

Електронна пошта

admin@example.com

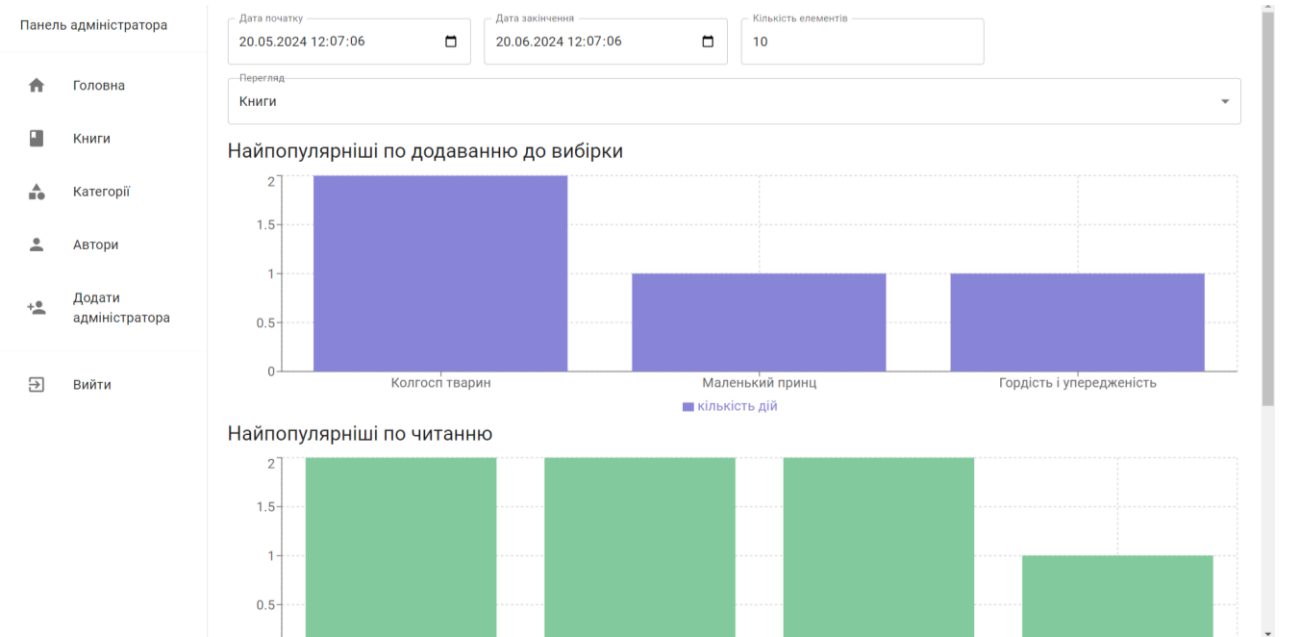
Пароль

.....

УВІЙТИ

Рисунок 3.21 – Форма входу

Після успішної авторизації користувач переходить на головну сторінку, де доступна статистика найпопулярніших книг, авторів та категорій (рис. 3.22).



З бокового меню можна перейти на сторінку перегляду категорій, які подані у вигляді таблиці, яку можна сортувати та фільтрувати (рис. 3.23).

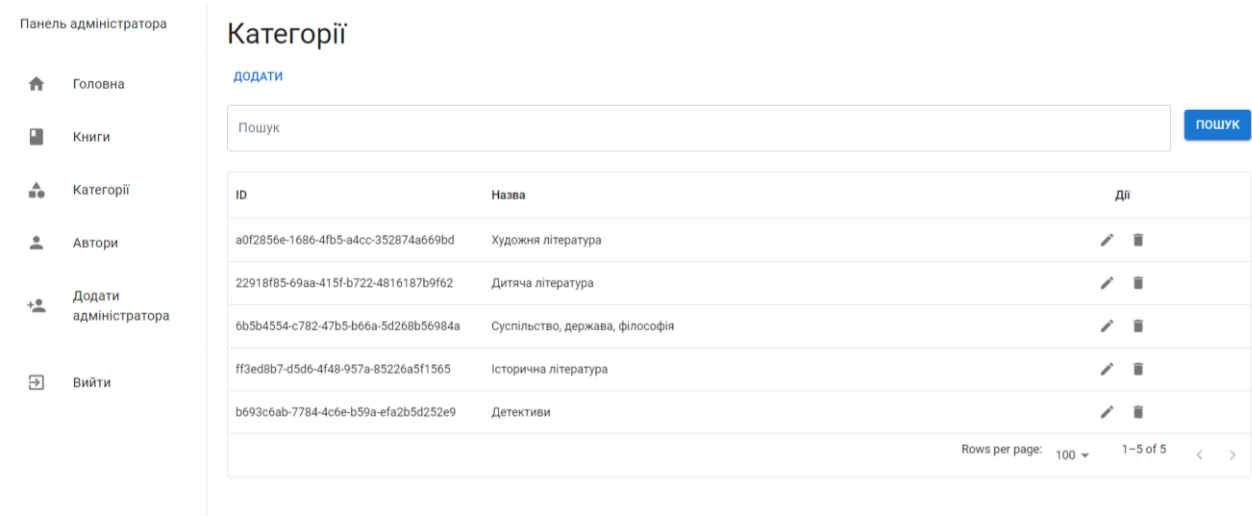


Рисунок 3.23 – Список категорій

На цій сторінці також можна додати нову категорію (рис. 3.24), редагувати (рис. 3.25) або видалити існуючу.

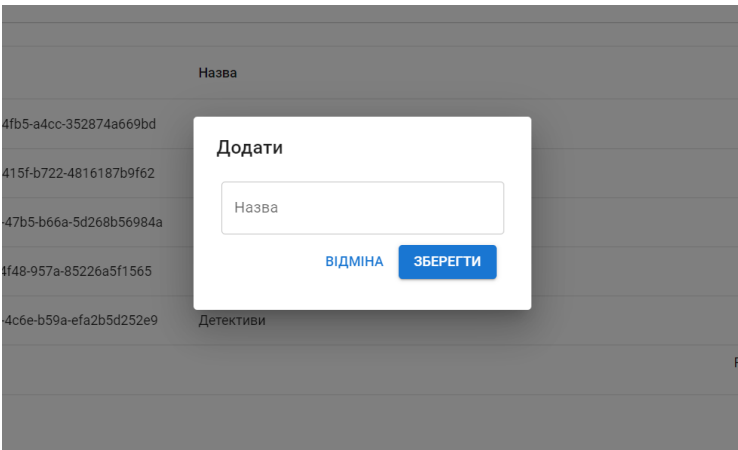


Рисунок 3.24 – Форма додавання категорії

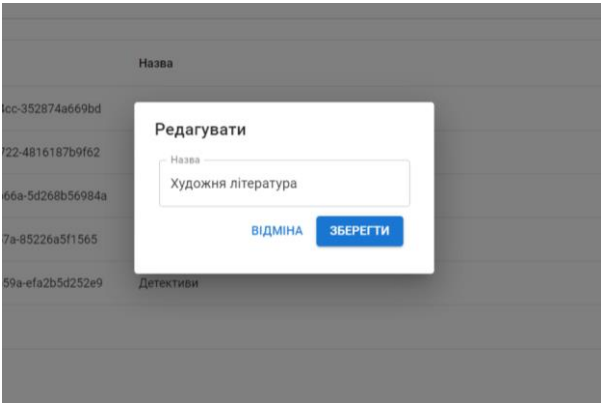


Рисунок 3.25 – Форма редагування категорії

Представлення авторів та дій з ними схоже на категорії (рис. 3.26, 3.27, 3.28).

Панель адміністратора

Головна

Книги

Категорії

Автори

Додати адміністратора

Вийти

Автори

ДОДАТИ

Пошук

ПОШУК

ID	Ім'я	Прізвище	Дата народження	Дії
c8bf4d5-6954-4ca...	Алан Александер	Мілн	1882-01-18	 
4c05307e-c449-44...	Антуан	де Сент-Екзюпері	1900-06-29	 
5145c89e-dc86-45...	Аристотель		0001-01-01	 
9cb479a1-0b76-4b...	Артур Конан	Дойль	1859-05-22	 
97d06a7e-fefa-4f5...	Вальтер	Скотт	1771-08-15	 
30205b0f-c4b6-45...	Вільям	Шекспір	1564-04-23	 
95d6c98b-497c-47...	Данте	Аліг'єрі	1265-05-18	 
ae3b0858-cace-4c...	Джейн	Остін	1775-12-16	 
f58afe21-f390-40b...	Джордж	Оруелл	1903-06-25	 
64940746-f795-48...	Франц	Кафка	1883-07-03	 

Рисунок 3.26 – Список авторів

Додати

Ім'я

Прізвище

Дата народження
дд.мм.rrrr

ВІДМІНА

ЗБЕРЕГТИ

Рисунок 3.27 – Форма додавання автора

Редагувати

Ім'я
Алан Александер

Прізвище
Мілн

Дата народження
18.01.1882

ВІДМІНА

ЗБЕРЕГТИ

Рисунок 3.28 – Форма редагування автора

Список книг теж схожий на попередні, відрізняється лише додатковою дією – заміна файлу (рис. 3.29).

Панель адміністратора

Головна

Книги

Категорії

Автори

Додати адміністратора

Вийти

КНИГИ

ДОДАТИ

Пошук

ПОШУК

ID	Назва	Опис	Ціна	Автори	Категорії	Обкладинка	Книга	Дії
1631f079-f...	Вінні-Пух та його д...	Це – книжка про с...	7.99	Алан Александер ...	Дитяча література	6b72d5c9-...	eddef1a2-...	
dd531d12-...	Метафізика	«Метафізика» – од...	0	Аристотель	Суспільство, держа...	bba9f3ef-b...	6c16fe9a-f...	
2e336a60-...	Пригоди Шерлока ...	На схилі своїх літ А...	11	Артур Конан Дойль	Детективи	02ec578b-...	715f0db9-...	
5905869b-...	Айвенго	Один із найпопуля...	0	Вальтер Скотт	Художня література	0fc02bb1-...	32a2e86e-...	
dfee9392-...	Пригоди Шерлока ...	Понад століття том...	10.55	Артур Конан Дойль	Детективи	aef0dc13-...	26e79efb-...	
d30f15b7-...	Гамлет, принц данс...	Це перший у треть...	15	Вільям Шекспір	Художня література	646a99b9-...	d0e44903-...	
0fdd70a3-...	Гордість і упередж...	Світ творів англійс...	25	Джейн Остін	Художня література	ce541d7f-...	91b6d738-...	
9141cbfb-...	Колгосп тварин	«Колгосп Тварин» ...	0	Джордж Оруелл	Художня література	c0891627-...	2a3a5db2-...	
6e1c6752-...	Божественна коме...	З ім'ям Данте (126...	0	Данте Аліґ'єрі	Художня література	f9b2512b-...	8248a28a-...	
2faeac26-...	Пригоди Шерлока ...	Опубліковане 1893 ...	10.99	Артур Конан Дойль	Детективи	a972113d-...	11ccb75e-...	

Рисунок 3.29 – Список книг

Основна відмінність форми додавання від авторів і категорій полягає у наявності полів для завантаження файлів (рис. 3.30), у той час як форма редагування містить лише текстові поля (рис. 3.31).

Додати книгу

Назва

Опис

Ціна

Автори

Категорії

Обкладинка

Книга

Вибрати файл

Файл не вибрано

Вибрати файл

Файл не вибрано

ВІДМІНА

ЗБЕРЕГТИ

Рисунок 3.30 – Форма додавання книги

Рисунок 3.31 – Форма редагування книги

Якщо адміністратор бажає змінити файл книги, він може зробити це через окрему форму (рис. 3.32). Завантажений файл зберігається як обкладинка чи книга в залежності від формату.

Рисунок 3.32 – Форма заміни файлу книги

Будь-які дії з категоріями, авторами та книгами супроводжуються сповіщеннями про успішність виконання дії (рис. 3.33, 3.34).

Рисунок 3.33 – Сповіщення про успішне видалення запису

Категорії

Зміни збережено

ДОДАТИ

Пошук

ID	Назва
a0f2856e-1686-4fb5-a4cc-352874a669bd	Художня література

Рисунок 3.34 – Сповіщення про успішне додавання\редагування запису

У веб-додатку також є пошук, який працює по такому самому принципу, як і в мобільному додатку (рис. 3.35).

Рисунок 3.35 – Автодоповнення

Крім того, адміністратор має можливість додати нового адміністратора. Ця функція забезпечує розширення команди адміністраторів для кращого керування контентом

Рисунок 3.36 – Форма реєстрації адміністратора

3.3 Тестування сервісу

Тестування сервісу є важливим етапом у розробці та забезпеченні якості програмного комплексу. Воно включає в себе різні методи перевірки коректності функціонування системи, зокрема валідацію форм, яка є ключовою складовою взаємодії користувача з додатком.

Валідація форм дозволяє забезпечити правильність і повноту даних, які вводяться користувачами, що є критично важливим для безперебійної роботи сервісу та його безпеки. У цьому пункті детально описано процес валідації форм, включаючи перевірку введених даних на відповідність встановленим вимогам, обмеженням та правилам.

Основна мета валідації форм полягає в запобіганні введення некоректних або неповних даних, що можуть призвести до помилок у системі або порушення її роботи. Валідація забезпечує, щоб усі обов'язкові поля були заповнені, дані мали правильний формат, а введена інформація відповідала очікуваним параметрам.

Розглянемо валідацію у веб-додатку. При введенні неправильних значень у форму авторизації, під полями виводиться повідомлення (рис. 3.37).

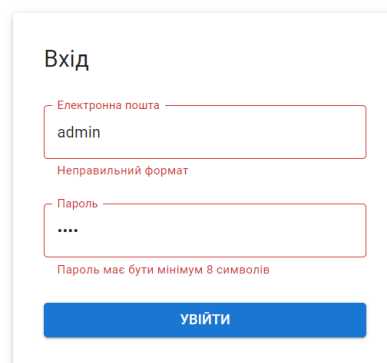


Рисунок 3.37 – Валідація форми входу (неправильний формат даних)

Якщо відправити форму пустою, з'являться повідомлення про те, що поля мають бути заповнені (рис. 3.38).

Рисунок 3.38 – Валідація форми входу (пусті поля)

Якщо ж користувача з введеною електронною поштою не існує або пароль неправильний, виводиться сповіщення про помилку (рис. 3.39).

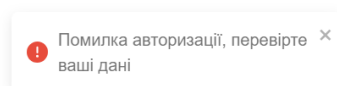


Рисунок 3.39 – Валідація форми входу (неіснуючий користувач)

Усі форми додавання/редагування проходять перевірки на те, чи поля заповнені (рис. 3.40), окрім полів «Прізвище» та «Дата народження» для автора, оскільки ці дані можуть бути невідомі. Додаткову перевірку проходять поля для введення ціни (значення має бути невід’ємним) та завантаження файлів (файл має бути не більше 20 МБ) (рис. 3.41, 3.42).

Рисунок 3.40 – Валідація форми додавання книг

Рисунок 3.41 – Валідація значення поля «Ціна»

Рисунок 3.42 – Валідація розміру файлу

У мобільному додатку форми авторизації так само перевіряються на пусті значення та неправильні дані (рис. 3.43, 3.44).

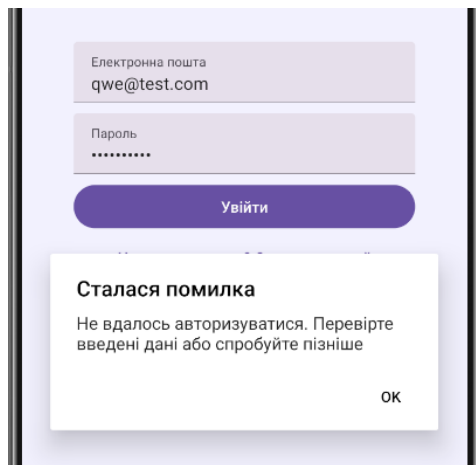


Рисунок 3.43 – Валідація форми авторизації (неправильний пароль)

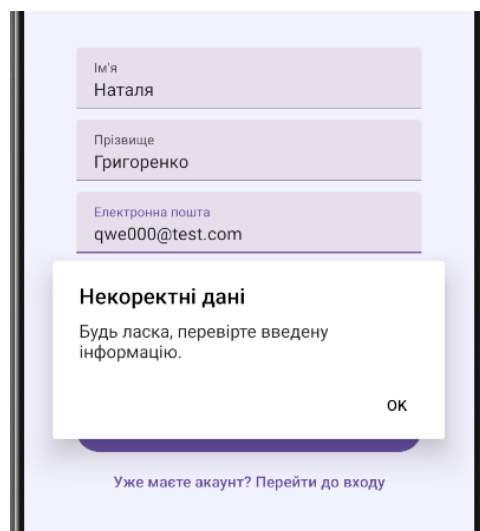


Рисунок 3.44 – Валідація форми авторизації (не співпадають значення полів паролю і підтвердження)

При написанні коментарів також є перевірка на неприйнятні слова. Якщо користувач написав одне з них, кнопка стає неактивною та виводиться відповідне повідомлення (рис. 3.45).

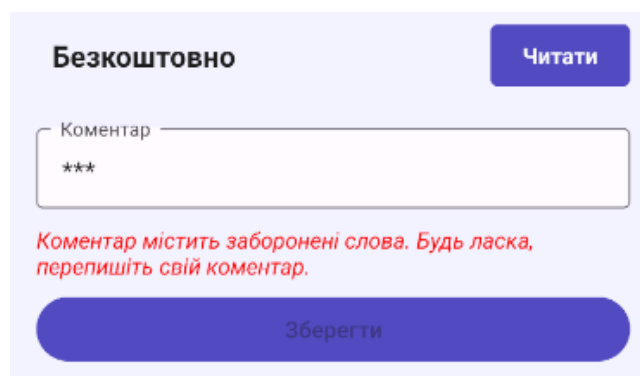


Рисунок 3.45 – Перевірка на неприйнятні слова

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

Висновки до третього розділу

У цьому розділі детально розглянуто інтерфейс та порядок роботи з сервісом. Спочатку описано взаємодію компонентів системи, а саме клієнтської частини та серверу. Також розглянута структура інтерфейсу системи, висвітлено основні компоненти інтерфейсу, їх призначення та взаємодія між собою, що дозволило зрозуміти логіку побудови системи.

Останнім етапом у цьому розділі стало тестування роботи програмного комплексу. Проведено тестування з метою перевірки коректності функціонування валідації даних. Результати тестування показали, що система працює стабільно, забезпечуючи належний рівень точності та надійності у виконанні своїх основних функцій.

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розроблено сервіс з надання послуг електронної бібліотеки, що включає веб- та мобільний додатки. Для цього проведено аналіз потреб користувачів та аналогів, доступних на українському ринку, розроблено архітектуру системи, яка забезпечує оптимальну роботу сервісу, реалізовано API та клієнтські додатки і проведено їхнє тестування.

Перший етап проекту передбачав аналіз потреб користувачів та огляд існуючих аналогів на українському ринку, що дозволив зрозуміти основні вимоги до функціональності та інтерфейсу. На основі цього аналізу була обрана оптимальна архітектура системи, враховуючи необхідність забезпечення високої швидкодії та надійності. В якості основних технологій для розробки були вибрані ASP.NET та SQL Server для реалізації бекенду, React для веб-додатку та React Native для мобільного додатку.

Детальне проектування системи включало в себе визначення варіантів використання, розробку діаграм класів для опису структури програмного рішення, проектування бази даних та розробку алгоритмів роботи системи. Цей етап завершився успішною реалізацією модулів, які відповідають за функціонал системи, та створенням ефективного інтерфейсу для взаємодії з користувачем.

Останнім етапом було описано структуру інтерфейсу сервісу та його функціональні можливості. Також проведено тестування коректності роботи програми.

У результаті виконання всіх поставлених завдань і досягнення мети, сервіс електронної бібліотеки успішно реалізований і готовий до використання, надаючи зручний доступ до бібліотечних ресурсів.

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Google Play Market. Play Книги [Електронний ресурс] – Режим доступу до ресурсу:

https://play.google.com/store/apps/details?id=com.google.android.apps.books&pcampaignid=web_share

2. Google Play Market. Yakaboo [Електронний ресурс] – Режим доступу до ресурсу:

https://play.google.com/store/apps/details?id=ua.yakaboo&pcampaignid=web_share

3. Google Play Market. Букнет [Електронний ресурс] – Режим доступу до ресурсу: <https://play.google.com/store/apps/details?id=ua.booknet&hl=uk>

4. Архітектура програмного забезпечення: все що треба знати [Електронний ресурс]. – Режим доступу до ресурсу:

<https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya>

5. 5 essential patterns of software architecture [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://www.redhat.com/architect/5-essential-patterns-software-architecture>

6. Монолітна архітектура ПЗ [Електронний ресурс]. – Режим доступу до ресурсу: <https://qalight.ua/baza-znaniy/shho-take-monolitna-arhitektura/>

7. Мікросервісна архітектура ПЗ [Електронний ресурс]. – Режим доступу до ресурсу: <https://qalight.ua/baza-znaniy/shho-take-mikroservisna-arhitektura-pz/>

8. Клієнт-серверна архітектура [Електронний ресурс]. – Режим доступу до ресурсу: <https://training.qatestlab.com/blog/technical-articles/client-server-architecture/>

9. ASP.NET documentation [Електронний ресурс]. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0&WT.mc_id=dotnet-35129-website

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

10. React [Електронний ресурс]. – Режим доступу до ресурсу:
<https://react.dev/>
11. React Native [Електронний ресурс]. – Режим доступу до ресурсу:
<https://reactnative.dev/>
12. Microsoft SQL documentation [Електронний ресурс]. – Режим доступу до ресурсу: <https://learn.microsoft.com/uk-ua/sql/?view=sql-server-ver16>
13. UML Use Case Diagram Tutorial [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.lucidchart.com/pages/uml-use-case-diagram>
14. What is Activity Diagram? [Електронний ресурс]. – Режим доступу до ресурсу:
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/>
15. UML Sequence Diagrams [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.uml-diagrams.org/sequence-diagrams.html>
16. Introduction to Identity on ASP.NET Core [Електронний ресурс]. – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio>
17. react-native-pdf [Електронний ресурс]. – Режим доступу до ресурсу:
<https://www.npmjs.com/package/react-native-pdf?activeTab=readme>
18. Build forms in React, without the tears [Електронний ресурс]. – Режим доступу до ресурсу: <https://formik.org/>
19. Yup [Електронний ресурс]. – Режим доступу до ресурсу:
<https://www.npmjs.com/package/yup>
20. Material UI - Overview [Електронний ресурс]. – Режим доступу до ресурсу: <https://mui.com/material-ui/getting-started/>
21. Entity Framework documentation hub [Електронний ресурс]. – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/ef/>
22. AutoMapper [Електронний ресурс]. – Режим доступу до ресурсу:
<https://automapper.org/>

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

23. Recharts [Електронний ресурс]. – Режим доступу до ресурсу:
<https://recharts.org/en-US/>

24. React-Toastify [Електронний ресурс]. – Режим доступу до ресурсу:
<https://www.npmjs.com/package/react-toastify?activeTab=readme>

25. Making your React Native apps look and feel native [Електронний ресурс]. – Режим доступу до ресурсу: <https://reactnativepaper.com/>

26. React Native Async Storage [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.npmjs.com/package/@react-native-async-storage/async-storage>

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

ДОДАТКИ

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

Сервіс AuthService, який відповідає за авторизацію та автентифікацію

```
using AutoMapper;
using LibraryApp.API.Data;
using LibraryApp.API.Models;
using LibraryApp.API.Models.Database;
using LibraryApp.API.Models.DTO;
using LibraryApp.API.Models.Requests;
using LibraryApp.API.Models.Response;
using LibraryApp.API.Services.Interfaces;
using Microsoft.AspNetCore.Identity;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace LibraryApp.API.Services.Implementations
{
    public class AuthService : IAuthService
    {
        private readonly IConfiguration _configuration;
        private readonly AppDbContext _context;
        private readonly UserManager<User> _userManager;
        private readonly IMapper _mapper;

        public AuthService(IConfiguration configuration, AppDbContext context,
            UserManager<User> userManager, IMapper mapper)
        {
            _configuration = configuration;
            _context = context;
            _userManager = userManager;
            _mapper = mapper;
        }

        public async Task<Token> CreateJWTTokenAsync(User user, List<string> roles)
        {
            var claims = new List<Claim>
            {
                new Claim(ClaimTypes.NameIdentifier, user.Id.ToString()),
                new Claim(ClaimTypes.Email, user.Email),
                new Claim(ClaimTypes.Name, user.UserName)
            };

            foreach (var role in roles)
            {
                claims.Add(new Claim(ClaimTypes.Role, role));
            }

            var key = new
                SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Jwt:Key"]));
            var credentials = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

            var token = new JwtSecurityToken(
                issuer: _configuration["Jwt:Issuer"],
                audience: _configuration["Jwt:Audience"],
                claims: claims,
                expires: DateTime.Now.AddHours(1),
                signingCredentials: credentials);

            if (user.RefreshToken == null || user.RefreshTokenExpiration <= DateTime.Now)
            {
                var refreshTokenValue = CreateRefreshToken();
            }
        }
    }
}
```

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

```

        user.RefreshToken = refreshTokenValue;
        user.RefreshTokenExpiration = DateTime.Now.AddDays(1);
        await _userManager.UpdateAsync(user);
        await _context.SaveChangesAsync();
    }

    var accessToken = new JwtSecurityTokenHandler().WriteToken(token);

    return new Token { AccessToken = accessToken, RefreshToken =
user.RefreshToken };
}

public async Task<Token> RefreshTokenAsync(Token token)
{
    var principal = GetPrincipalFromExpiredToken(token.AccessToken);
    var user = await _userManager.FindByNameAsync(principal.Identity.Name);
    if (user == null || user.RefreshToken != token.RefreshToken ||
user.RefreshTokenExpiration <= DateTime.Now)
    {
        throw new SecurityTokenException("Invalid refresh token");
    }

    var roles = await _userManager.GetRolesAsync(user);
    return await CreateJWTTokenAsync(user, roles.ToList());
}

private string CreateRefreshToken()
{
    var randomNumber = new byte[32];
    using (var rng = System.Security.Cryptography.RandomNumberGenerator.Create())
    {
        rng.GetBytes(randomNumber);
        return Convert.ToBase64String(randomNumber);
    }
}

private ClaimsPrincipal GetPrincipalFromExpiredToken(string token)
{
    var tokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = false,
        ValidateIssuerSigningKey = true,
        ValidIssuer = _configuration["Jwt:Issuer"],
        ValidAudience = _configuration["Jwt:Audience"],
        IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Jwt:Key"]))
    };

    var tokenHandler = new JwtSecurityTokenHandler();
    var principal = tokenHandler.ValidateToken(token, tokenValidationParameters,
out var securityToken);
    var jwtSecurityToken = securityToken as JwtSecurityToken;

    if (jwtSecurityToken == null ||
!jwtSecurityToken.Header.Alg.Equals(SecurityAlgorithms.HmacSha256,
StringComparison.InvariantCultureIgnoreCase))
    {
        throw new SecurityTokenException("Invalid token");
    }

    return principal;
}

```

```

public async Task<IdentityResult> RegisterAdminAsync(RegisterRequest register)
{
    var identityUser = _mapper.Map<Admin>(register);
    identityUser.UserName = Guid.NewGuid().ToString();

    var result = await _userManager.CreateAsync(identityUser,
register.Password);
    if (result.Succeeded)
    {
        result = await _userManager.AddToRoleAsync(identityUser, "Admin");
        await _userManager.UpdateAsync(identityUser);
    }
    return result;
}

public async Task<IdentityResult> RegisterUserAsync(RegisterRequest register)
{
    var identityUser = _mapper.Map<AuthorizedUser>(register);
    identityUser.UserName = Guid.NewGuid().ToString();

    var result = await _userManager.CreateAsync(identityUser, register.Password);
    if (result.Succeeded)
    {
        result = await _userManager.AddToRoleAsync(identityUser, "User");
        await _userManager.UpdateAsync(identityUser);

        var selection = new List<Selection> {
identityUser }
            //new Selection { Name = "Куплені", Created = DateTime.Now, Owner =
DateTime.Now, Owner = identityUser }
        };
        await _context.Selections.AddRangeAsync(selection);
        await _context.SaveChangesAsync();
    }

    return result;
}

public async Task<bool> ValidateUserCredentialsAsync(LoginRequest login)
{
    var user = await _userManager.FindByEmailAsync(login.Email);
    if (user == null)
    {
        return false;
    }

    return await _userManager.CheckPasswordAsync(user, login.Password);
}

public async Task<LoginResponse> LoginAsync(LoginRequest login)
{
    var user = await _userManager.FindByEmailAsync(login.Email);
    if (user != null)
    {
        var roles = await _userManager.GetRolesAsync(user);

        var passwordCheck = await _userManager.CheckPasswordAsync(user,
login.Password);

        if (passwordCheck && roles != null)
        {
            var jwtToken = await CreateJWTTokenAsync(user, roles.ToList());

```

```

        var userDTO = _mapper.Map<UserDTO>(user);

        return new LoginResponse
        {
            User = userDTO,
            AccessToken = jwtToken.AccessToken,
            RefreshToken = jwtToken.RefreshToken
        };
    }

    return null;
}
}
}

```

Сервіс BookService, який відповідає за дії з книгами

```

using AutoMapper;
using LibraryApp.API.Data;
using LibraryApp.API.Models.Database;
using LibraryApp.API.Models.Requests;
using LibraryApp.API.Models.Response;
using LibraryApp.API.Services.Interfaces;
using Microsoft.EntityFrameworkCore;

namespace LibraryApp.API.Services.Implementations
{
    public class BookService : IBookService
    {
        private readonly AppDbContext _context;
        private readonly IMapper _mapper;
        private readonly IAuthService _authorService;
        private readonly ICategoryService _categoryService;
        private readonly IFileService _fileService;

        public BookService(AppDbContext context, IMapper mapper, IAuthService
authorService, ICategoryService categoryService, IFileService fileService)
        {
            _context = context;
            _mapper = mapper;
            _authorService = authorService;
            _categoryService = categoryService;
            _fileService = fileService;
        }

        public async Task<ResponseBookDTO> CreateBookAsync(RequestBookDTO dto)
        {
            var authors = await _context.Authors.Where(a =>
dto.AuthorIds.Contains(a.Id)).ToListAsync();
            if (authors.Count != dto.AuthorIds.Count)
            {
                throw new ArgumentException("Some authors not found");
            }

            var categories = await _context.Categories.Where(c =>
dto.CategoryIds.Contains(c.Id)).ToListAsync();
            if (categories.Count != dto.CategoryIds.Count)
            {
                throw new ArgumentException("Some categories not found");
            }
        }
    }
}

```

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

```

        var book = _mapper.Map<Book>(dto);
        book.Authors = authors;
        book.Categories = categories;
        book.Created = DateTime.Now;

        var newFile = await _fileService.UploadFileAsync(dto.CoverFile);
        book.CoverFile = _mapper.Map<CoverFile>(newFile);
        newFile = await _fileService.UploadFileAsync(dto.BookContent);
        book.BookContent = _mapper.Map<BookContent>(newFile);

        await _context.Books.AddAsync(book);
        await _context.SaveChangesAsync();

        return _mapper.Map<ResponseBookDTO>(book);
    }

    public async Task<bool> DeleteBookAsync(Guid id)
    {
        var book = await _context.Books
            .Include(b => b.BookContent)
            .Include(b => b.CoverFile)
            .FirstOrDefaultAsync(b => b.Id == id);

        if (book == null)
        {
            return false;
        }

        var oldBookFilePath = Path.Combine("Uploads", book.BookContent.Name);
        var oldCoverFilePath = Path.Combine("Uploads", book.CoverFile.Name);

        _context.BookContents.Remove(book.BookContent);
        _context.CoverFiles.Remove(book.CoverFile);

        if (System.IO.File.Exists(oldBookFilePath))
        {
            System.IO.File.Delete(oldBookFilePath);
        }

        if (System.IO.File.Exists(oldCoverFilePath))
        {
            System.IO.File.Delete(oldCoverFilePath);
        }

        _context.Books.Remove(book);
        await _context.SaveChangesAsync();

        return true;
    }

    public async Task<ResponseBookDTO> GetBookByIdAsync(Guid id)
    {
        var book = await _context.Books
            .Include(b => b.Authors)
            .Include(b => b.BookContent)
            .Include(b => b.CoverFile)
            .Include(b => b.Categories)
            .Include(b => b.Comments)
            .ThenInclude(c => c.User)
            .FirstOrDefaultAsync(b => b.Id == id);

        if (book == null)
        {

```

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

        return null;
    }

    return _mapper.Map<ResponseBookDTO>(book);
}

public async Task<IEnumerable<ResponseBookDTO>> GetBooksAsync()
{
    var query = _context.Books
        .Include(b => b.Authors)
        .Include(b => b.Categories)
        .Include(b => b.BookContent)
        .Include(b => b.CoverFile)
        .Include(b => b.Comments).ThenInclude(c => c.User);
    var totalItems = await query.CountAsync();
    var books = await query
        .ToListAsync();

    var result =
        _mapper.Map<IEnumerable<ResponseBookDTO>>(books).ToList();

    return result;
}

public async Task<IEnumerable<ResponseBookDTO>>
GetBooksByCategoryAsync(Guid id, int? number = null)
{
    var category = await _context.Categories.FindAsync(id);
    if (category == null)
        throw new ArgumentException("No such category");

    var books = _context.Books
        .Include(b => b.Authors)
        .Include(b => b.BookContent)
        .Include(b => b.CoverFile)
        .Include(b => b.Categories)
        .Include(b => b.Comments).ThenInclude(c => c.User)
        .Where(b => b.Categories.Contains(category));

    if (number != null)
    {
        books = books.Take(5);
    }

    return _mapper.Map<IEnumerable<ResponseBookDTO>>(books).ToList();
}

public async Task<ResponseBookDTO> UpdateBookAsync(Guid id, RequestBookDTO
bookDTO)
{
    if (id != bookDTO.Id)
    {
        throw new ArgumentException("Book ID mismatch");
    }

    var book = await _context.Books
        .Include(b => b.Authors)
        .Include(b => b.BookContent)
        .Include(b => b.CoverFile)
        .Include(b => b.Categories)
        .FirstOrDefaultAsync(b => b.Id == id);

    if (book == null)
    {

```

```

        return null;
    }

    var authors = await _context.Authors.Where(a =>
bookDTO.AuthorIds.Contains(a.Id)).ToListAsync();
    if (authors.Count != bookDTO.AuthorIds.Count)
    {
        throw new ArgumentException("Some authors not found");
    }

    var categories = await _context.Categories.Where(a =>
bookDTO.CategoryIds.Contains(a.Id)).ToListAsync();
    if (categories.Count != bookDTO.CategoryIds.Count)
    {
        throw new ArgumentException("Some categories not found");
    }

    _mapper.Map(bookDTO, book);
    book.Authors = authors;
    book.Categories = categories;
    book.Created = book.Created;

    _context.Entry(book).State = EntityState.Modified;

    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!_context.Books.Any(e => e.Id == id))
        {
            return null;
        }
        else
        {
            throw;
        }
    }

    return _mapper.Map<ResponseBookDTO>(book);
}

file) public async Task<ResponseBookDTO> UpdateBookFileAsync(Guid id, IFormFile
{
    var book = await _context.Books
        .Include(b => b.BookContent)
        .Include(b => b.CoverFile)
        .FirstOrDefaultAsync(b => b.Id == id);

    if (book == null)
    {
        return null;
    }

    var newFile = await _fileService.UploadFileAsync(file);

    bool isCover = false;
    if (newFile.FileType.Contains("image"))
        isCover = true;

    string oldFilePath;

```

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ІПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

```

        if (isCover)
        {
            oldFilePath = Path.Combine("Uploads", book.CoverFile.Name);

            _context.CoverFiles.Remove(book.CoverFile);

            book.CoverFile = _mapper.Map<CoverFile>(newFile);
        }
        else
        {
            oldFilePath = Path.Combine("Uploads", book.BookContent.Name);

            _context.BookContents.Remove(book.BookContent);

            book.BookContent = _mapper.Map<BookContent>(newFile);
        }

        if (System.IO.File.Exists(oldFilePath))
        {
            System.IO.File.Delete(oldFilePath);
        }

        _context.Entry(book).State = EntityState.Modified;

        try
        {
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            throw new DbUpdateConcurrencyException("Concurrency error
occurred while updating book");
        }

        return _mapper.Map<ResponseBookDTO>(book);
    }

    public async Task<IEnumerable<ResponseBookDTO>>
GetBooksByKeywordAsync(string keywords)
    {
        if (keywords == "none")
        {
            return await GetBooksAsync();
        }

        var keywordList = keywords.Split(new[] { ' ', ',' },
StringSplitOptions.RemoveEmptyEntries);

        var exactMatchBooks = new List<Book>();
        var otherBooks = new List<Book>();

        var phraseExactMatchQuery = _context.Books
            .Include(b => b.Authors)
            .Include(b => b.Categories)
            .Include(b => b.BookContent)
            .Include(b => b.CoverFile)
            .Include(b => b.Comments).ThenInclude(c => c.User)
            .Where(b => b.Name.ToLower() == keywords.ToLower());

        var phraseExactMatches = await phraseExactMatchQuery.ToListAsync();
        exactMatchBooks.AddRange(phraseExactMatches);

        foreach (var keyword in keywordList)
        {

```

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

        var exactMatchQuery = _context.Books
            .Include(b => b.Authors)
            .Include(b => b.Categories)
            .Include(b => b.BookContent)
            .Include(b => b.CoverFile)
            .Where(b => b.Name.ToLower() == keyword.ToLower() ||
                b.Authors.Any(a =>
a.FirstName.ToLower() == keyword.ToLower() || a.LastName.ToLower() == keyword.ToLower())
                ||
                b.Categories.Any(c =>
c.Name.ToLower() == keyword.ToLower()));

        var exactMatches = await
exactMatchQuery.OrderBy(b=>b.Name).ToListAsync();
        foreach (var match in exactMatches)
        {
            if (!exactMatchBooks.Contains(match))
            {
                exactMatchBooks.Add(match);
            }
        }

        var otherQuery = _context.Books
            .Include(b => b.Authors)
            .Include(b => b.Categories)
            .Include(b => b.BookContent)
            .Include(b => b.CoverFile)
            .Where(b => !exactMatchBooks.Contains(b) &&
                (b.Name.ToLower().Contains(keyword.ToLower()) ||
                b.Authors.Any(a =>
a.FirstName.ToLower().Contains(keyword.ToLower()) ||
a.LastName.ToLower().Contains(keyword.ToLower())) ||
                b.Categories.Any(c =>
c.Name.ToLower().Contains(keyword.ToLower()))));

        var partialMatches = await otherQuery.OrderBy(b =>
b.Name).ToListAsync();
        foreach (var match in partialMatches)
        {
            if (!otherBooks.Contains(match))
            {
                otherBooks.Add(match);
            }
        }

        var combinedList = exactMatchBooks.Union(otherBooks).ToList();
        var result = _mapper.Map<IEnumerable<ResponseBookDTO>>(combinedList);

        return result;
    }

    public async Task<IEnumerable<SearchSuggestion>> GetSuggestionsAsync(string
term)
    {
        if (string.IsNullOrEmpty(term) || term == "none")
        {
            return new List<SearchSuggestion>();
        }

        term = term.ToLower();

        var suggestions = await _context.Books

```

```

        .Where(b =>
            b.Name.ToLower().StartsWith(term) ||
            b.Authors.Any(a =>
                a.FirstName.ToLower().Contains(term) ||
                a.LastName.ToLower().Contains(term)) ||
            b.Categories.Any(c =>
                c.Name.ToLower().Contains(term)))
        .Select(book => new SearchSuggestion
        {
            Text = book.Name
        })
        .Distinct()
        .Take(10)
        .ToListAsync();

    return suggestions;
}

public async Task<Dictionary<string, List<ResponseBookDTO>>>
GetRelatedBooks()
{
    var shuffledCategories = await _context.Categories.OrderBy(c =>
Guid.NewGuid()).ToListAsync();
    var selectedCategories = shuffledCategories.Take(4).ToList();

    var res = new Dictionary<string, List<ResponseBookDTO>>();

    foreach (var category in selectedCategories)
    {
        var categoryBooks = await GetBooksByCategoryAsync(category.Id,
5);

        res.Add(category.Name,
_mapper.Map<IEnumerable<ResponseBookDTO>>(categoryBooks).ToList());
    }

    return res;
}

public async Task<IEnumerable<ResponseBookDTO>> GetNewestBooks()
{
    var newestBooks = await _context.Books
.Include(b => b.Authors)
.Include(b => b.Categories)
.Include(b => b.BookContent)
.Include(b => b.CoverFile)
.Include(b => b.Comments)
.ThenInclude(c => c.User)
.OrderByDescending(b => b.Created).Take(5).ToListAsync();

    var result =
_mapper.Map<IEnumerable<ResponseBookDTO>>(newestBooks).ToList();

    return result;
}

public async Task<IEnumerable<ResponseBookDTO>> GetRandomBooks(int count=7)
{
    if (count <= 0)
    {
        throw new ArgumentException("Count must be greater than 0.",
nameof(count));
    }

    int totalBooks = await _context.Books

```

```

        .CountAsync();
        if (totalBooks == 0)
        {
            throw new InvalidOperationException("No books available in the
database.");
        }

        if (count > totalBooks)
        {
            count = totalBooks;
        }

        var res = await _context.Books
            .Include(b => b.Authors)
            .Include(b => b.Categories)
            .Include(b => b.BookContent)
            .Include(b => b.CoverFile)
            .Include(b => b.Comments)
            .ThenInclude(c => c.User)
            .OrderBy(b => Guid.NewGuid())
            .Take(count)
            .ToListAsync();

        return _mapper.Map<IEnumerable<ResponseBookDTO>>(res);
    }
}

```

Додаток Б

Інтерфейс читання книги

```

import React, { useState, useRef, useEffect } from "react";
import {
    View,
    Dimensions,
    StyleSheet,
    Text,
    BackHandler,
    Alert,
} from "react-native";
import { link } from "../constants/link";
import { SafeAreaView } from "react-native";
import Pdf from "react-native-pdf";
import Slider from "@react-native-community/slider";
import { Colors } from "../constants/styles";
import AsyncStorage from "@react-native-async-storage/async-storage";
import {
    Button,
    Checkbox,
    Dialog,
    IconButton,
    Portal,
} from "react-native-paper";
import { getBookStamp, setBookStamp } from "../util/book-data";
import { useAuth } from "../store/auth-context";
import useFetchWithToken from "../store/useFetchWithToken";

```

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

```

const EbookReader = ({ route, navigation }) => {
  const { id, url, bookId } = route.params;
  const resource = { uri: `${link}/uploads/${url}`, cache: true };
  const [numberOfPages, setNumberOfPages] = useState(0);
  const [currentPage, setCurrentPage] = useState();
  const [sliderVisible, setSliderVisible] = useState(true);
  const [paging, setPaging] = useState(true);
  const [horizontalScroll, setHorizontalScroll] = useState(true);
  const [visible, setVisible] = useState(false);
  const [isComponentRendered, setIsComponentRendered] = useState(false);
  const [oldPageNumber, setOldPageNumber] = useState(1);
  const fetchWithToken = useFetchWithToken();

  const showDialog = () => setVisible(true);
  const hideDialog = () => setVisible(false);

  const pdfRef = useRef(null);

  const saveProgress = async (page) => {
    if (page !== undefined) {
      try {
        await AsyncStorage.setItem(`viewingStamp=${url}`, page.toString());
      } catch (error) {
        console.error("Error saving data: ", error);
      }
    }
  };

  const toggleSliderVisibility = () => {
    setSliderVisible(!sliderVisible);
  };

  useEffect(() => {
    const getReadingProgress = async () => {
      try {
        const res = await AsyncStorage.getItem(`viewingStamp=${url}`);
        if (res !== null) {
          setOldPageNumber(parseInt(res));
          setCurrentPage(parseInt(res));
        } else {

          const response = await fetchWithToken(getBookStamp, url);
          if (response !== null) {
            setOldPageNumber(parseInt(res));
            setCurrentPage(parseInt(res));
          }
        }
      } catch (error) {
        setCurrentPage(1);
      }
    }
  });

```

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		90

```

};

const loadSettings = async () => {
  try {
    const pagingSettings = await AsyncStorage.getItem("paging");
    const horizontalScrollSettings = await AsyncStorage.getItem(
      "horizontalScroll"
    );
    if (pagingSettings !== null) setPaging(JSON.parse(pagingSettings));
    if (horizontalScrollSettings !== null)
      setHorizontalScroll(JSON.parse(horizontalScrollSettings));
  } catch (e) {
    console.error("Failed to load settings.", e);
  }
};

getReadingProgress();
loadSettings();
setIsComponentRendered(true);
}, []);

useEffect(() => {
  if (isComponentRendered) {
    saveProgress(currentPage);
  }
  const backHandler = BackHandler.addEventListener(
    "hardwareBackPress",
    () => {
      Alert.alert(
        "Вихід з перегляду книги",
        "Ви впевнені, що хочете повернутися на попередню сторінку?",
        [
          {
            text: "Відміна",
            onPress: () => null,
            style: "cancel",
          },
          {
            text: "Вийти",
            onPress: async () => {
              await saveProgressToDB(currentPage, numberOfPages, id, bookId);
              navigation.goBack();
            },
          },
        ],
        { cancelable: false }
      );
      return true;
    }
  );
});

```

```

    return () => backHandler.remove();
  }, [currentPage, isComponentRendered]);

const saveProgressToDB = async (value, numberOfPages, fileId, bookId) => {
  try {
    console.log(value, numberOfPages, fileId, bookId);
    const response = await fetchWithToken(setBookStamp,
      value=value,
      numberOfPages=numberOfPages,
      fileId=fileId,
      bookId=bookId
    );
    console.log("Data saved successfully");
  } catch (error) {
    console.error("Error saving data: ", error);
  }
};

const handleCheckboxChange = async (key, value) => {
  try {
    if (key === "paging") {
      setPaging(value);
    } else if (key === "horizontalScroll") {
      setHorizontalScroll(value);
    }
    await AsyncStorage.setItem(key, JSON.stringify(value));
  } catch (e) {
    console.error("Failed to save settings.", e);
  }
};

return (
  <SafeAreaView
    style={{ flex: 1, position: "relative", backgroundColor: "white" }}
  >
    {sliderVisible && (
      <View style={styles.header}>
        <IconButton
          icon="arrow-left"
          size={24}
          iconColor={Colors.primary500}
          style={{ position: "relative" }}
          onPress={async () => {
            await saveProgressToDB(currentPage, numberOfPages, id, bookId);
            navigation.goBack();
          }}
        />
        <IconButton
          icon="cog"

```

```

        size={24}
        iconColor={Colors.primary500}
        style={{ position: "relative" }}
        onPress={showDialog}
    />
    <Portal>
        <Dialog visible={visible} onDismiss={hideDialog}>
            <Dialog.Title>Налаштування перегляду</Dialog.Title>
            <Dialog.Content>
                <View style={styles.checkboxContainer}>
                    <Checkbox
                        status={paging ? "checked" : "unchecked"}
                        onPress={() => handleCheckboxChange("paging", !paging)}
                    />
                    <Button
                        onPress={() => handleCheckboxChange("paging", !paging)}
                    >
                        Поділ на сторінки
                    </Button>
                </View>
                <View style={styles.checkboxContainer}>
                    <Checkbox
                        status={horizontalScroll ? "checked" : "unchecked"}
                        onPress={() =>
                            handleCheckboxChange(
                                "horizontalScroll",
                                !horizontalScroll
                            )
                        }
                    />
                    <Button
                        onPress={() =>
                            handleCheckboxChange(
                                "horizontalScroll",
                                !horizontalScroll
                            )
                        }
                    >
                        Горизонтальне прокручування
                    </Button>
                </View>
            </Dialog.Content>
            <Dialog.Actions>
                <Button onPress={hideDialog}>ОК</Button>
            </Dialog.Actions>
        </Dialog>
    </Portal>
</View>
)}}

```

```

<Pdf
  ref={pdfRef}
  style={{
    width: Dimensions.get("window").width,
    height: Dimensions.get("window").height,
    marginTop: 35,
  }}
  page={oldPageNumber}
  horizontal={horizontalScroll}
  enablePaging={paging}
  trustAllCerts={false}
  source={resource}
  onLoadComplete={({numPages, filePath}) => {
    setNumberOfPages(numPages);
  }}
  onPageChanged={({page, numberOfPages}) => {
    if (currentPage !== page) setCurrentPage(page);
  }}
  onPageSingleTap={toggleSliderVisibility}
/>

{sliderVisible && (
  <View style={styles.progressContainer}>
    <Text style={styles.progressText}>
      {currentPage} / {numberOfPages}
    </Text>
    <Slider
      style={styles.slider}
      minimumValue={1}
      maximumValue={numberOfPages}
      value={currentPage}
      onValueChange={(value) => setCurrentPage(Math.floor(value))}
      onSlidingComplete={(value) => {
        if (pdfRef.current) {
          setCurrentPage(Math.floor(value));
          pdfRef.current.setPage(Math.floor(value));
        }
      }}
      minimumTrackTintColor={Colors.primary800}
      maximumTrackTintColor={Colors.primary800}
      thumbTintColor={Colors.primary800}
    </Slider>
  </View>
)}
</SafeAreaView>
);
};
export default EbookReader;

const styles = StyleSheet.create({

```

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		94

```
container: {
  flex: 1,
  justifyContent: "center",
  alignItems: "center",
},
checkboxContainer: {
  flexDirection: "row",
  alignItems: "center",
},
header: {
  position: "absolute",
  top: 35,
  zIndex: 1,
  width: "100%",
  backgroundColor: "rgba(250, 250, 250, 0.9)",
  paddingHorizontal: 10,
  paddingVertical: 15,
  flexDirection: "row",
  justifyContent: "space-between",
  elevation: 10,
},
progressContainer: {
  position: "absolute",
  bottom: 0,
  width: "100%",
  backgroundColor: "rgba(250, 250, 250, 0.9)",
  padding: 10,
  elevation: 20,
},
progressText: {
  color: Colors.primary800,
  textAlign: "center",
  marginBottom: 10,
},
slider: {
  width: "100%",
  height: 40,
},
});
```

Додаток В

Форма додавання/редагування книги

```
import React, { useEffect, useState } from "react";
import {
  Dialog,
  DialogTitle,
  DialogContent,
  DialogActions,
  Button,
```

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		95

```

    TextField,
    Box,
    InputLabel,
  } from "@mui/material";
import { Formik, Field, Form, ErrorMessage } from "formik";
import * as Yup from "yup";
import { useAuth } from "../hooks/AuthProvider";
import { SearchableSelect } from "../SearchableSelect";
import { addBook, getAuthors, getCategories } from "../data/info";
import useFetchWithToken from "../hooks/useFetchWithToken";

const validationSchema = Yup.object({
  name: Yup.string().required("Введіть назву книги"),
  description: Yup.string().required("Введіть опис"),
  price: Yup.number()
    .typeError("Ціна має бути числом")
    .transform((originalValue, parsedValue) => {
      if (typeof parsedValue === "string") {
        return parseFloat(parsedValue.replace(/,/g, "."));
      }
      return parseFloat(parsedValue);
    })
    .min(0, "Ціна має бути 0 або більше"),
  authorIds: Yup.array()
    .min(1, "Виберіть хоча б одного автора")
    .required("Виберіть автора"),
  categoryIds: Yup.array()
    .min(1, "Виберіть хоча б одну категорію")
    .required("Виберіть категорію"),
  coverFile: Yup.mixed()
    .test(
      "fileSize",
      "Файл надто великий",
      (value) => !value || (value && value.size <= 10485760)
    )
    .required("Додайте обкладинку"),
  bookContent: Yup.mixed()
    .test(
      "fileSize",
      "Файл надто великий",
      (value) => !value || (value && value.size <= 20971520)
    )
    .required("Додайте електронну книгу"),
});

const editValidationSchema = Yup.object({
  name: Yup.string().required("Введіть назву книги"),
  description: Yup.string().required("Введіть опис"),
  price: Yup.number()

```

```

.typeError("Ціна має бути числом")
.transform((originalValue, parsedValue) => {
  if (typeof parsedValue === "string") {
    return parseFloat(parsedValue.replace(/,/g, "."));
  }
  return parsedValue;
})
.min(0, "ціна має бути 0 або більше"),
.required("Введіть ціну"),
authorIds: Yup.array()
  .min(1, "Виберіть хоча б одного автора")
  .required("Виберіть автора"),
categoryIds: Yup.array()
  .min(1, "виберіть хоча б одну категорію")
  .required("Виберіть категорію"),
});

export const PublicationForm = ({ formValues, open, handleClose, notify }) => {
  const [authors, setAuthors] = useState([]);
  const [genres, setGenres] = useState([]);
  const user = useAuth();
  const fetchWithToken = useFetchWithToken();

  useEffect(() => {
    const fetchAuthors = async () => {
      const data = await fetchWithToken(getAuthors);
      setAuthors(data);
    };

    const fetchGenres = async () => {
      const data = await fetchWithToken(getCategories);
      setGenres(data);
    };

    fetchAuthors();
    fetchGenres();
  }, []);

  const handleSubmit = async (values, { setSubmitting }) => {
    try {
      var body;
      if (formValues === null) {
        var formData = new FormData();

        if (formValues) formData.append("id", formValues?.id);
        formData.append("name", values.name);

        let priceFloat;
        if (!/^-\?\d+$/i.test(values.price)) {
          priceFloat = values.price.toString().includes(",")

```

```

        ? values.price
        : values.price.replace(".", "");
    } else priceFloat = values.price;
    console.log(priceFloat);
    formData.append("price", priceFloat);

    formData.append("description", values.description);
    formData.append("authorIds", values.authorIds);
    formData.append("categoryIds", values.categoryIds);

    if (formValues === null && values.coverFile && values.bookContent) {
        formData.append("coverFile", values.coverFile);
        formData.append("bookContent", values.bookContent);
    }

    body = formData;
    console.table(Object.fromEntries(formData));
} else {
    let priceFloat = values.price.includes(",")
        ? parseFloat(values.price.replace(",", "."))
        : parseFloat(values.price);
    const requestBookDTO = {
        id: formValues?.id,
        name: values.name,
        description: values.description,
        authorIds: values.authorIds,
        price: priceFloat,
        categoryIds: values.categoryIds,
    };

    body = JSON.stringify(requestBookDTO);
    console.log(requestBookDTO);
}

const method = formValues ? "PUT" : "POST";
const url = formValues
    ? `https://localhost:7271/api/Books/${formValues.id}`
    : "https://localhost:7271/api/Books/Add";
const contentType = formValues
    ? {
        "Content-Type": "application/json",
    }
    : {};
const data = await fetchWithToken(
    addBook,
    url,
    method,
    contentType,
    body
);

```

```

        notify("Зміни збережено", "success");
        setSubmitting(false);
        handleClose();
    } catch (error) {
        notify("Відбулась помилка, спробуйте ще раз", "error");
        setSubmitting(false);
    }
};

const initialAddValues = {
    name: "",
    description: "",
    price: 0,
    authorIds: [],
    categoryIds: [],
    coverFile: null,
    bookContent: null,
};

const initialEditValues = {
    name: formValues?.name,
    description: formValues?.description,
    price: formValues?.price.toString(),
    authorIds: formValues?.authors?.map((author) => author.id),
    categoryIds: formValues?.categories?.map((category) => category.id),
};

return (
    <Dialog open={open} onClose={handleClose} fullWidth maxWidth="sm">
        <DialogTitle>
            {formValues ? "Редагувати книгу" : "Додати книгу"}
        </DialogTitle>
        <DialogContent>
            <Formik
                initialValues={
                    formValues === null ? initialAddValues : initialEditValues
                }
                validationSchema={
                    formValues === null ? validationSchema : editValidationSchema
                }
                validateOnChange={true}
                onSubmit={(values, actions) => {
                    console.log("onSubmit called");
                    handleSubmit(values, actions);
                }}
            >
                {({ isSubmitting, setFieldValue, values, errors, touched }) => {
                    console.log("Form values:", values);
                    console.log("Form errors:", errors);
                    //console.log("Is submitting:", isSubmitting);
                }}
            </Formik>
        </DialogContent>
    </Dialog>
);

```

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		99

```

return (
  <Form>
    <Box mb={2}>
      <Field
        name="name"
        as={TextField}
        label="Назва"
        fullWidth
        variant="outlined"
        error={touched.name && Boolean(errors.name)}
        helperText={<ErrorMessage name="name" />}
      />
    </Box>

    <Box mb={2}>
      <Field
        name="description"
        as={TextField}
        label="Опис"
        multiline
        rows={4}
        fullWidth
        variant="outlined"
        error={touched.description && Boolean(errors.description)}
        helperText={<ErrorMessage name="description" />}
      />
    </Box>

    <Box mb={2}>
      <Field
        name="price"
        as={TextField}
        label="Ціна"
        fullWidth
        variant="outlined"
        error={touched.price && Boolean(errors.price)}
        helperText={<ErrorMessage name="price" />}
      />
    </Box>

    <Box mb={2}>
      <SearchableSelect
        name="authorIds"
        options={authors.map((author) => ({
          value: author.id,
          label: `${author.firstName} ${
            author.lastName !== "Unknown" ? author.lastName : ""
          }`,
        })`},
      )}}
      label="Автори"
    </Box>
  </Form>
)

```

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		100

```

    />
  </Box>

  <Box mb={2}>
    <SearchableSelect
      name="categoryIds"
      options={genres.map((genre) => ({
        value: genre.id,
        label: genre.name,
      })))}
      label="Категорії"
    />
  </Box>

  {formValues === null && (
    <>
      <Box mb={2}>
        <InputLabel>Обкладинка</InputLabel>
        <input
          id="coverFile"
          name="coverFile"
          type="file"
          onChange={(event) => {
            setFieldValue(
              "coverFile",
              event.currentTarget.files[0]
            );
          }}
          accept=".jpg, .png, .gif"
        />
        <ErrorMessage
          name="coverFile"
          component="div"
          style={{ color: "red" }}
        />
      </Box>
      <Box mb={2}>
        <InputLabel>Книга</InputLabel>
        <input
          id="bookContent"
          name="bookContent"
          type="file"
          onChange={(event) => {
            setFieldValue(
              "bookContent",
              event.currentTarget.files[0]
            );
          }}
          accept=".pdf"
        />
      </Box>
    </>
  )}

```

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		101

```

        <ErrorMessage
            name="bookContent"
            component="div"
            style={{ color: "red" }}
        />
    </Box>
</>
)}

<DialogActions>
    <Button onClick={handleClose} color="secondary">
        Відміна
    </Button>
    <Button
        variant="contained"
        color="primary"
        type="submit"
        disabled={isSubmitting}
    >
        Зберегти
    </Button>
</DialogActions>
</Form>
);
}}
</Formik>
</DialogContent>
</Dialog>
);
};

```

		Рябова Є.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		102