НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

комп'ютерних наук
(назва кафедри)

_____ Голуб Б.Л.
(підпис) (ПІБ)

"_04___" _червня___ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

«Інформаційна система для управління онлайн бібліотекою»

Спеціальність 122 – «Комп'ютерні науки»

Гарант освітньої програми

_____Д.е.н., професор_____ _____ ___Руденський Р.А.____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

_____ _____ ___Панкратьєв В.О.____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконала _____ _____Косогор О.В._____
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерних наук
к.т.н., доцент _____ Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"16" грудня 2024 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студентці
Косогор Ользі Володимирівні
(прізвище, ім'я, по батькові)

Спеціальність 122 – «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи «Інформаційна система для управління онлайн бібліотекою»
затверджена наказом ректора НУБіП України № 2246 "С" від "16".12.2024р.

Термін подання завершеної роботи на кафедру 2025. 05. 25
(рік, місяць, число)

Вихідні дані до бакалаврської кваліфікаційної роботи
Публічні джерела,

Перелік питань, які потрібно розробити:
Системний аналіз предметної області, інформаційне забезпечення, прикладне програмне забезпечення, рекомендації щодо впровадження та експлуатації системи

Перелік графічних документів (за потреби) діаграми та моделі

Керівник бакалаврської кваліфікаційної роботи _____ Панкратьєв В.О.
(підпис) (прізвище та ініціали)

Завдання прийняла до виконання _____ Косогор О.В.
(підпис) (прізвище та ініціали студента)

Дата отримання завдання "16" грудня 2024 р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення.

ПС – програмна система.

SQL (Structured Query Language) – структурована мова запитів.

СУБД – система управління базами даних.

UML (Unified Modeling Language) – уніфікована мова моделювання.

GUI (Graphical User Interface) – графічний інтерфейс користувача.

IDE (Integrated Development Environment) – інтегроване середовище розробки.

RAM (Random Access Memory) – оперативна пам’ять.

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується стрімким поширенням цифрових рішень у всіх сферах людської діяльності, зокрема в галузі освіти, наукової діяльності та корпоративного знаннєвого обміну. В умовах зростаючого обсягу електронних інформаційних ресурсів постає потреба в ефективному керуванні цифровими бібліотеками, що забезпечують швидкий доступ до структурованих колекцій електронних книг, документів та інших навчально-методичних матеріалів. Зокрема, навчальні заклади, наукові установи, компанії та окремі користувачі все частіше звертаються до цифрових бібліотек як до зручного інструменту для зберігання, пошуку, читання й обміну знаннями.

Актуальність завдання, що вирішується в межах кваліфікаційної роботи, полягає в необхідності створення універсального, масштабованого та інтуїтивно зрозумілого програмного продукту для управління онлайн-бібліотекою. Серед існуючих рішень багато платформ є або надто складними у використанні для пересічного користувача, або такими, що не враховують особливості локального впровадження в навчальних, корпоративних або приватних умовах. Саме тому розробка настільного клієнт-серверного додатку з можливістю гнучкого адміністрування, організації зручного каталогу, підтримки електронного читання й керування власними списками книг має високу практичну значущість.

Метою розробки програмного додатку є створення десктопної інформаційної системи, що забезпечить повний цикл роботи з електронною бібліотекою: від додавання та організації контенту до зручного пошуку, читання і завантаження книг. Рішення має бути адаптоване для різних категорій користувачів: викладачів, студентів, співробітників компаній або просто ентузіастів, які впорядковують особисту колекцію книг. У процесі розробки враховувалися потреби таких користувачів як адміністратори (які керують бібліотекою) і кінцеві користувачі (які читають або завантажують книги, створюють власні списки тощо).

Для досягнення поставленої мети в роботі використано сучасні методи та технології, що забезпечують гнучкість і надійність програмного забезпечення. Зокрема:

- мова програмування C# використовується для реалізації клієнтської частини у середовищі Windows Forms, що дозволяє створювати зручний графічний інтерфейс користувача;
- PostgreSQL — як система управління базами даних, що забезпечує зберігання всієї структурованої інформації про книги, користувачів і списки;
- Supabase — платформа для хостингу бази даних і організації бекенд-функцій, яка забезпечує підключення до хмарної інфраструктури через API;
- додатково застосовуються засоби для валідації даних, обробки запитів, роботи з файлами, а також забезпечення безпеки автентифікації та авторизації.

Результати роботи були **апробовані** на VII Всеукраїнській науково-практичній конференції студентів і аспірантів “Теоретичні та прикладні аспекти розробки комп’ютерних систем”. За результатами дослідження підготовлено тези, у яких висвітлено підходи до розробки системи, архітектурні рішення, а також ключові функціональні можливості майбутнього програмного забезпечення.

Пояснювальна записка дипломної кваліфікаційної роботи викладена на 82 сторінках та містить:

- 17 використаних джерела інформації;
- 4 додатки, що включають фрагменти програмного коду, приклади SQL-запитів тощо.

Основна частина роботи поділяється на чотири розділи:

1. **Системний аналіз предметної області** — викладено постановку задачі, проведено аналіз джерел та аналогічних систем, змодельовано предметну область для подальшої реалізації.

2. **Інформаційне забезпечення** — описано логічну модель даних, обґрунтовано вибір СУБД, розглянуто процес створення інформаційної бази.
3. **Прикладне програмне забезпечення** — охоплює структуру програмного забезпечення, обґрунтування вибору інструментів, алгоритмізацію окремих модулів та реалізацію функціоналу.
4. **Рекомендації щодо впровадження та експлуатації системи** — наведено результати тестування, вимоги до системи для розгортання, описано склад інсталяційного пакету та загальні рекомендації з використання додатку.

Таким чином, у роботі вирішується важлива практична задача — створення доступного програмного продукту для ефективного керування електронними бібліотеками з широкими можливостями як для користувачів, так і для адміністраторів системи.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

У сучасних умовах стрімкого розвитку цифрових технологій та активного переходу багатьох сфер діяльності в онлайн-простір, особливої актуальності набуває автоматизація бібліотечних процесів, зокрема створення та впровадження інформаційних систем для управління електронними бібліотеками. Електронна бібліотека - інформаційна система, яка складається з впорядкованого фонду електронних ресурсів, каталогу на цей фонд та комплексу апаратно-програмних засобів, що підтримують стабільне функціонування пошукової системи і дають можливість оперативного поповнення, реєстрації, довготривалого зберігання фонду електронної бібліотеки та розподіленого доступу до нього через мережу Інтернет [1].

Предметна область охоплює функціонування бібліотечної системи в онлайн-середовищі, яка передбачає організацію обліку книг, електронного архівування, управління каталогами, реєстрації користувачів, контролю доступу, формування запитів на отримання чи завантаження матеріалів, ведення статистики використання ресурсів тощо. Зважаючи на зміну поведінкових моделей користувачів та зростання попиту на швидкий, зручний і цілодобовий доступ до інформації, цифрові бібліотеки стають не лише альтернативою традиційним закладам, а й основним джерелом отримання знань у багатьох галузях.

У межах предметної області ключовими об’єктами є:

- користувачі (читачі, адміністратори, модератори);
- книжкові ресурси (електронні книги, журнали, документи);
- каталоги (жанри, автори, мова, рік видання, формати файлів тощо);
- функції пошуку і фільтрації (за назвою, автором, жанром, доступністю);
- процеси реєстрації, авторизації та управління доступом;
- функціонал додавання, редагування, видалення книжок;

- механізм збереження та завантаження файлів;
- системні журнали та звіти про активність.

Онлайн бібліотечні системи можуть обслуговувати як загальну читацьку аудиторію, так і бути частиною корпоративного чи навчального середовища (наприклад, в університетах, школах, наукових інституціях). Залежно від масштабу реалізації, така система може підтримувати тисячі одиниць літератури, десятки тисяч користувачів, мати багатоетапну систему прав доступу та інтегруватися з іншими інформаційними сервісами (аналітичними модулями, системами електронного навчання тощо).

У рамках даної предметної області особливо важливою є побудова ефективної архітектури інформаційної системи, що дозволяє забезпечити надійне збереження контенту, зручність користування, масштабованість і безпеку. Актуальними завданнями виступають: підтримка різних форматів файлів (PDF, EPUB, DOCX), пошук і фільтрація за різними критеріями, швидке завантаження, керування правами доступу, адміністрування системи.

Таким чином, предметна область охоплює сукупність процесів і ресурсів, спрямованих на організацію, підтримку та розвиток інформаційної платформи для зберігання, поширення та використання цифрового бібліотечного контенту, що є основою для подальшого аналізу, моделювання та створення програмної системи управління онлайн бібліотекою.

1.2 Аналіз вимог до програмної системи

Аналіз вимог є ключовим етапом системного аналізу, що дозволяє сформувати уявлення про функціональні й нефункціональні характеристики майбутньої програмної системи, визначити коло користувачів, сценарії її використання та очікувані результати. На основі цього аналізу здійснюється подальше проєктування та реалізація системи.

Загальна характеристика системи

Інформаційна система для управління онлайн бібліотекою належить до класу прикладних автоматизованих інформаційних систем, призначених для організації зберігання, пошуку, каталогізації та надання доступу до електронних ресурсів бібліотеки. Система має на меті автоматизувати рутинні операції, забезпечити ефективну взаємодію між користувачем і електронним книжковим фондом, спростити адміністративне управління бібліотекою.

За характером оброблюваної інформації розроблювана система є документно-орієнтованою, а за масштабом — придатною як для локального використання (наприклад, у межах навчального закладу), так і для роботи в глобальному мережевому середовищі. За ступенем автоматизації вона передбачає повний цикл обробки електронних документів — від завантаження до доступу кінцевих користувачів.

Функціональні вимоги

Головною метою діяльності електронної бібліотеки є максимально повне та оперативне обслуговування користувачів [2]. У відповідності до поставленої мети, система повинна реалізовувати такі основні функції:

- реєстрація та автентифікація користувачів;
- розмежування прав доступу: читачі, редактори, адміністратори;
- додавання нових книг до бази, включаючи завантаження електронних файлів та введення супровідних метаданих (назва, автор, жанр, рік видання, короткий опис тощо);
- каталогізація книг та підтримка фільтрації за різними критеріями (автор, жанр, рік, формат);
- пошук книг за ключовими словами;
- перегляд детальної інформації про книгу та завантаження її електронної версії;

- можливість редагування й видалення записів (для користувачів з відповідними правами);
- формування звітів про кількість доданих/завантажених книг, активність користувачів тощо;
- ведення журналу дій (логування системних подій);
- надання адміністративного інтерфейсу для управління користувачами й параметрами системи.

Нефункціональні вимоги

Крім реалізації функціональних можливостей, система повинна відповідати ряду нефункціональних вимог, а саме:

- юзабіліті (зручність користування): зрозумілий і логічний інтерфейс користувача з підтримкою адаптивного дизайну;
- продуктивність: швидке виконання основних операцій, таких як пошук, перегляд і завантаження ресурсів;
- надійність: забезпечення збереження даних у випадку збоїв, перевірка цілісності файлів;
- безпека: захист персональних даних користувачів, контроль доступу до адміністративного функціоналу, шифрування з’єднань;
- масштабованість: підтримка зростання кількості користувачів і обсягів інформації без суттєвого зниження продуктивності;
- супроводжуваність і модифікованість: легкість у підтримці, оновленні, доповненні функціоналу.

Таким чином, вимоги до інформаційної системи визначають її структуру, обсяг функціональності, сценарії використання, а також технічні та експлуатаційні параметри, яким вона повинна відповідати. Сформульовані вимоги виступають основою для побудови моделі системи та подальшої її реалізації у вигляді повнофункціонального програмного продукту.

1.3 Моделювання предметної області

Для забезпечення повноти й чіткості розуміння логіки функціонування інформаційної системи управління онлайн бібліотекою, доцільно застосувати засоби об’єктно-орієнтованого моделювання предметної області. Основним інструментом такого моделювання є мова UML (Unified Modeling Language), яка призначена для забезпечення стандартного способу візуалізації проектування системи [3]. Вона дозволяє описати функціональність, структуру і поведінку системи за допомогою візуальних діаграм.

Діаграма прецедентів (Use Case Diagram)

Діаграма прецедентів дозволяє візуалізувати функціональні вимоги до системи у вигляді сценаріїв взаємодії між користувачами (акторами) та системою. На діаграмі прецедентів (додаток В) зображено взаємодію двох основних акторів: **Користувача** та **Адміністратора** з функціональністю системи.

Користувач має доступ до таких функціональних можливостей:

- реєстрація та авторизація;
- редагування власного профілю;
- перегляд каталогу книг, пошук і перегляд детальної інформації про книги;
- читання або завантаження електронних книг;
- оцінювання книги та написання відгуку;
- управління персональними списками (створення, редагування, видалення).

Адміністратор має розширені повноваження, які, окрім можливостей користувача, включають:

- керування книжковим фондом (додавання, редагування, видалення книг);

- перегляд та адміністрування акаунтів користувачів;
- редагування структури каталогу бібліотеки (жанри, серії тощо);
- ведення звітності (створення звітів про діяльність бібліотеки);
- управління системними налаштуваннями.

Діаграма діяльності (Activity Diagram)

Для візуалізації логіки виконання типових сценаріїв взаємодії користувача із системою використовуються діаграми діяльності. На рисунку 1.2 зображено діаграму діяльності для одного із ключових сценаріїв взаємодії користувача із системою – «Завантаження книги користувачем».

Сценарій розпочинається з того, що авторизований користувач обирає книгу для завантаження. Система перевіряє, чи доступна книга для завантаження. Якщо ресурс недоступний, користувачу виводиться повідомлення про тимчасову неможливість завантаження.

У випадку, якщо книга доступна, система формує запит до сервера, який, у свою чергу, виконує перевірку прав доступу користувача до запитуваного ресурсу. У разі позитивного результату система генерує посилання на файл та передає його користувачу. Якщо ж права доступу відсутні, користувачу виводиться повідомлення про обмеження доступу.

Вся логіка сценарію — від вибору книги до завершення процесу — подана у вигляді діаграми діяльності (рис.1.2), яка відображає основні гілки прийняття рішень та взаємодію між користувачем і сервером системи.

Рис.1.2 Діаграма діяльності для процесу завантаження книги

Діаграма послідовності (Sequence Diagram)

Для наочного відображення взаємодії між об’єктами системи в межах одного з ключових сценаріїв використовується діаграма послідовності. На діаграмі послідовності (рис.1.3) зображено процес додавання нової книги адміністратором.

Сценарій починається з того, що адміністратор відкриває форму додавання книги в інтерфейсі. Далі вводяться всі необхідні метадані (назва, опис, рік видання), обираються автори та жанри, а також завантажується файл електронної книги.

Після введення даних система формує запит, який надсилається на сервер. Серверна частина виконує перевірку коректності отриманої інформації. Якщо дані містять помилки або не відповідають вимогам, адміністратор отримує повідомлення про помилку з поясненням. У разі успішної валідації система зберігає як метадані книги, так і файл, після чого надсилає підтвердження про успішне завершення операції.

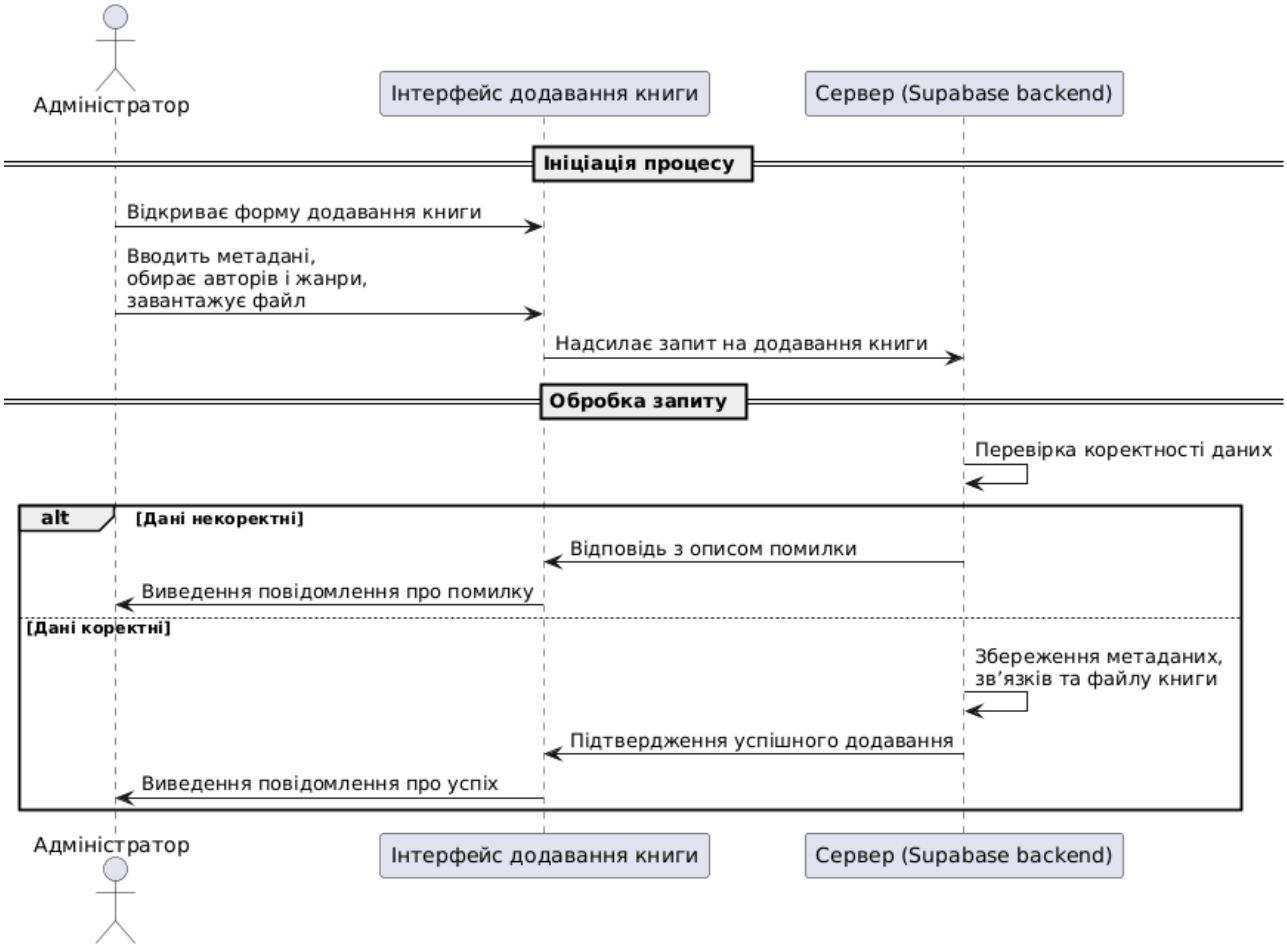


Рис.1.3 Діаграма послідовності процесу додавання книги

Застосування UML-діаграм дозволяє формалізувати логіку роботи системи, чітко визначити ролі користувачів, їхні дії, а також структуру даних. Це значно спрощує процес розробки, тестування та подальшого супроводу програмної системи.

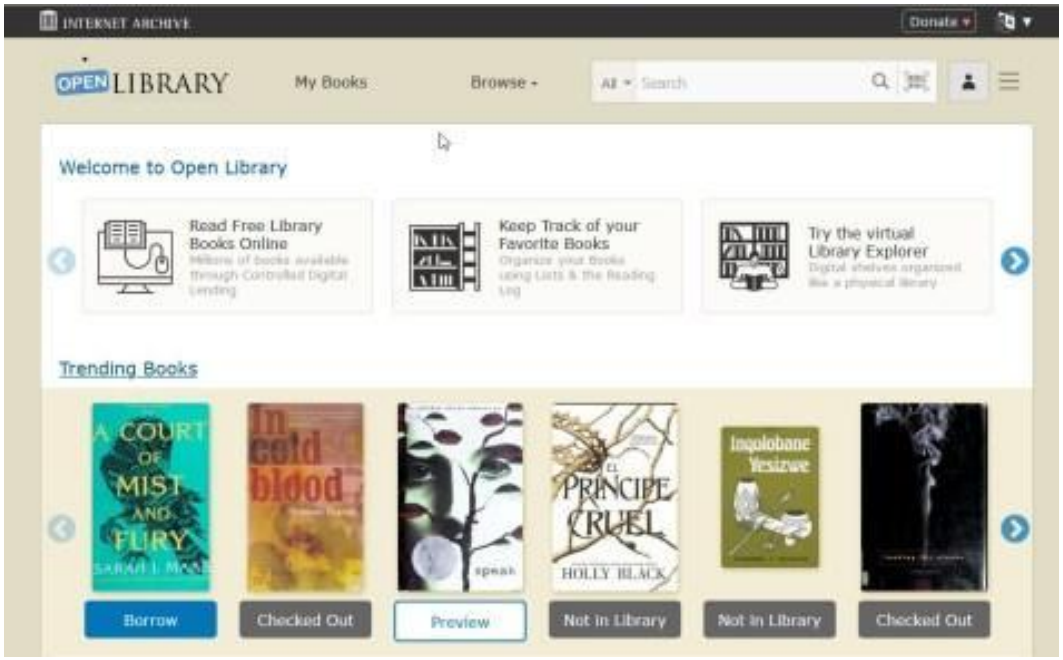
1.4 Огляд інформаційних джерел та існуючих рішень

У процесі розробки інформаційної системи доцільно проаналізувати існуючі програмні продукти зі схожим функціональним призначенням. Такий огляд дозволяє виявити сильні та слабкі сторони подібних рішень, визначити найкращі практики реалізації функцій, а також сформулювати вимоги до власного проєкту. Нижче подано аналіз чотирьох програмних продуктів, що частково або повністю реалізують функціональність, пов'язану з електронними бібліотеками, каталогами книг та інструментами для взаємодії з ними.

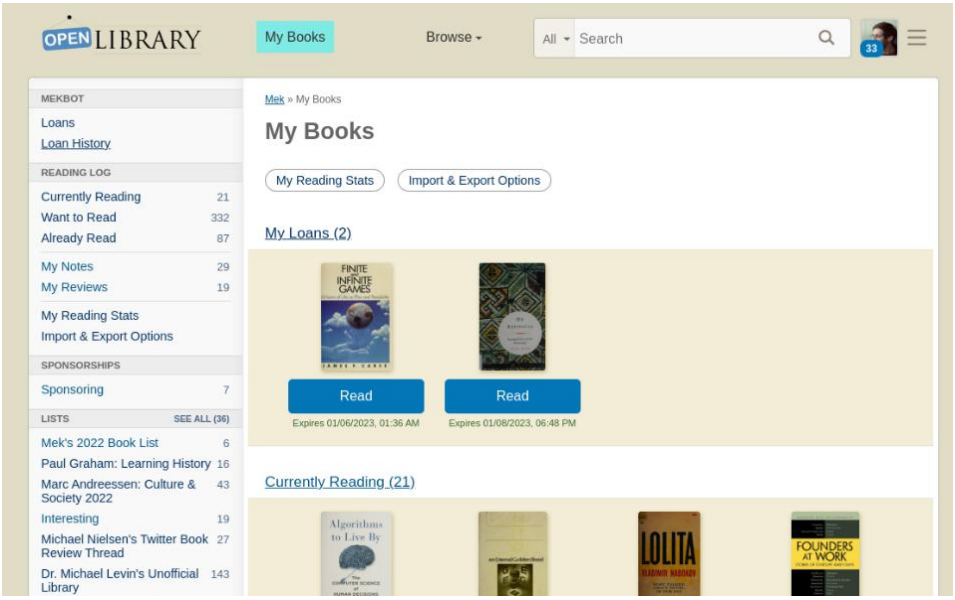
Open Library

Open Library — це відкрита веб-платформа, яка позиціонує себе як "інтернет-бібліотека для всіх". Вона надає доступ до мільйонів цифрових копій книг, які можна читати онлайн або позичити у цифровому форматі. Основна функціональність включає пошук книг за назвою, автором або темою, перегляд картки книги з описом та можливістю читання або завантаження, додавання до списків та перегляд історії користувача. Open Library також підтримує функції авторизації, додавання рецензій і оцінок, проте не має повноцінного настільного застосунку. [7]

а)



б)



в)

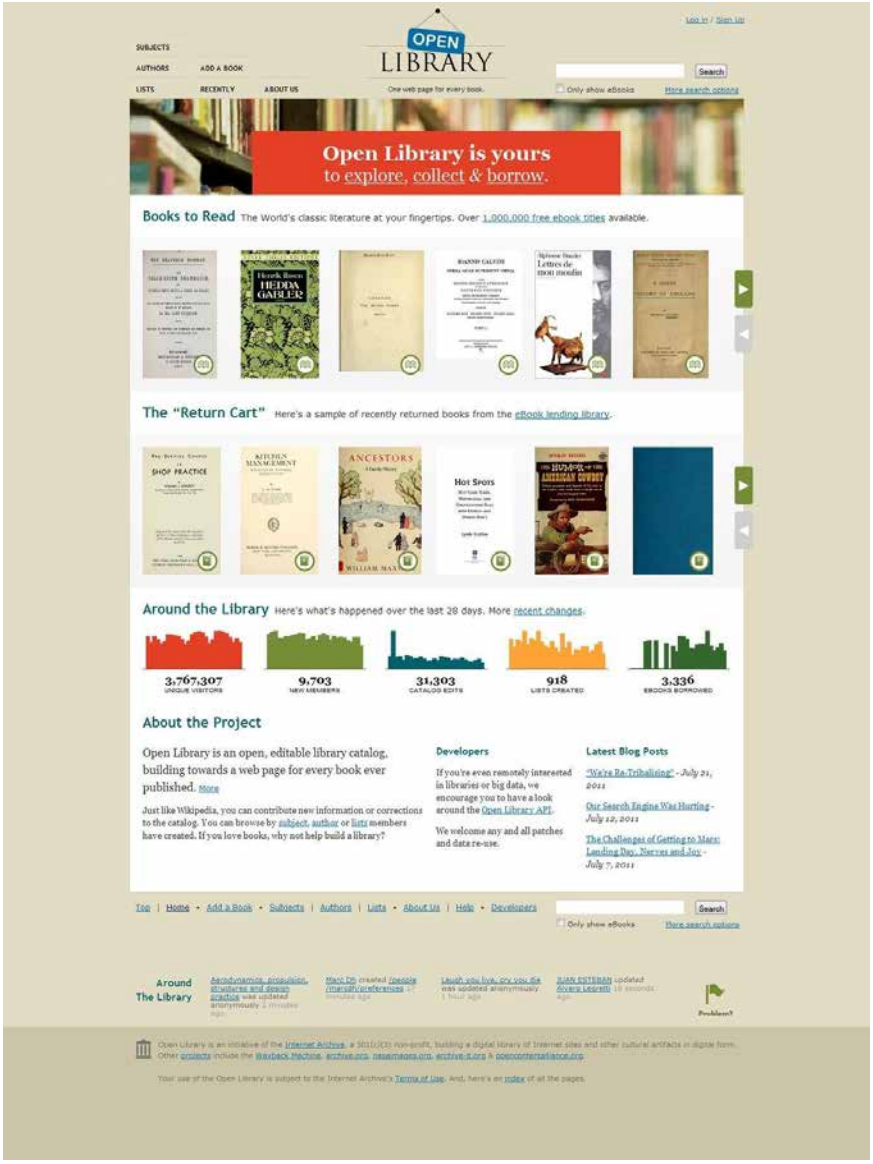
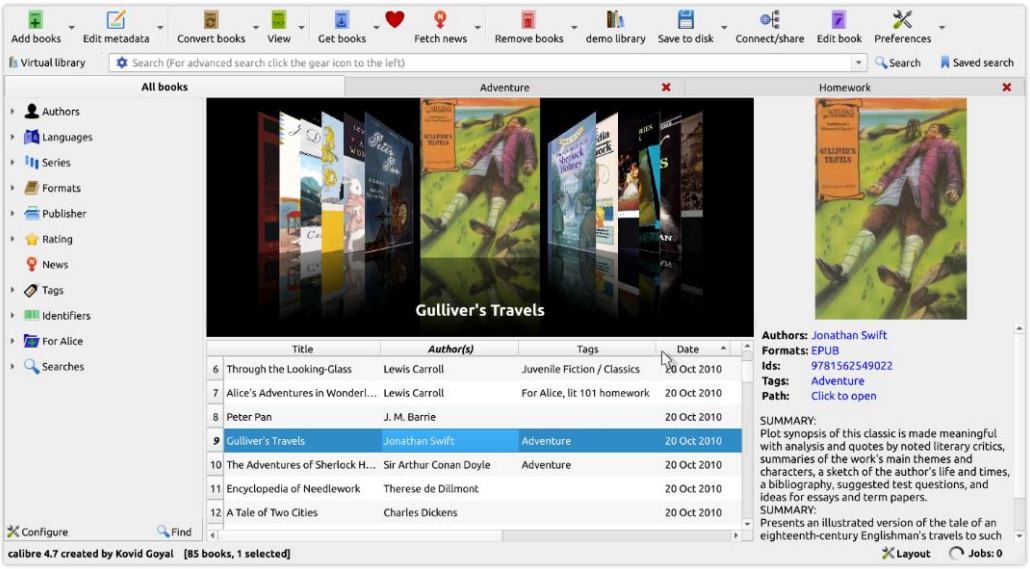


Рис.1.4 Інтерфейс Open Library

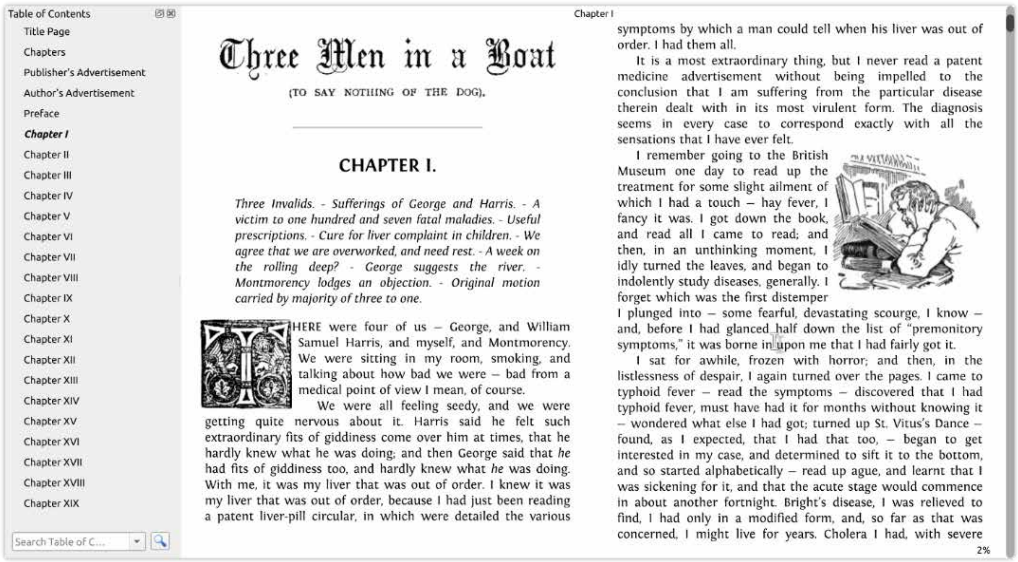
Calibre

Calibre — це потужний десктопний додаток для керування особистими електронними бібліотеками [5]. Він підтримує імпорт та експорт книг у численних форматах, включає вбудований рідер, інструменти для редагування метаданих, конвертацію між форматами, створення категорій та тегів. Calibre також дозволяє синхронізацію з електронними пристроями для читання та має можливість розгортання локального сервера бібліотеки. Недоліком є складний інтерфейс для звичайних користувачів і відсутність інструментів спільного користування бібліотекою в режимі онлайн.

а)



б)



в)

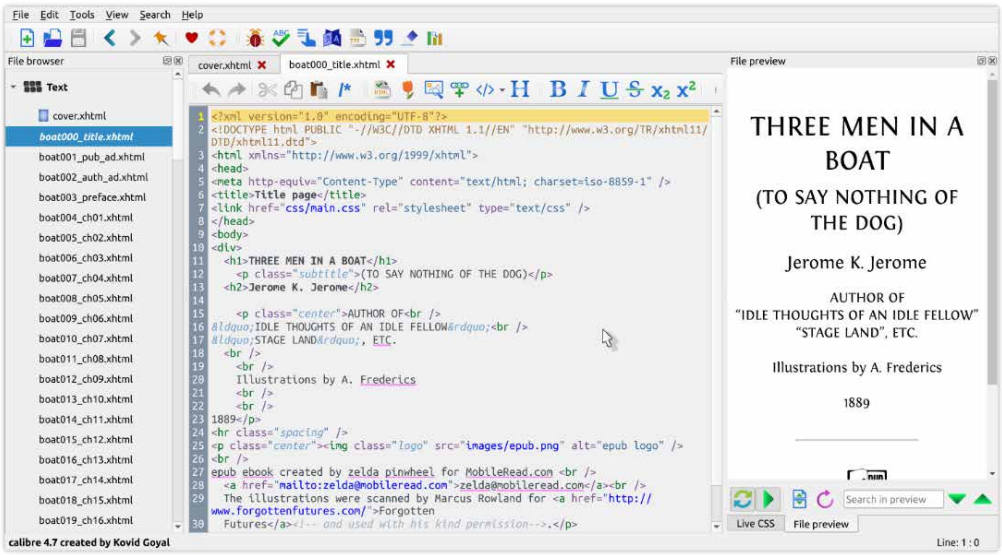
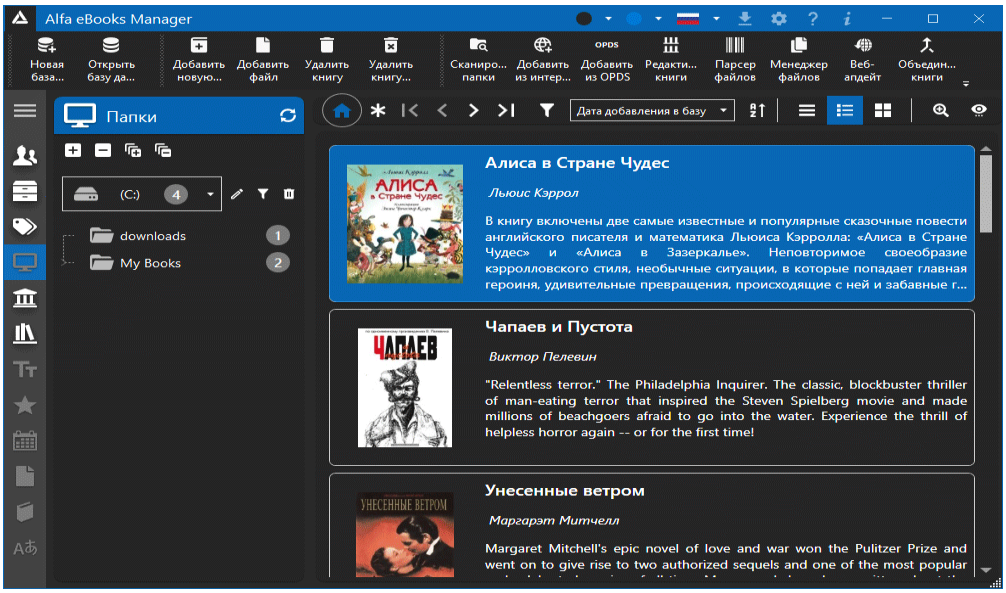


Рис.1.5 Інтерфейс Calibre

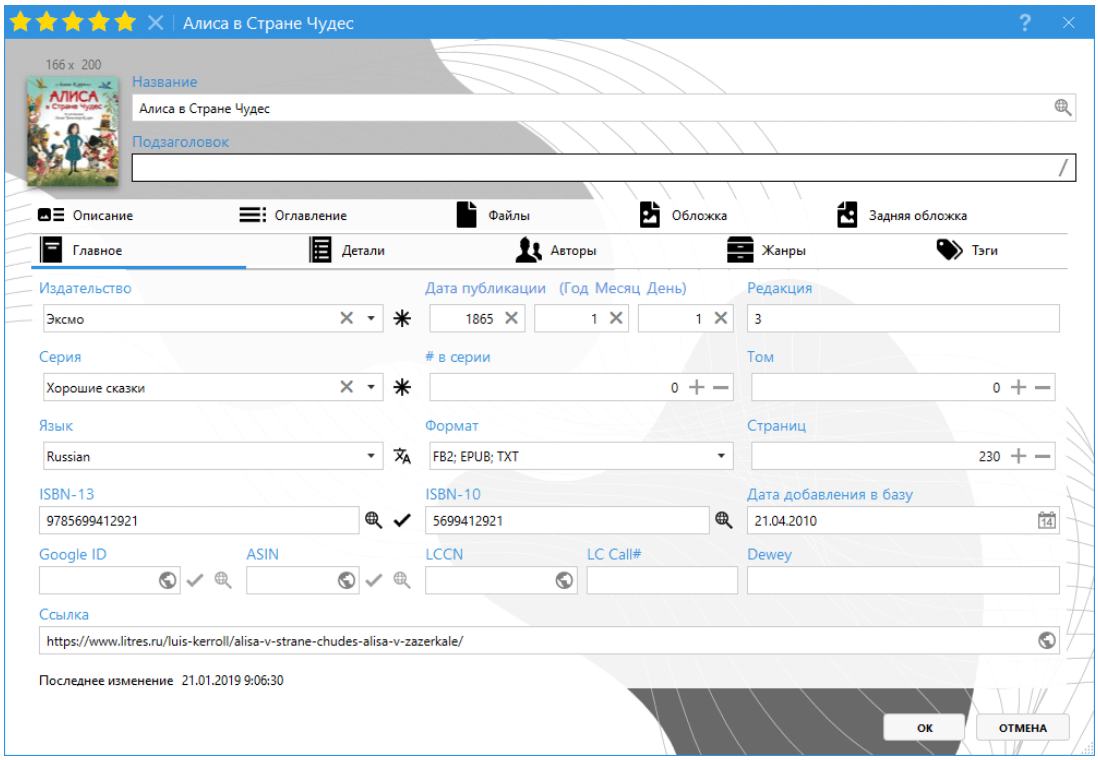
Alfa eBooks Manager

Alfa eBooks Manager — комерційне рішення для організації локальної бібліотеки електронних книг на ПК. Додаток дозволяє створювати структуру електронної колекції, імпортувати книги з різних джерел, призначати їм метаінформацію, встановлювати обкладинки та категорії [6]. Є можливість відкриття книг у сторонніх рідерах, ведення статистики читання, підтримка ISBN та інтеграція з онлайн-сервісами. Alfa eBooks Manager має зручний інтерфейс, але більшість функцій доступні лише у платній версії.

а)


Alfa eBooks Manager
Папки
Дата добавления в базу
Алиса в Стране Чудес
Льюис Кэррол
Чапаев и Пустота
Виктор Пелевин
Унесенные ветром
Маргарэт Митчелл

б)


Алиса в Стране Чудес
Название
Подзаголовок
Описание
Оглавление
Файлы
Обложка
Задняя обложка
Главное
Детали
Авторы
Жанры
Тэги
Издательство
Эксмо
Дата публикации (Год Месяц День)
Редакция
Серия
Хорошие сказки
в серии
Том
Язык
Russian
Формат
FB2; EPUB; TXT
Страниц
ISBN-13
9785699412921
ISBN-10
5699412921
Дата добавления в базу
21.04.2010
Google ID
ASIN
LCCN
LC Call#
Dewey
Ссылка
https://www.litres.ru/luis-kerroll/alisa-v-strane-chudes-alisa-v-zazerkale/
Последнее изменение 21.01.2019 9:06:30
OK
ОТМЕНА

в)

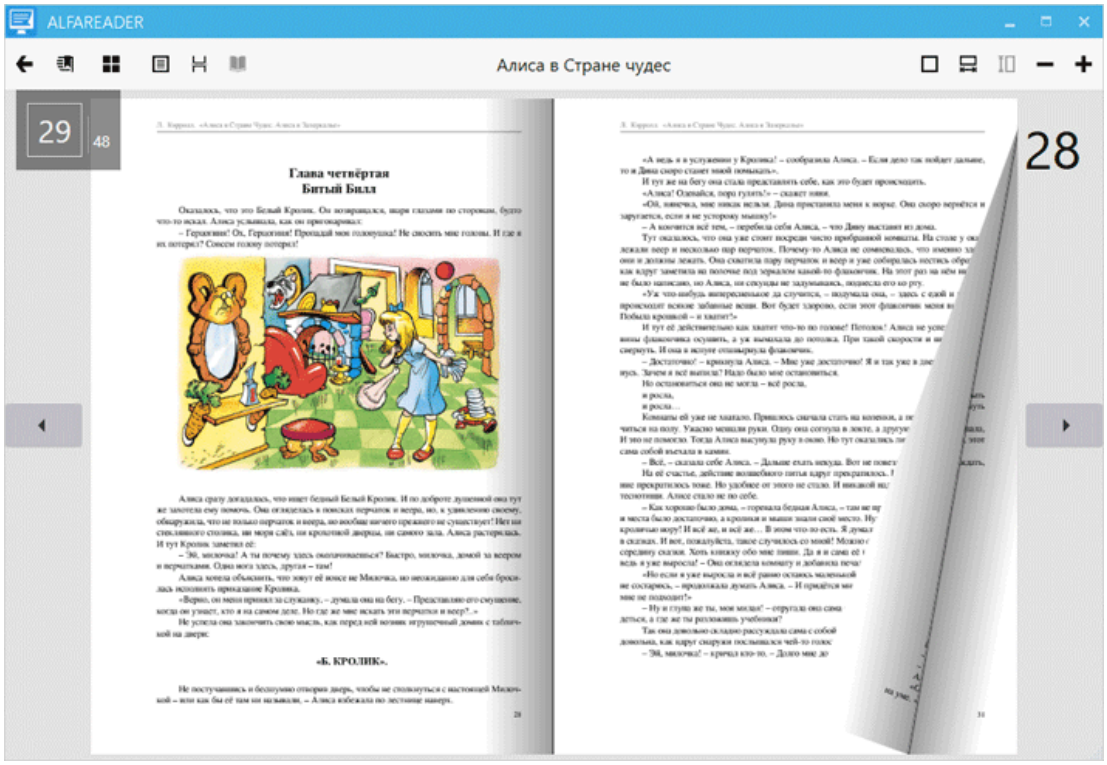
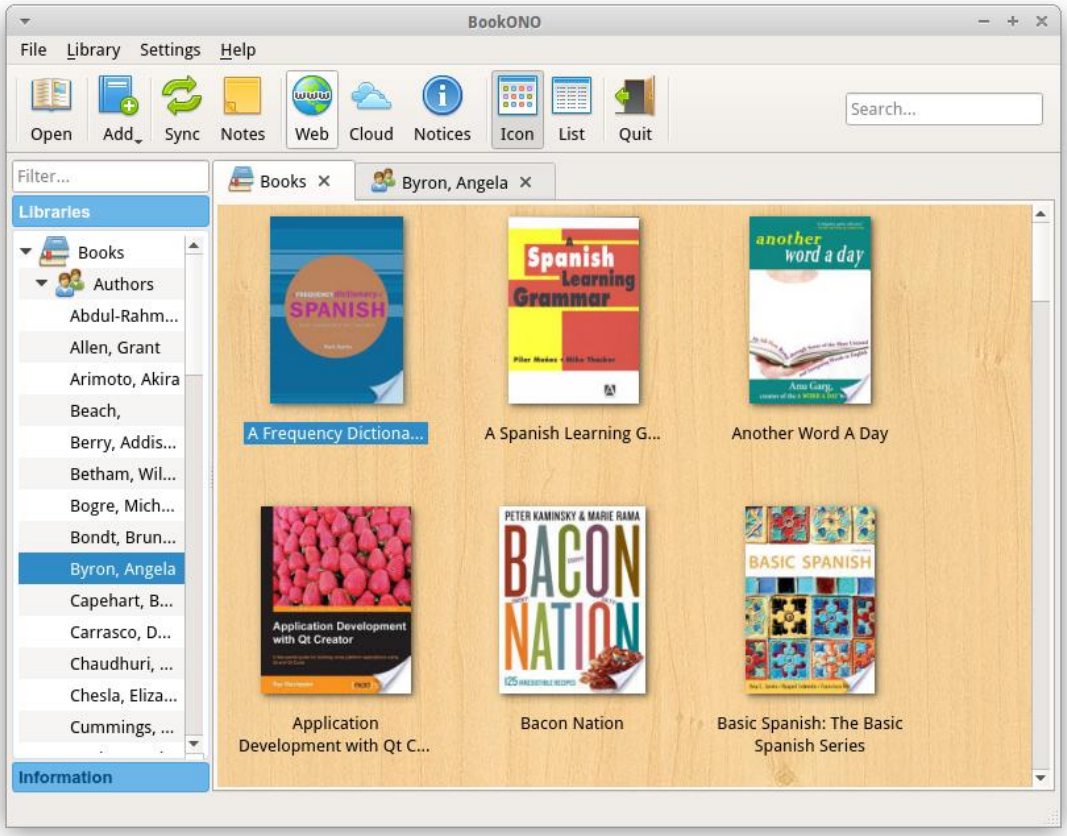


Рис.1.6 Інтерфейс Alfa eBooks Manager

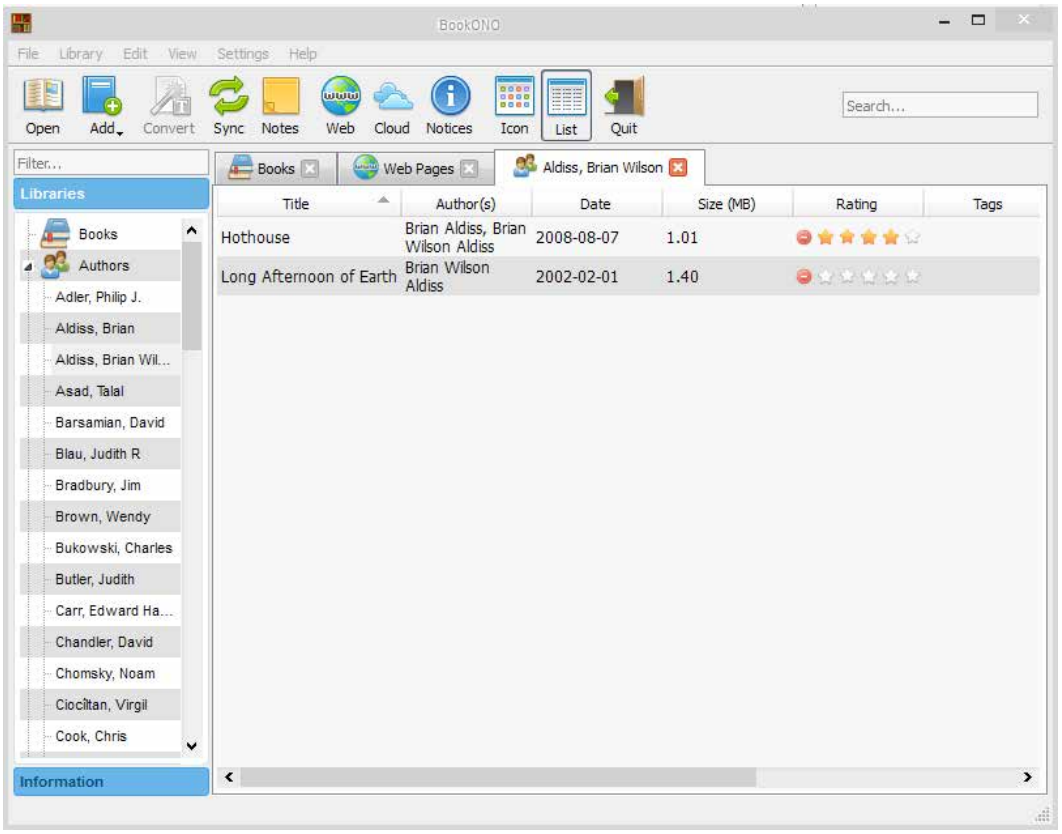
BookONO

BookONO — це сучасний застосунок для Windows та Android, призначений для читання, керування та каталогізації електронних книг [4]. Додаток підтримує створення власних списків, сортування книг за категоріями, додавання нотаток, оцінювання, а також синхронізацію з хмарними сервісами. Інтерфейс є інтуїтивно зрозумілим і орієнтованим на комфорт користувача. Особливістю BookONO є підтримка аудіокниг і сучасного візуального стилю, однак у безкоштовній версії функціональність обмежена.

а)



б)



в)

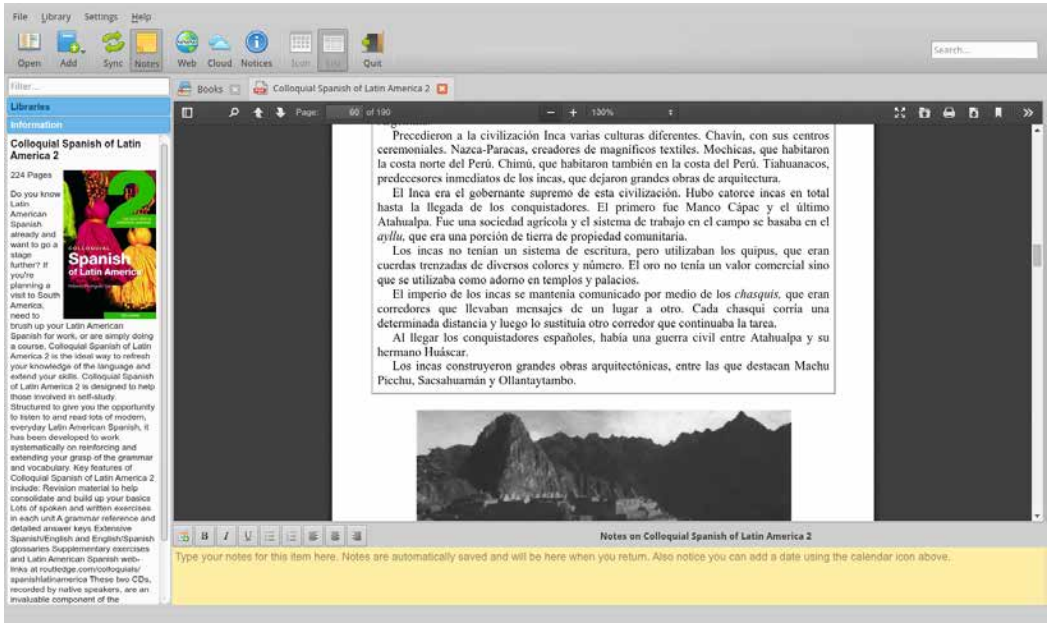


Рис.1.7 – Інтерфейс BookONO

Таблиця 1.1

Порівняння програм аналогів у вигляді таблиці

Критерій	Open Library	Calibre	Alfa eBooks Manager	BookONO
Тип	Веб-застосунок	Десктопний	Десктопний	Десктоп/мобільний
Доступ до книг	Онлайн/цифрова оренда	Імпорт локальних файлів	Імпорт локальних файлів	Імпорт локальних файлів
Каталогізація	Обмежена	Повна (теги, категорії)	Повна	Повна
Пошук/фільтрація	Так	Так	Так	Так
Читання книг	У веб-рідері	Вбудований рідер	Через сторонній рідер	Вбудований рідер
Управління профілем	Так	Ні	Ні	Так
Списки користувача	Так	Ні	Частково	Так
Платформа	Веб	Windows, Linux, MacOS	Windows	Windows, Android
Ліцензія	Безкоштовна	Безкоштовна (open-source)	Безкоштовна / платна	Безкоштовна / платна
Оцінки, рецензії	Так	Ні	Ні	Так
Синхронізація з хмарою	Ні	Частково	Ні	Так (обмежено)

1.5 Постановка завдання

Метою даної дипломної роботи є проєктування та розробка інформаційної системи OnLineLibrary – десктопного застосунку для управління електронною бібліотекою, що забезпечує зручну роботу з каталогом книг, можливість їх систематизації, перегляду, пошуку, керування списками, а також адміністрування бази даних книг і користувачів.

Система призначена для персонального або локального використання кінцевими користувачами (читачами) та адміністраторами бібліотеки. Очікується, що застосунок забезпечить автоматизацію основних операцій,

пов’язаних з управлінням книгами та взаємодією з користувачем, забезпечуючи високий рівень зручності, швидкодії та розширюваності.

Інформаційна система повинна реалізовувати наступні **функціональні** вимоги:

Для користувача:

- реєстрація нового облікового запису;
- авторизація в системі;
- редагування профілю користувача;
- перегляд каталогу книг із можливістю фільтрації;
- пошук книг за основними параметрами (назва, автор, жанр) ;
- перегляд детальної інформації про книгу;
- завантаження або відкриття файлу книги;
- робота з персональними списками: створення, редагування, видалення списків; додавання/вилучення книг.

Для адміністратора:

- додавання, редагування та видалення книг;
- керування каталогами: жанри, автори, формати;
- перегляд і керування користувачами.

Нефункціональні вимоги:

- архітектура: клієнт-серверна модель з підключенням до бази Supabase через REST API;
- платформа: Windows-додаток, реалізований за допомогою .NET (WinForms);
- інтерфейс: багатовкладковий, локалізований (українська мова), з підтримкою навігації та повідомлень про помилки;
- збереження даних: централізоване, у реляційній базі з підтримкою зовнішніх ключів (зв’язки між таблицями);
- безпека: перевірка введених даних, базова автентифікація користувачів, обмеження доступу за роллю;

- Масштабованість: модульна структура, що дозволяє надалі додати підтримку нових форматів, рекомендаційної системи тощо.

Розробка вважається завершеною, якщо виконано наступне:

- реалізовані всі заявлені функціональні модулі;
- програма коректно працює із базою Supabase;
- пройдено модульне та інтеграційне тестування;
- забезпечено базову документацію користувача;
- програма успішно встановлюється на Windows 10+.

Таким чином, результатом виконання даної роботи стане прототип інформаційної системи OnLineLibrary, що дозволить ефективно керувати електронною бібліотекою, забезпечуючи зручність, простоту використання та можливість подальшого розвитку функціоналу.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних

Логічна модель даних є основою для створення структури інформаційної бази та визначає взаємозв'язки між об'єктами предметної області. Вона розроблена з дотриманням принципів реляційної моделі та відповідає вимогам третьої нормальної форми (3NF), що гарантує відсутність надлишковості даних і забезпечує логічну цілісність.

Для побудови логічної моделі даних було використано *ER-модель (Entity-Relationship)*, яка описує сутності системи, їх атрибути та зв'язки між ними. Вона дозволяє чітко відобразити логічну структуру бази даних незалежно від обраної системи управління базами даних (СУБД).

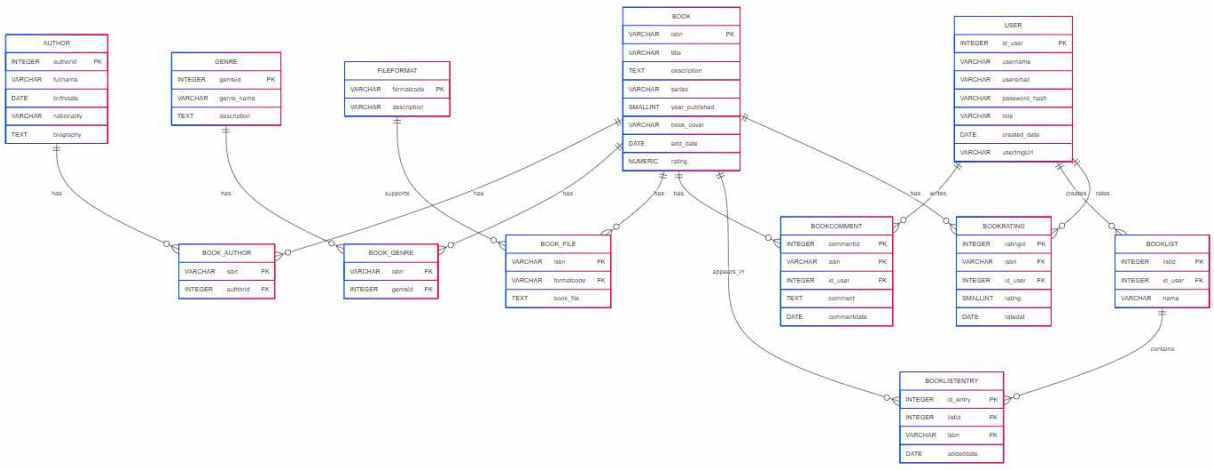


Рис.2.1 ER модель інформаційної бази

Основні сутності та їх атрибути:

- Author — зберігає інформацію про авторів: authorid (PK), fullname, birthdate, nationality, biography;
- Book — містить основні дані про книги: isbn (PK), title, description, series, year_published, book_cover, add_date, rating;
- User — зберігає дані зареєстрованих користувачів: id_user (PK), username, useremail, password_hash, role, created_date, userImgUrl;

- Book_Author — відображає зв’язок "багато до багатьох" між книгами та авторами: isbn (FK), authorid (FK);
- Genre — містить список жанрів: genreid (PK), genre_name, description;
- Book_Genre — зв’язок між книгами та жанрами: isbn (FK), genreid (FK);
- Book_File — зберігає файли книг і формат: isbn (FK), formatcode (FK), book_file;
- FileFormat — формати збереження файлів: formatcode (PK), description;
- BookComment — коментарі до книг: commentid (PK), isbn (FK), id_user (FK), comment, commentdate;
- BookRating — оцінки книг користувачами: ratingid (PK), isbn (FK), id_user (FK), rating, ratedat;
- BookList — списки книг, створені користувачами: listid (PK), id_user (FK), name;
- BookListEntry — елементи списків: id_entry (PK), listid (FK), isbn (FK), addeddate.

2.2 Вибір системи управління інформаційною базою

На цьому етапі проєктування системи було здійснено обґрунтований вибір системи управління базою даних (СУБД), яка є основою для зберігання, обробки та організації інформації. Для реалізації інформаційної бази програмного забезпечення було обрано СУБД *PostgreSQL* у зв’язці з платформою *Supabase*.

PostgreSQL — це потужна об’єктно-реляційна система управління базами даних з відкритим вихідним кодом, яка підтримує повний набір реляційних операторів, типів даних і механізмів забезпечення цілісності даних [8]. Основними перевагами PostgreSQL, що обумовили її вибір, є:

- висока відповідність стандартам SQL (SQL:2011) та підтримка складних транзакцій;
- підтримка складних типів даних, зокрема JSON, масивів, типів користувача;

- розширюваність: користувачі можуть створювати власні функції, оператори, типи та агрегати;
- надійність і стабільність, які доведено широким використанням у промислових рішеннях;
- можливість масштабування як вертикального (шляхом розширення потужності сервера), так і горизонтального (за допомогою розподілених рішень та реплікації).

Ця СУБД повністю відповідає вимогам до сучасних web-орієнтованих рішень, а також забезпечує високу продуктивність при роботі з великим обсягом даних, що є важливим для цифрових бібліотек та електронних книжкових платформ.

Supabase — це сучасна Backend-as-a-Service (BaaS) платформа, побудована поверх PostgreSQL, яка надає інструменти для швидкої розробки серверної частини застосунків [9]. У контексті даного програмного рішення Supabase використовується як хмарна оболонка над PostgreSQL для забезпечення:

- швидкого розгортання бази даних без необхідності ручної конфігурації серверів;
- автоматичного RESTful API для кожної таблиці бази даних, що значно пришвидшує розробку;
- механізмів автентифікації та авторизації користувачів із можливістю ролей і політик доступу;
- вбудованої реплікації, резервного копіювання і журналювання змін, що підвищує надійність системи;
- інтеграції з зовнішніми клієнтами, зокрема через WebSocket, GraphQL та інші інтерфейси.

Такий вибір дозволяє поєднати надійність, гнучкість і масштабованість PostgreSQL з простотою налаштування і використання Supabase як інфраструктурного рішення. Крім того, це відкриває широкі можливості для подальшого розвитку системи, включаючи підключення сторонніх сервісів,

використання функцій хмарного зберігання (наприклад, для обкладинок книг або самих файлів книг), а також побудову системи сповіщень, аналітики тощо.

Таким чином, обраний технологічний стек, який поєднує PostgreSQL як реляційну СУБД і Supabase як платформу хмарної інфраструктури, дозволяє реалізувати високопродуктивну, масштабовану та безпечну інформаційну базу, що відповідає як поточним, так і перспективним вимогам до системи управління електронною бібліотекою.

2.3 Створення інформаційної бази

На основі розробленої концептуальної моделі було реалізовано фізичну структуру інформаційної бази із використанням платформи Supabase, яка функціонує на основі СУБД PostgreSQL. Процес створення бази даних включав створення основних та допоміжних таблиць, визначення зв’язків між ними, встановлення обмежень цілісності, а також конфігурування первинних і зовнішніх ключів згідно з логікою предметної області.

У системі наявні такі ключові таблиці:

- User — зберігає інформацію про зареєстрованих користувачів, включаючи логін, email, роль, дату реєстрації та (опціонально) хеш пароля;
- Book — основна таблиця з інформацією про книги (ISBN, назва, опис, серія, рік публікації, обкладинка, рейтинг, дата додавання);
- Author — містить відомості про авторів (ПІБ, дата народження, національність, біографія);
- Genre — перелік жанрів з описами;
- FileFormat — допустимі формати файлів книг (наприклад, PDF, EPUB, FB2);
- Book_Author — проміжна таблиця для реалізації зв’язку багато-до-багатьох між книгами та авторами;
- Book_Genre — проміжна таблиця для зв’язку книг із жанрами;

- Book_File — зберігає файли книг певного формату, що пов’язуються з конкретною книгою;
- BookComment — коментарі користувачів до книг;
- BookRating — оцінки книг, виставлені користувачами;
- BookList — списки книг, створені користувачами (наприклад, «Улюблене», «До прочитання» тощо);
- BookListEntry — зв’язок книг із конкретними списками користувачів.

Повний SQL-скрипт створення таблиць наведено у Додатку А.

Для зручності візуалізації структури створеної бази даних було виконано підключення до Supabase через середовище pgAdmin, та побудовано діаграму з основними таблицями та зв’язками між ними (додаток Г).

У Supabase використовувався SQL Editor для виконання запитів зі створення таблиць, а також компоненти Storage — для зберігання файлів книг — і Authentication — для керування обліковими записами користувачів. Крім того, було налаштовано Row-Level Security для контролю доступу до даних залежно від ролі користувача.

Результатом є функціональна, масштабована інформаційна база, що повністю відображає структуру предметної області цифрової бібліотеки та слугує надійною основою для реалізації бізнес-логіки застосунку.

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна структура програмного забезпечення

Організаційна структура прикладного програмного забезпечення інформаційної системи управління онлайн-бібліотекою побудована за принципами багаторівневої архітектури, що забезпечує розділення відповідальностей між різними частинами системи. Це дозволяє спростити розробку, тестування, масштабування та супровід ПЗ.

Архітектура умовно поділена на три логічні рівні: Client, ApplicationLogic, Server.

Рівень **Client** (клієнтська частина) відповідає за відображення інтерфейсу користувача та забезпечує взаємодію з користувачем. Він включає такі пакети:

- UserInterface – реалізує графічний інтерфейс користувача. Відображає вікна, панелі, кнопки, поля вводу та інші візуальні елементи, з якими взаємодіє користувач;
- UserActions – містить логіку обробки дій користувача: натискання кнопок, введення даних, вибір із меню тощо. Ці дії спрямовуються на відповідні компоненти логіки застосунку або до серверної частини;
- GUIComponents – набір багаторазових графічних компонентів, таких як таблиці, спливаючі вікна, модальні діалоги тощо. Забезпечує ефективне створення інтерфейсу за допомогою готових елементів.

Рівень **ApplicationLogic** (логіка додатку) - це центральна частина системи, яка обробляє бізнес-логіку та керує поведінкою застосунку відповідно до дій користувача:

- Authentication – відповідає за автентифікацію та авторизацію користувачів, включає механізми входу, реєстрації, зберігання токенів та перевірку доступу;

- ProfileManagement – керує персональними даними користувача: ім’ям, email, роллю, фото профілю. Забезпечує функції редагування профілю та збереження змін;
- BookManagement – реалізує операції з книгами: додавання, редагування, перегляд і видалення. Включає перевірку правильності введених даних та взаємодію з каталогом;
- CatalogManagement – відповідає за структуру каталогів, класифікацію книг за жанрами, форматами, категоріями. Забезпечує навігацію та організований доступ до всіх доступних книг;
- SearchAndFilter – реалізує пошук книг за різними параметрами (назва, автор, жанр), а також механізми фільтрації, сортування і повнотекстового пошуку;
- PersonalLists – дає змогу користувачеві створювати особисті добірки: обране, прочитане, «на потім». Дані синхронізуються із сервером;
- AdminControl – надає адміністративні можливості, зокрема модерацію контенту, перегляд логів, управління користувачами та їх ролями. Доступне лише адміністраторам.

Рівень **Server** (серверна частина) відповідає за обробку запитів клієнта, реалізацію бізнес-логіки та збереження даних у базі:

- ServerAPI – набір API-контролерів, які обробляють HTTP-запити клієнта. Визначає маршрути, методи (GET, POST, PUT, DELETE), формує відповіді та статуси;
- BusinessLogic – вміщує основні сервіси, які реалізують бізнес-правила, перевірки, обчислення, обробку вхідних запитів і доступу;
- DatabaseAccess – шар взаємодії з базою даних Supabase (на базі PostgreSQL). Виконує операції створення, читання, оновлення та видалення (CRUD), формує SQL-запити або використовує ORM;
- DataModels – описує структури даних системи (моделі сутностей): книга, користувач, жанр, список. Визначає типи даних, валідаційні обмеження та зв’язки між сутностями.

Структурна схема взаємодії між цими компонентами наведена на рис.3.1.

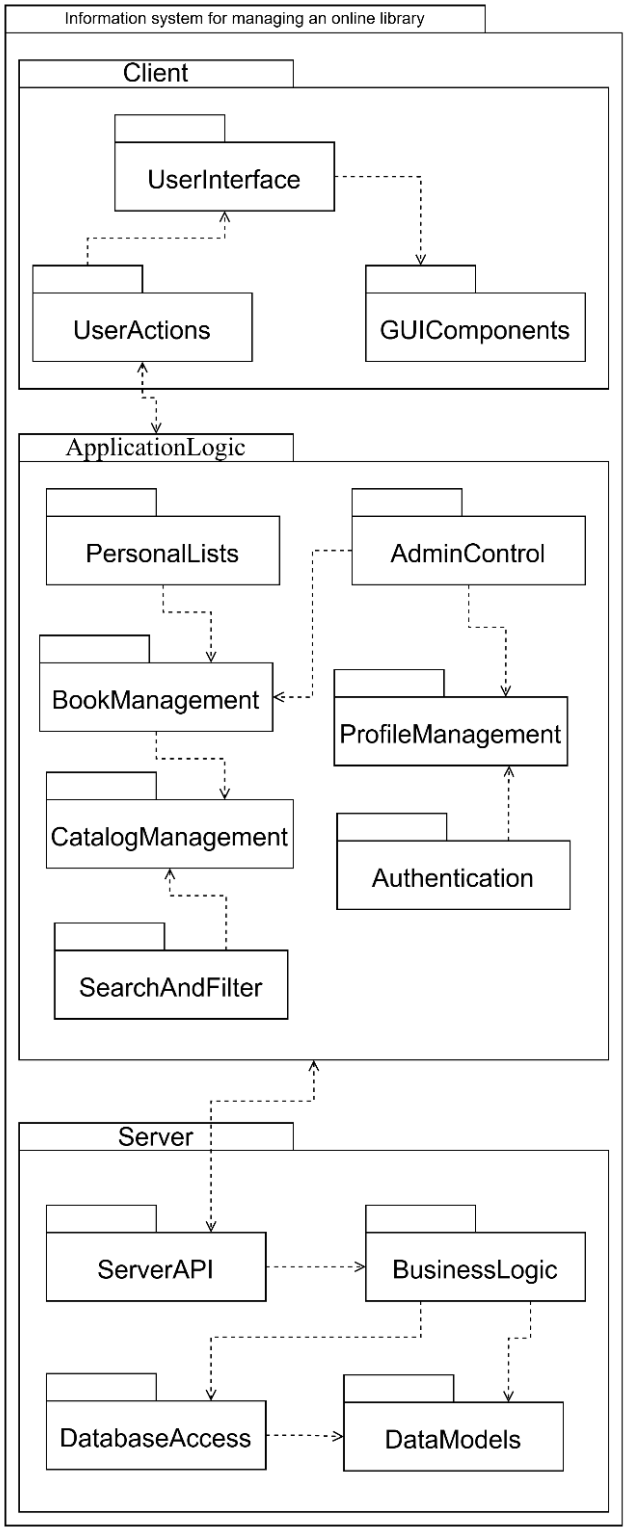


Рис.3.1 Діаграма пакетів інформаційної системи

Такий поділ дозволяє ефективно масштабувати систему, забезпечує зрозумілу структуру, спрощує тестування окремих модулів та сприяє гнучкому розвитку функціоналу системи в майбутньому.

3.2 Вибір інструментарію для створення ППЗ

У процесі розробки інформаційної системи управління онлайн-бібліотекою було прийнято рішення використовувати перевірений, надійний та гнучкий набір інструментів, який дозволяє ефективно реалізувати функціональність, зручний графічний інтерфейс та забезпечити надійну взаємодію з базою даних. Нижче наведено обґрунтування вибору основного інструментарію, що використовувався на всіх етапах створення програмного забезпечення.

Мова програмування: C#. Мова C# є основною мовою розробки для даного проєкту. Вона надає потужні об’єктно-орієнтовані засоби, підтримує сучасні парадигми програмування (включно з асинхронністю, LINQ-запитами, лямбда-виразами тощо), має багатий стандартний набір бібліотек, а також ідеально підходить для створення десктопних додатків у середовищі Windows. Крім того, C# повністю інтегрується з .NET-екосистемою та має активну спільноту підтримки.

Середовище розробки: Visual Studio. В якості середовища розробки було обрано Microsoft Visual Studio — один із найпотужніших IDE для C# та .NET-платформи. Visual Studio забезпечує:

- вбудований дизайнер форм, який значно пришвидшує розробку інтерфейсу;
- підтримку налагодження, підказок IntelliSense, підключення баз даних та засобів контролю версій;
- зручну інтеграцію з пакетами NuGet для швидкого підключення додаткових бібліотек.

Технологія створення інтерфейсу: Windows Forms. Для розробки графічного інтерфейсу користувача (GUI) було обрано технологію Windows

Forms (WinForms). Це одна з найдавніших, але водночас стабільних і перевірених технологій створення десктопних додатків під Windows.

Основними перевагами Windows Forms у контексті даного проєкту є:

- Простота та швидкість розробки.

  Завдяки візуальному дизайнеру форм у Visual Studio, розробник може швидко створити прототипи і функціональні вікна без необхідності писати значну кількість коду вручну;
- Широкий набір вбудованих елементів управління.

  Стандартні компоненти (наприклад, Button, Label, TextBox, DataGridView, ComboBox, TabControl, PictureBox тощо) охоплюють практично всі потреби інтерфейсу сучасного десктопного додатку;
- Інтеграція з .NET Framework.

  Це забезпечує простий доступ до бібліотек для роботи з базами даних, обробки файлів, логіки авторизації тощо;
- Подійна модель програмування.

  Обробка взаємодій користувача з інтерфейсом (наприклад, натискання кнопки або вибір елемента зі списку) реалізується через зручні події, що підвищує читаність і структурованість коду;
- Висока сумісність з операційною системою Windows.

  Оскільки кінцеві користувачі працюють у середовищі Windows, Windows Forms забезпечує стабільність та передбачуваність поведінки додатку.

Обмеження Windows Forms, які були враховані:

- недостатня підтримка сучасних UI/UX практик (матеріальний дизайн, анімації);
- ускладнене масштабування під змінні розміри екранів або DPI;
- менші можливості кастомізації у порівнянні з WPF або сучасними веб-фреймворками.

Попри це, у межах поставленої задачі — створення класичного бізнес-додатку з чіткою структурою та зрозумілим інтерфейсом — Windows Forms є цілком обґрунтованим вибором.

Додатково були застосовані такі бібліотеки та зовнішні сервіси:

- **Supabase** — використовується як хмарна платформа для зберігання й обробки даних (автентифікація, база даних, завантаження файлів). Вона забезпечує RESTful API, зручну інтеграцію з .NET через запити HTTP, підтримку SQL і управління сховищем файлів (наприклад, зображень книг або самих електронних книг);
- **Json.NET (Newtonsoft.Json)** — застосовується для серіалізації/десеріалізації даних у форматі JSON при обміні інформацією з Supabase;
- **System.Net.Http** — стандартна бібліотека для виконання HTTP-запитів до зовнішніх сервісів, зокрема, для роботи з API Supabase;
- **System.Text.RegularExpressions** — для валідації полів введення (наприклад, перевірка ISBN чи формату електронної пошти).

Для зберігання інформації використовується **PostgreSQL**, яка є основною СУБД у Supabase. Вона має високу продуктивність, підтримує транзакції, зовнішні ключі та інші реляційні механізми, що забезпечують цілісність даних.

Таким чином, поєднання C#, Windows Forms, Visual Studio, Supabase та супутніх бібліотек дало змогу реалізувати повнофункціональний, зручний і надійний програмний продукт для управління онлайн-бібліотекою. Вибір цього інструментарію був зумовлений оптимальним балансом між простотою розробки, функціональністю, продуктивністю та підтримкою сучасних вимог до десктопних систем.

3.3 Алгоритмізація та програмування програмних модулів

У процесі розробки програмного забезпечення «OnLineLibrary» — системи управління онлайн-бібліотекою — було застосовано модульний підхід до проєктування та програмування. Такий підхід передбачає поділ застосунку на ізольовані, взаємопов’язані програмні модулі з чіткими зонами відповідальності, що забезпечує гнучкість, зручність супроводу, тестованість і можливість масштабування системи.

Архітектурна модель

Система побудована за класичною трирівневою клієнт-серверною архітектурною моделлю, яка включає:

1. **Рівень представлення (UI)** — реалізований як Windows Forms-додаток. Забезпечує користувацький ввід, навігацію, перегляд і керування даними бібліотеки.
2. **Рівень логіки доступу до даних** — окремий програмний модуль SupabaseService, що інкапсулює логіку взаємодії з хмарною базою даних Supabase через REST API.
3. **Модельний рівень** — набір класів, що описують структуру основних сутностей предметної області (книги, автори, жанри, користувачі, формати тощо) та забезпечують типізовану обробку відповідей від серверної частини.

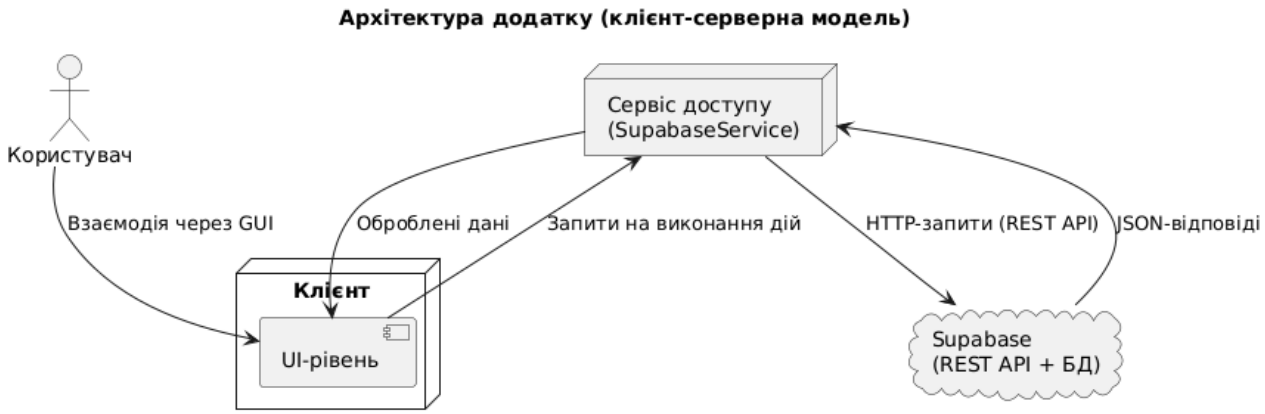


Рис.3.2 Архітектурна структура програми (Client–Server модель)

Розробка інтерфейсу користувача

Інтерфейс користувача (англ. User Interface, UI, дружній інтерфейс) — засіб зручної взаємодії користувача з інформаційною системою [10]. Він являє собою сукупність елементів для введення, обробки та відображення інформації, оптимально пристосованих для потреб і комфорту користувача.

У цьому підпункті розглядається реалізація інтерфейсу для основних форм розробленого додатку. Інтерфейс було створено з використанням технології WinForms, що дозволяє забезпечити зручну та інтуїтивно зрозумілу взаємодію користувача з системою. Розробка UI базувалась на принципах простоти, зручності та наочності. Навігація здійснюється через меню та кнопки з чіткими позначеннями.

Для прикладу, нижче наведено дизайн деяких вікон системи.

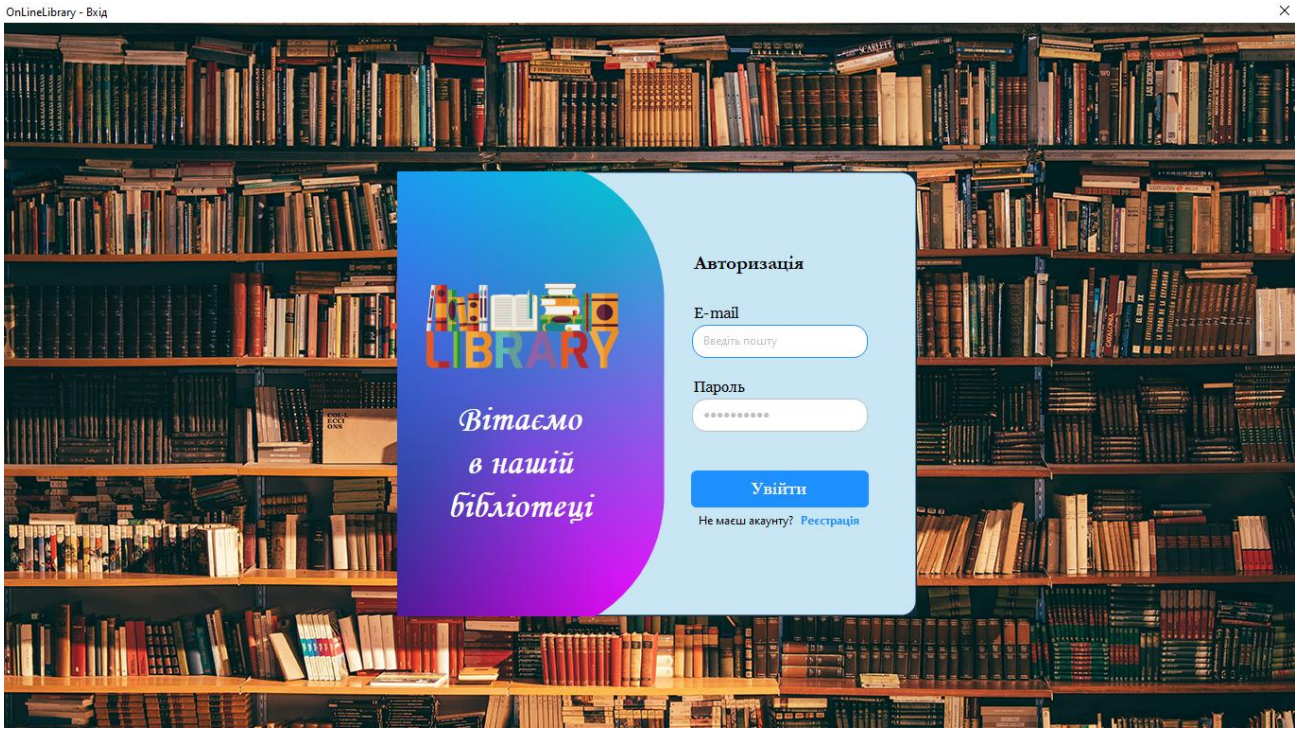


Рис.3.3 Вікно авторизації в системі

Рис.3.4 Вікно реєстрації

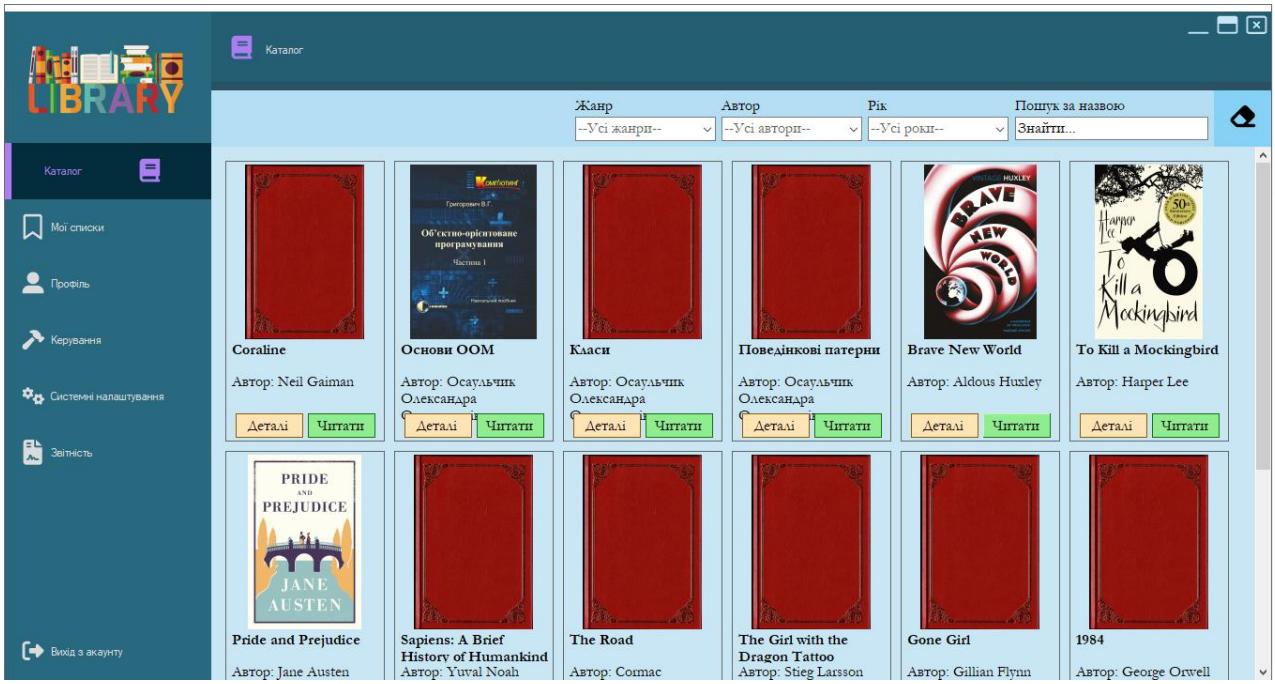


Рис.3.5 Вікно каталогу книг

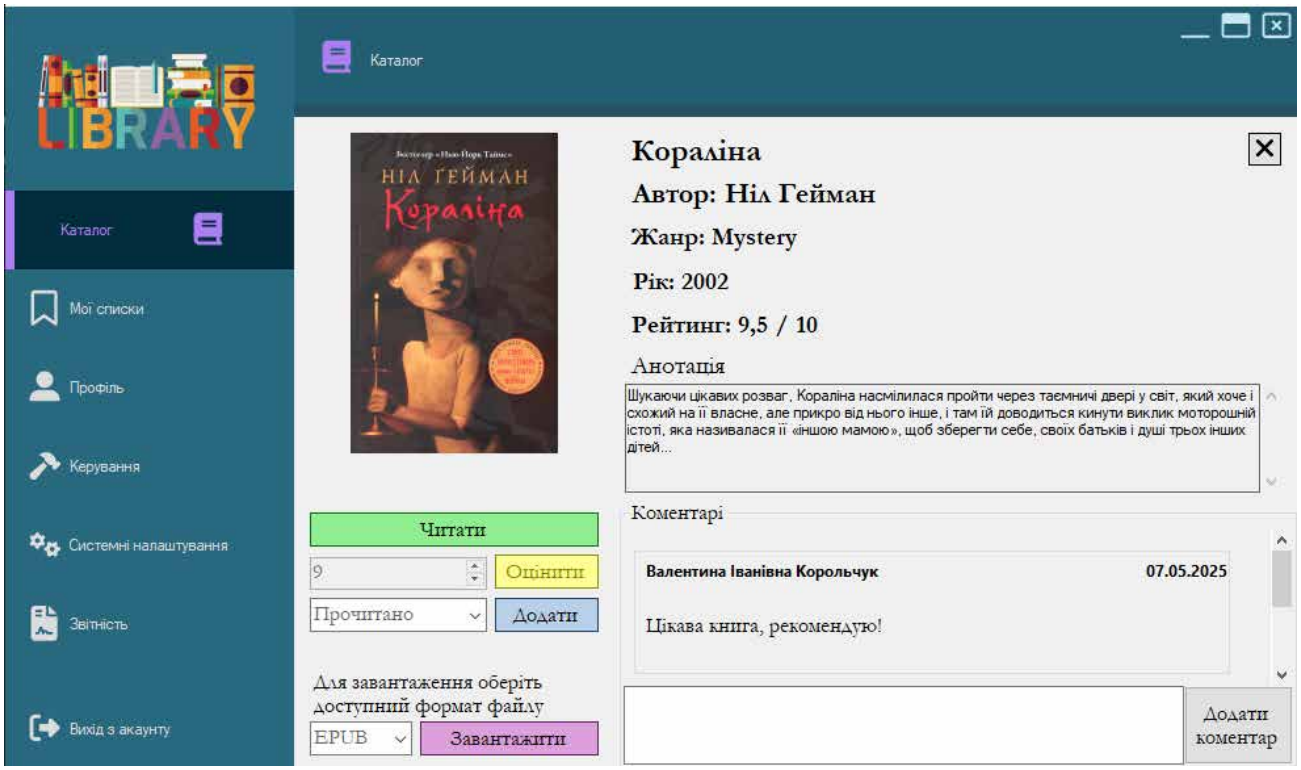


Рис.3.6 Вікно деталей книги

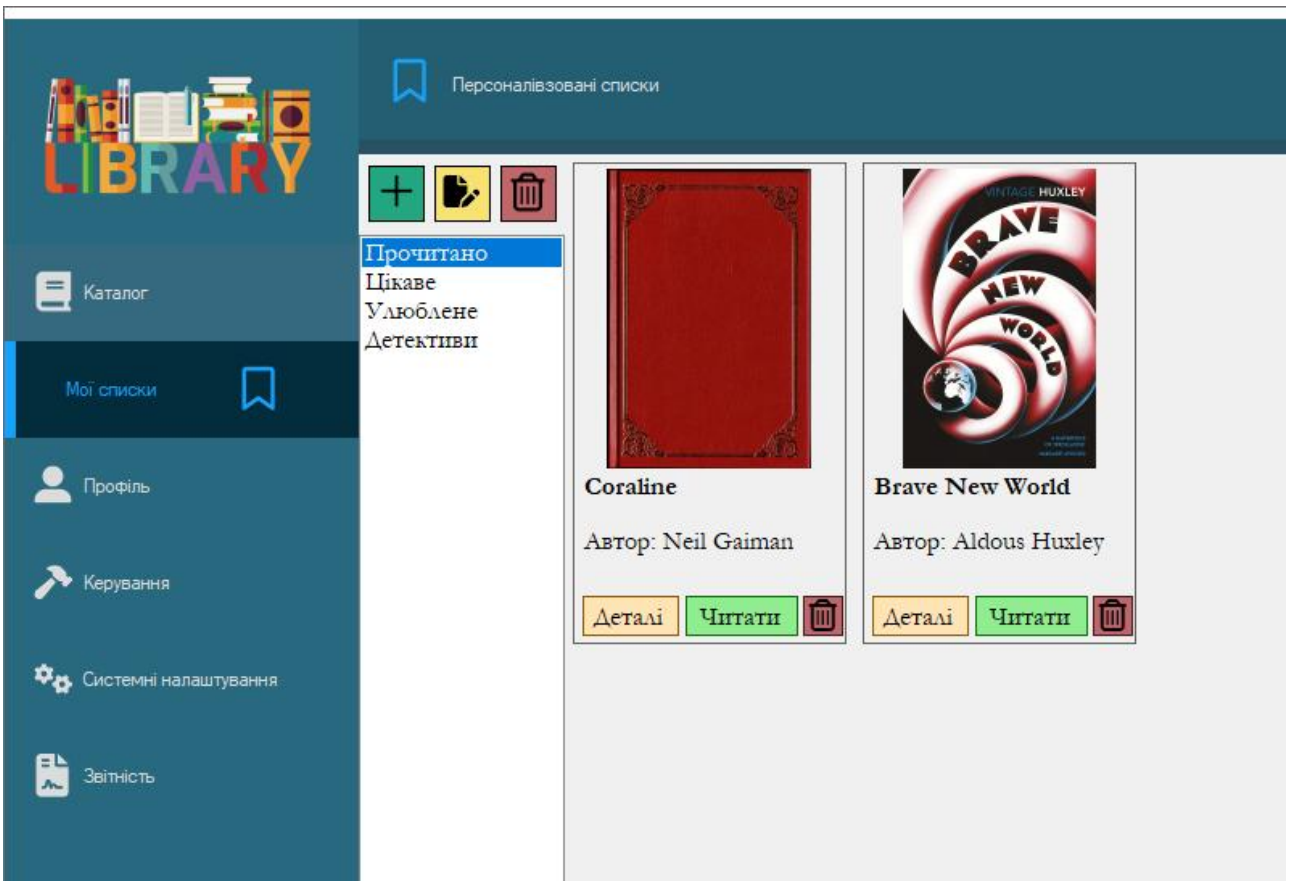


Рис.3.7 Вікно користувацьких списків

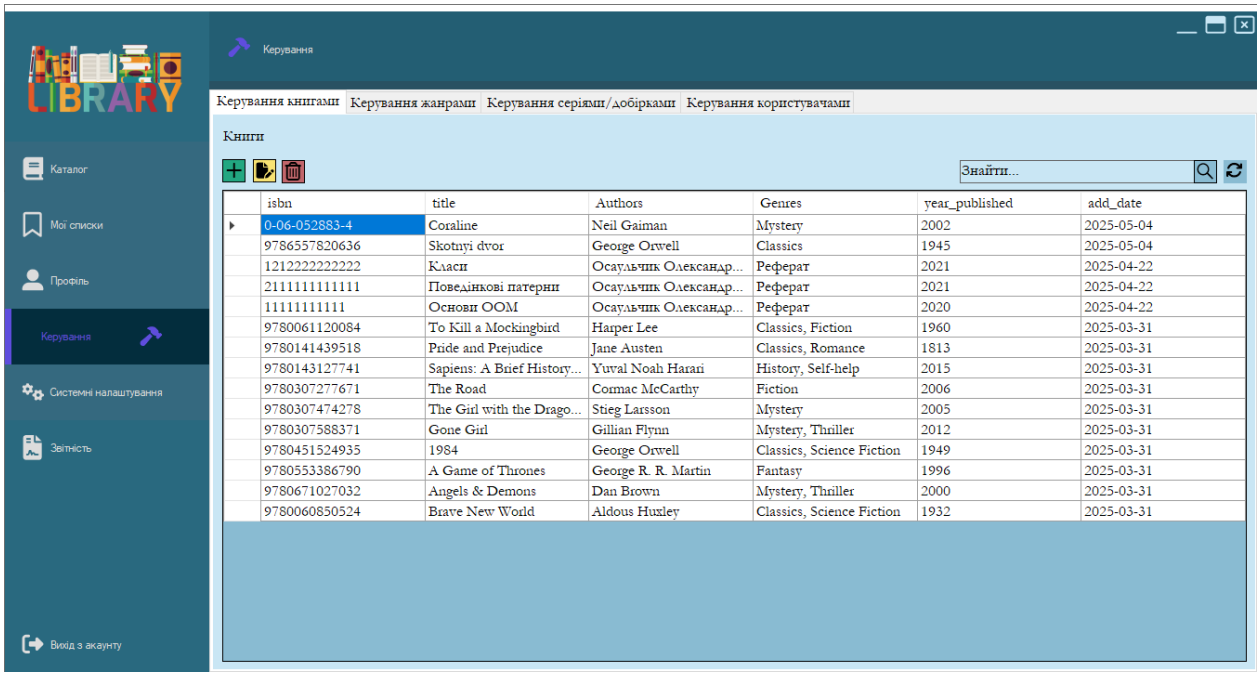


Рис.3.8 Вікно керування бібліотекою

Організація підключення до бази даних

Для взаємодії з хмарною базою даних, розгорнутою на платформі Supabase, реалізовано спеціалізований сервісний клас SupabaseService, у якому безпосередньо конфігурується HTTP-клієнт. На рис.3.9 наведено конструктор класу.

```
private readonly string baseUrl =
"https://muonuwsnudasdlvviuqe.supabase.co";
private readonly string apiKey = "...";

private readonly HttpClient _client;

public SupabaseService()
{
    _client = new HttpClient
    {
        BaseAddress = new Uri($"{baseUrl}/rest/v1/")
    };
    _client.DefaultRequestHeaders.Add("apikey", apiKey);
    _client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", apiKey);
}
```

Рис.3.9 Конструктор SupabaseService

Такий підхід є логічно обґрунтованим і технічно доцільним з кількох причин:

- Цілісність та локалізація логіки. Вся конфігурація HTTP-клієнта зосереджена безпосередньо в сервісному класі, де реалізовано запити до API. Це дозволяє підтримувати високий рівень інкапсуляції й уникнути надмірної фрагментації коду.
- Уніфікація запитів. Усі запити до API Supabase використовують одні й ті самі заголовки, токен авторизації та базову адресу. Це дозволяє уникнути дублювання коду й зменшує імовірність помилок при конфігурації клієнта.
- Масштабність проєкту. Враховуючи поточний обсяг і складність проєкту, винесення конфігурації підключення в окремий клас не додає суттєвих переваг, тоді як інтеграція в SupabaseService забезпечує достатній рівень керованості та зручності підтримки.
- Підвищення тестованості. Завдяки чіткій структурі, клас легко ізолюється для модульного тестування, а також може бути розширений або модифікований без порушення архітектурної цілісності застосунку.

Таким чином, обраний варіант реалізації є оптимальним для даного контексту, забезпечує гнучкість, підтримуваність та відповідність принципам розробки сучасного програмного забезпечення.

Клас SupabaseService: центральна логіка доступу до даних

Клас SupabaseService, більш детальний лістинг якого наведене в додатку Б, є центральним компонентом взаємодії клієнтської частини (інтерфейсу користувача) з базою даних. Його основна мета — забезпечити повноцінну, структуровану та зручну роботу з усіма даними онлайн-бібліотеки.

Він реалізує ключові сценарії, які охоплюють повний цикл CRUD-операцій та інші важливі функції системи, зокрема:

- реєстрація, автентифікація та авторизація користувачів, включаючи перевірку унікальності email та хешування паролів (BCrypt);

- додавання, редагування та видалення сутностей, зокрема книг, авторів, жанрів, користувачів, форматів файлів;
- виконання складених транзакцій, як-от додавання книги з прив’язками до авторів, жанрів, формату та файлу;
- завантаження файлів у Supabase Storage, включаючи електронні книги у різних форматах (PDF, EPUB, тощо) та зображення обкладинок;
- пошук та фільтрацію даних за критеріями (жанр, автор, рік, ключові слова);
- формування звітів, зокрема отримання агрегованої інформації про кількість завантажень, популярність авторів/жанрів тощо;
- реалізацію захисту доступу до окремих функцій системи залежно від ролі користувача (адміністратор / звичайний користувач).

Частину реалізації взаємодії з базою даних ми розглянемо на прикладі складеної операції додавання книги до системи (рис.3.10).

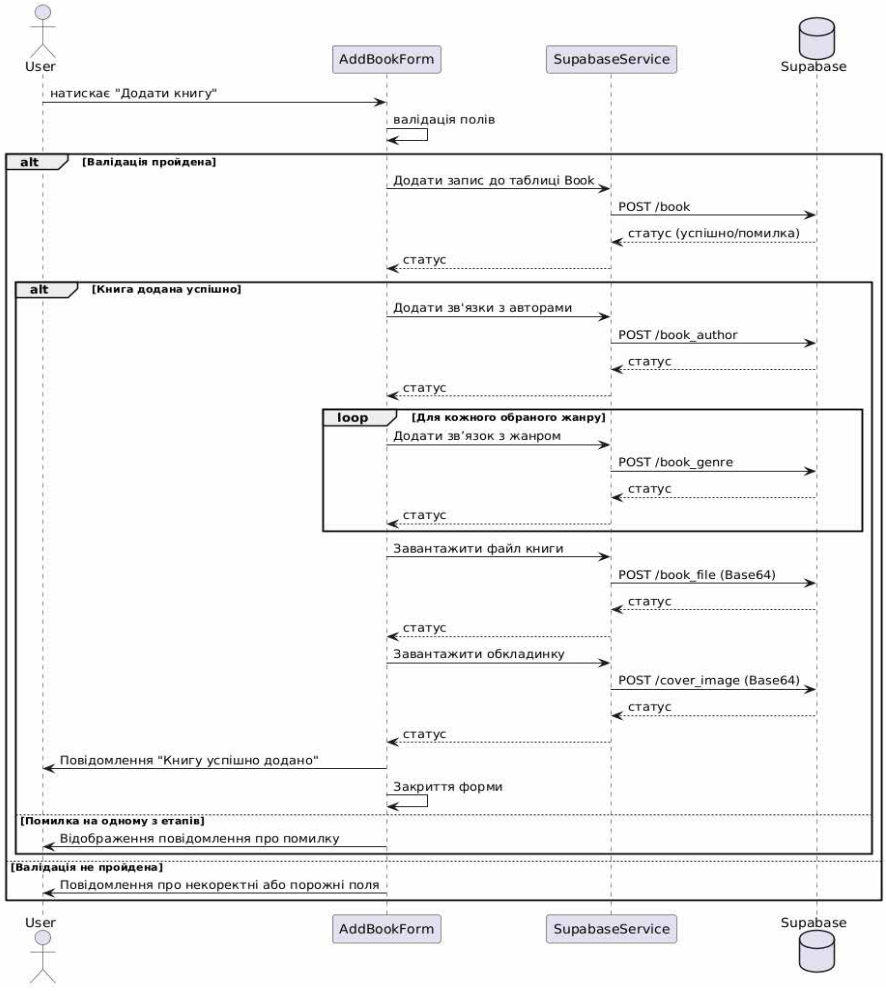


Рис.3.10 Алгоритм додавання книги

На рис.3.10 відображено поетапну логіку взаємодії з базою під час додавання нової книги. Процес ініціюється після натискання кнопки «Додати книгу» у відповідній формі, й далі виконується така послідовність:

1. Перевірка валідності полів введення;
2. Вставка нового запису до таблиці book;
3. Додавання зв’язків із таблицями book_author і book_genre;
4. Завантаження обкладинки до Supabase Storage, зберігання посилання в полі cover_url таблиці book;
5. Завантаження електронного файлу книги до Supabase Storage, зберігання у таблиці book_file.

У разі помилки на будь-якому з кроків відображається відповідне повідомлення, і виконання послідовності переривається.

Лістинг методів із SupabaseService, що слугують для додавання нової книги до бібліотеки:

1. **Метод InsertBookAsync** — додавання нової книги до таблиці book. Цей метод, зображений на рис.3.11, додає основну інформацію про книгу (ISBN, назву, опис, рік) до таблиці book, а також зберігає дату додавання.

```
public async Task<bool> InsertBookAsync(string isbn, string title, int
yearPublished, string description, string coverUrl)
{
    var book = new
    {
        isbn = isbn,
        title = title,
        year_published = yearPublished,
        description = description,
        cover_url = coverUrl,
        add_date = DateTime.UtcNow
    };

    var content = new StringContent(JsonConvert.SerializeObject(book),
Encoding.UTF8, "application/json");
    var response = await _client.PostAsync("book", content);
    return response.IsSuccessStatusCode;
}
```

Рис.3.11 Метод InsertBookAsync

2. **Метод InsertBookAuthorAsync —** прив’язка книги до автора. Цей метод створює запис у зв’язувальній таблиці book_author. Лістинг методу наведено на рис.3.12.

```
public async Task<bool> InsertBookAuthorAsync(string isbn, int authorId)
{
    var relation = new
    {
        isbn = isbn,
        authorid = authorId
    };

    var content = new StringContent(JsonConvert.SerializeObject(relation),
Encoding.UTF8, "application/json");
    var response = await _client.PostAsync("book_author", content);
    return response.IsSuccessStatusCode;
}
```

Рис.3.12 Метод InsertBookAuthorAsync

3. **Метод InsertBookGenreAsync,** що зображений на рис.3.13, відповідає за прив’язка книги до жанру. Може викликатися кілька разів — для кожного вибраного жанру окремо.

```
public async Task<bool> InsertBookGenreAsync(string isbn, int genreId)
{
    var relation = new
    {
        isbn = isbn,
        genreid = genreId
    };

    var content = new StringContent(JsonConvert.SerializeObject(relation),
Encoding.UTF8, "application/json");
    var response = await _client.PostAsync("book_genre", content);
    return response.IsSuccessStatusCode;
}
```

Рис.3.13 Метод InsertBookGenreAsync

4. **Метод UploadCoverImageAsync** — завантаження обкладинки до Supabase Storage. Цей метод завантажує зображення обкладинки в Supabase Storage та повертає пряме посилання. На зображенні 3.14 наведено лістинг даного методу.

```
public async Task<string?> UploadCoverImageAsync(string filePath, string
isbn)
{
    var bytes = File.ReadAllBytes(filePath);
    var base64 = Convert.ToBase64String(bytes);

    var content = new StringContent(base64, Encoding.UTF8,
"application/octet-stream");
    var fileName = $"covers/{isbn}_{Path.GetFileName(filePath)}";

    var response = await _client.PostAsync($"storage/v1/object/{fileName}",
content);

    return response.IsSuccessStatusCode
        ? $"https://your-supabase-url.storage.supabase.io/{fileName}"
        : null;
}
```

Рис.3.14 Метод UploadCoverImageAsync

5. **Метод InsertBookFileAsync,** що наведений на рис.3.15, відповідає за збереження електронного файлу книги. Додає файл книги до Storage і створює запис у таблиці book_file.

```
public async Task<bool> InsertBookFileAsync(string isbn, string formatCod
byte[] fileBytes)
{
    var base64File = Convert.ToBase64String(fileBytes);

    var fileEntry = new
    {
        isbn = isbn,
        formatcode = formatCode,
        file = base64File
    };

    var content = new StringContent(JsonConvert.SerializeObject(fileEntry),
Encoding.UTF8, "application/json");
    var response = await _client.PostAsync("book_file", content);
    return response.IsSuccessStatusCode;
}
```

Рис.3.15 Метод InsertBookFileAsync

В цілому, клас SupabaseService дозволяє централізовано керувати всіма сценаріями доступу до даних, підтримуючи як прості запити, так і багатокрокові транзакції з використанням Supabase REST API. Це забезпечує модульність, повторне використання коду, зручність тестування та масштабування системи в майбутньому. Окремі методи, як-от завантаження обкладинки чи файлу книги, чітко відокремлені та адаптовані під архітектуру Supabase, що сприяє збереженню чистоти архітектури проєкту.

Моделі даних (Book, Author, User тощо)

Кожна сутність реалізована як клас-модель із анотаціями JsonProperty, що забезпечує автоматичну десеріалізацію JSON-об’єктів від Supabase у типізовані об’єкти C#. Це дозволяє працювати з даними у зручній формі, без ручної обробки JSON.

Для прикладу розглянемо модель Book, що містить такі поля: isbn, title, authors, genres, year_published, book_cover, book_file тощо. На рис.3.16 наведено лістинг реалізації описаної моделі.

```
public class Book
{
    [JsonProperty("isbn")]
    public string ISBN { get; set; }
    [JsonProperty("title")]
    public string Title { get; set; }
    [JsonProperty("description")]
    public string Description { get; set; }
    [JsonProperty("series")]
    public string Series { get; set; }
    [JsonProperty("year_published")]
    public short? YearPublished { get; set; }
    [JsonProperty("book_cover")]
    public string BookCoverUrl { get; set; }
    public Image BookCoverImage { get; set; }
    [JsonProperty("add_date")]
    public DateTime AddDate { get; set; }
    [JsonProperty("average_rating")]
    public decimal? AverageRating { get; set; }
    [JsonProperty("authors")]
    public string Authors { get; set; }
    [JsonProperty("genres")]
    public string Genres { get; set; }
    [JsonProperty("available_formats")]
    public string AvailableFormats { get; set; }
}
```

Рис.3.16 Модель Book

Усі запити виконуються асинхронно, що забезпечує відгукливість додатку. Така реалізація дозволяє централізовано керувати всією логікою взаємодії з базою даних у межах одного сервісу, що є ознакою якісного та підтримуваного архітектурного підходу.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Завершальним етапом розробки програмної системи є тестування, яке дозволяє перевірити її функціональність, надійність та відповідність вимогам. У рамках цього етапу було проведено комплексне тестування системи, що включало:

- функціональне тестування;
- перевірку коректності обробки введених даних;
- тестування реакції на помилки;
- оцінку інтерфейсу користувача;
- базову перевірку безпеки та продуктивності.

Тестування функціональності

Було перевірено роботу основного функціоналу системи відповідно до вимог проєкту.

Реєстрація та авторизація користувачів. Для реєстрації користувача необхідно заповнити всі поля відповідної форми. У разі введення коректних даних система успішно створює обліковий запис. Якщо ж допущено помилку (наприклад, порушено формат email або залишено обов’язкове поле порожнім), додаток надає відповідне повідомлення. З метою підвищення безпеки, під час реєстрації з роллю адміністратора потрібно додатково ввести спеціальний адміністративний пароль. Цей пароль є частиною службової інформації й надається замовнику разом з інсталяційним пакетом системи.

а)


OnLineLibrary - Вхід
LIBRARY
Вітаємо
в нашій
бібліотеці
Реєстрація
Зареєструватись як
Читач
Ім'я користувача
Іванов Іван
E-mail
ivanon@gmail.com
Пароль
1111
Підтвердження
1111
Зареєструватись
Уже маєш акаунт? Авторизація
Реєстрація успішна!
OK

б)


OnLineLibrary - Вхід
Вітаємо
в нашій
бібліотеці
Реєстрація
Зареєструватись як
Бібліотекар
adminPassword123
Ім'я користувача
алентина Іванівна Корольчук
E-mail
korolchykValent@gmail.com
Пароль
1234
Підтвердження
1234
Зареєструватись
Уже маєш акаунт? Авторизація
Реєстрація успішна!
OK

в)

Рис.4.1 Реєстрація користувача в ролі читача (а), бібліотекаря (б) та авторизація в системі (в)

Додавання, редагування та видалення книжок. Даний функціонал доступний лише адміністратору. Меню керування містить декілька вкладок, що відповідають за управління книгами, жанрами, серіями, користувачами, тощо. Для прикладу, розглянемо процес додавання нової книги до системи.

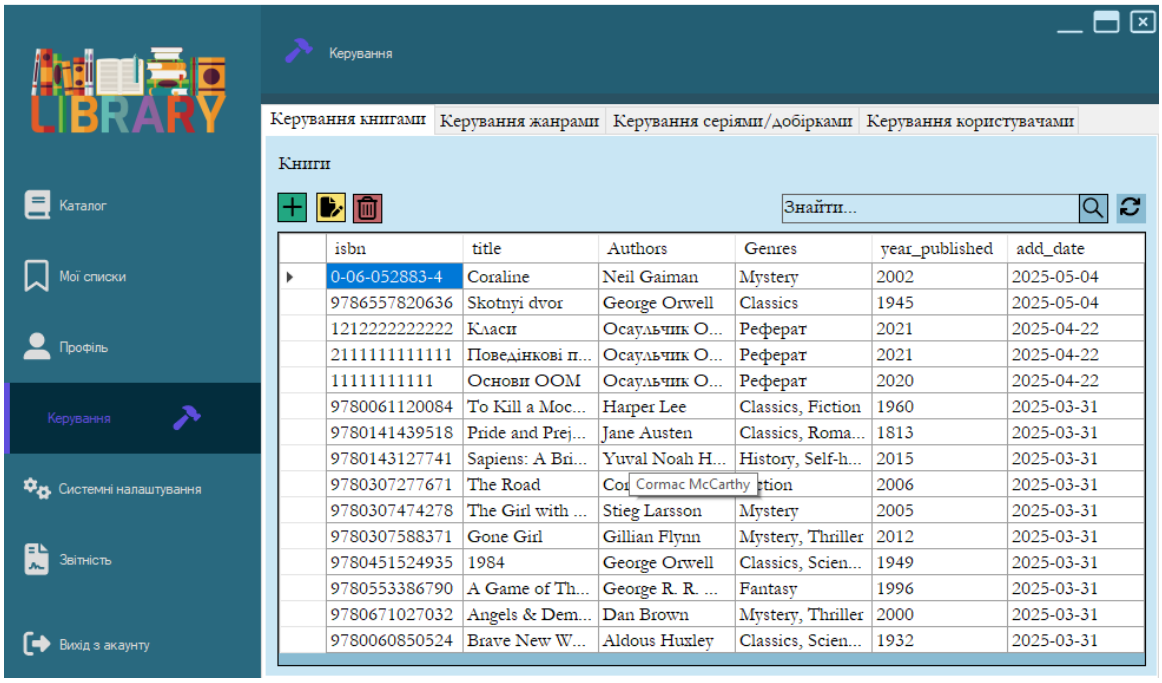


Рис.4.2 Вікно керування книгами

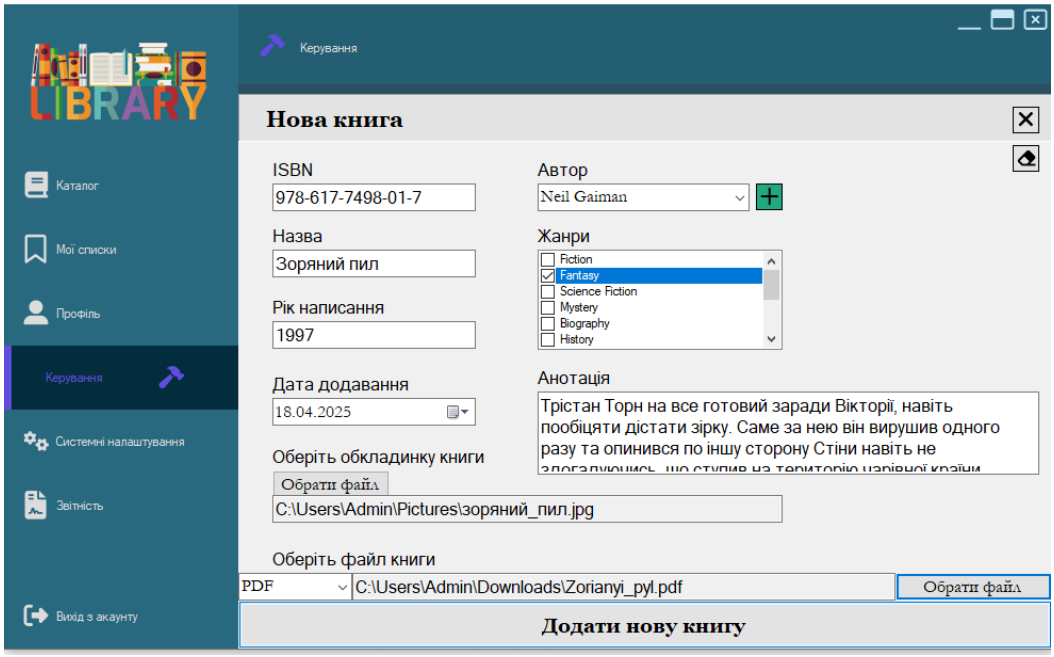


Рис.4.3 Внесення усіх метаданих книги

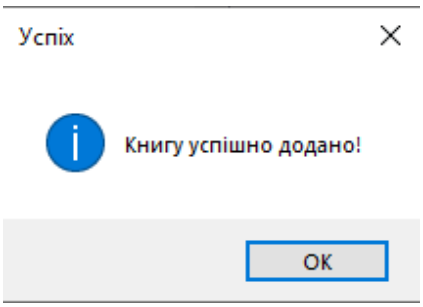


Рис.4.4 Повідомлення про успішне додавання

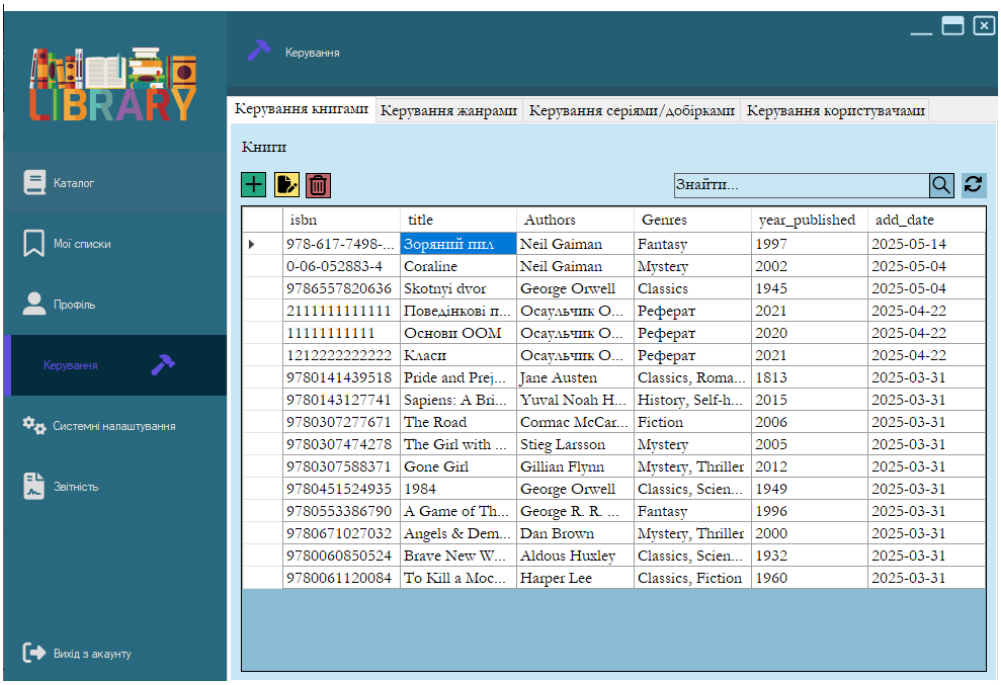


Рис.4.5 Поява нової книги в списку

Завантаження та перегляд файлів книжок. Для завантаження книги необхідно у вікні детальної інформації про книгу обрати один з доступних варіантів та натиснути кнопку завантажити. Читання книги можливе як з вікна детальної інформації, так і з вікна користувацьких списків.

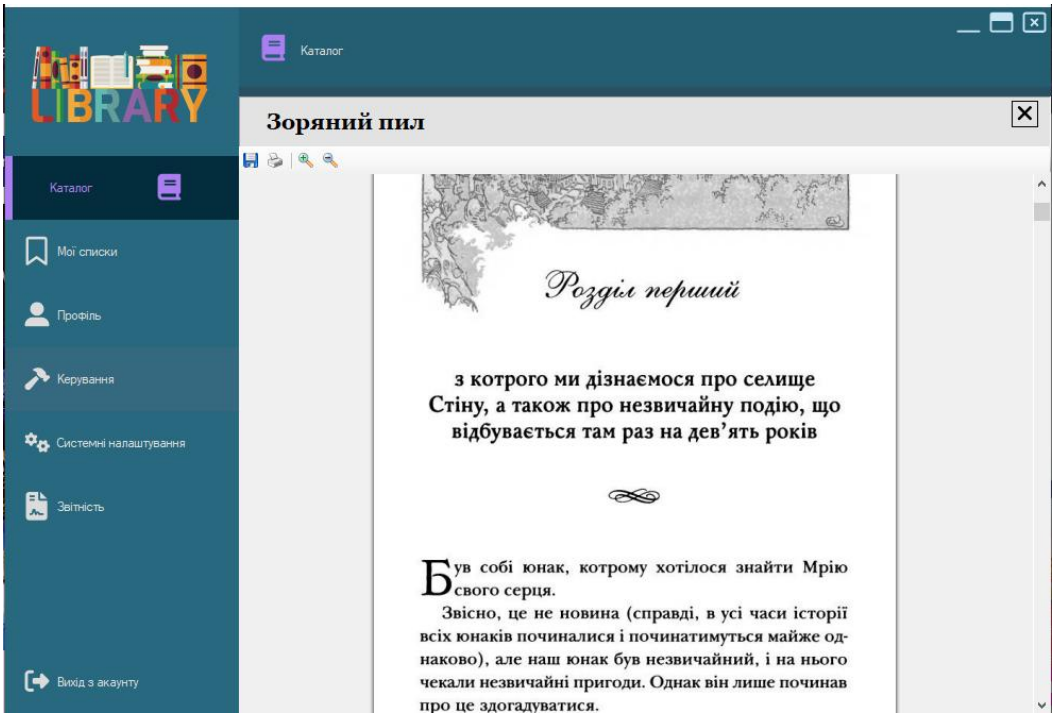


Рис.4.6 Відкриття книги для прочитання

Пошук книг за назвою, автором або жанром. Вікно каталогу містить фільтраційну панель завдяки якій можна шукати книги за різними параметрами.

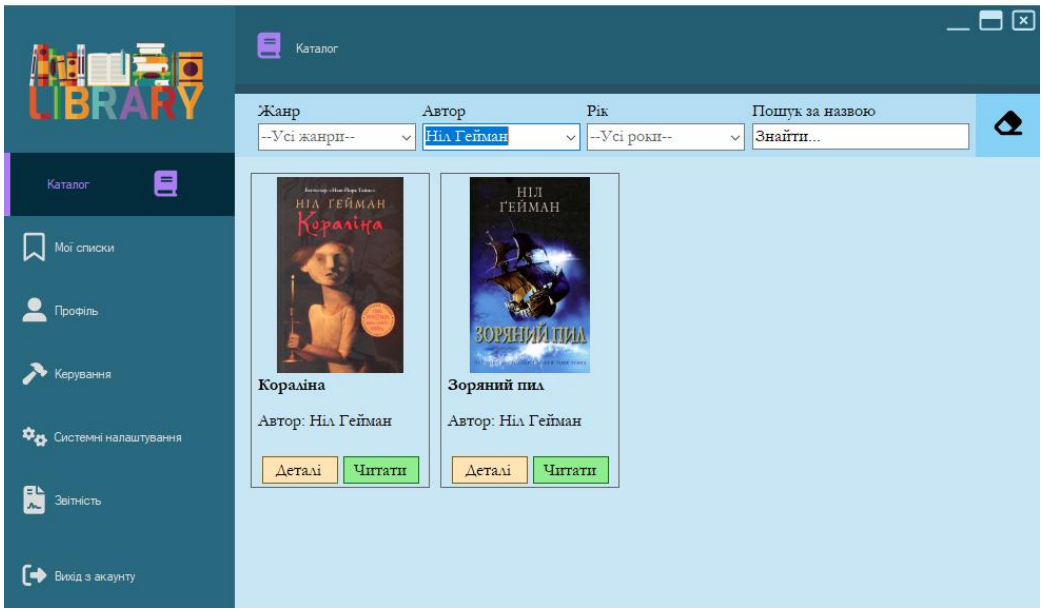


Рис.4.7 Фільтрація книг за автором

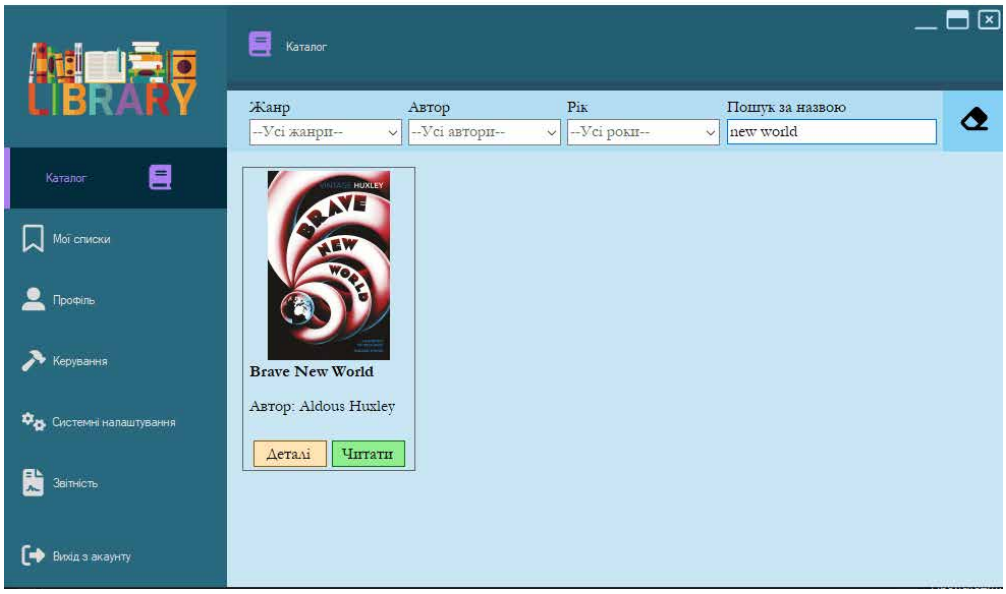


Рис.4.8 Фільтрація книг за назвою

Створення персональних списків книг (наприклад, “Улюблене”, “До прочитання”). Вікно деталей книги містить випадаючий список з усіма створеними списками користувача та кнопку «Додати». Необхідно просто обрати список, до якого потрібно додати книгу і натиснути кнопку. Створити новий список, редагувати чи видалити наявний можна за допомогою відповідних

кнопок над переліком списків. Щоб видалити книгу зі списку, необхідно натиснути відповідну кнопку на картці та підтвердити видалення.

а) б)

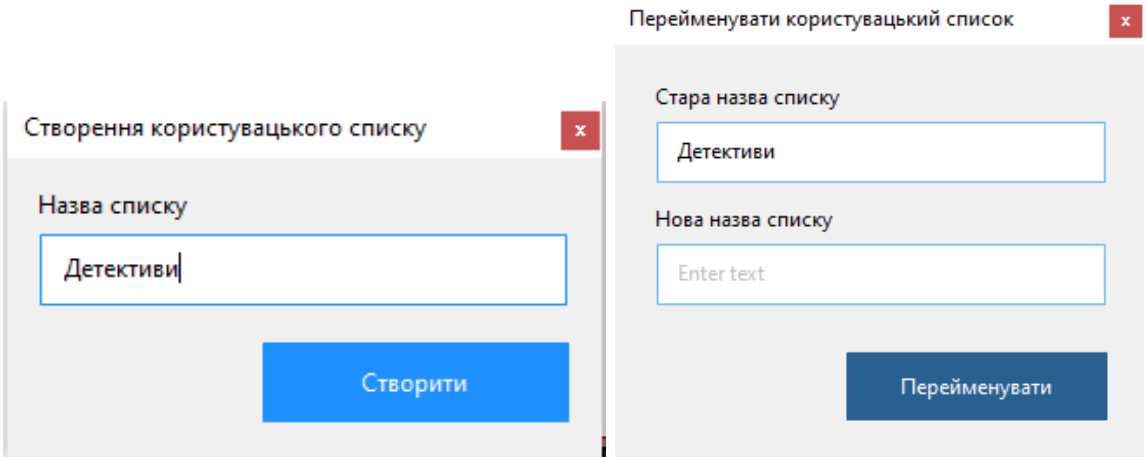


Рис.4.9 Контекстні меню створення (а) та редагування (б) списків

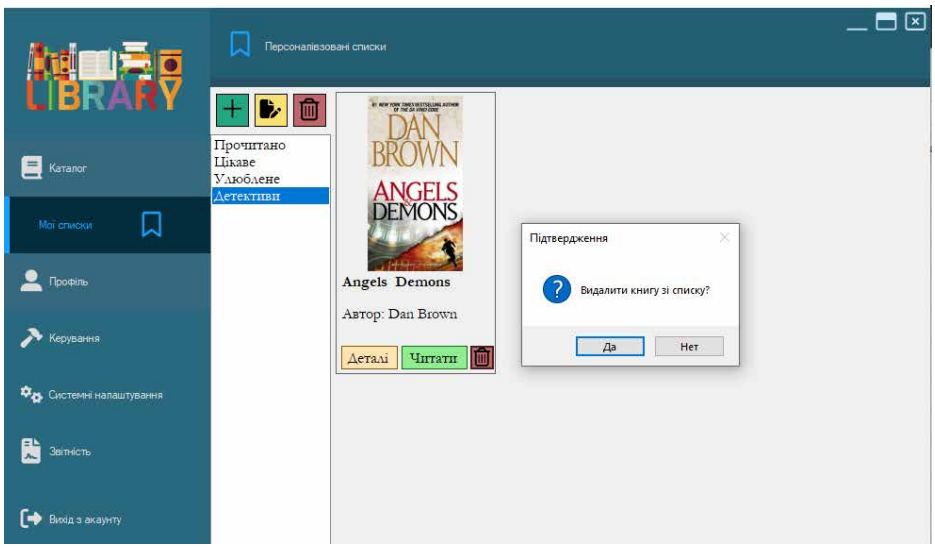


Рис.4.10 Підтвердження видалення книги з обраного списку

Додавання коментарів та оцінювання книг. Ці дії теж доступні у вікні деталей. Для оцінки є спеціальне поле, де обирається цифра від 1 до 10 та натискається кнопка «Оцінити». Після оцінювання ці елементи стають неактивними, а в полі оцінки відображається поставлена. Рейтинг книги також оновлюється, адже він формується на основі середнього значення усіх поставлених книзі оцінок. Для написання коментаря необхідно у відповідному

текстовому полі ввести його текст та натиснути «Додати коментар». Написаний коментар з’явиться в полі коментарів, де буде також відображатись ваше ім’я та дата додавання коментаря.

а)

б)

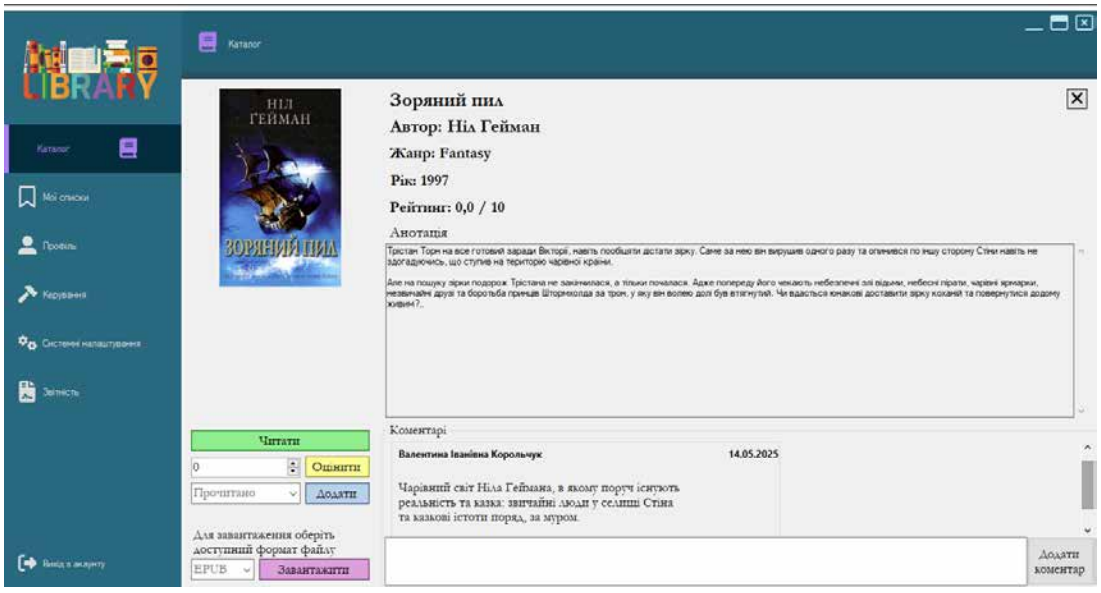


Рис.4.11 Процес додавання коментаря до книги

Рис.4.12 Оцінювання книги

Після проведення цих та інших тестів функціональності ми переконалися, що всі основні функції системи працюють бездоганно і відповідають вимогам специфікації.

Тестування обробки введених даних

Особливу увагу приділено валідації полів при введенні інформації:

- перевірка обов’язкових полів (наприклад, назва книги, ISBN, формат файлу);
- обробка некоректного формату email, дат, або повторного ISBN;
- перевірка допустимих значень рейтингів та коментарів.

У разі неправильного введення система надає зрозуміле повідомлення про помилку (рис 4.13) без перезавантаження сторінки, що покращує UX.

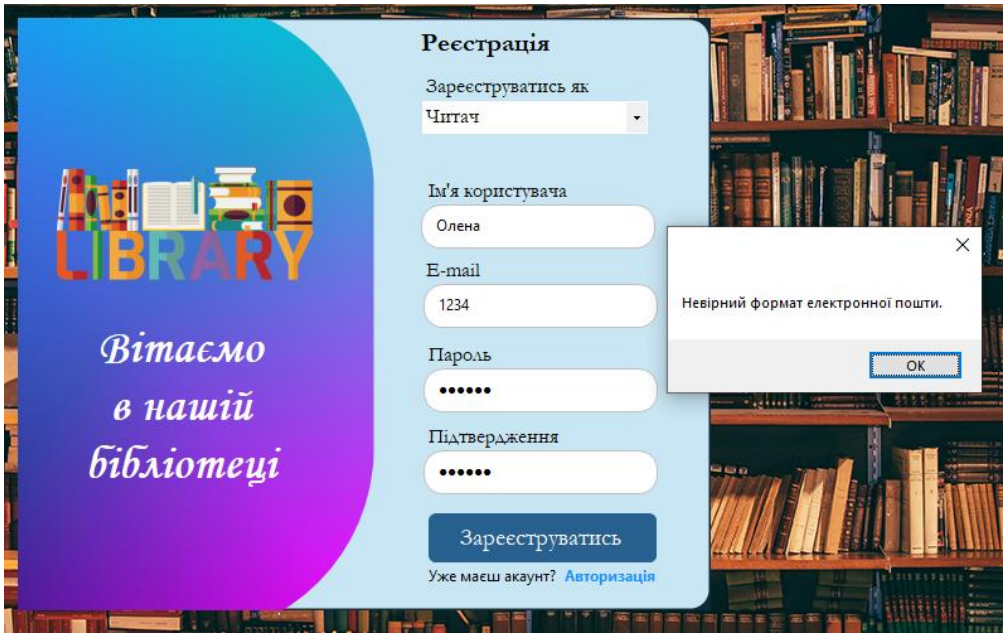


Рис.4.13 Валідація email під час реєстрації

Тестування реакції на помилки

Було перевірено поведінку системи в умовах непередбачуваних або помилкових дій користувача, наприклад:

- спроба реєстрації уже зареєстрованого e-mail;
- спроба додавання вже існуючу книгу (з тим самим ISBN);
- завантаження файлу непідтримуваного формату;
- введення SQL-ін’єкцій у поля введення;
- спроба доступу до закритих функцій без авторизації.

Далі наведено приклад повідомлення про помилку (рис 4.14).

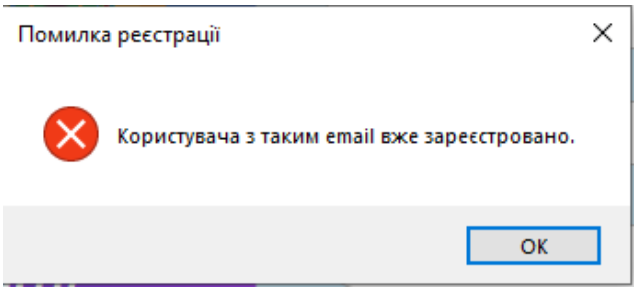


Рис. 4.14 Повідомлення про помилку

Система правильно обробляє помилки, не порушуючи загальну стабільність, і забезпечує інформативне повідомлення для користувача.

Тестування інтерфейсу користувача

Інтерфейс системи був перевірений на відповідність базовим принципам зручності, в тому числі:

- логічна структура сторінок;
- зрозумілий і мінімалістичний дизайн;
- реакція на дії користувача (наприклад, підтвердження додавання до списку);
- коректне відображення на різних розмірах екранів (адаптивність).

Результати показали, що інтерфейс є інтуїтивно зрозумілим, а навігація — зручною для користувача.

4.2 Вимоги до апаратного та програмного забезпечення

Для забезпечення ефективного функціонування програмної системи OnLineLibrary визначено вимоги до апаратного та програмного забезпечення з урахуванням архітектури системи, її функціонального призначення та передбачуваного навантаження. Враховуючи клієнт-серверну модель взаємодії, система складається з двох основних вузлів — клієнтського (робоча станція користувача) та серверного (віддалене сховище даних та бекенд-логіка).

На діаграмі розміщення (рис. 4.15) зображено логічну структуру системи, яка включає наступні компоненти:

- Клієнтський додаток (WinForms-додаток, що встановлюється локально)
- Веб-сервіси для обробки запитів та взаємодії з базою даних
- Хмарна платформа Supabase як серверна частина, що забезпечує зберігання даних, аутентифікацію та RESTful API

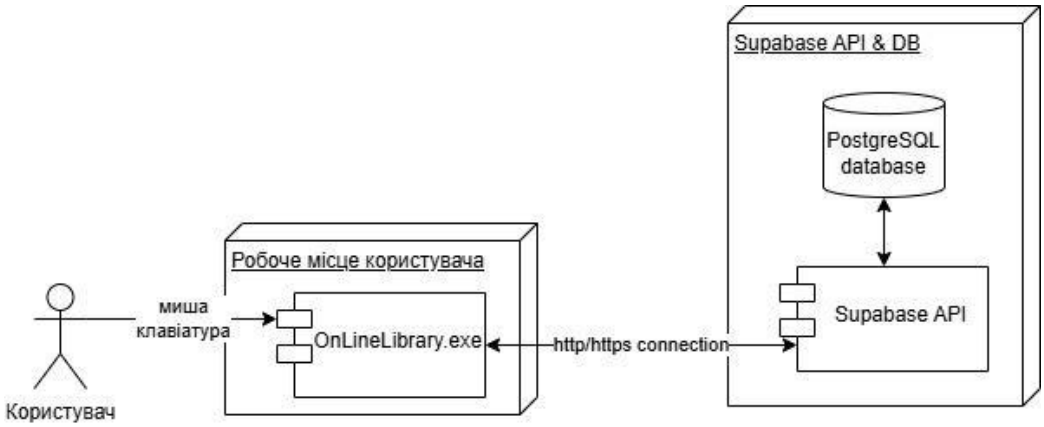


Рис.4.15 Діаграма розміщення компонентів системи

Таблиця 4.1

Апаратні вимоги клієнтської частини наведені у таблиці

Компонент	**Мінімальні вимоги**	**Рекомендовані вимоги**
Процесор (CPU)	2-ядерний, 2.0 GHz	4-ядерний, 2.5 GHz або вищий
Оперативна пам’ять (RAM)	4 ГБ	8 ГБ або більше
Відеокарта	Інтегрована	Дискретна або інтегрована з підтримкою DirectX 11
Вільне місце на диску	500 МБ	1 ГБ або більше
Мережеве підключення	Інтернет з’єднання зі швидкістю не менше 5 Мбіт/с	Стабільне широкосмугове з’єднання

Серверна частина розміщується на керованому хостингу, тому апаратні ресурси адмініструються хмарною платформою. Система не потребує локального серверного обладнання від користувача або компанії-замовника.

Таблиця 4.2

Програмні вимоги клієнтсьокої частини наведені в таблиці

Програмне забезпечення	**Мінімальна версія**	**Примітка**
Операційна система	Windows 10 або новіша	Рекомендується Windows 11
.NET Framework	.NET 6.0 або новіша	Потрібна для запуску WinForms-додатку
СУБД або драйвер доступу	PostgreSQL ODBC драйвер	Для інтеграції з Supabase (PostgreSQL)
Браузер (опціонально)	Google Chrome, Firefox, Edge	Для доступу до веб-інтерфейсів Supabase
Антивірус/фаєрвол	Необов’язково	Має дозволяти вихідні з’єднання до API

Програмні вимоги серверної частини (Supabase):

- СУБД: PostgreSQL 14+;
- API-інтерфейс: RESTful та Realtime WebSocket (за замовчуванням у Supabase);
- сертифікація безпеки: HTTPS, JWT-аутентифікація.

4.3 Склад інсталяційного пакету

Інсталяційний пакет системи OnLineLibrary забезпечує автоматизовану установку всіх необхідних компонентів клієнтської частини на робочих станціях користувачів. Відповідно до побудованої топології (див. рис.4.15), серверна частина не вимагає локальної інсталяції — вона розгортається на Supabase, а клієнт працює як автономний десктоп-застосунок з доступом до REST API.

Нижче наведено перелік об’єктів, що входять до складу інсталяційного пакету, та їх призначення.

Таблиця 4.3

Склад інсталяційного пакету наведено в таблиці

Назва файлу/папки	Тип	Опис
OnLineLibrary.Setup.msi	Інсталятор	Головний інсталяційний файл у форматі MSI, створений за допомогою WiX або InstallShield. Запускає весь процес встановлення.
OnLineLibrary.exe	Виконуваний файл	Основний клієнтський додаток (WinForms), що містить усі UI-модулі.
Newtonsoft.Json.dll	Бібліотека	Для серіалізації/десеріалізації JSON при комунікації з Supabase.
BCrypt.Net-Next.dll	Бібліотека	Для хешування паролів згідно зі стандартом BCrypt.
System.Net.Http.dll	Системна бібліотека	Використовується HttpClient для HTTP(S)-запитів.
appsettings.json	Конфігурація	Файл конфігурації клієнта: BaseUrl, ApiKey, timeout, логування.
README.txt	Документація	Інструкції з установки, налаштування й запуску додатку.
LICENSE.txt	Ліцензійний файл	Текст ліцензії використання програмного продукту.
ThirdPartyNotices.txt	Документація	Перелік використаних сторонніх бібліотек із версіями та ліцензіями.
Certificates/	Папка	Сертифікати SSL (якщо використовується власний HTTPS-проксі).
Uninstall.exe	Деінсталятор	Утиліта для повного видалення додатку та очищення реєстру.
Logs/	Папка	Директорія для лог-файлів (створюється після першого запуску).

Процес інсталяції

1. **Запуск MSI.** Подвійний клік — інсталятор перевіряє наявність .NET 6.0 (при потребі пропонує інсталювати);
2. **Копіювання файлів.** Усі вищезгадані файли та каталоги копіюються до обраного каталогу (за замовчуванням C:\Program Files\OnLineLibrary\);
3. **Налаштування конфігурації.** Інсталятор пропонує ввести або підтвердити значення BaseUrl і ApiKey, які зберігаються в appsettings.json;
4. **Створення ярликів.** Автоматичне створення ярлика на робочому столі та в меню «Пуск»;
5. **Завершення.** Можливість відразу запустити OnLineLibrary.

Таким чином, інсталяційний пакет містить всі необхідні компоненти для швидкого, безпечного та відтворюваного розгортання клієнтської частини OnLineLibrary на робочих станціях користувачів.

ВИСНОВКИ

Дана бакалаврська кваліфікаційна робота присвячена розробці інформаційної системи для управління онлайн-бібліотекою. Метою проєкту було створення ефективного, зручного у використанні та функціонально повного застосунку для організації, перегляду, зберігання та завантаження електронних книг, орієнтованого на потреби індивідуальних користувачів або невеликих локальних спільнот.

У процесі реалізації поставленої мети було послідовно розглянуто всі ключові етапи життєвого циклу програмного забезпечення.

У першому розділі виконано аналіз предметної області, виявлено основні функціональні потреби цільових користувачів, класифіковано ролі та сценарії взаємодії з системою. Побудовано UML-діаграми (прецедентів, діяльності, послідовності), що дозволили формалізувати поведінку системи та уточнити її функціональні вимоги.

У другому розділі розроблено концептуальну модель даних та на її основі реалізовано фізичну структуру інформаційної бази в середовищі Supabase, що базується на СУБД PostgreSQL. Створено повний набір таблиць із коректно налаштованими зв’язками, обмеженнями цілісності та політиками безпеки. Особливу увагу приділено реалізації зв’язків багато-до-багатьох, зберіганню бінарних файлів книг у Supabase Storage, а також контролю доступу до ресурсів через механізми автентифікації та RLS.

У третьому розділі було безпосередньо реалізовано програмну систему. Архітектура побудована за трирівневою моделлю: інтерфейс користувача (Windows Forms), сервіс доступу до даних (SupabaseService) та модельний рівень. Розроблено набір функціональних форм, які забезпечують додавання книг, пошук, фільтрацію, коментування, перегляд та завантаження. Інкапсульована логіка взаємодії з базою дозволила забезпечити чистоту архітектури, повторне використання коду та зручність у тестуванні.

У четвертому розділі проведено комплексне тестування системи, яке охоплювало функціональні сценарії, валідацію введених даних, обробку помилок, зручність інтерфейсу, базові аспекти продуктивності та безпеки. Результати тестування підтвердили стабільну роботу основних компонентів, коректність обробки даних та високу зручність у використанні системи. Також описано вимоги до апаратного й програмного забезпечення, побудовано діаграму розміщення компонентів та сформовано інсталяційний пакет.

Підсумовуючи, можна стверджувати, що розроблена система *OnLineLibrary* успішно пройшла всі етапи проєктування, реалізації та тестування. Вона забезпечує повноцінне функціональне середовище для роботи з електронними книгами, поєднуючи зручність інтерфейсу, безпечну взаємодію з хмарною базою даних та підтримку мультимедійного контенту.

Запропоноване рішення є технічно завершеним і може бути використано як основа для розгортання невеликої бібліотеки або персонального архіву літератури. Його перевагами є модульність, масштабованість, простота розгортання та адаптивність до майбутнього розширення.

У подальшому розвиток системи може включати:

- реалізацію веб- або мобільного клієнта;
- інтеграцію з зовнішніми бібліотечними API (наприклад, Google Books або OpenLibrary);
- впровадження рекомендаційних механізмів на основі оцінок і вподобань;

Таким чином, поставлена в роботі мета повністю досягнута, а система має потенціал до масштабування та впровадження в умовах реального використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Електронна бібліотека [Електронний ресурс] – Режим доступу: https://ube.nlu.org.ua/article/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%20%D0%B1%D1%96%D0%B1%D0%BB%D1%96%D0%BE%D1%82%D0%B5%D0%BA%D0%B0
2. Електронна бібліотека як середовище адаптивного агрегування інформації [Електронний ресурс] – Режим доступу: http://dwl.kiev.ua/art/bib2013/dwl_obar_2013.pdf
3. Уніфікована мова моделювання (Unified Modeling Language - UML) [Електронний ресурс] – Режим доступу: https://www.maxzosim.com/unifikovana-mova-modeluvannia/
4. BookONO E-book Manager [Електронний ресурс] – Режим доступу: https://www.etopian.com/software/bookono/
5. Calibre ebook management [Електронний ресурс] – Режим доступу: https://calibre-ebook.com/
6. Alfa eBooks Manager [Електронний ресурс] – Режим доступу: https://www.alfaebooks.ru/
7. Open Library [Електронний ресурс] – Режим доступу: https://openlibrary.org/
8. PostgreSQL Documentation [Електронний ресурс] – Режим доступу: https://www.postgresql.org/docs/current/
9. Supabase Documentation [Електронний ресурс] – Режим доступу: https://supabase.com/docs
10. Інтерфейс користувача [Електронний ресурс] – Режим доступу: https://ube.nlu.org.ua/article/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%20%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0
11. Посібник «Проектування інформаційних систем» Макаренко [Електронний ресурс] – Режим доступу:

https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20160217112601/content-20160217112601.pdf

12. Visual Studio IDE documentation [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/uk-ua/visualstudio/ide/?view=vs-2022
13. Microsoft Docs: Windows Forms documentation [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/
14. Основи UML [Електронний ресурс] – Режим доступу: https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-elements.html
15. Нейгел К., Ивьен Б., Глинн Д., Уотсон К., Скиннер М. C# 5.0 и платформа .NET 4.5 для профессионалов. Учеб. пособие. – М.: Вильямс, 2014. – 1440с.
16. Хейлперс, Тим. "C# 9 and .NET 5 - Modern Cross-Platform Development". - Birmingham: Packt Publishing, 2020. - 822 с.
17. John Sharp. "Microsoft Visual C# Step by Step" – 10th Edition – Microsoft Press, 2022. – 832с.

ДОДАТОК А

Sql скрипт для створення бази даних та її таблиць, а також внесення умовно-постійної інформації

Сторінок - 5

Київ – 2025

```
-- Створення бази
CREATE DATABASE OnLineLibraryDb;

-- Таблиця користувачів
CREATE TABLE "User" (
    ID_user SERIAL PRIMARY KEY,
    UserName VARCHAR(100) NOT NULL,
    UserEmail VARCHAR(150) NOT NULL,
    Password_hash VARCHAR(255), -- або NULL, якщо плануєш підтримувати авторизацію без
пароля
    Role VARCHAR(50) CHECK (Role IN ('Admin', 'User')) DEFAULT 'User',
    Created_date DATE NOT NULL
);

-- Таблиця авторів
CREATE TABLE Author (
    AuthorId SERIAL PRIMARY KEY,
    FullName VARCHAR(150) NOT NULL,
    BirthDate DATE,
    Nationality VARCHAR(100),
    Biography TEXT
);

-- Таблиця жанрів
CREATE TABLE Genre (
    GenreId SERIAL PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Description TEXT
);

-- Таблиця книг
CREATE TABLE Book (
    ISBN VARCHAR(30) PRIMARY KEY,
    Title VARCHAR(255) NOT NULL,
    Description TEXT,
    Series VARCHAR(100),
    Year_published SMALLINT,
    Book_cover VARCHAR(255),
    Add_date DATE NOT NULL,
    Rating NUMERIC(3,2)
);

-- Зв’язок книга - автор
CREATE TABLE Book_Author (
    ISBN VARCHAR(30),
    AuthorId INT,
    PRIMARY KEY (ISBN, AuthorId),
    FOREIGN KEY (ISBN) REFERENCES Book(ISBN) ON DELETE CASCADE,
    FOREIGN KEY (AuthorId) REFERENCES Author(AuthorId) ON DELETE CASCADE
);

-- Зв’язок книга - жанр
CREATE TABLE Book_Genre (
    ISBN VARCHAR(30),
    GenreId INT,
    PRIMARY KEY (ISBN, GenreId),
    FOREIGN KEY (ISBN) REFERENCES Book(ISBN) ON DELETE CASCADE,
    FOREIGN KEY (GenreId) REFERENCES Genre(GenreId) ON DELETE CASCADE
);
```

```
-- Таблиця форматів файлів
CREATE TABLE FileFormat (
    FormatCode VARCHAR(20) PRIMARY KEY,
    Description VARCHAR(100)
);

-- Таблиця файлів книг
CREATE TABLE Book_file (
    ISBN VARCHAR(30),
    FormatCode VARCHAR(20),
    Book_file BYTEA NOT NULL,
    PRIMARY KEY (ISBN, FormatCode),
    FOREIGN KEY (ISBN) REFERENCES Book(ISBN) ON DELETE CASCADE,
    FOREIGN KEY (FormatCode) REFERENCES FileFormat(FormatCode)
);

-- Коментарі
CREATE TABLE BookComment (
    CommentId SERIAL PRIMARY KEY,
    ISBN VARCHAR(30),
    ID_user INT,
    Comment TEXT NOT NULL,
    CommentDate DATE NOT NULL,
    FOREIGN KEY (ISBN) REFERENCES Book(ISBN) ON DELETE CASCADE,
    FOREIGN KEY (ID_user) REFERENCES "User"(ID_user) ON DELETE SET NULL
);

-- Оцінки
CREATE TABLE BookRating (
    RatingId SERIAL PRIMARY KEY,
    ISBN VARCHAR(30),
    ID_user INT,
    Rating SMALLINT CHECK (Rating BETWEEN 1 AND 5),
    RatedAt DATE NOT NULL,
    FOREIGN KEY (ISBN) REFERENCES Book(ISBN) ON DELETE CASCADE,
    FOREIGN KEY (ID_user) REFERENCES "User"(ID_user) ON DELETE SET NULL
);

-- Списки книг
CREATE TABLE BookList (
    ListId SERIAL PRIMARY KEY,
    ID_user INT,
    Name VARCHAR(100) NOT NULL,
    FOREIGN KEY (ID_user) REFERENCES "User"(ID_user) ON DELETE CASCADE
);

-- Записи списку
CREATE TABLE BookListEntry (
    Id_entry SERIAL PRIMARY KEY,
    ListId INT,
    ISBN VARCHAR(30),
    AddedDate DATE NOT NULL,
    FOREIGN KEY (ListId) REFERENCES BookList(ListId) ON DELETE CASCADE,
    FOREIGN KEY (ISBN) REFERENCES Book(ISBN) ON DELETE CASCADE
);

-- Заповнення таблиці форматів файлів
INSERT INTO FileFormat (FormatCode, Description) VALUES
('PDF', 'Формат електронних документів від Adobe'),
```

```
('EPUB', 'Стандартний формат для електронних книг'),
('MOBI', 'Формат для електронних книг, використовуваний Amazon Kindle'),
('FB2', 'Формат електронних книг FictionBook'),
('TXT', 'Звичайний текстовий формат без форматування');

-- Заповнення таблиці жанрів
INSERT INTO Genre (Genre_name, Description) VALUES
('Fiction', 'Художня література'),
('Fantasy', 'Фентезі: вигадані світи, магія'),
('Science Fiction', 'Наукова фантастика'),
('Mystery', 'Детективи, загадки'),
('Biography', 'Біографії та мемуари'),
('History', 'Історична література'),
('Romance', 'Романтична література'),
('Horror', 'Жахи'),
('Self-help', 'Саморозвиток'),
('Classics', 'Класика світової літератури'),
('Thriller', 'Напружені трилери');

-- Книга 1
INSERT INTO Book (ISBN, Title, Year_published, Description, Add_date)
VALUES ('9780143127741', 'Sapiens: A Brief History of Humankind', 2015, 'Історія людства
від первісного суспільства до сучасності', '2025-03-31');

INSERT INTO Author (FullName, BirthDate, Nationality)
VALUES ('Yuval Noah Harari', '1976-02-24', 'Ізраїль');

INSERT INTO Book_Author (ISBN, AuthorId)
SELECT '9780143127741', AuthorId FROM Author WHERE FullName = 'Yuval Noah Harari';

INSERT INTO Book_Genre (ISBN, GenreId)
SELECT '9780143127741', GenreId FROM Genre WHERE Genre_name IN ('History', 'Self-help');

-- Книга 2
INSERT INTO Book (ISBN, Title, Year_published, Description, Add_date)
VALUES ('9780553386790', 'A Game of Thrones', 1996, 'Перша книга з циклу "Пісня льоду й
полум’я"', '2025-03-31');

INSERT INTO Author (FullName, BirthDate, Nationality)
VALUES ('George R. R. Martin', '1948-09-20', 'США');

INSERT INTO Book_Author (ISBN, AuthorId)
SELECT '9780553386790', AuthorId FROM Author WHERE FullName = 'George R. R. Martin';

INSERT INTO Book_Genre (ISBN, GenreId)
SELECT '9780553386790', GenreId FROM Genre WHERE Genre_name = 'Fantasy';

-- Книга 3
INSERT INTO Book (ISBN, Title, Year_published, Description, Add_date)
VALUES ('9780061120084', 'To Kill a Mockingbird', 1960, 'Класика американської літератури
про расизм і справедливість', '2025-03-31');

INSERT INTO Author (FullName, BirthDate, Nationality)
VALUES ('Harper Lee', '1926-04-28', 'США');

INSERT INTO Book_Author (ISBN, AuthorId)
SELECT '9780061120084', AuthorId FROM Author WHERE FullName = 'Harper Lee';

INSERT INTO Book_Genre (ISBN, GenreId)
```

```
SELECT '9780061120084', GenreId FROM Genre WHERE Genre_name IN ('Classics', 'Fiction');

-- Книга 4
INSERT INTO Book (ISBN, Title, Year_published, Description, Add_date)
VALUES ('9780451524935', '1984', 1949, 'Антиутопічний роман про тоталітаризм', '2025-03-
31');

INSERT INTO Author (FullName, BirthDate, Nationality)
VALUES ('George Orwell', '1903-06-25', 'Велика Британія');

INSERT INTO Book_Author (ISBN, AuthorId)
SELECT '9780451524935', AuthorId FROM Author WHERE FullName = 'George Orwell';

INSERT INTO Book_Genre (ISBN, GenreId)
SELECT '9780451524935', GenreId FROM Genre WHERE Genre_name IN ('Science Fiction',
'Classics');

-- Книга 5
INSERT INTO Book (ISBN, Title, Year_published, Description, Add_date)
VALUES ('9780307474278', 'The Girl with the Dragon Tattoo', 2005, 'Детективний роман з
елементами трилеру', '2025-03-31');

INSERT INTO Author (FullName, BirthDate, Nationality)
VALUES ('Stieg Larsson', '1954-08-15', 'Швеція');

INSERT INTO Book_Author (ISBN, AuthorId)
SELECT '9780307474278', AuthorId FROM Author WHERE FullName = 'Stieg Larsson';

INSERT INTO Book_Genre (ISBN, GenreId)
SELECT '9780307474278', GenreId FROM Genre WHERE Genre_name = 'Mystery';

-- Книга 6
INSERT INTO Book (ISBN, Title, Year_published, Description, Add_date)
VALUES ('9780141439518', 'Pride and Prejudice', 1813, 'Класичний роман про любов і
соціальні норми', '2025-03-31');

INSERT INTO Author (FullName, BirthDate, Nationality)
VALUES ('Jane Austen', '1775-12-16', 'Велика Британія');

INSERT INTO Book_Author (ISBN, AuthorId)
SELECT '9780141439518', AuthorId FROM Author WHERE FullName = 'Jane Austen';

INSERT INTO Book_Genre (ISBN, GenreId)
SELECT '9780141439518', GenreId FROM Genre WHERE Genre_name IN ('Romance', 'Classics');

-- Книга 7
INSERT INTO Book (ISBN, Title, Year_published, Description, Add_date)
VALUES ('9780060850524', 'Brave New World', 1932, 'Фантастичне бачення майбутнього
суспільства', '2025-03-31');

INSERT INTO Author (FullName, BirthDate, Nationality)
VALUES ('Aldous Huxley', '1894-07-26', 'Велика Британія');

INSERT INTO Book_Author (ISBN, AuthorId)
SELECT '9780060850524', AuthorId FROM Author WHERE FullName = 'Aldous Huxley';

INSERT INTO Book_Genre (ISBN, GenreId)
SELECT '9780060850524', GenreId FROM Genre WHERE Genre_name IN ('Science Fiction',
'Classics');
```

```
-- Книга 8
INSERT INTO Book (ISBN, Title, Year_published, Description, Add_date)
VALUES ('9780307277671', 'The Road', 2006, 'Постапокаліптичний роман про батька та сина',
'2025-03-31');

INSERT INTO Author (FullName, BirthDate, Nationality)
VALUES ('Cormac McCarthy', '1933-07-20', 'США');

INSERT INTO Book_Author (ISBN, AuthorId)
SELECT '9780307277671', AuthorId FROM Author WHERE FullName = 'Cormac McCarthy';

INSERT INTO Book_Genre (ISBN, GenreId)
SELECT '9780307277671', GenreId FROM Genre WHERE Genre_name = 'Fiction';

-- Книга 9
INSERT INTO Book (ISBN, Title, Year_published, Description, Add_date)
VALUES ('9780307588371', 'Gone Girl', 2012, 'Психологічний трилер про зникнення дружини',
'2025-03-31');

INSERT INTO Author (FullName, BirthDate, Nationality)
VALUES ('Gillian Flynn', '1971-02-24', 'США');

INSERT INTO Book_Author (ISBN, AuthorId)
SELECT '9780307588371', AuthorId FROM Author WHERE FullName = 'Gillian Flynn';

INSERT INTO Book_Genre (ISBN, GenreId)
SELECT '9780307588371', GenreId FROM Genre WHERE Genre_name IN ('Thriller', 'Mystery');


-- Книга 10
INSERT INTO Book (ISBN, Title, Year_published, Description, Add_date)
VALUES ('9780671027032', 'Angels & Demons', 2000, 'Пригодницький роман з елементами
історії й релігії', '2025-03-31');

INSERT INTO Author (Fullname, Birthdate, Nationality)
VALUES ('Dan Brown', '1964-06-22', 'США');

INSERT INTO Book_Author (ISBN, AuthorID)
SELECT '9780671027032', AuthorId FROM Author WHERE Fullname = 'Dan Brown';

INSERT INTO Book_Genre (ISBN, GenreID)
SELECT '9780671027032', GenreId FROM Genre WHERE Genre_name IN ('Mystery', 'Thriller');
```

ДОДАТОК Б

Важливі методи класу SupabaseService

Сторінок - 11

Київ – 2025

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Drawing;
using System.Windows.Forms;

namespace OnLineLibrary
{
    public class SupabaseService
    {

        private readonly string baseUrl = "https://muonuwsnudasdlvviuqe.supabase.co";
        private readonly string apiKey = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...";

        private readonly HttpClient _client;

        public SupabaseService()
        {
            _client = new HttpClient
            {
                BaseAddress = new Uri($"{baseUrl}/rest/v1/")
            };
            _client.DefaultRequestHeaders.Add("apikey", apiKey);
            _client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", apiKey);
        }

        /// <summary>
        /// Авторизація користувача
        /// </summary>
        public async Task<User> LoginUserAsync(string email, string plainPassword)
        {
            var encodedEmail = Uri.EscapeDataString(email);
            var response = await
_client.GetAsync($"user?useremail=eq.{encodedEmail}");

            if (!response.IsSuccessStatusCode)
                throw new Exception("Помилка при перевірці email.");

            var json = await response.Content.ReadAsStringAsync();

            if (!json.TrimStart().StartsWith("["))
            {
                dynamic error = JsonConvert.DeserializeObject(json);
                throw new Exception($"Помилка при авторизації: {error?.message ??
"Невідома помилка"}");
            }

            var users = JsonConvert.DeserializeObject<List<User>>(json);

            if (users == null || users.Count == 0)
                return null; // Користувача не знайдено

            var user = users[0];
```

```
            // Перевірка пароля
            bool isPasswordCorrect = BCrypt.Net.BCrypt.Verify(plainPassword,
user.password_hash);

            return isPasswordCorrect ? user : null;
        }

        /// <summary>
        /// Реєстрація нового користувача
        /// </summary>
        public async Task<bool> RegisterUserAsync(string username, string email,
string plainPassword, string selectedRole)
        {
            // Перевірка, чи email вже існує
            var encodedEmail = Uri.EscapeDataString(email);
            var checkResponse = await
_client.GetAsync($"user?useremail=eq.{encodedEmail}");

            if (!checkResponse.IsSuccessStatusCode)
                throw new Exception("Не вдалося перевірити email.");

            var existingJson = await checkResponse.Content.ReadAsStringAsync();

            if (existingJson.TrimStart().StartsWith("["))
            {
                var existingUsers =
JsonConvert.DeserializeObject<List<User>>(existingJson);
                if (existingUsers != null && existingUsers.Count > 0)
                    return false; // Email вже зареєстрований
            }
            else
            {
                dynamic error = JsonConvert.DeserializeObject(existingJson);
                throw new Exception($"Помилка при перевірці: {error?.message ??
"Невідома помилка"}");
            }

            // Хешування пароля (bcrypt)
            string passwordHash = BCrypt.Net.BCrypt.HashPassword(plainPassword);

            // Переклад ролі з української на англійську
            string role = selectedRole == "Бібліотекар" ? "Admin" : "User";

            // Формування нового користувача
            var user = new
            {
                username = username,
                useremail = email,
                password_hash = passwordHash,
                role = role,
                created_date = DateTime.UtcNow
            };

            var content = new StringContent(JsonConvert.SerializeObject(user),
Encoding.UTF8, "application/json");
            var response = await _client.PostAsync("user", content); // перевір, щоб
endpoint точно був "user"

            return response.IsSuccessStatusCode;
        }

        /// <summary>
```

```
        /// Отримати всі книги
        /// </summary>
        public async Task<List<Book>> GetAllBooksAsync()
        {
            var books = new List<Book>();

            var response = await _client.GetAsync("full_book_view?select=*");

            if (response.IsSuccessStatusCode)
            {
                var json = await response.Content.ReadAsStringAsync();
                books = JsonConvert.DeserializeObject<List<Book>>(json);

                var defaultImage = Properties.Resources.обгортка;

                foreach (var book in books)
                {
                    if (!string.IsNullOrEmpty(book.BookCoverUrl))
                    {
                        try
                        {
                            var imageBytes = await
_client.GetByteArrayAsync(book.BookCoverUrl);
                            using (var ms = new MemoryStream(imageBytes))
                            {
                                book.BookCoverImage = Image.FromStream(ms);
                            }
                        }
                        catch
                        {
                            book.BookCoverImage = defaultImage;
                        }
                    }
                    else
                    {
                        book.BookCoverImage = defaultImage;
                    }
                }
            }
            else
            {
                throw new Exception($"Помилка отримання даних:
{response.StatusCode}");
            }

            return books;
        }

        /// <summary>
        /// Отримати список авторів
        /// </summary>
        public async Task<List<Author>> GetAuthorsAsync()
        {
            var response = await
_client.GetAsync("author?select=authorid,fullname,birthdate,nationality,biography");

            if (!response.IsSuccessStatusCode)
                return new List<Author>();

            var json = await response.Content.ReadAsStringAsync();
            return JsonConvert.DeserializeObject<List<Author>>(json) ?? new
List<Author>();
```

```
        }

        /// <summary>
        /// Отримати список жанрів
        /// </summary>
        public async Task<List<Genre>> GetGenresAsync()
        {
            var response = await
_client.GetAsync("genre?select=genreid,genre_name,description");

            if (!response.IsSuccessStatusCode)
                return new List<Genre>();

            var json = await response.Content.ReadAsStringAsync();
            return JsonConvert.DeserializeObject<List<Genre>>(json) ?? new
List<Genre>();
        }

        /// <summary>
        /// для вставки даних до бд
        /// </summary>
        public async Task<bool> PostAsync(string tableName, object data)
        {
            var json = JsonConvert.SerializeObject(new[] { data }); // Supabase очікує
масив
            var content = new StringContent(json, Encoding.UTF8, "application/json");

            var response = await _client.PostAsync(tableName, content);
            return response.IsSuccessStatusCode;
        }


        /// <summary>
        /// вставка книг
        /// </summary>
        public async Task<bool> InsertBookAsync(string isbn, string title, int
year_published, string description, string add_date)
        {
            try
            {
                var book = new
                {
                    isbn = isbn,
                    title = title,
                    year_published = year_published,
                    description = string.IsNullOrEmpty(description) ? null :
description,
                    add_date = DateTime.Parse(add_date).ToString("yyyy-MM-dd") // або
просто DateTime тип
                };

                var json = JsonConvert.SerializeObject(book);
                var content = new StringContent(json, Encoding.UTF8,
"application/json");

                var response = await _client.PostAsync("book", content);

                if (!response.IsSuccessStatusCode)
                {
                    string error = await response.Content.ReadAsStringAsync();
```

```
                    MessageBox.Show($"Помилка при додаванні книги:
{response.StatusCode}\n{error}", "HTTP Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                }

                return response.IsSuccessStatusCode;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Внутрішня помилка при додаванні книги:
{ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
            }
        }

        /// <summary>
        /// додавання авторів до книги
        /// </summary>
        public async Task<bool> InsertBookAuthorAsync(string isbn, int authorid)
        {
            try
            {
                var bookAuthor = new
                {
                    isbn = isbn,
                    authorid = authorid
                };

                var json = JsonConvert.SerializeObject(bookAuthor);
                var content = new StringContent(json, Encoding.UTF8,
"application/json");

                var response = await _client.PostAsync("book_author", content);

                if (!response.IsSuccessStatusCode)
                {
                    string error = await response.Content.ReadAsStringAsync();
                    MessageBox.Show($"Помилка при додаванні автора до книги:
{response.StatusCode}\n{error}", "HTTP Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                }

                return response.IsSuccessStatusCode;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Внутрішня помилка при додаванні автора до книги:
{ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
            }
        }

        /// <summary>
        /// додавання жанру до книги
        /// </summary>
        public async Task<bool> InsertBookGenreAsync(string isbn, int genreid)
        {
            try
            {
                var bookGenre = new
                {
                    isbn = isbn,
```

```
                    genreid = genreid
                };

                var json = JsonConvert.SerializeObject(bookGenre);
                var content = new StringContent(json, Encoding.UTF8,
"application/json");

                var response = await _client.PostAsync("book_genre", content);

                if (!response.IsSuccessStatusCode)
                {
                    string error = await response.Content.ReadAsStringAsync();
                    MessageBox.Show($"Помилка при додаванні жанру до книги:
{response.StatusCode}\n{error}", "HTTP Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                }

                return response.IsSuccessStatusCode;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Внутрішня помилка при додаванні жанру до книги:
{ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
            }
        }

        /// <summary>
        /// додавання файлу книги
        /// </summary>
        public async Task<bool> InsertBookFileAsync(string isbn, string formatCode,
byte[] book_file)
        {
            try
            {
                var bookFile = new
                {
                    isbn = isbn,
                    formatcode = formatCode,
                    book_file = Convert.ToBase64String(book_file) // кодуємо у Base64
                };

                var json = JsonConvert.SerializeObject(bookFile);
                var content = new StringContent(json, Encoding.UTF8,
"application/json");

                var response = await _client.PostAsync("book_file", content);

                if (!response.IsSuccessStatusCode)
                {
                    string error = await response.Content.ReadAsStringAsync();
                    MessageBox.Show($"Помилка при додаванні файлу книги:
{response.StatusCode}\n{error}", "HTTP Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                }

                return response.IsSuccessStatusCode;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Внутрішня помилка при додаванні файлу книги:
{ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
                return false;
            }
        }

        // завантаження файлу
        public async Task<byte[]> DownloadBookFileAsync(string fileUrl)
        {
            return await _client.GetByteArrayAsync(fileUrl);
        }

        //метод для отримання файлу за ISBN
        public async Task<(byte[] FileBytes, string FormatCode)>
GetBookFileByIsbnAsync(string isbn)
        {
            var response = await
_client.GetAsync($"book_file?isbn=eq.{isbn}&select=formatcode,book_file");

            if (!response.IsSuccessStatusCode)
                throw new Exception($"Не вдалося завантажити файл книги:
{response.StatusCode}");

            var json = await response.Content.ReadAsStringAsync();
            var data = JsonConvert.DeserializeObject<List<BookFileRecord>>(json);
            if (data == null || data.Count == 0)
                throw new Exception("Файл книги не знайдено.");

            var base64String = data[0].book_file;

            try
            {
                var fileBytes = Convert.FromBase64String(base64String); //  !
декодуємо
                return (fileBytes, data[0].formatcode);
            }
            catch (FormatException)
            {
                throw new Exception("Файл у базі має некоректне кодування (не
Base64).");
            }
        }

        //додавання файлу книги в базу
        private class BookFileRecord
        {
            public string formatcode { get; set; }
            public string book_file { get; set; } // Base64 string
        }


        //---------  форма Користувацьких списків  ---------
        // Отримання списків користувача
        public async Task<List<BookList>> GetUserBookListsAsync(int userId)
        {
            var response = await _client.GetAsync($"booklist?id_user=eq.{userId}");
            response.EnsureSuccessStatusCode();
            var json = await response.Content.ReadAsStringAsync();
            return JsonConvert.DeserializeObject<List<BookList>>(json);
        }

        // Отримання книг у списку
        public async Task<List<Book>> GetBooksInListAsync(int listId)
        {
```

```
            var isbns = await GetISBNsInListAsync(listId);
```



```
            return await GetBooksByISBNsAsync(isbns);
        }

        //Метод для отримання повної інформації по full_book_view
        private async Task<List<Book>> GetBooksByISBNsAsync(List<string> isbns)
        {
            if (isbns == null || isbns.Count == 0)
                return new List<Book>();

            // Екрануємо ISBN у лапки для PostgREST
            var encodedIsbns = string.Join(",", isbns.Select(isbn => $"\"{isbn}\""));
            var url = $"full_book_view?isbn=in.({encodedIsbns})";

            var response = await _client.GetAsync(url);
            response.EnsureSuccessStatusCode();

            var json = await response.Content.ReadAsStringAsync();
            var books = JsonConvert.DeserializeObject<List<Book>>(json);

            var defaultImage = Properties.Resources.обгортка;

            foreach (var book in books)
            {
                if (!string.IsNullOrEmpty(book.BookCoverUrl))
                {
                    try
                    {
                        var imageBytes = await
_client.GetByteArrayAsync(book.BookCoverUrl);
                        using (var ms = new MemoryStream(imageBytes))
                        {
                            book.BookCoverImage = Image.FromStream(ms);
                        }
                    }
                    catch
                    {
                        book.BookCoverImage = defaultImage;
                    }
                }
                else
                {
                    book.BookCoverImage = defaultImage;
                }
            }

            return books;
        }

        // Створення нового списку
        public async Task CreateBookListAsync(BookList list)
        {
            // Створюємо анонімний об’єкт без поля listid
            var payload = new[]
            {
                new
                {
                    id_user = list.id_user,
                    name = list.name
                }
            };
```

```
            var json = JsonConvert.SerializeObject(payload);
```



```
            var content = new StringContent(json, Encoding.UTF8, "application/json");

            var response = await _client.PostAsync("booklist", content);
            var responseText = await response.Content.ReadAsStringAsync();

            if (!response.IsSuccessStatusCode)
            {
                // Викидаємо виняток із повідомленням від Supabase
                throw new Exception($"Помилка створення списку: {response.StatusCode}
- {responseText}");
            }
        }


        // Видалення списку та його елементів
        public async Task DeleteBookListAsync(int listId)
        {

            var request1 = new HttpRequestMessage(HttpMethod.Delete,
$"booklistentry?listid=eq.{listId}");
            request1.Headers.Add("Prefer", "return=representation");
            var entryResponse = await _client.SendAsync(request1);
            var entryContent = await entryResponse.Content.ReadAsStringAsync();

            if (!entryResponse.IsSuccessStatusCode ||
string.IsNullOrWhiteSpace(entryContent))
                throw new Exception($"Не вдалося видалити записи списку:
{entryContent}");

            var request2 = new HttpRequestMessage(HttpMethod.Delete,
$"booklist?listid=eq.{listId}");
            request2.Headers.Add("Prefer", "return=representation");
            var response = await _client.SendAsync(request2);
            var content = await response.Content.ReadAsStringAsync();

            if (!response.IsSuccessStatusCode || string.IsNullOrWhiteSpace(content))
                throw new Exception($"Не вдалося видалити список: {content}");

        }

        // видалення книги зі списку
        public async Task DeleteBookFromListAsync(int listId, string isbn)
        {
            var url = $"booklistentry?listid=eq.{listId}&isbn=eq.{isbn}";
            var request = new HttpRequestMessage(HttpMethod.Delete, url);

            var response = await _client.SendAsync(request);
            if (!response.IsSuccessStatusCode)
            {
                throw new Exception($"Не вдалося видалити книгу:
{response.StatusCode}");
            }
        }

        // ---------- форма оцінка, рейтинг, коментарі ----------
        public async Task<double> GetAverageBookRatingAsync(string isbn)
        {
            var response = await
_client.GetAsync($"bookrating?select=rating&isbn=eq.{isbn}");

            if (!response.IsSuccessStatusCode)
```

```
                return 0;
```



```
            var json = await response.Content.ReadAsStringAsync();
            var ratings = JsonConvert.DeserializeObject<List<BookRating>>(json);

            return ratings != null && ratings.Count > 0
                ? ratings.Average(r => r.rating)
                : 0;
        }

        public async Task AddBookRatingAsync(BookRating rating)
        {
            var payload = new[]
            {
                new
                {
                    isbn = rating.isbn,
                    id_user = rating.id_user,
                    rating = rating.rating,
                    ratedat = rating.ratedat
                }
            };
            var json = JsonConvert.SerializeObject(payload);
            var content = new StringContent(json, Encoding.UTF8, "application/json");

            var response = await _client.PostAsync("bookrating", content);
            var responseText = await response.Content.ReadAsStringAsync();

            if (!response.IsSuccessStatusCode)
            {
                throw new Exception($"Помилка додавання оцінки: {response.StatusCode}
- {responseText}");
            }
        }

        public async Task AddBookCommentAsync(BookComment comment)
        {
            var payload = new[]
            {
                new
                {
                    isbn = comment.isbn,
                    id_user = comment.id_user,
                    comment = comment.comment,
                    commentdate = comment.commentdate
                }
            };
            var json = JsonConvert.SerializeObject(payload);
            var content = new StringContent(json, Encoding.UTF8, "application/json");
            var response = await _client.PostAsync("bookcomment", content);
            var responseText = await response.Content.ReadAsStringAsync();

            if (!response.IsSuccessStatusCode)
            {
                throw new Exception($"Помилка додавання коментаря:
{response.StatusCode} - {responseText}");
            }
        }

        public async Task AddBookToListAsync(int listid, string isbn)
        {
            var payload = new[]
```

```
            {
                new
                {
                    listid = listid,
                    isbn = isbn,
                    addeddate = DateTime.UtcNow
                }
            };
            var json = JsonConvert.SerializeObject(payload);
            var content = new StringContent(json, Encoding.UTF8, "application/json");

            var response = await _client.PostAsync("booklistentry", content);
            var responseText = await response.Content.ReadAsStringAsync();

            if (!response.IsSuccessStatusCode)
            {
                throw new Exception($"Помилка додавання до списку:
{response.StatusCode} - {responseText}");
            }
        }
    }
}
```

ДОДАТОК В

Діаграма прецедентів користувачів системи

Сторінок - 1

Київ – 2025

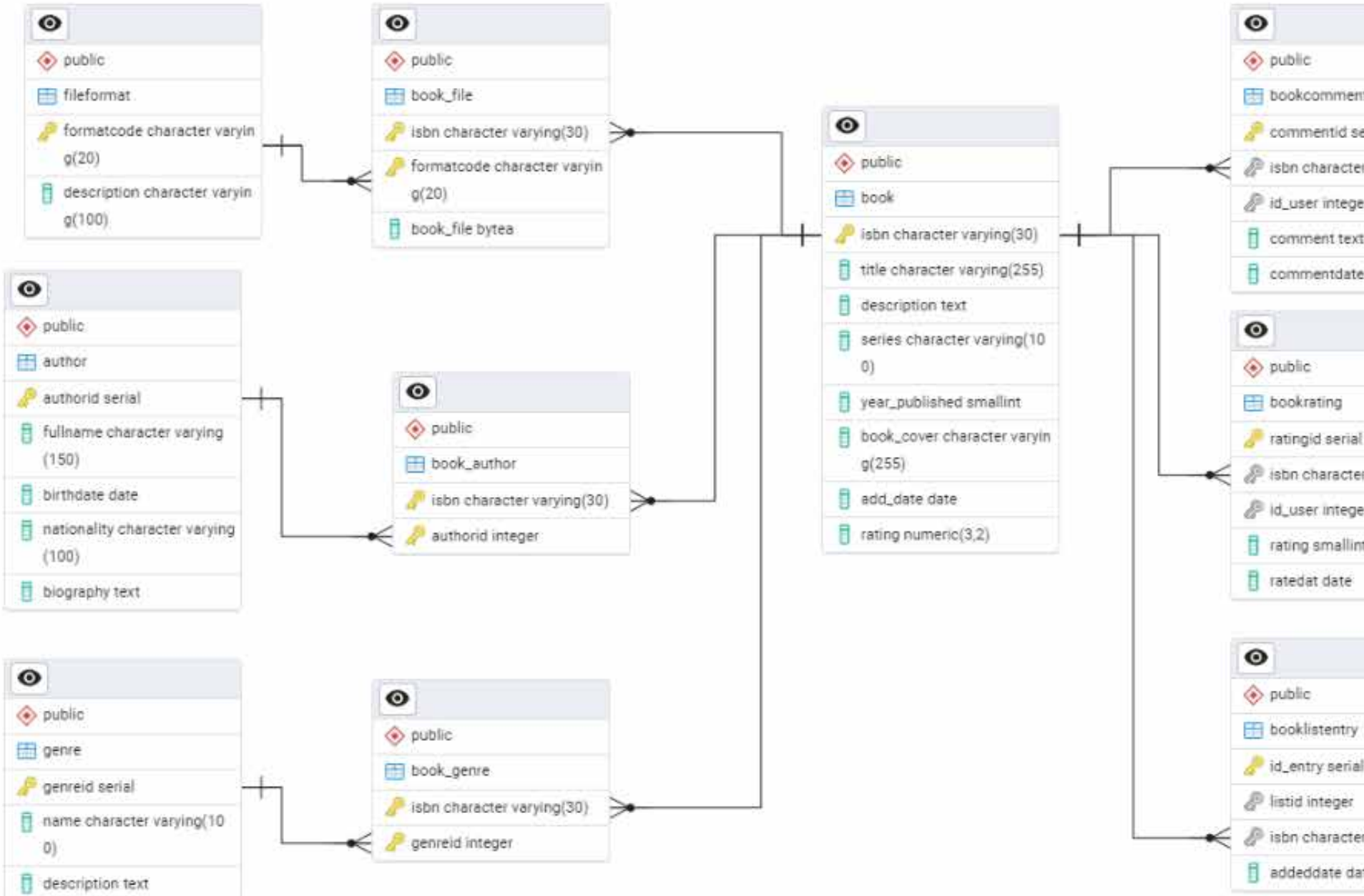

Редагування профілю
Перегляд каталогу книг
Пошук книг за параметрами
Перегляд деталей книги
Читання книги
Оцінювання книги
Написання відгуку
Керування власними списками
Завантаження файлу книги
Додавання нової книги
Редагування інформації про книгу
Видалення книги з бібліотеки
Керування книжковим фондом
<<include>>
Створення нового списку
Додавання книги до списку
Видалення книги з списку
Видалення особистого
Перегляд списку користувачів
Видалення користувачів
Керування каталогом
Ведення звітності
Управління системними налаштуваннями
Адміністр
истувач

ДОДАТОК Г

Створена у середовищі pgAdmin модель бази

Сторінок - 1

Київ – 2025


public
fileformat
formatcode character varying(20)
description character varying(100)
public
book_file
isbn character varying(30)
formatcode character varying(20)
book_file bytea
public
book
isbn character varying(30)
title character varying(255)
description text
series character varying(100)
year_published smallint
book_cover character varying(255)
add_date date
rating numeric(3,2)
public
bookcomment
commentid se
isbn character
id_user intege
comment text
commentdate
public
author
authorid serial
fullname character varying(150)
birthdate date
nationality character varying(100)
biography text
public
book_author
isbn character varying(30)
authorid integer
public
bookrating
ratingid serial
isbn character
id_user intege
rating smallint
ratedat date
public
genre
genreid serial
name character varying(100)
description text
public
book_genre
isbn character varying(30)
genreid integer
public
booklistentry
id_entry serial
listid integer
isbn character
addeddate da