

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ЗАКЛАД  
„ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ТАРАСА ШЕВЧЕНКА”

Навчально-науковий інститут фізики, математики та інформаційних  
технологій

Кафедра інформаційних технологій та систем

Залеський Олександр Володимирович

**“Проектування та розробка пошукового додатку «Бібліотека» з  
системою рекомендацій книг”.**

Бакалаврська робота  
за напрямом підготовки 121 Інженерія програмного забезпечення

Особистий підпис – \_\_\_\_\_

Науковий керівник – \_\_\_\_\_  
(підпис)

доцент кафедри ІТС  
М.А. СЕМЕНОВ  
(посада, науковий ступінь,  
наукове звання, ініціали, прізвище)

Зав. кафедри – \_\_\_\_\_  
(підпис)

зав. кафедри ІТС, кандидат  
педагогічних наук, доцент,  
М.А. Семенов  
(посада, науковий ступінь,  
наукове звання, ініціали, прізвище)

ПОЛТАВА – 2025

Міністерство освіти і науки України  
Державний заклад „Луганський національний університет  
імені Тараса Шевченка”

Факультет (інститут)

Навчально-науковий інститут фізики,  
математики та інформаційних технологій

Кафедра, циклова комісія

Інформаційних технологій та систем

Освітній ступень

Бакалавр

Напрямок підготовки (спеціальність)

F2 «Інженерія програмного забезпечення»

(код, назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри ІТС

М.А. Семенов

(підпис)

(ініціали, прізвище)

“ ” 2025 р.

**ЗАВДАННЯ  
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Залеського Олександра Володимировича

(прізвище, ім'я, по батькові )

**1. Тема проекту (роботи)** Проектування та розробка пошукового додатку  
«Бібліотека» з системою рекомендацій книг

Керівник кваліфікаційної роботи

Семенов М.А

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по університету

Від“ ” 2025 року №

**2. Строк подання студентом проекту (роботи)**

**3. Вихідні дані до роботи (проекту)** у результаті виконання роботи  
повинно бути розроблено додаток «Бібліотека»

(визначаються кількісні або (та) якісні показники, яким повинен відповідати об'єкт розробки)

**4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно  
розробити)** АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.

ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

ПРОЕКТУВАННЯ ТА КОНСТРУЮВАННЯ ДОДАТКА.

(визначаються назви розділів або (та) перелік питань, які повинні увійти до тексту ПЗ)

**5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)**

**6. Консультанти розділів проекту (роботи)**

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання „\_\_\_\_\_” \_\_\_\_\_ 2025 р.

**КАЛЕНДАРНИЙ ПЛАН**

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
	Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника.	До 15 жовтня	
	Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно-експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень.	Другий тиждень листопада (10 листопада )	
	Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником.	До 15 грудня	
	Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання.	До 28 січня	
	Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи.	Перший тиждень березня	
	Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації.	До 31 березня	
	Попередній захист роботи на кафедрі	квітень	
	Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії	За 10 днів до державної атестації	
	Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом.	За 5 днів до державної атестації	

**Студент**

\_\_\_\_\_

підпис

**О.В. Залеський**

\_\_\_\_\_

(ініціали, прізвище)

**Керівник проекту (роботи)**

\_\_\_\_\_

підпис

**М.А. Семенов**

## АНОТАЦІЯ

**ЗАЛЄСЬКИЙ О.В**

**Тема:** Проєктування додатку для онлайн бібліотеки.

**Спеціальність:** 121 "Інженерія програмного забезпечення".

**Установа:** ДЗ ЛНУ імені Тараса Шевченка, 2023р.

**Кваліфікаційна робота містить:** 68 стор., 10 рис., 6 джерел, 3 додатка включаючи окремий файл.

**Мета роботи** – вивчення та аналіз процесу проєктування програмного забезпечення для онлайн бібліотеки

**Методи дослідження:** *теоретичні:* аналіз науково-технічних джерел з проблем дослідження; *емпіричні:* порівняний аналіз можливостей засобів онлайн бібліотеки; *експериментальні:* тестування розробленого додатку.

Результати роботи. виконано основні етапи роботи які включають в себе детальний огляд актуальних аспектів, визначення цілей проєкту, встановлення вимог користувачів та технічного завдання, обґрунтування методів проєктування та архітектурних рішень.

**Ключові слова:** проєктування, розробка, знаходження результативних підходів, дослідження області, встановлення пріоритетів, визначення та аналіз вимог.

## **ABSTRACT**

Zalieskyi O.V.

**Theme:** Design of an application for an online library.

**Speciality:** 121 "Software Engineering"

**Institution:** Luhansk Taras Shevchenko National University, 2023.

Diploma work contains: 68 pages, 10 figures, 6 sources, 3 appendices including a separate file.

The aim of the work is to study and analyze the software design process for an online library system.

**Research methods:**

Theoretical: analysis of scientific and technical sources related to the topic;

Empirical: comparative analysis of the capabilities of various online library tools;

Experimental: testing of the developed application.

**Results:**

The main stages of the work have been completed, including a detailed review of relevant aspects, definition of project goals, identification of user and technical requirements, and justification of design methods and architectural solutions.

**Keywords:** design, development, effective approaches, domain research, priority setting, requirements identification and analysis.

ІТС.ПІ4.0721-01-ВІ

ВІДОМІСТЬ ПРОЕКТУ. РОЗРОБКА ОНЛАЙН БІБЛІОТЕКИ ДЛЯ  
ЗБЕРІГАННЯ ТА ОБРОБКИ ЛІТЕРАТУРНИХ РЕСУРСІВ.

Позначення	Найменування	Кількість прим/стор	Місцезнаходження / Примітка
Документація проекту			
ІТС.ПІ4.0721- 02-ТЗ	Розробка онлайн бібліотеки. Технічне завдання.	1/6	Формат А4
ІТС.ПІ4.0721- 03-ПЗ	Розробка онлайн бібліотеки. Пояснювальна записка.	1/55	Формат А4
ІТС.ПІ4.0721- 04-КК	Розробка онлайн бібліотеки. Керівництво користувача.	1/6	Формат А4
ІТС.ПІ4.0721- 05-МТ	Розробка онлайн бібліотеки. Програма та методика тестування.	1/4	Формат А4

**Міністерство освіти і науки України**  
**Державний заклад «Луганський національний університет**  
**імені Тараса Шевченка»**  
Факультет (інститут) Навчально-науковий інститут фізики, математики та  
інформаційних технологій  
(повна назва)  
Кафедра Кафедра інформаційних технологій та систем  
(повна назва)

## **ТЕХНІЧНЕ ЗАВДАННЯ**

на виконання програмної розробки (ПР):

### **«ПРОЕКТУВАННЯ ТА РОЗРОБКА ПОШУКОВОГО ДОДАТКУ «БІБЛІОТЕКА» З СИСТЕМОЮ РЕКОМЕНДАЦІЙ КНИГ»**

#### **ПОГОДЖЕНО**

Керівник кваліфікаційної роботи

Семенов М.А.

“ ” 2025\_\_р

#### **ВИКОНАВЕЦЬ**

Студент групи 4ІПЗ

Залеський О.В.

“21” Травня 2025р

# **ЗМІСТ ТЕХНІЧНОГО ЗАВДАННЯ**

## **ЗМІСТ**

<b>ВСТУП .....</b>	<b>3</b>
<b>1 ЗАГАЛЬНІ ВІДОМОСТІ .....</b>	<b>9</b>
1.1 Технічне завдання до «онлайн бібліотеки» .....	9
<b>2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ .....</b>	<b>11</b>
<b>3 ХАРАКТЕРИСТИКА ОБ'ЄКТІВ АВТОМАТИЗАЦІЇ .....</b>	<b>13</b>
<b>4 ВИМОГИ ДО СИСТЕМИ .....</b>	<b>14</b>
4.1 Вимоги до структури і функціонування системи: .....	14
4.2 Вимоги до персоналу: .....	14
4.3 Показники призначення: .....	15
4.5 Підсистема "Каталог Книг": .....	15
4.6 Підсистема "Кошик": .....	15
4.7 Підсистема "Оформлення замовлення": .....	15
4.8 Підсистема "Адміністрування": .....	16
4.9 Перелік можливих відмов: .....	16
4.10 Критерії відмов: .....	16
4.11 Вимоги до видів забезпечення: .....	17
<b>5 СКЛАД І ЗМІСТ РОБІТ ЗІ СТВОРЕННЯМ СИСТЕМИ .....</b>	<b>18</b>
5.1 Перелік стадій та етапів робіт: .....	18
5.2 Терміни виконання: .....	18
5.3 Програма забезпечення надійності: .....	19
5.4 Програма метрологічного забезпечення: .....	19
<b>6 ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ ІС .....</b>	<b>20</b>
6.1 Вимоги до випробувань системи: .....	20
6.2 Загальні вимоги до приймання робіт за стадіями: .....	20
<b>7 ВИМОГИ ДО СКЛАДУ ТА ЗМІСТУ РОБІТ З ПІДГОТОВКИ ОБ'ЄКТА АВТОМАТИЗАЦІЇ ДО ВВЕДЕННЯ В ДІЮ.....</b>	<b>22</b>
<b>8 ВИМОГИ ДО ДОКУМЕНТУВАННЯ .....</b>	<b>24</b>
8.1 Перелік документів, які підлягають розробці для Онлайн Бібліотеки: .....	24
8.2 Перелік документів на машинних носіях: .....	24
<b>9 ДЖЕРЕЛА РОЗРОБКИ .....</b>	<b>25</b>



## **ВСТУП**

### **Спільна мета створення ІС:**

Створення та управління онлайн-бібліотекою, включаючи збір, зберігання, організацію та надання доступу до книжок, журналів, статей та іншої літератури в електронному форматі.

Перелік об'єктів, на яких передбачається використання системи:

Користувачі (читачі), які мають доступ до бібліотеки через веб-інтерфейс або мобільний додаток.

Адміністратори, які керують вмістом бібліотеки, включаючи додавання, редагування та видалення книг, управління користувачами тощо.

Найменування і необхідні значення показників об'єкта, які повинні бути досягнуті при впровадженні ІС:

Технічні показники:

Доступність 24/7 системи.

Захист від несанкціонованого доступу та збереження конфіденційності користувачів.

Швидкість завантаження та доступу до контенту.

Технологічні показники:

Підтримка різних форматів файлів для електронних книг (PDF, EPUB, MOBI тощо).

Можливість пошуку та сортування книг за різними категоріями (жанри, автори, рейтинги тощо).

Виробничо-економічні показники:

Масштабованість системи для забезпечення росту кількості користувачів та контенту.

Ефективність використання ресурсів сервера та мінімізація витрат на зберігання та підтримку системи.

Це загальні параметри, які можна розширити або деталізувати залежно від специфічних вимог та потреб користувачів бібліотеки.

### **Склад підсистем та функціональних завдань:**

Підсистема управління контентом (CMS), яка включає інструменти для додавання та редагування книг, описів, зображень, відгуків клієнтів та іншої інформації на сайті.

Підсистема аналітики та звітності, яка дозволяє аналізувати та відстежувати читачів, відвідуваність додатку, популярність книг та іншу статистику.

Підсистема авторизації та безпеки, яка забезпечує безпеку додатку та даних клієнтів, а також авторизацію користувачів та управління правами доступу.

Підсистема обробки повідомлень та клієнтської підтримки, яка дозволяє клієнтам зв'язуватися з підтримкою, отримувати відповіді на запитання та сповіщення .

### **Вимоги до інформаційної бази, математичного та програмного забезпечення, комплексу технічних засобів:**

Інформаційна база повинна забезпечувати можливість збереження та обробки інформації про книги, клієнтів, замовлення, оплати та доставки, а також статистичну звітність та аналіз продажів.

Математичне та програмне забезпечення повинно забезпечувати швидку та безперебійну роботу системи, а також високий рівень захисту даних від несанкціонованого доступу та атак хакерів.

Комплекс технічних засобів повинен забезпечувати високу швидкість передачі даних, безперебійну роботу системи та захист від вірусів та шкідливих програм.

### **Загальні вимоги до системи:**

Система повинна бути легкою у використанні та навігації, зручною для користувачів.

Система повинна забезпечувати можливість швидкого та зручного пошуку книг та інформації про них.

Система повинна забезпечувати безперебійний доступ до інтернет-додатку та можливість здійснення оплати та оформлення замовлень книг онлайн.

Система повинна забезпечувати захист персональних даних користувачів та захист від шахрайства та фішингу.

### **Перелік завдань створення системи і виконавців:**

- Розробка дизайну та інтерфейсу сайту - веб-дизайнер
- Розробка програмного забезпечення - програміст
- Розробка бази даних - бази даних адміністратор
- Налагодження роботи додатку та програмного забезпечення - системний - адміністратор

Розробка тестових сценаріїв та тестування сайту та програмного забезпечення

При розробці програмного проєкту для онлайн бібліотеки необхідно враховувати наступні вимоги:

### **Функціональні вимоги:**

- Реєстрація та авторизація користувачів;
- Пошук книг з можливістю фільтрації за ціною, категорією, автором тощо;
- Додавання книг в кошик;
- Оформлення замовлення з можливістю введення адреси доставки та вибору способу оплати;

- Можливість переглянути історію замовлень.
- Нефункціональні вимоги:
- Безпека та захист персональних даних користувачів;
- Висока швидкість роботи сайту;
- Інтуїтивно зрозумілий та простий інтерфейс користувача;
- Сумісність з різними браузерами та пристроями.

### **Вимоги до розробки:**

- Використання сучасних технологій розробки програмного забезпечення;
- Використання баз даних для зберігання інформації про книги, користувачів та їх замовлення;
- Розробка захисту від зломів та хакерських атак;
- Розробка бекенду та фронтенду сайту.

### **Вимоги до тестування:**

- Проведення регулярного тестування додатку для виявлення та виправлення помилок;
- Проведення тестування на різних браузерах та пристроях;
- Проведення тестування на швидкість роботи додатку та оптимізацію його роботи.
- Вимоги до супроводу:
- Підтримка користувачів та вирішення їх проблем;
- Розробка нових функцій та можливостей сайту;
- Постійне оновлення програмного забезпечення сайту.

### **Етапи створення системи та терміни їх виконання:**

Етап 1: Аналіз вимог та планування проекту. Термін виконання - 1 місяць.

Етап 2: Розробка дизайну та інтерфейсу. Термін виконання - 2 місяці.

Етап 3: Розробка бази даних та програмного забезпечення. Термін виконання - 3 місяці.

Етап 4: Тестування та налагодження системи. Термін виконання - 1 місяць.

Етап 5: Впровадження та підтримка системи. Термін виконання - безстроково.

### **Попередній розрахунок витрат та рівень економічної ефективності:**

- Загальні витрати на проект складаються з витрат на розробку, тестування, впровадження та підтримку системи.
- Вартість розробки системи: 100 000 грн.
- Вартість тестування та налагодження системи: 50 000 грн.
- Вартість впровадження системи: 20 000 грн.
- Вартість підтримки системи: 10 000 грн. на рік.
- Рівень економічної ефективності проекту може бути визначений з урахуванням доходів від продажу товарів через інтернет-магазин та економії витрат на зарплати працівників фізичного магазину.
- Розрахунок прибутку від продажу товарів залежить від обсягу продажів та маржинальної прибутковості. Продажі можуть бути прогнозовані на основі аналізу ринку та конкурентів.
- Економія витрат на зарплати працівників фізичного магазину може бути оцінена на основі середньої зарплати фахівців, які працюють у фізичному магазині.
- Попередній розрахунок витрат та рівень економічної ефективності:

- Загальні витрати на проект складаються з витрат на розробку, тестування, впровадження та підтримку системи.
- Вартість розробки системи: 100 000 грн.
- Вартість тестування та налагодження системи: 50 000 грн.
- Вартість впровадження системи: 20 000 грн.
- Вартість підтримки системи: 10 000 грн. на рік.
- Рівень економічної ефективності проекту може бути визначений з урахуванням доходів від продажу товарів через інтернет-магазин та економії витрат на зарплати працівників фізичного магазину.
- Розрахунок прибутку від продажу товарів залежить від обсягу продажів та маржинальної прибутковості. Продажі можуть бути прогнозовані на основі аналізу ринку та конкурентів.
- Економія витрат на зарплати працівників фізичного магазину може бути оцінена на основі середньої зарплати фахівців, які працюють у фізичному магазині.

## **1 ЗАГАЛЬНІ ВІДОМОСТІ**

### **1.1 Технічне завдання до « додатку онлайн бібліотеки»**

1.Повна назва системи та її умовне позначення: Назва: Онлайн Бібліотека Умовне позначення: ВІВ-UA

2.Шифр теми або шифр (номер) договору: Шифр договору: ТМ-2023-01

3.Найменування підприємств розробника і замовника системи, їх реквізити:  
Розробник: ТОВ "Бібліотека Юа" Адреса: вул. Козаченко, 1, м. Київ, 01001, Україна Ідентифікаційний код: 1234567890 Замовник: ФОП Літвінов І.І. Адреса: вул. Молодіжна, 22\7, м. Золоте, 79000, Україна Ідентифікаційний код: 0987654321

4. Перелік документів, на підставі яких створюється ІС:

- Технічне завдання
- Функціональна специфікація
- Дизайн-проект
- Технічний проект
- Програмний код
- Тестова документація
- Інструкції користувача

5.Планові терміни початку і закінчення робіт: Початок робіт: 01.04.2023  
Закінчення робіт: 01.09.2023

6.Відомості про джерела і порядок фінансування робіт: Джерела фінансування: ФОП Літвінов І.І. Порядок фінансування: за договором

7.Порядок оформлення і подання замовнику результатів робіт зі створення системи, її частин та окремих засобів: Результати робіт повинні бути подані замовнику у вигляді:

7.1 Програмне забезпечення: - Встановлюваний пакет програмного забезпечення має бути переданий замовнику на CD або DVD диску. - Замовник повинен отримати інструкції щодо процедури інсталяції та налаштування програмного забезпечення.

7.2. База даних: - База даних має бути передана замовнику на CD або DVD диску. - Замовник повинен отримати інструкції щодо процедури імпорту даних до бази даних та налаштування з'єднання з базою даних.

7.3. Документація: - Документація має бути надіслана замовнику в електронному вигляді на CD або DVD диску. - Документація має містити опис програмного забезпечення, структуру бази даних, опис інтерфейсу користувача та іншу необхідну інформацію.

7.4. Інструкції користувача: - Інструкції користувача мають бути передані замовнику в електронному вигляді на CD або DVD диску. - Інструкції користувача повинні містити інформацію про використання програмного забезпечення, в тому числі про його функціональні можливості, налаштування та використання інтерфейсу користувача.

7.5. Передача результатів робіт: - Результати робіт повинні бути передані замовнику у відповідному вигляді не пізніше ніж за 10 днів до дати прийняття в експлуатацію. - Замовник повинен підтвердити отримання результатів робіт відповідним підписом і датою.



## **2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ**

### **1. Вид діяльності, яка автоматизується:**

Розробка ОНЛАЙН БІБЛІОТЕКИ, що дозволяє здійснювати покупки та замовлення книг зручно та швидко через мережу Інтернет.

### **2. Перелік об'єктів, на яких передбачається використання системи:**

- Система буде використовуватися клієнтами для здійснення покупок книг через мережу Інтернет.

- Система буде використовуватися адміністраторами для керування товарами, замовленнями, клієнтами, платежами та доставкою.

### **3. Найменування і необхідні значення технічних, технологічних, виробничо-економічних та інших показників об'єкта, які повинні бути досягнуті при впровадженні ІС:**

Технічні показники:

- Дадоток повинен працювати на платформі електронної комерції з можливістю керування контентом, підтримкою мобільних пристроїв та забезпечення безпеки даних.

- Дадоток повинен мати вбудовану систему керування запасами, яка автоматично відстежує кількість товару на складі та повідомляє адміністратора про необхідність поповнення запасів.

- Дадоток повинен мати систему оплати з можливістю різних способів оплати (банківська карта, електронний гаманець тощо) та забезпеченням безпеки платежів.

- Дадоток повинен мати систему доставки, яка автоматично вибирає оптимальний спосіб доставки товару та обраховує вартість доставки.

### **5. Технологічні показники:**

- Дадоток повинен бути розроблений з використанням сучасних технологій програмування та відповідати вимогам сучасних веб-стандартів.
- Дадоток повинен бути оптимізований для швидкої роботи на різних пристроях з різною швидкістю Інтернет-з'єднання.
- Дадоток повинен бути підтримувати роботу з базами даних з використанням сучасних технологій управління даними та забезпечення безпеки даних.

#### 6. Виробничо-економічні показники:

- Дадоток повинен забезпечувати зручний та швидкий процес здійснення покупок та замовлень, що забезпечує високий рівень задоволеності клієнтів та збільшення прибутку.
- Дадоток повинен забезпечувати ефективне керування запасами та оптимізацію процесу доставки товарів, що дозволяє зменшити витрати на складське господарство та доставку.
- Дадоток повинен забезпечувати високий рівень безпеки даних клієнтів та операцій з оплатою та доставкою товарів, що дозволяє зберегти репутацію компанії та запобігти втратам.

#### 7. Інші показники:

- Дадоток повинен бути доступним для користувачів з будь-якої точки світу та забезпечувати можливість здійснення покупок та замовлень у зручний для них час.
- Дадоток повинен мати зручний та інтуїтивно зрозумілий інтерфейс, що дозволяє клієнтам легко здійснювати покупки та замовлення без додаткових зусиль та непотрібного навантаження на мозок.

### **3 ХАРАКТЕРИСТИКА ОБ'ЄКТІВ АВТОМАТИЗАЦІЇ**

1.Об'єкт автоматизації: Онлайн Бібліотека, що надає можливість замовлення та оплати товарів онлайн.

#### **2.Умови експлуатації:**

Дадоток має працювати на постійній основі, 24 години на добу, 7 днів на тиждень.

Дадоток має бути доступний з будь-якого пристрою, що має підключення до Інтернету, тому повинен бути оптимізований для роботи на різних пристроях, таких як комп'ютери, планшети та мобільні пристрої.

Дадоток повинен мати захист від хакерських атак та інших видів кіберзлочинності.

Дадоток повинен підтримувати можливість швидкої та безпечної оплати товарів за допомогою кредитних карток та інших електронних засобів оплати.

Навколишнє середовище має бути забезпечене стабільним та надійним зв'язком з Інтернетом, щоб забезпечити швидку та безперебійну роботу магазину.

## **4 ВИМОГИ ДО СИСТЕМИ**

### **4.1 Вимоги до структури і функціонування системи:**

1. Система повинна містити такі підсистеми: система керування замовленнями, система управління клієнтськими даними, система управління запасами та система оплати.
2. Рівні ієрархії повинні бути наступними: вищий рівень - система управління, нижчий рівень - підсистеми.
3. Система має мати централізований підхід до керування та управління всіма підсистемами.
4. Інформаційний обмін між підсистемами повинен бути автоматизований та забезпечувати швидкий доступ до даних для кожної підсистеми.
5. Режим функціонування системи повинен бути 24/7.
6. Система повинна мати можливість інтеграції з іншими суміжними системами, такими як системи доставки та логістики.
7. Перспективи розвитку системи повинні передбачати можливість розширення функціоналу, додавання нових функцій та збільшення кількості продуктів в магазині.

### **4.2 Вимоги до персоналу:**

1. Кількість персоналу: не менше 10 осіб.
2. Кваліфікація користувачів: середній рівень знань і вмінь в роботі з комп'ютером та Інтернетом, знання основ електронної комерції.
3. Режим роботи: 24/7 доступ до системи.
4. Порядок підготовки: користувачі повинні отримати навчання з використання системи та основ електронної комерції.

#### **4.3 Показники призначення:**

1.Ступінь пристосованості системи до змін процесів управління: система повинна бути гнучкою та легко змінюватись для адаптації до змін у внутрішніх та зовнішніх процесах.

2.Ступінь пристосованості системи до значень параметрів: система повинна бути спроможною обробляти велику кількість параметрів, що пов'язані з товаром, замовленнями та оплатою. Також повинна бути забезпечена можливість швидкої зміни цін та наявності товару.

Вимоги до функцій ОНЛАЙН БІБЛІОТЕКИ можуть бути наступними:

#### **4.5 Підсистема "Каталог Книг":**

- Автоматизація процесу додавання, редагування та видалення книг з каталогу
- Відображення характеристик товарів та їх цін
- Можливість сортування товарів за різними параметрами
- Можливість пошуку товарів за назвою або описом
- Керування запасами товарів на складі

#### **4.6 Підсистема "Кошик":**

- Можливість додавання товарів в кошик та зміна їх кількості
- Розрахунок вартості замовлення з урахуванням цін на товари та вартості доставки
- Можливість видалення товарів з кошика
- Реалізація опції "зберегти кошик" для користувачів, які не завершили оформлення замовлення

#### **4.7 Підсистема "Оформлення замовлення":**

- Збір інформації про замовника (ім'я, адреса, телефон, електронна пошта)
- Вибір способу доставки та способу оплати
- Підтвердження замовлення користувачем

- Реалізація опції "зберегти замовлення" для користувачів, які хочуть завершити замовлення пізніше

#### **4.8 Підсистема "Адміністрування":**

- Можливість додавання, редагування та видалення акаунтів користувачів та їх прав
- Аналіз статистики продажів товарів
- Керування запасами книг на складі
- Можливість налаштування вартості доставки та інших параметрів магазину

Часовий регламент та вимоги до якості реалізації кожної функції можуть бути визначені окремо залежно від обсягу проекту та ресурсів, які доступні для розробки.

#### **4.9 Перелік можливих відмов:**

1. Недоступність сайту для користувачів
2. Пошкодження бази даних
3. Втрата даних
4. Помилки в роботі системи оплати
5. Проблеми з доставкою товарів

#### **4.10 Критерії відмов:**

1. Час відновлення роботи сайту після відмови
2. Кількість відвідувачів сайту, які не змогли здійснити покупку через відмову
3. Кількість втрачених замовлень і продажів через відмову системи
4. Час відновлення бази даних після відмови
5. Рівень надійності системи оплати
6. Час доставки товарів до клієнтів

#### **4.11 Вимоги до видів забезпечення:**

Математичне забезпечення повинно містити достатній склад математичних моделей методів, які будуть використовуватися для розв'язання завдань автоматизації. Вимоги до математичного забезпечення повинні включати деталізований перелік математичних методів та алгоритмів, що використовуються для роботи системи, та область їхнього застосування.

Інформаційне забезпечення повинно містити склад, структуру та організацію даних, обмін даними між компонентами системи, інформаційну сумісність із суміжними системами, які використовуються класифікатори, систему управління базами даних (СУБД), контроль даних і ведення інформаційних масивів, процедури надання юридичної сили вихідних документів. Також повинні бути вказані вимоги до забезпечення безпеки даних та забезпечення захисту від несанкціонованого доступу до інформації.

Надійність: програмні засоби повинні бути стійкими до відмов та помилок, що забезпечується за допомогою відповідних алгоритмів контролю та захисту від помилок.

Ефективність: програмні засоби повинні працювати швидко і ефективно з великою кількістю даних та високою швидкістю обробки.

Масштабованість: програмні засоби повинні забезпечувати можливість масштабування функціональності та продуктивності з урахуванням зростаючих потреб користувачів та обсягів даних.

Сумісність: програмні засоби повинні бути сумісними з іншими програмними засобами, що використовуються в системі, та стандартами, що застосовуються в галузі.

Безпека: програмні засоби повинні забезпечувати захист від несанкціонованого доступу, вірусів, зловмисного програмного забезпечення та інших загроз безпеці.

## **5 СКЛАД І ЗМІСТ РОБІТ ЗІ СТВОРЕННЯМ СИСТЕМИ**

### **5.1 Перелік стадій та етапів робіт:**

- 1.Розробка технічного завдання і визначення вимог до функціональності та якості системи.
- 2.Розробка архітектури та дизайну системи.
- 3.Розробка програмного забезпечення.
- 4.Встановлення та налаштування серверного та мережевого обладнання.
- 5.Тестування та налагодження системи.
- 6.Запуск та введення системи в експлуатацію.
- 7.Підтримка та розвиток системи.

### **5.2 Терміни виконання:**

Розробка технічного завдання і визначення вимог до функціональності та якості системи - 2 тижні.

Розробка архітектури та дизайну системи - 3 тижні.

Розробка програмного забезпечення - 10 тижнів.

Встановлення та налаштування серверного та мережевого обладнання - 1 тиждень.

Тестування та налагодження системи - 2 тижні.

Запуск та введення системи в експлуатацію - 1 тиждень.

Підтримка та розвиток системи - постійно.

Склад організацій – виконавців робіт:

-Керівник проекту.

-Аналітики та програмісти.

-Адміністратори серверів та мережі.

-Менеджери з продажу.



-Фахівці з підтримки користувачів.

-Вид і порядок експертизи технічної документації:

-Перевірка повноти та коректності технічної документації (керівництва з встановлення, інструкції з експлуатації, каталоги запчастин, технічні описи тощо).

-Перевірка відповідності технічної документації стандартам та нормативним вимогам.

-Визначення наявності технічних ризиків та способів їх усунення.

-Визначення відповідності технічної документації вимогам безпеки та екології.

### **5.3 Програма забезпечення надійності:**

Перевірка продукту на відповідність вимогам безпеки та екології.

Визначення частоти виникнення небажаних наслідків та їх наслідків.

Визначення чутливості продукту до змін зовнішніх умов (температура, вологість, тиск, вібрація тощо).

Визначення дійсного терміну служби продукту та його вузлів та деталей.

Розробка методів перевірки надійності продукту.

### **5.4 Програма метрологічного забезпечення:**

Визначення метрологічних вимог до продукту та засобів вимірювання.

Перевірка точності засобів вимірювання, їх метрологічного забезпечення та калібрування.

Розробка технології вимірювання параметрів продукту.

Контроль за процесами вимірювання та перевірки точності продукту.

Оцінка впливу некоректної метрологічної підтримки на точність та надійність продукту.

## **6 ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ ІС**

### **6.1 Вимоги до випробувань системи:**

Випробування повинні охоплювати всі функціональні можливості системи, включаючи функції додавання товарів до кошика, оформлення замовлень, оплату, доставку та повернення товару, а також збір і аналіз даних про клієнтів і замовлення.

Методи випробувань повинні бути заздалегідь визначені та детально описані в технічній документації.

Випробування повинні проводитись як на стороні клієнта, так і на стороні сервера, щоб переконатись, що система працює коректно на різних пристроях і з різною швидкістю Інтернет-з'єднання.

Випробування повинні включати тестування на захист від хакерських атак, злому паролів, витоку конфіденційної інформації та інших потенційних загроз.

### **6.2 Загальні вимоги до приймання робіт за стадіями:**

Кожен етап робіт повинен бути затверджений замовником і описаний в детальному плані робіт.

При виконанні кожного етапу робіт повинні бути виконані всі відповідні вимоги до якості та документації.

Перед переходом до наступного етапу робіт, поточний етап повинен бути повністю завершений і прийнятий замовником.

При прийнятті робіт замовником, повинен бути складений акт приймання-передачі робіт, який містить повний перелік виконаних робіт і результатів випробувань, а також підписи відповідальних осіб з обох сторін.

Статус приймальної комісії в Онлайн Бібліотеці може бути визначений як орган, який здійснює приймання робіт за стадіями і відповідальний за перевірку виконання всіх вимог до системи, програмного та технічного забезпечення.

Приймальна комісія повинна складатися з фахівців зі знанням відповідних технічних вимог, які мають достатній досвід роботи зі схожими проектами і можуть забезпечити адекватну оцінку виконаної роботи. Крім того, приймальна комісія повинна мати достатню повноваження для відхилення робіт, які не відповідають вимогам технічної документації і вимогам програмного та технічного забезпечення.

## **7 ВИМОГИ ДО СКЛАДУ ТА ЗМІСТУ РОБІТ З ПІДГОТОВКИ ОБ'ЄКТА АВТОМАТИЗАЦІЇ ДО ВВЕДЕННЯ В ДІЮ**

Перетворення вхідної інформації до машино-читасмого вигляду може включати обробку та форматування даних, отриманих від користувачів, щоб вони могли бути збережені та використані системою магазину.

Зміни в об'єкті автоматизації можуть включати розробку та впровадження нових функцій та можливостей для користувачів, а також оновлення та вдосконалення існуючих функцій, щоб підвищити їх ефективність та надійність. Наприклад, можуть бути додані нові методи оплати, вдосконалено функції пошуку та фільтрації товарів, оновлено дизайн та інтерфейс магазину для поліпшення користувацького досвіду тощо.

Терміни і порядок комплектування та навчання персоналу для інтернет-магазину побутової техніки:

### **Комплектування:**

Збір заявок на необхідне обладнання та інші матеріали для роботи інтернет-магазину.

Придбання та доставка обладнання та матеріалів.

Розміщення та підключення обладнання.

Навчання персоналу:

Розробка програми навчання з урахуванням функцій та особливостей інтернет-магазину.

Проведення навчання нового персоналу перед початком роботи.

Проведення регулярних навчань для всього персоналу з метою оновлення знань та вмінь з роботи в онлайн бібліотеці.

Терміни для комплектування можуть залежати від доступності обладнання та матеріалів на ринку та складності їх доставки. Терміни для навчання персоналу

можуть бути різними залежно від обсягу та складності навчальної програми. Однак, важливо встановити реалістичні терміни та стежити за їх виконанням для успішного запуску Онлайн Бібліотеки.

## **8 ВИМОГИ ДО ДОКУМЕНТУВАННЯ**

### **8.1 Перелік документів, які підлягають розробці для Онлайн Бібліотеки:**

- Технічне завдання.
- Проектна документація.
- Документація на програмне забезпечення та апаратні засоби.
- Документація на процеси та процедури роботи інтернет магазину.
- Інструкції користувача та технічні описи для кожного товару.
- Документація з питань безпеки та охорони праці.

### **8.2 Перелік документів на машинних носіях:**

- Проектна документація в електронному вигляді.
- Програмне забезпечення та драйвери для апаратних засобів.
- Конфігураційні файли налаштування робочих станцій та серверів.
- Резервні копії бази даних.
- Документація з питань безпеки та охорони праці в електронному вигляді.

## **9 ДЖЕРЕЛА РОЗРОБКИ**

Документи на підставі яких розробляється технічне завдання та система Онлайн Бібліотеки, можуть бути розроблені наступні документи та інформаційні матеріали:

1. Аналіз ринку схожих додатків і інтернет-торгівлі книжок, включаючи статистику продажів, тенденції розвитку, конкурентів, потенційних покупців та їх поведінку.
2. Вимоги до функціональності та характеристик додатку, які можуть бути зібрані від потенційних користувачів, партнерів та замовників.
3. Опис бізнес-процесів та функціональних вимог до програмного забезпечення магазину книг, таких як створення та управління продуктами, замовлення, оплата та доставка, зв'язок з користувачами та партнерами.
4. Вимоги до дизайну та інтерфейсу користувача, що включають розташування елементів, кольорову гаму, шрифти та інші аспекти.
5. Вимоги до інформаційної безпеки, які охоплюють захист персональних даних користувачів, захист від хакерських атак та забезпечення цілісності даних.
6. План тестування та випробування програмного забезпечення, який містить стратегію тестування, тестові сценарії та критерії успішності.
7. План впровадження та підтримки додатку, який охоплює, періодичність оновлення та підтримки програмного забезпечення, відповідальність за технічну підтримку та надання послуг користувачам.





**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**ДЗ «ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**  
**ІМЕНІ ТАРАСА ШЕВЧЕНКА»**

Навчально-науковий інститут математики та інформаційних технологій  
(назва факультету, інституту)  
Кафедра інформаційних технологій та систем  
(назва кафедри)

**Пояснювальна записка до кваліфікаційної роботи**  
за першим (бакалаврським) рівнем освіти

на тему:

**Проектування та розробка пошукового додатку «Бібліотека» з системою  
рекомендацій книг**

Виконав: здобувач вищої освіти 4 курсу спеціальності  
F2 «Інженерія програмного забезпечення»  
(шифр і назва напрямку підготовки, спеціальності)

\_\_\_\_\_ Олександр ЗАЛЄСЬКИЙ  
(прізвище та ініціали)

Керівник \_\_\_\_\_ Микола СЕМЕНОВ  
(прізвище та ініціали)

Полтава – 2025

## ЗМІСТ

ВСТУП .....	3
1. РОЗДІЛ 1. ХАРАКТЕРИСТИКА СИСТЕМИ, ТА АНАЛІЗ ВИМОГ .....	5
1.1 Призначення , область застосування .....	5
1.2 Визначення цілей .....	7
1.3 Аналіз Вимог .....	9
1.4 Вимоги до інтерфейсу користувача .....	10
1.5 Вимоги до апаратних, програмних та комунікаційних інтерфейсів .....	12
1.6 Вимоги до адаптації на місці .....	13
1.7 Функції продукту .....	14
1.8 Обмеження .....	15
2. РОЗДІЛ 2. ОГЛЯД МЕТОДІВ ПРОЄКТУВАННЯ ПРОГРАМНИХ СИСТЕМ .....	17
2.1. Огляд існуючих методів та основних принципів проєктування, архітектури програмного забезпечення, інженерних підходів і принципів керування ПЗ .....	17
2.2 Обґрунтування вибору методів та підходів до проєктування ПЗ .....	28
3. РОЗДІЛ 3. ОПИС ТА ОБҐРУНТУВАННЯ ПРОЄКТНИХ РІШЕНЬ ПО ПРОЄКТУВАННЮ ПРОГРАМНОЇ СИСТЕМИ .....	34
3.1 Структурні схеми системи (модулі системи та компоненти) .....	34
3.2 Опис Архітектурної системи .....	39
3.3 Структура бази даних , моделювання бази даних (інфологічна модель) .....	40
4. РОЗДІЛ 4. ПРАКТИЧНА РЕАЛІЗАЦІЯ (РОЗРОБКА БАЗИ ДАНИХ БІБЛІОТЕКИ) .....	44
4.1 Розробка бази даних програми .....	45
4.1.1 Концептуальне проєктування бази даних .....	45
4.1.2 Логічне проєктування баз даних .....	46
4.1.3 Фізичне проєктування бази даних .....	47
4.2 Проєктування структури програми та паттерни .....	48
4.3 Опис класів та методів застосування .....	51
4.4 Опис SQL-запитів до бази даних системи .....	52
5. РОЗДІЛ 5. ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ .....	59
5.1 Апаратні та програмні засоби створення та експлуатації програми .....	59
5.2 Інструкція користувача .....	59
5.3 Опис контрольних прикладів .....	60
ВИСНОВОК .....	62
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....	64
ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ (Окремий файл) .....	65
ДОДАТОК Б ЕКРАННІ ФОРМИ .....	66
ДОДАТОК В Фрагменти Лістингу .....	70

## ВСТУП

У сучасному цифровому епохальному світі, де доступ до інформації є ключовим аспектом розвитку, онлайн бібліотеки стали важливою складовою навчального та професійного простору. Проєктування та створення програмного забезпечення для онлайн бібліотек відкриває безмежні можливості для надання широкого спектру літературних творів, наукових публікацій та ресурсів у доступній та зручній формі.

**Мета даної бакалаврської роботи** полягає у вивченні та аналізі процесу проєктування програмного забезпечення для онлайн бібліотеки. Розглянувши теоретичні аспекти проєктування програм, а також проведення аналізу вимог, архітектури системи, процесу розробки, тестування та управління якістю, мета роботи - надати інсайти та рекомендації для створення ефективної та функціональної онлайн бібліотеки.

**Завдання** включають аналіз вимог, визначення технічного завдання, вибір оптимальних методів проєктування, огляд архітектури та реалізацію функціоналу.

**Аналіз потреб користувачів:** Вивчення вимог та очікувань цільової аудиторії від онлайн бібліотеки, врахування їхніх потреб для розробки користувацького інтерфейсу.

**Проєктування функціональності:** Визначення основних функцій та можливостей системи, створення сценаріїв взаємодії користувача з додатком.

**Вибір технологій:** Дослідження та вибір оптимальних технологій для реалізації функціональності додатку, зокрема баз даних, мов програмування та інтерфейсних рішень.

**Розробка та втілення концепції:** Створення прототипу онлайн бібліотеки, розробка програмного забезпечення, тестування та впровадження системи.

**Оцінка ефективності:** Проведення оцінки функціональності, користувацької зручності та загальної продуктивності онлайн бібліотеки.

**Цілі які ставляться на бакалаврський проєкт :**

*Розробка функціонального та ефективного онлайн інструменту для доступу до літературних ресурсів.*

*Дослідження та застосування сучасних методів проектування та програмування для створення продуктивної системи.*

*Забезпечення зручного та інтуїтивно зрозумілого інтерфейсу для користувачів.*

*Підвищення доступності до навчальних матеріалів та літературних творів за допомогою технологій.*

**У контексті навчання** дослідництва та практичного застосування технологій, створення онлайн бібліотек є важливим етапом у забезпеченні загального доступу до знань та інформації для широкого кола користувачів. Відповідно, ця робота буде зосереджена на висвітленні ключових аспектів проектування програмного забезпечення для створення ефективної та легко доступної онлайн бібліотеки.

Зрозуміння процесу створення онлайн бібліотеки та факторів, які впливають на їхню функціональність та успішну роботу, відіграє ключову роль у розвитку сучасної освітньої та інформаційної інфраструктури. Результати даного дослідження можуть стати важливим внеском у сферу програмування та розробки онлайн інструментів для доступу до знань.

Для **досягнення** поставленої мети та вирішення завдань будуть використані методи дослідження, що включають аналіз вимог користувачів, дослідження існуючих систем онлайн бібліотек, вибір оптимальних

технологій для реалізації програмного продукту та оцінка його функціональності та продуктивності.

Описана бакалаврська робота включає в себе ретельний аналіз та систематизацію знань про процес проектування програмного забезпечення для створення онлайн бібліотек, що може стати корисним для фахівців у галузі ІТ, студентів технічних спеціальностей та всіх зацікавлених у цій темі.

## **1. РОЗДІЛ 1. ХАРАКТЕРИСТИКА СИСТЕМИ, ТА АНАЛІЗ ВИМОГ**

### **1.1 Призначення , область застосування**

Бібліотеки залишаються одними з найбільш поширених місць, де можна отримати різноманітну літературу - книги, довідники та інші видання. Незважаючи на те, що деякі заклади переходять на онлайн-предоставлення книг через хмарні сховища, це поки що не є загальнопоширеним. Принципи отримання, зберігання та видачі книг за всі ці роки практично не змінилися.

Сучасна інтерпретація принципів роботи бібліотек переросла в автоматизовану бібліотечну інформаційну систему (АБІС). Однак, для точного визначення АБІС потрібно уточнити круг професійних завдань. Зокрема, основне завдання полягає в автоматизації традиційних бібліотечних технологій. До складових АБІС відносять реляційну базу даних та програму для взаємодії з цією базою даних [1].

На сьогодні на ринку вже існує безліч рішень щодо автоматизації роботи бібліотек: "ОРАС-Global", "АБІС Лібра" та інші продукти.

Аналізуючи існуюче програмне забезпечення, можна сказати, що ці програми занадто складні для звичайних користувачів через складний інтерфейс та потребують складної налаштування. Ці рішення не підходять для невеликих бібліотек через зайву функціональність. Крім того, всі

програми є платними, а при наявності невеликої кількості даних такі витрати не рентабельні [2].

Тому необхідно розглянути можливість створення власної системи автоматизації роботи онлайн бібліотеки.

Система, над якою проводиться аналіз вимог, є онлайн бібліотекою (програмним забезпеченням), призначеною для надання користувачам доступу до електронних книг та ресурсів через інтернет. Основні мети та області застосування цієї системи включають:

Надання доступу до літературних ресурсів: Онлайн бібліотека робить доступними книги у електронному форматі для читання та вивчення для широкого кола користувачів.

Зручний пошук та вибір книг: Система надає можливість швидкого пошуку книг за різними критеріями (назва, автор, жанр тощо) і забезпечує зручний інтерфейс для вибору та відображення деталей книги.

Читання книг онлайн та офлайн: Користувачі можуть читати книги безпосередньо на платформі онлайн або завантажувати їх для читання в офлайн-режимі.

Управління контентом: Система надає можливість користувачам додавати книги до списку вибраного, оцінювати та залишати відгуки про книги.

Персоналізовані рекомендації: На основі історії читання система рекомендує користувачам книги, які можуть їм сподобатися.

Області застосування включають освіту, науку, загальну читацьку аудиторію, студентів, дослідників та будь-яку особу, яка зацікавлена у доступі до різноманітного літературного контенту в електронній формі. Така система сприяє зручності, доступності та широкому розповсюдженню знань і інформації.

## 1.2 Визначення цілей

1. **Надати доступ до бібліотечних ресурсів:** Розробити платформу, яка забезпечить користувачам (читачам) можливість переглядати, шукати та отримувати доступ до різноманітних книг, журналів, наукових робіт тощо.
2. **Організувати зручний пошук та навігацію:** Створити ефективні та інтуїтивно зрозумілі інструменти пошуку, які дозволять користувачам знаходити потрібні книги за назвою, автором, жанром або іншими параметрами.
3. **Реалізувати особисті кабінети користувачів:** Забезпечити можливість реєстрації та авторизації, а також створення особистих профілів, де користувачі можуть переглядати історію взаємодії з бібліотекою, додавати обрані книги, залишати відгуки тощо.
4. **Забезпечити можливість видачі та повернення книг:** Реалізувати механізми онлайн замовлення та отримання книг, а також процес повернення та оновлення статусу книг у бібліотеці.
5. **Забезпечити безпеку даних та конфіденційність користувачів:** Застосувати найкращі практики щодо зберігання та захисту особистих даних користувачів, включаючи паролі, інформацію про платежі тощо.
6. **Розширити функціональні можливості:** Постійно вдосконалювати та розширювати функціонал платформи, додавати нові функції на основі вимог користувачів і новітніх технологій.
7. **Оптимізувати взаємодію та продуктивність:** Забезпечити швидку та ефективну роботу платформи, щоб забезпечити максимальний комфорт для користувачів при роботі з бібліотекою.

Дана програма призначена для використання бібліотекарями та користувачами з метою скорочення часових затрат на заповнення даних про

книги та читачів, бібліотекарів, а також обліку виданих книг. Вихідними даними для програми є:

1. інформація про читачів;
2. інформація про книги;
3. інформація про видання книг.

Метою проєкту є - розробка додатку, що дозволяє вести базу даних бібліотеки, вести облік книг, видач книг, читачів, довідкової інформації про працівників з можливістю перегляду, створення, зміни та видалення записів у БД, а також пошук по таблицям.

Також необхідно реалізувати зменшення часових витрат бібліотекарів на розгортання ПЗ;

- Надати можливість простого в освоєнні графічного інтерфейсу користувача;
- Реалізувати значне зменшення затрат часу працівників бібліотеки на заповнення відповідних обліків та таблиць;
- надання інформації у зручному форматі.

Для досягнення поставленої мети, будуть використані технології та інструменти, що дозволяють ефективно розробляти та підтримувати програмне забезпечення.

- Створення зручного та доступного середовища: Забезпечення легкого доступу до книжного контенту для користувачів будь-де і будь-коли.
- Покращення користувацького досвіду: Забезпечення інтуїтивного та зручного інтерфейсу, що робить використання бібліотеки приємним та ефективним для користувачів.
- Ефективний пошук та навігація: Розробка механізмів швидкого та точного пошуку книг, які дозволять користувачам легко знаходити необхідний контент.



- Забезпечення безпеки та конфіденційності: Захист особистої інформації користувачів та даних системи від несанкціонованого доступу.
- Розширення функціональності: Поступове розширення можливостей системи за допомогою нових функцій, які забезпечать додаткові можливості для користувачів.
- Забезпечення стабільності та надійності: Розробка програмного продукту, який працює стабільно та надійно під час використання користувачами.
- Задоволення потреб користувачів: Створення середовища, що задовольняє потреби та очікування користувачів, забезпечуючи позитивний досвід використання.

### **1.3 Аналіз Вимог**

Процес проектування вимог програмного забезпечення для онлайн бібліотеки є ключовим етапом, що включає ряд кроків та методів для створення функціональної та ефективної системи.[5]

Оцінка та документування вимог до системи від користувачів, що включає функціональні та нетехнічні вимоги, необхідність системи безпеки та масштабованості. [5]

*Проектування архітектури системи:*

Розробка структури системи, включаючи визначення компонентів, модулів, їх взаємодію та інтерфейси, що забезпечують функціональність системи.

*Проектування інтерфейсу користувача (UI/UX):*

Розробка зручного та інтуїтивно зрозумілого інтерфейсу для користувачів, який відповідає їх очікуванням та забезпечує комфортне взаємодію з системою.

**Проектування бази даних:** Створення структури бази даних, вибір типів даних, таблиць та зв'язків між ними, що дозволяє зберігати та ефективно обробляти інформацію.

*Розробка деталей архітектури:* Уточнення технічних деталей архітектури, вибір технологій та інструментів для реалізації функціональності системи.

*Проектування модулів та компонентів:* Розробка окремих модулів, їх функціональності та взаємодії з іншими частинами системи.

*Тестування проектування:* Перевірка архітектури на відповідність вимогам, виявлення та виправлення можливих недоліків та недоречностей у проекті.

*Документування проекту:* Підготовка документації, що описує проектні рішення, структуру системи, інтерфейси та інші технічні аспекти.

Процес проектування ПЗ для онлайн бібліотеки передбачає систематичний та послідовний підхід до розробки, що дозволяє створити функціональну та ефективну систему, відповідну вимогам користувачів та бізнес-потребам

#### **1.4 Вимоги до інтерфейсу користувача**

Користувацький інтерфейс повинен містити:

1. **Зручний та інтуїтивний інтерфейс:** Інтерфейс повинен бути простим у використанні та інтуїтивно зрозумілим для всіх категорій користувачів. [1]
2. **Ергономічний дизайн:** Важливо, щоб елементи управління були розміщені з урахуванням логічного порядку дій користувача та не заважали йому в користуванні.
3. **Адаптивний дизайн:** Інтерфейс повинен бути адаптований під різні типи пристроїв (комп'ютер, планшет, мобільний телефон) для зручного використання на будь-якому пристрої. [1]
4. **Чітка навігація:** Забезпечити легку та швидку навігацію користувача по різним розділам та функціям платформи. [1]

5. **Мінімалістичний дизайн:** Уникати перевантаження інформацією, застосовувати прості та чисті дизайнерські рішення.
6. **Функціональність пошуку та фільтрації:** Забезпечити можливість швидкого пошуку та фільтрації книг за різними критеріями.
7. **Підтримка особистих кабінетів користувачів:** Надати можливість реєстрації, авторизації, перегляду особистих даних та історії взаємодії з бібліотекою. [1]
8. **Пояснення для користувачів:** Забезпечити пояснення та інструкції для користувачів щодо користування різними функціями та можливостями платформи. [1]
9. **Безпека та конфіденційність:** Захист особистих даних користувачів, зокрема паролів, платіжної інформації та іншої конфіденційної інформації.
10. **Контрастність та доступність:** Забезпечити достатню контрастність, щоб інтерфейс був доступний для користувачів з різними видами обмежень.

#### **Авторизація та Реєстрація:**

Проста форма для входу та створення облікового запису з обов'язковими полями (ім'я, електронна пошта, пароль).

#### **Головна Сторінка:**

Чіткий та привабливий дизайн, який відображає важливі функції та актуальну інформацію про нові книги або акції.

Меню або панель навігації для швидкого доступу до основних розділів бібліотеки. [1]

#### **Пошук та Фільтрація:**

Простий та зручний пошук книг за назвою, автором, жанром тощо.

Фільтри для точного та швидкого пошуку, наприклад, за рейтингом, роком видання, мовою тощо. [1]

**Сторінка Книги:**

Інформація про книгу (назва, автор, опис, обкладинка, рейтинг, відгуки).

Можливість читати книгу онлайн або завантажувати.

Кнопки для додавання книги до обраного, оцінювання та написання відгуку.

**Профіль Користувача:**

Огляд особистих даних користувача та можливість зміни профілю.

Історія прочитаних книг, обране, активні позички тощо.

**Адаптивність та Відповідність:**

Адаптований дизайн для різних пристроїв (планшети, смартфони, ПК).

Відповідність вимогам щодо доступності та можливість використання основних функцій навіть для користувачів з обмеженими можливостями.

Вимоги до інтерфейсу користувача мають за мету забезпечити зручність використання, чіткість структури та інтуїтивно зрозумілі дії для користувачів будь-якого рівня.

### **1.5 Вимоги до апаратних, програмних та комунікаційних інтерфейсів**

Для роботи програми необхідно мати встановлену систему управління базами даних MySQL (локально), програмну платформу .NET 6, а також обчислювальну систему наступної мінімальної апаратної конфігурації:

Попри наявність новіших версій платформи, .NET 6 було обрано через його стабільність, сумісність з MySQL та підтримку довгострокової підтримки (LTS), що забезпечує безпечну експлуатацію.

1. процесор з тактовою частотою 1,0 ГГц;
2. ОЗУ об'ємом не менше 1 Гб;
3. вільне місце на жорсткому диску 300 Мб;
4. відеокарта з об'ємом пам'яті не менше 514 Мб;
5. операційна система Windows 10;

6. наявність засобів введення-виведення: миша, клавіатура, монітор;  
Необхідно забезпечити програмне взаємодію системи з системою управління базами даних MySQL.

### **1.6 Вимоги до адаптації на місці**

Перед використанням потрібно експортувати базу даних Library.sql, що додається до програми, та переконатися в успішному завершенні цієї операції. Перевірити доступ програми до бази даних. Для встановлення програми достатньо завантажити теку з файлами програми. Запуск програми здійснюється при запуску файлу LibraryDB.exe.

Вимоги до адаптації на місці можуть стосуватися здатності системи адаптуватися до специфічних вимог чи умов конкретного місця або регіону. Ось деякі можливі вимоги:

#### **1. Міжнародна підтримка:**

- Мульти-мовна підтримка для користувачів з різних країн.
- Можливість зміни мови інтерфейсу системи.

#### **2. Культурні аспекти:**

- Урахування особливостей культури та релігійних переконань у представленні матеріалів або вмісту бібліотеки.
- Забезпечення відповідності законодавству та моральним нормам різних країн.

#### **3. Локальні потреби:**

- Адаптація пошукових алгоритмів до специфічних особливостей місцевих потреб користувачів.
- Врахування регіональних попередніх вподобань у відображенні рекомендацій та пропозицій.

#### **4. Технічні аспекти:**

- Підтримка різних типів пристроїв, які є популярними в конкретній локації (наприклад, підтримка великої кількості смартфонів, планшетів чи мобільних пристроїв).
- Оптимізація для швидкої роботи на мобільних мережах, які можуть бути менш стабільними або повільними у деяких регіонах.

## **5. Системні вимоги:**

- Мінімальні вимоги до апаратних можливостей для забезпечення швидкої та ефективної роботи системи на різних пристроях.

Ці вимоги до адаптації на місці важливі для забезпечення успішної експлуатації продукту в різних країнах чи місцевостях з різними культурними, технічними та соціальними характеристиками.

### **1.7 Функції продукту**

Приклад додатку має бути здатний працювати з таблицями бази даних для обробки інформації про книги, читачів та їх видачі. У програмі повинна бути реалізована можливість пошуку інформації в таблиці за різними запитамі. Також має бути функція генерації деяких звітів, таких як неповнолітні читачі, боржники та книги, що закінчилися.

Функції продукту "Онлайн бібліотека" можуть включати різноманітні можливості, спрямовані на задоволення потреб користувачів у доступі до літературних ресурсів та зручного їх використання. Ось деякі базові функції:

#### **1. Реєстрація та Авторизація:**

Можливість створення облікового запису та авторизації для доступу до функцій бібліотеки.

#### **2. Пошук та Фільтрація:**

Пошук книг за різними критеріями: назва, автор, жанр, рік видання тощо.

Фільтрація результатів пошуку для точнішого вибору.

#### **3. Читання та Відгуки:**

Можливість читання книг онлайн без необхідності завантаження.

Додавання відгуків та оцінок книг.

#### **4. Управління Контентом:**

Додавання нових книг та їх видалення.

Можливість редагування інформації про книгу: опис, обкладинка, автор тощо.

#### **5. Особистий Кабінет Користувача:**

Моніторинг прочитаних книг.

Можливість створення власного списку обраних книг.

Управління особистими даними та паролем.

#### **6. Сповіщення та Рекомендації:**

Система сповіщень про нові книги, оновлення або акції.

Автоматичні рекомендації книг на основі історії читання та вибору користувача.

#### **7. Система Користувацьких Прав:**

Налаштування приватності, управління доступом до книг та профілю.

#### **8. Підтримка Різних Форматів:**

Підтримка різних форматів книг: електронні книги у PDF, EPUB тощо.

Ці функції дозволяють користувачам зручно та ефективно взаємодіяти з онлайн бібліотекою, забезпечуючи доступ до різноманітного контенту та можливості управління ним.

### **1.8 Обмеження**

Заборонено перейменовувати, редагувати та видаляти файли всередині файлової системи програми;

Програма не працює з іншими СУБД, і також потребує локального підключення до СУБД MySQL;

Продукт успішно працює лише в ОС Windows 10,11 з встановленою платформою .NET 6.0;

Продукт не передбачає автоматичного переходу на платформи, які не перераховані в даному документі.



## 2. РОЗДІЛ 2. ОГЛЯД МЕТОДІВ ПРОЄКТУВАННЯ ПРОГРАМНИХ СИСТЕМ

### 2.1. Огляд існуючих методів та основних принципів проєктування, архітектури програмного забезпечення, інженерних підходів і принципів керування ПЗ

Процес розробки програмного забезпечення для онлайн бібліотеки базується на низці принципів, які гарантують його якість, ефективність та стабільність. Ось кілька ключових принципів, які варто врахувати:

*Принцип модульності:* Розбиття системи на окремі модулі (або компоненти) дозволяє легко управляти та розвивати код, забезпечуючи його вищу структурованість та повторне використання[5].

*Принцип однорідності (Uniformity Principle):* Усі частини системи повинні дотримуватися спільних стандартів та правил програмування для спрощення зрозумілості коду для розробників [5].

*Принцип розділення відповідальності (Separation of Concerns):* Різні аспекти функціональності (такі як інтерфейс, бізнес-логіка, доступ до даних) повинні бути чітко відокремлені, що сприяє покращенню супроводжуваності та розвитку системи [5].

*Принцип єдиної відповідальності (Single Responsibility Principle):* Кожен модуль або клас повинен мати лише одну конкретну відповідальність, що спрощує тестування та зменшує взаємозалежність компонентів [5].

*Принцип відкритості та закритості (Open/Closed Principle):* Система повинна бути відкритою для розширення новим функціоналом, але закритою для змін, тобто допускати розширення без необхідності зміни вже наявного коду [5].

*Принцип інверсії залежностей* (Dependency Inversion Principle): Класи модулів мають залежати від абстракцій, а не від конкретних реалізацій, що сприяє зменшенню зв'язаності та підвищує гнучкість системи.

Ці принципи становлять основу для створення ефективної та стабільної архітектури програмного забезпечення для онлайн бібліотеки, допомагаючи забезпечити гнучкість, розширюваність та якість системи [5].

Огляд існуючих *методів*:

1. **Waterfall (Каскадний)**: Традиційний метод, що передбачає послідовні етапи розробки від визначення вимог до випуску продукту.

Модель каскадного (Waterfall) процесу розробки програмного забезпечення - це лінійний підхід, де кожен етап розробки проходить послідовно один за одним. Ця модель передбачає, що кожен етап повинен бути повністю завершений, перш ніж почнеться наступний.

Основні етапи моделі каскадного процесу розробки (Waterfall) включають такі кроки:[4]

**Визначення вимог:** Збирання та аналіз вимог від клієнта та створення специфікацій програми. [4].

**Проектування:** Розробка архітектури програми на основі вимог та створення детальних планів реалізації.

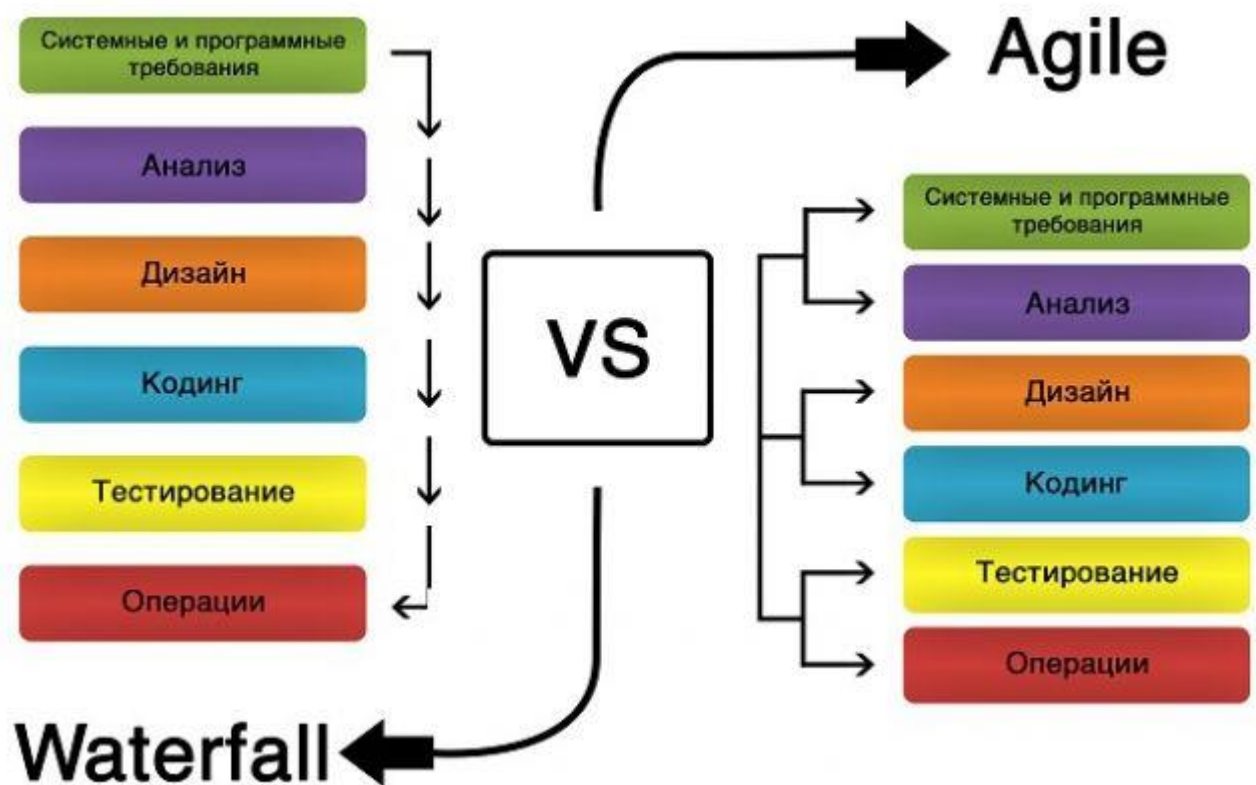
**Реалізація (розробка):** Написання коду та створення програмного продукту на основі визначених вимог і планів.

**Тестування:** Перевірка програми на відповідність вимогам, виявлення та виправлення помилок. [4].

**Впровадження (випуск):** Введення програми в експлуатацію та постачання продукту клієнту або користувачам.

**Підтримка:** Надання підтримки продукту, виправлення помилок, оновлення та вдосконалення програмного забезпечення [4].

Основна перевага моделі каскадного процесу полягає в простоті та чіткості послідовності етапів, що полегшує управління проектом та робить його більш передбачуваним. Однак ця модель може бути менш гнучкою у порівнянні з іншими методологіями, оскільки будь-які зміни вимог після початку процесу розробки можуть призвести до складнощів та витратних коригувань на пізніших етапах.



2. **Agile (Гнучкий):** Група методів розробки ПЗ, таких як Scrum, Kanban, XP тощо, які дозволяють гнучко відповідати змінам у вимогах та швидко адаптуватися [3]

Методологія Agile (гнучкий) - це набір принципів розробки програмного забезпечення, які ставлять акцент на гнучкість, співпрацю замість контрактів, реагування на зміни, а також визнання цінності робочого продукту. Agile спрямований на постійну ітеративну

розробку, сприяє постійному вдосконаленню і враховує зміни вимог у процесі розробки [2].

Основні принципи методології Agile включають:

**Гнучкість:** Здатність швидко реагувати на зміни та адаптуватися до нових умов, навіть у середині розробки. [4]

**Ітераційний** підхід: Розробка програмного забезпечення відбувається через короткі, ітеративні цикли розробки. [4]

**Співпраця** замість контракту: Більша увага приділяється співпраці та комунікації замість формальних контрактів. [4]

**Самоорганізація** і співпраця команди: Команди самі приймають рішення щодо планування та виконання завдань. [4]

**Постійна** відмовка і покращення (постійне вдосконалення): Команда постійно аналізує свою роботу та шукає способи поліпшення. [3]

**Продуктивна комунікація:** Підтримка ефективної та частої комунікації серед членів команди та замовників. [5]

Методологія Agile дозволяє підтримувати гнучкість та швидкість реагування на зміни у вимогах клієнта або ринкових умовах, проте вимагає активної участі замовника та досить стабільних вимог для успішної реалізації проекту.

3. **Incremental (Інкрементальний):** Розробка в кілька ітерацій, додавання нових функцій та можливостей на кожній ітерації. [4]

Методологія Інкрементального розроблення є однією зі стратегій розробки програмного забезпечення, де процес розробки розділяється на послідовні фази, а кожна нова фаза розширює функціональність попередньої. У цій методології розробки програмне забезпечення будується через поетапне додавання нового функціоналу, кожен з яких є інкрементом або приростом.

Основні аспекти інкрементального підходу включають:

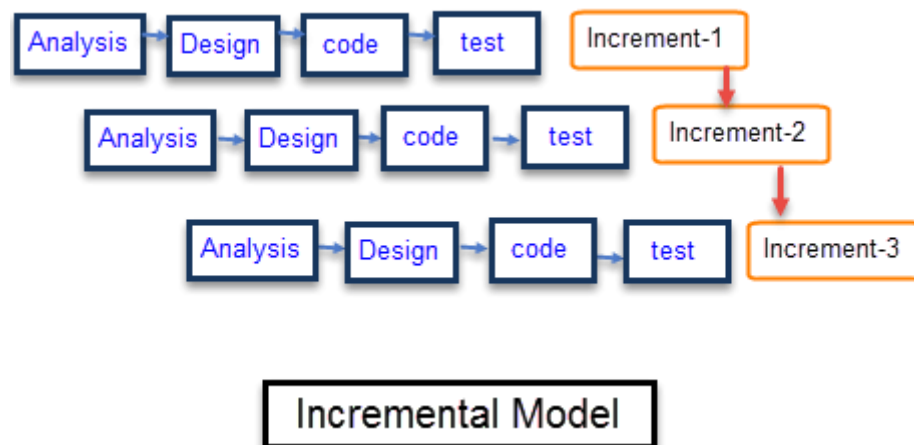
**Поступовий** розвиток продукту: Програмне забезпечення розробляється етапами, де кожен новий інкремент (приріст) розширює функціональність продукту. [4]

**Можливість** внесення змін: Дозволяє замовнику вносити зміни вимог під час розробки.

**Постійні** ітерації і оцінка результатів: Поетапний процес дозволяє проводити регулярні перевірки та тестування, що полегшує виявлення помилок та швидке їх виправлення.

**Поступове** уточнення вимог: Кожен інкремент додається на основі попереднього, дозволяючи уточнювати вимоги та функціонал по мірі розвитку проекту. [3]

Цей метод підходить для проектів, де вимоги часто змінюються або можуть бути уточнені під час розробки. Він дає можливість постійно вдосконалювати та розширювати функціонал програмного продукту через додавання нових інкрементів з покращеними можливостями.



4. **Spiral (Спіральний)**: Це комбінація каскадної моделі з елементами ітераційного розробки, що включає постійну оцінку ризиків та протилежності. [4]

Методологія "Спіральний" (Spiral) є гнучкою моделлю розробки програмного забезпечення, яка поєднує в собі елементи інших

методологій розробки та має ітеративний характер. Модель "Спіральний" передбачає постійну адаптацію до нових вимог та управління ризиками.

Основні етапи методології "Спіральний" включають:

**Оцінка** ризиків (Identification of Risks): Аналіз потенційних ризиків, які можуть виникнути на різних етапах розробки програми.

**Розробка** стратегії (Developing Strategies): Розробка стратегії для управління ризиками та визначення шляхів їх уникнення або зменшення. [4]

**Виконання** (Execution): Реалізація програми та проведення тестування, використання аналізу результатів для планування наступного ітераційного циклу.

**Оцінка** (Evaluation): Аналіз результатів інкрементальних ітерацій та прийняття рішення про продовження або коригування розробки з урахуванням отриманих результатів.

**Кожен** ітераційний цикл у методології "Спіральний" може включати в себе віддалені етапи розробки, такі як збір вимог, проектування, реалізація, тестування та випуск продукту. Головний аспект цієї моделі полягає в тому, що ризики активно враховуються і зменшуються на кожному етапі розробки.

**Метод** "Спіральний" відмінно підходить для проектів, які потребують великої уваги до управління ризиками, а також тих, де вимоги можуть бути змінені під час розробки, дозволяючи фокусуватися на гнучкості та адаптації.

5. **RAD (Радикальна розробка)**: Метод, орієнтований на швидкий процес розробки через використання прототипів та інтенсивну залученість замовника. [4]

**Методологія RAD (Rapid Application Development)** - це стратегія розробки програмного забезпечення, яка спрямована на швидке розгортання системи за допомогою швидких ітерацій та великого акценту на роботі з клієнтом. [4]

Основні особливості методології RAD:

**Швидкість** розробки: Ця методологія спрямована на швидке створення продукту. Короткі ітерації дозволяють швидко реагувати на зміни та додавати новий функціонал.

**Процес** взаємодії з клієнтом: Основний акцент робиться на активній співпраці та залученні клієнта на кожному етапі розробки.

**Прототипування:** Використання прототипів для демонстрації можливостей та отримання відгуку клієнта на ранніх стадіях розробки.

**Робота** в команді: Команди розробки працюють над великою кількістю завдань одночасно, прискорюючи процес розробки.

**Рекурсивність** ітерацій: Після отримання відгуку від клієнта, продукт може проходити через кілька ітерацій для уточнення та вдосконалення.

Методологія RAD найбільш ефективно використовується для проектів, де вимоги користувачів не зовсім чіткі та можуть змінюватися під час розробки. Ця методологія дозволяє швидко створювати та адаптувати програмне забезпечення з великою увагою до вимог клієнтів. Однак, вона може бути менш ефективною для проектів з жорсткими обмеженнями щодо бюджету та ресурсів.



6. **DevOps:** Підхід, що поєднує розробку та операції з метою автоматизації процесів виробництва ПЗ. [4]

DevOps підхід розглядається як сучасна практика забезпечення безперервної інтеграції, тестування та розгортання систем, що може бути корисним при подальшому масштабуванні або підтримці онлайн бібліотеки.

**DevOps** - це підхід до розробки програмного забезпечення, що поєднує розробку (Development) та експлуатацію (Operations). Основною метою DevOps є створення швидких, стабільних та автоматизованих процесів розробки та розгортання програмного забезпечення.

Ключові аспекти методології DevOps:

**Співпраця** і комунікація: Злагоджена робота між розробниками, тестувальниками та адміністраторами систем з метою вдосконалення процесів розробки та впровадження. [4]

**Автоматизація:** Використання інструментів для автоматизації процесів розгортання, тестування та моніторингу програмного забезпечення.

Неперервна інтеграція та неперервна доставка (CI/CD): Постійна інтеграція нового коду та автоматична доставка змін до продукції для забезпечення швидкої відповіді на зміни.

**Моніторинг** та забезпечення якості: Систематичний моніторинг роботи програми для виявлення помилок та забезпечення високої якості програмного забезпечення. [4]

**Еластичність** та масштабованість: Можливість швидко змінювати розмір та потужність системи в залежності від потреб.

DevOps сприяє зменшенню часу розгортання програмного забезпечення, поліпшенню комунікації між розробниками та адміністраторами, покращує якість та надійність програмного забезпечення. Цей підхід є особливо корисним у проектах, де потрібна



швидка реакція на зміни вимог та швидкий цикл розгортання нових функцій чи виправлень.

**7. Lean Software Development:** Орієнтований на зменшення витрат та мінімізацію марнотратства, фокусуючись на потребах клієнта.

Lean Software Development - це методологія, яка базується на принципах Lean Manufacturing та фокусується на ефективності виробництва і використанні ресурсів для створення програмного забезпечення. [4]

**Основні** принципи Lean Software Development:

**Визначення** цінності (Value): Цінність визначається перспективою клієнта - те, що клієнт готовий оплатити.

**Ідентифікація** потоку цінності (Value Stream): Визначення кроків, які необхідно зробити для перетворення вхідних вимог у цінні продукти. [4]

**Створення** потягучості (Pull): Акцент на виробництві лише тих функцій чи функціоналів, які реально потрібні клієнтам.

**Вдосконалення** процесу (Perfection): Постійна праця над вдосконаленням процесів розробки для досягнення максимальної ефективності.

**Управління** якістю (Quality): Акцент на постійному контролі якості продукту та процесів його розробки.

**Вдалий** відбір (Eliminate Waste): Мінімізація витрат шляхом усунення непродуктивних процесів. [4]

**Змінність** (Amplify Learning): Створення сприятливого середовища для навчання та вдосконалення.

**Методологія** Lean Software Development дозволяє оптимізувати розробку шляхом уникнення витрат часу і ресурсів на непотрібні процеси, надає можливість швидко реагувати на зміни та постійно покращувати якість програмного забезпечення. Цей підхід особливо

корисний у проектах, де важлива оптимізація витрат та швидкість реакції на зміни.

**8. FDD (Feature-Driven Development):** Зосереджений на поступовому визначенні, плануванні, дизайні та реалізації функцій.

Огляд існуючих архітектур програмного забезпечення передбачає розгляд різних архітектурних стилів та шаблонів, які використовуються для розробки програмних продуктів. Ось деякі з них:

**MVC (Model-View-Controller):** Розділення програми на три основні компоненти: модель (дані), представлення (інтерфейс користувача) та контролер (логіка).

**Microservices (Мікросервіси):** Архітектурний підхід, що передбачає розбиття програми на невеликі автономні сервіси, що працюють разом.

**SOA (Service-Oriented Architecture):** Архітектурний підхід, де програма складається з різних сервісів, які комунікують між собою через мережу.

**Event-Driven Architecture (EDA):** Архітектурний стиль, де взаємодія між компонентами програми здійснюється за допомогою подій.

**Layered Architecture (Шарова архітектура):** Розділення програми на логічні шари (наприклад, презентаційний, бізнес-логіка, доступ до даних).

**Hexagonal Architecture (Ports and Adapters):** Фокусується на відокремленні бізнес-логіки від зовнішніх елементів через використання портів та адаптерів.

**Component-Based Architecture (Архітектура на основі компонентів):** Підхід, де програма розбита на окремі компоненти, які можуть бути розроблені та підтримувані окремо.

**CQRS (Command Query Responsibility Segregation):** Розділення запитів на читання та запис на окремі моделі, що дозволяє оптимізувати читання та запис.

**Патерни** проектування в програмуванні - це загальні рекомендації та шаблони, які виникли на основі накопиченого досвіду розробки програмного

забезпечення. Ось кілька типових паттернів, які можна використовувати у розробці онлайн бібліотеки:

**1. Модель-Вид-Контролер (Model-View-Controller, MVC):**

- **Опис:** Розділяє програму на три основні складові: модель (дані), вид (представлення) та контролер (логіка).
- **Використання в онлайн бібліотеці:** Модель може включати дані про книги та користувачів, Вид - відображення інформації користувачам, а Контролер - обробку запитів користувачів.

**2. Сервісний шар (Service Layer):**

- **Опис:** Визначає сервіси, які надають бізнес-логіку та функціональність системи.
- **Використання в онлайн бібліотеці:** Сервіси можуть забезпечувати функції каталогування книг, авторизації користувачів та інші операції.

**3. Репозиторій (Repository):**

- **Опис:** Відокремлює доступ до даних від бізнес-логіки та забезпечує уніфікований інтерфейс для роботи з базою даних.
- **Використання в онлайн бібліотеці:** Репозиторії можуть використовуватися для взаємодії з базою даних книг, користувачів тощо.

**4. Фабричний метод (Factory Method):**

- **Опис:** Надає спосіб створення об'єктів без прив'язки до конкретних класів.
- **Використання в онлайн бібліотеці:** Може використовуватися для створення об'єктів книг, користувачів, які можуть бути різних типів.

**5. Одинак (Singleton):**

- **Опис:** Гарантує, що певний клас має лише один екземпляр та надає глобальний доступ до цього екземпляра.
- **Використання в онлайн бібліотеці:** Може застосовуватися для доступу до певних об'єктів, наприклад, системного журналу або менеджера сесій.

#### 6. Фасад (Facade):

- **Опис:** Надає простий інтерфейс до складних підсистем, щоб полегшити їхнє використання.
- **Використання в онлайн бібліотеці:** Може використовуватися для створення простого інтерфейсу для клієнтів бібліотеки для реєстрації, пошуку книг тощ

## 2.2 Обґрунтування вибору методів та підходів до проєктування ПЗ

Обґрунтування методів та підходів до проєктування програмного забезпечення для онлайн бібліотеки є важливим етапом, що визначає правильний напрямок розробки продукту.

Результативність будь-якого методу залежить від впровадження його принципів та відповідного підходу до роботи з ним. Тому вибір методу повинен базуватися на потребах проєкту та можливостях команди розробників.

Нижче наведено ті самі методи та підходи, які будуть використовуватися для розробки та проєктування для додатку онлайн бібліотеки:

#### Метод:

1. **Agile (Scrum):** Гнучкий метод розробки, що дозволяє швидко адаптуватися до змін у вимогах та забезпечити більшу прозорість процесу розробки.

#### Архітектура:

**Модульна архітектура** є підхід, що полягає у розбитті програми на окремі, незалежні від інших частин, модулі. Кожен модуль відповідає за виконання певного завдання чи функції, і має свій власний інтерфейс для взаємодії з іншими модулями.

Основні переваги модульної архітектури включають:

**Модульність і відокремленість:** Модулі можуть бути розроблені, тестовані та вдосконалені незалежно один від одного, спрощуючи процес розробки.

**Легше супроводження і розширення:** Модульна структура сприяє легкому виявленню помилок та їх локалізації, а також спрощує додавання нової функціональності.

**Використання інтерфейсів:** Використання чітко визначених інтерфейсів між модулями дозволяє розділити відповідальності та знизити залежність між ними.

**Підтримка тестування:** Модулі можуть бути протестовані окремо, що полегшує виявлення помилок та забезпечення їх правильного функціонування.

**Підтримка багаторазового використання коду:** Модулі, які відповідають за конкретні функції, можуть бути використані у різних частинах програми або навіть в різних проектах.

У **модульній** архітектурі кожен модуль може мати свої внутрішні складові, такі як класи, функції, методи, які допомагають виконувати визначені завдання. Важливо планувати структуру модулів заздалегідь, визначаючи їхні функціональні області та взаємодію між ними, щоб досягти більшої ефективності та гнучкості в розробці програмного забезпечення.

**Інженерний підхід:**

1. **Model-View-Controller (MVC):** Розділення програми на модель (дані), представлення (інтерфейс користувача) та контролер (логіка).
2. **Test-Driven Development (TDD):** Розробка через написання тестів перед розробкою коду, що сприяє високій якості програмного продукту.

Методологія Agile: Обґрунтування використання методології Agile базується на необхідності гнучкості та можливості швидко вносити зміни в систему відповідно до змінних вимог користувачів. Цей підхід дозволить швидше адаптуватися до змін та покращувати продукт на кожному етапі розробки.

Розробка через тестування (Test-Driven Development, TDD): Обґрунтування використання TDD полягає в підході, коли спочатку розробляються тести для функціоналу, а потім сам функціонал. Це дозволяє створювати стабільний та відповідний функціонал та зменшує кількість помилок у програмному коді.

Модульне тестування (Unit Testing): Обґрунтування використання модульних тестів полягає в тому, щоб перевірити окремі модулі (функції, класи тощо) на коректність роботи. Цей підхід дозволяє виявляти помилки та забезпечує більшу надійність програмного забезпечення.

Принципи SOLID та Clean Code: Обґрунтування застосування принципів SOLID (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion) та практик "чистого коду" (Clean Code) визначається бажанням створити систему, яка буде легко змінюватися та розширюватися з мінімальними зусиллями.

Проектування за допомогою UML-діаграм: Обґрунтування використання UML-діаграм (Unified Modeling Language) полягає в можливості візуалізації структури системи, її компонентів та взаємодій між ними, що

сприяє кращому розумінню системи та зручності комунікації в команді розробників.

Обґрунтування вибору цих підходів полягає у прагненні до ефективного та гнучкого розробництва, яке дозволить створити високоякісне програмне забезпечення для онлайн бібліотеки, що відповідає сучасним стандартам розробки та вимогам користувачів.

Вибір конкретного методу розробки програмного забезпечення для онлайн бібліотеки може бути залежний від різних факторів, таких як розмір команди розробників, складність проекту, обсяг та специфіка вимог клієнта, час та бюджет, доступні ресурси, технічні знання та навички команди тощо. Проте, я вважаю що вибраний метод а саме методологія Agile, зокрема Scrum або Kanban, може бути вибраним методом для розробки програмного забезпечення онлайн бібліотеки. Agile надає можливість гнучко реагувати на зміни вимог, швидше адаптувати продукт до потреб користувачів, ефективно керувати процесом розробки та сприяє постійному вдосконаленню продукту.

Scrum чи Kanban - це конкретні реалізації методології Agile. Scrum використовує ітераційний підхід з короткими циклами розробки (спринтами), плануванням, регулярними відгуками та постійним вдосконаленням. Kanban спрямований на створення потокової моделі розробки з акцентом на візуалізацію робочого процесу та його оптимізацію.

Обираючи між Scrum та Kanban, важливо врахувати специфіку проекту та потреби команди. Scrum може бути корисним для проектів, які вимагають стриманого планування та чіткого управління спринтами. У той час як Kanban може бути ефективним у проектах, де важлива постійна відкритість до змін та потоковий процес розробки.

Обґрунтування використання методу Scrum для проектування програмного забезпечення онлайн бібліотеки може бути наступним:

Гнучкість та адаптивність до змін:

Підхід Scrum дозволяє гнучко реагувати на зміни вимог та пріоритетів, що є ключовим у розробці програмного продукту, особливо для онлайн бібліотеки, де можуть з'являтися нові технології або змінюватися потреби користувачів.

Ітераційний підхід:

Робота у вигляді ітераційних спринтів у Scrum дозволяє розробляти та випускати функціонал частіше, забезпечуючи постійний прогрес у розвитку продукту.

Зосередження на важливому функціоналі:

Scrum спрямований на розробку функцій з високим пріоритетом, що дозволяє концентрувати зусилля на важливих для користувача аспектах онлайн бібліотеки, таких як зручний пошук, легкість використання та доступ до книг.

Регулярні відгуки та вдосконалення:

Регулярність відгуків під час планування, спринт-рев'ю та ретроспектив дозволяє виявляти недоліки швидше, вносити зміни та покращувати якість продукту.

Ефективне управління ризиками:

Scrum сприяє ранньому виявленню проблем, що дозволяє управляти ризиками та вирішувати їх у ранній стадії проекту.

Комунікація та співпраця в команді:

Scrum підтримує активну комунікацію та співпрацю між учасниками команди, що є ключовим для успішного розвитку продукту та уникнення можливих непорозумінь.

Використання методу Scrum у процесі розробки онлайн бібліотеки допоможе ефективно управляти проектом, забезпечуючи якісний та користувацько-орієнтований результат, відповідно до вимог користувачів та бізнес-потреб.

**Спринт 1:**



Мета: Базовий функціонал системи.

Реєстрація та авторизація користувачів.

Основна структура користувацького інтерфейсу.

Можливість перегляду списку книг.

### **Спринт 2:**

Мета: Пошук та фільтрація книг.

Реалізація пошуку за назвою, автором та жанром.

Можливість фільтрувати книги за різними категоріями.

### **Спринт 3:**

Мета: Функціонал читання та відгуки.

Додавання можливості читати книги онлайн.

Відгуки та оцінки книг.

### **Спринт 4:**

Мета: Додатковий функціонал та покращення.

Додаткові функції для користувачів, такі як додавання книг до обраного, рекомендації тощо.

Виправлення помилок та вдосконалення інтерфейсу.

Кожен спринт має свою мету та конкретний функціонал, який повинен бути реалізований протягом цього періоду. Перед кожним спринтом варто відпрацювати деталі, збір вимог, розробку та тестування, щоб забезпечити успішне виконання завдань та досягнення поставлених цілей.

### **3. РОЗДІЛ 3. ОПИС ТА ОБГРУНТУВАННЯ ПРОЄКТНИХ РІШЕНЬ ПО ПРОЄКТУВАННЮ ПРОГРАМНОЇ СИСТЕМИ**

#### **3.1 Структурні схеми системи (модулі системи та компоненти)**

Створення онлайн бібліотеки передбачає вибір оптимальної архітектури системи, що визначатиме основні принципи організації, взаємодії та компонентів програмного забезпечення. Рішення щодо архітектурного стилю базується на ряді факторів та потреб проєкту. Для онлайн бібліотеки розглядаються наступні архітектурні стилі:

**МОДУЛЬНА СТРУКТУРА** Цей стиль передбачає розділення системи на кілька частин. Клієнтська частина відповідає за інтерфейс користувача, взаємодію з ним, тоді як серверна частина керує базою даних, обробкою запитів та забезпеченням даних.

**Model-View-Controller (MVC):** Цей архітектурний стиль використовується для розділення програми на три основні компоненти: Модель, Що представляє дані та логіку, Перегляд, Що відображає інтерфейс

користувача, та Контролер, Що керує взаємодією між моделлю та переглядом.

Event-Driven Architecture (EDA): Цей стиль базується на обміні подіями між різними компонентами системи. Компоненти реагують на події та відповідно реагують на них.

При виборі архітектурного стилю для онлайн бібліотеки важливо врахувати рівень складності проєкту, потреби користувачів, масштабованість, продуктивність та інші фактори, що впливають на успішність розробки та функціональність системи.

Зважаючи на проектування додатку онлайн бібліотеки за допомогою **Service-Oriented Architecture**, основні етапи можуть виглядати наступним чином:

**1. Аналіз бізнес-потреб та вимог:**

- Оцінка потреб користувачів: можливість пошуку книг, реєстрація користувачів, оформлення запитів на книги тощо.
- Визначення функціональності: система каталогізації, процес видачі книг, профілі користувачів, аутентифікація та авторизація.

**2. Розділення функціональності на сервіси:**

- Каталогізація книг.
- Управління користувачами.
- Авторизація та аутентифікація.
- Видача та повернення книг.
- Сервіс пошуку книг.

**3. Проектування API сервісів:**

- Визначення та документування інтерфейсів для кожного сервісу, включаючи доступні функції та методи.

**4. Реалізація сервісів:**

- Розробка окремих сервісів згідно з визначеними API, включаючи реалізацію логіки та доступ до баз даних для кожного сервісу.

**5. Управління залежностями та взаємодією сервісів:**

- Забезпечення взаємодії між сервісами через визначені API, контроль над залежностями між ними.

**6. Тестування та валідація сервісів:**

- Проведення тестів на кожному з сервісів для переконання, що вони працюють правильно та взаємодіють належним чином.

**7. Розгортання та моніторинг:**

- Розгортання сервісів у продакшн та постійний моніторинг їхньої роботи для виявлення можливих проблем та оптимізації роботи системи.

**8. Підтримка та розвиток:**

- Підтримка роботи сервісів у виробничому середовищі та їхній подальший розвиток, виправлення помилок, впровадження нових можливостей.

**Структурні схеми** системи в контексті онлайн бібліотеки можуть бути представлені через кілька основних компонентів та їх взаємодію:

**Фронтенд** інтерфейсу користувача (Frontend):

**Головна сторінка:** Пошук книг, категорії, рекомендації, останні оновлення.

**Сторінки книг:** Відображення інформації про книгу, кнопки читання/завантаження, відгуки користувачів тощо.

**Авторизація/Реєстрація:** Форми для входу або створення облікового запису.

**Бекенд** (Backend):

Система управління базою даних: Зберігання інформації про книги, користувачів, їх вибори та взаємодію з системою.

Додаткові сервіси: Система авторизації, система рекомендацій, система обробки платежів (якщо передбачено).

Інтеграція зовнішніх сервісів:

Системи платіжних шлюзів: Якщо передбачається платний доступ до контенту.

Сервіси аналітики: Для відстеження активності користувачів, оцінювання популярності книг тощо.

API для інтеграції з іншими платформами або соціальними мережами.

Хмарні технології та зберігання даних:

Зберігання книг у хмарних сховищах: Для забезпечення доступу до контенту в режимі онлайн або офлайн.

Захист та резервне копіювання даних: Для забезпечення безпеки та надійності.

Ці структурні схеми відображають основні компоненти системи онлайн бібліотеки та їх взаємодію для забезпечення функціональності, безпеки та зручності користувачів. Вони дозволяють систематизувати роботу різних частин системи та забезпечити їх взаємодію для досягнення загальних цілей продукту.

Розробка онлайн бібліотеки передбачає створення ряду модулів та компонентів, які взаємодіють між собою для забезпечення функціональності системи. Ось загальний огляд основних модулів та компонентів, що планується розробляти:

#### **Модуль авторизації та аутентифікації:**

Відповідає за процеси реєстрації нових користувачів та їх авторизацію в системі.

#### **Модуль управління користувачами:**

Надає можливість переглядати та змінювати інформацію про користувачів, включаючи їх особисті дані та історію.

#### **Модуль управління книгами:**

Відповідає за додавання нових книг, редагування інформації про книги та їх видалення.

**Модуль пошуку та фільтрації:**

Забезпечує можливість пошуку книг за різними критеріями (назва, автор, жанр тощо) та їх фільтрацію.

**Модуль читання книг:**

Надає засоби для зручного читання книг онлайн та можливість завантаження для офлайн-читання.

**Модуль рейтингування та відгуків:**

Дозволяє користувачам залишати відгуки та оцінювати прочитані книги.

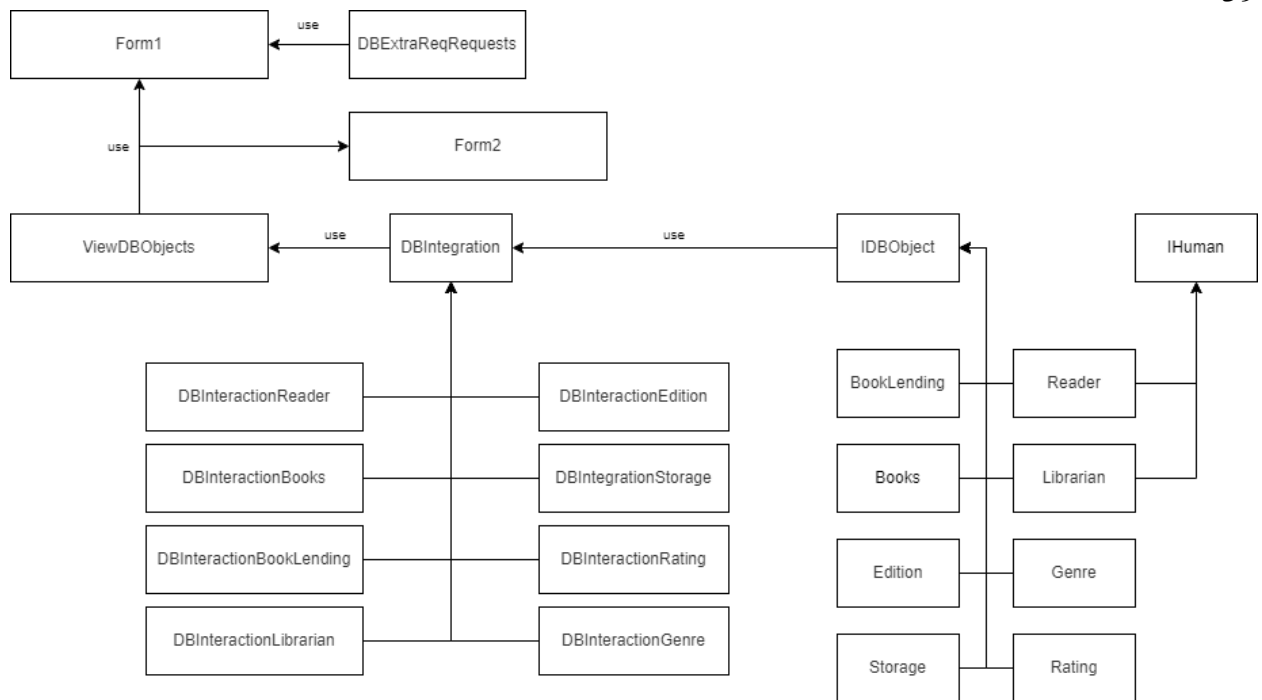
**Модуль рекомендацій:**

Надає персоналізовані рекомендації користувачам на основі їхньої історії читання.

**Модуль керування вибраним контентом:**

Забезпечує можливість користувачам додавати книги до списку вибраного для зручного доступу.

Кожен з цих модулів буде мати свою функціональність, взаємодіяти з базою даних та іншими компонентами системи. Проектування модулів та компонентів виконується з урахуванням їхньої незалежності, легкості розширення та можливості масштабування, щоб забезпечити ефективну та стабільну роботу всієї системи онлайн бібліотеки.



Малюнок 3.1 – Зв'язок файлів додатку

### 3.2 Опис Архітектурної системи

Архітектура онлайн бібліотеки включає різні компоненти, які спільно працюють для забезпечення функціональності та ефективності системи.

**Клієнтська частина (Frontend):** Цей компонент відповідає за інтерфейс користувача. Включає у себе веб-інтерфейс або мобільні додатки, які дозволяють користувачам взаємодіяти з бібліотекою, шукати книги, переглядати інформацію про них, оформляти замовлення та виконувати інші дії.

**Серверна частина (Backend):** Цей компонент забезпечує логіку бізнес-процесів. Включає в себе сервер, базу даних, додаткові сервіси для обробки запитів користувачів, управління даними та бізнес-логіку, наприклад, авторизацію, аутентифікацію, обробку оплати, пошук книг тощо.

**База даних:** Це централізоване сховище, де зберігається вся інформація про книги, користувачів, замовлення, рейтинги, коментарі та інша необхідна інформація для роботи системи.

Система авторизації та безпеки: Цей компонент відповідає за захист конфіденційності даних користувачів, автентифікацію, авторизацію та забезпечення безпеки системи в цілому.

Модуль управління контентом: Включає в себе функціонал для додавання, видалення та редагування інформації про книги, оновлення даних та управління контентом.

Сервіси та API: Додаткові сервіси або API, які можуть використовуватися для підключення додаткових функцій, співпраці з іншими системами, обміну даними тощо.

Архітектура системи онлайн бібліотеки ретельно спроектована для забезпечення ефективної взаємодії між різними компонентами, забезпечення безпеки, швидкодії та функціональності для користувачів.

### **3.3 Структура бази даних , моделювання бази даних (інфологічна модель)**

Структура бази даних для онлайн бібліотеки може бути представлена через низку сутностей та їх взаємозв'язки. Ось можлива інфологічна модель бази даних для системи онлайн бібліотеки:

*Користувачі (Users):*

user\_id (ідентифікатор користувача)

username (ім'я користувача)

email (електронна адреса користувача)

password\_hash (хеш пароля)

інші властивості (наприклад, роль користувача)

*Книги (Books):*

book\_id (ідентифікатор книги)

title (назва книги)

author (автор книги)

genre (жанр книги)



description (опис книги)

file\_path (шлях до файлу з книгою)

інші властивості (наприклад, рік видання, кількість сторінок тощо)

*Оцінки та відгуки (Ratings/Reviews):*

rating\_id (ідентифікатор оцінки)

user\_id (зовнішній ключ до таблиці користувачів)

book\_id (зовнішній ключ до таблиці книг)

rating (оцінка користувача)

review\_text (текстовий відгук користувача)

*Автори (Authors):*

author\_id (ідентифікатор автора)

author\_name (ім'я автора)

інші властивості (наприклад, рік народження, країна тощо)

*Категорії/Жанри (Categories/Genres):*

category\_id (ідентифікатор категорії)

category\_name (назва категорії)

**Ця інфологічна модель бази** даних визначає основні сутності та їх атрибути, а також взаємозв'язки між ними. Наприклад, кожен користувач може оцінювати та залишати відгуки про книги, а книги можуть бути призначені певним категоріям чи жанрам. Реалізація фізичної моделі бази даних (таблиці, ключі, індекси тощо) буде залежати від конкретної системи управління базами даних (наприклад, MySQL, PostgreSQL, MongoDB та ін.).

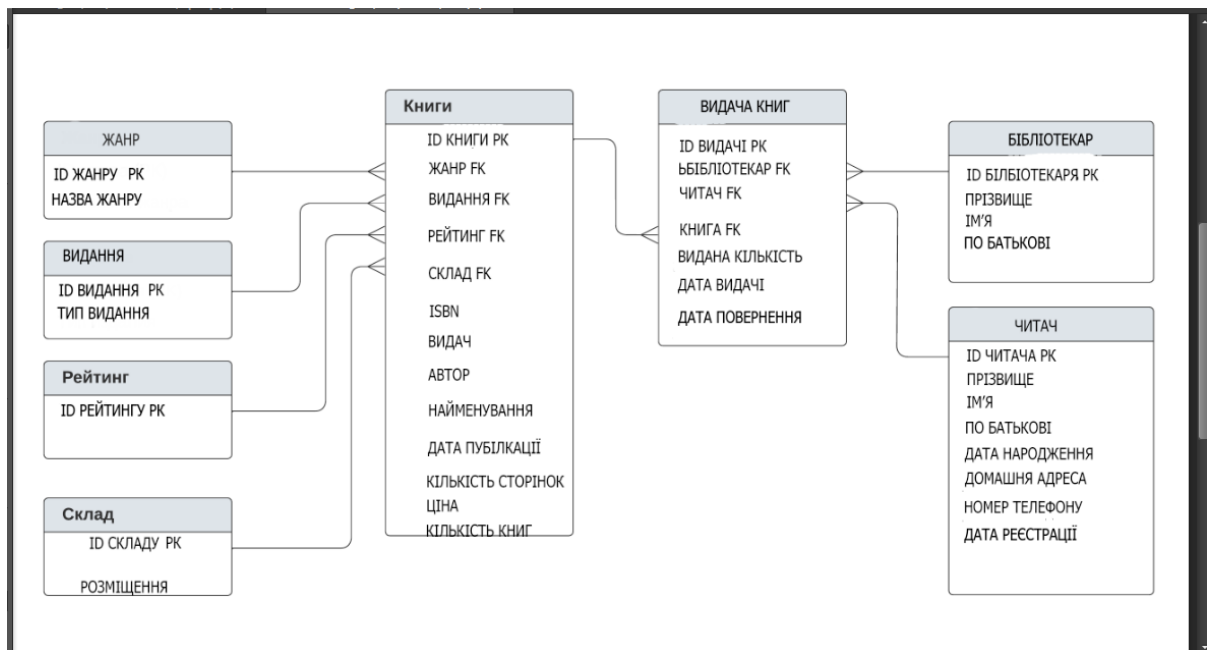
Ця **модель** може бути розширена або змінена залежно від конкретних вимог до системи та додаткових функцій, які можуть бути включені до онлайн бібліотеки.

**Логічне проектування** - створення схеми бази даних на основі конкретної моделі даних. У проекті використовується реляційна база даних. Для кожної виділеної сутності необхідно продумати поля, включаючи

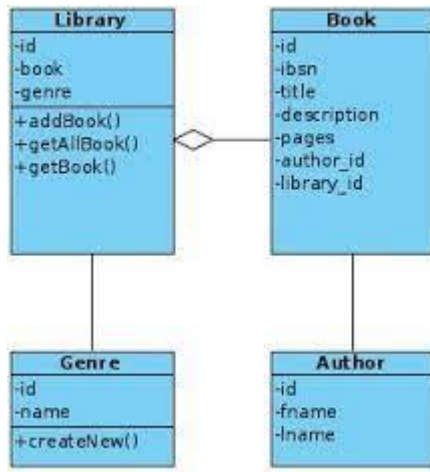
первинні та зовнішні ключі. Також були продумані зв'язки таблиць. У даному випадку всі зв'язки доцільно створити як "один до багатьох". У такого роду зв'язках рядок в першій таблиці може мати багато представлень в другій таблиці, але рядок у другій таблиці може мати тільки один рядок в першій таблиці. З урахуванням даних бізнес-процесів та предмета дослідження була створена логічна модель. Логічне представлення бази даних наведено на малюнку 4.2.

РК – ПЕРВИНИЙ КЛЮЧ

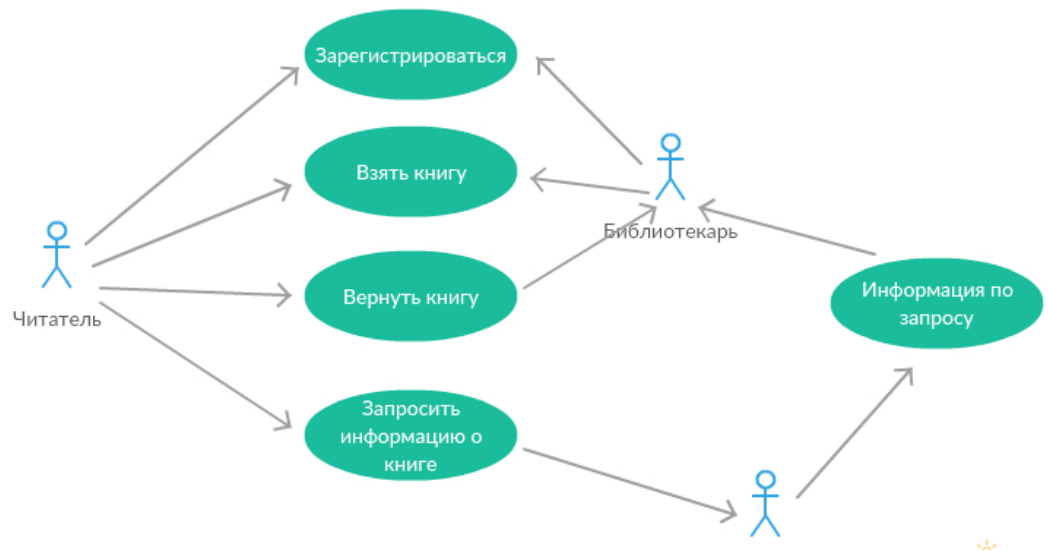
FK – ЦЕ КЛЮЧ, ВИКОРИСТОВУЄТЬСЯ ДЛЯ З'ЄДНАННЯ ДВОХ ТАБЛИЦЬ РАЗОМ.



Малюнок 3.2 – Інфологічна модель бази даних



Малюнок 3.2 – UML Diagram, архітектурна схема взаємодії компонентів онлайн бібліотеки.



Малюнок 3.3 UML DIAGRAM Прецидентів, Інфологічна модель бази даних онлайн бібліотеки

#### **4. РОЗДІЛ 4. ПРАКТИЧНА РЕАЛІЗАЦІЯ (РОЗРОБКА БАЗИ ДАНИХ БІБЛІОТЕКИ)**

В якості системи управління базами даних для створення та зберігання реляційної БД бібліотеки був обраний MySQL, а також утиліта Open Server Panel. MySQL - вільна реляційна система управління базами даних, розроблена та підтримувана американською компанією Oracle. Ця СУБД є хорошим рішенням для малого та середнього програмного забезпечення. MySQL можна використовувати як сервер, до якого звертаються клієнти (локально або віддалено), проте СУБД підтримує функціонал, який дозволяє включати MySQL всередину програми [3]. Крім цих переваг, система управління базами даних є однією з найбільш поширених та популярних на ринку [4]. Open Server Panel - утиліта, призначена для розробки, налагодження та тестування WEB-проектів, а також для надання веб-сервісів у локальних мережах. Цей програмний комплекс включає в себе набір серверного ПО та утиліт для адміністрування та налаштування всіх доступних компонентів. Дана утиліта підтримує і MySQL, забезпечуючи зручне та швидке локальне підключення. Для зручної взаємодії з необхідною СУБД з метою розробки, Open Server Panel містить спеціальний WEB-інтерфейс для адміністрування - phpMyAdmin, що полегшує роботу з створення БД [5]. Для створення додатку з графічним інтерфейсом користувача була обрана платформа Windows Forms, що працює на мові C#. В якості середовища розробки була обрана Microsoft Visual Studio 2022 Community. Для взаємодії з MySQL був обраний фреймворк MySQL Connector, оскільки він зручний у використанні.

Windows Forms - це платформа користувацького інтерфейсу для створення класичних додатків Windows за допомогою візуального

конструктора в Visual Studio. Платформа має можливість розміщення елементів керування шляхом перетягування, спрощуючи створення класичних додатків [6]. С# - сучасна об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. Дана мова характеризується легкістю в освоєнні, універсальністю та безпекою [7]. Visual Studio є найбільш функціонально насиченою IDE-середовищем, в якому можна розробляти додатки на С#. Visual Studio - це стартова платформа для написання, налагодження та збирання коду, а також подальшої публікації додатків. Інтегроване середовище розробки (IDE) є багатофункціональною програмою, яку можна використовувати для різних аспектів розробки програмного забезпечення.

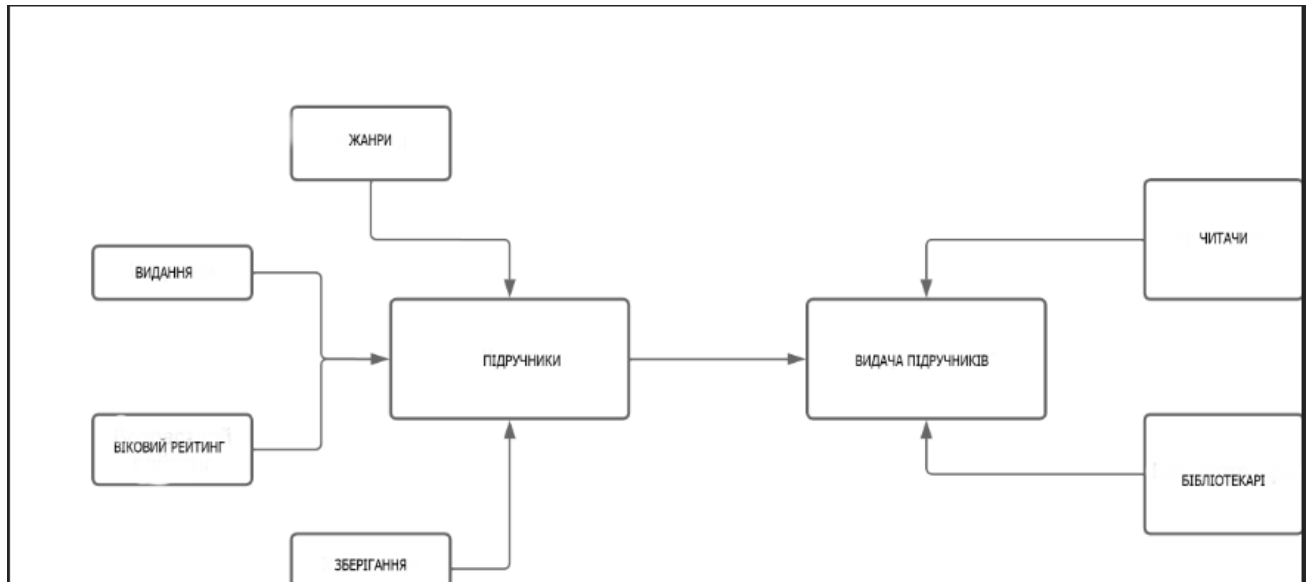
#### **4.1 Розробка бази даних програми**

##### ***4.1.1 Концептуальне проектування бази даних***

Концептуальне проектування - це створення семантичної моделі предметної області (інформаційної моделі самого високого рівня абстракції). Концептуальна модель бази даних включає в себе сутності та, в деяких випадках, теоретичні зв'язки даних сутностей. Під час вивчення предметної області було виділено наступні сутності, які представляють інформаційну цінність для задачі:

1. Видача книг - облік видання книг;
2. Читачі - всі зареєстровані читачі;
3. Книги - всі книги бібліотеки; Для виділення однакової інформації та оптимізації зберігання даних були виділені додаткові сутності:
4. Бібліотекарі - облік бібліотекарів, що беруть участь у виданні;
5. Жанри - всі види жанрів, пов'язаних з книгами;
6. Видання - всі види типів видань книг;
7. Віковий рейтинг - всі види вікових рейтингів;

8. Зберігання - всі розташування книг; Концептуальне представлення бази даних наведено на малюнку 4.1.



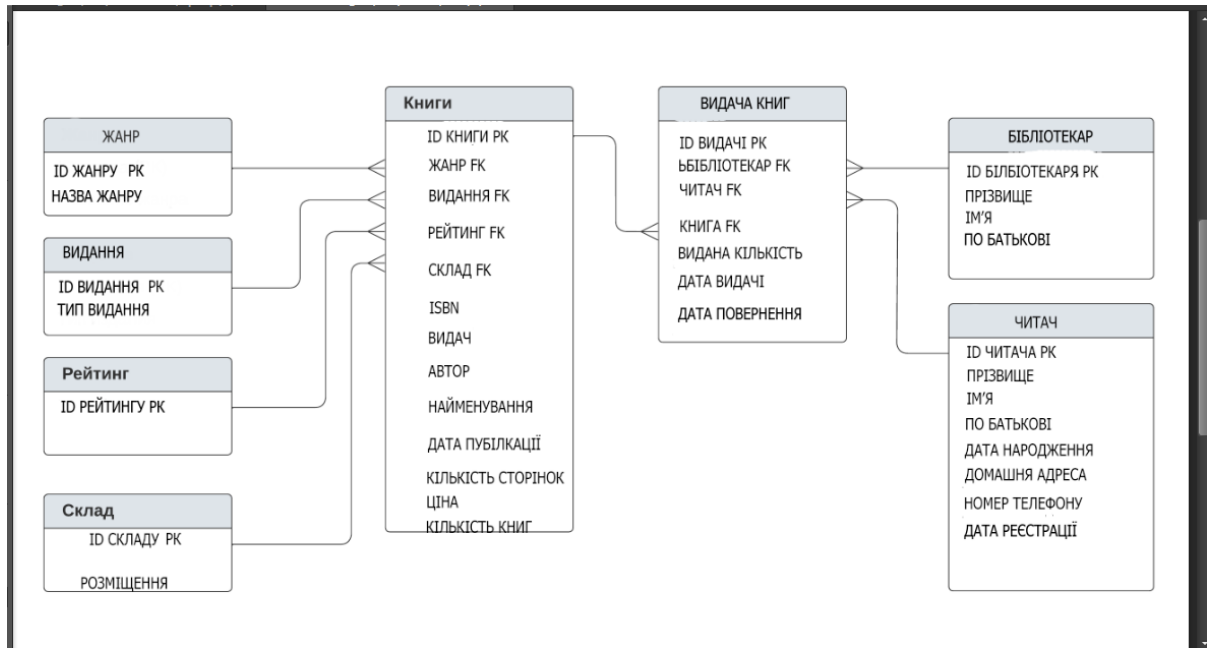
Малюнок 4.1 – Концептуальна модель бази даних

#### 4.1.2 Логічне проектування баз даних

Логічне проектування - створення схеми бази даних на основі конкретної моделі даних. У проекті використовується реляційна база даних. Для кожної виділеної сутності необхідно продумати поля, включаючи первинні та зовнішні ключі. Також були продумані зв'язки таблиць. У даному випадку всі зв'язки доцільно створити як "один до багатьох". У такого роду зв'язках рядок в першій таблиці може мати багато представлень в другій таблиці, але рядок у другій таблиці може мати тільки один рядок в першій таблиці. З урахуванням даних бізнес-процесів та предмета дослідження була створена логічна модель. Логічне представлення бази даних наведено на рисунку 4.2.

РК – ПЕРВИНИЙ КЛЮЧ

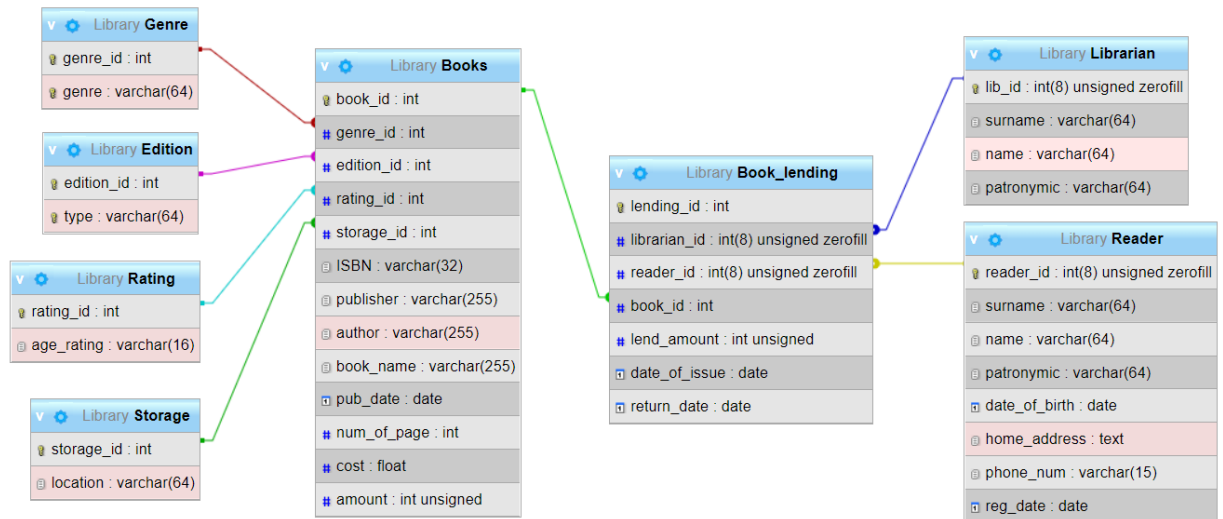
ФК – ЦЕ КЛЮЧ, ВИКОРИСТОВУЄТЬСЯ ДЛЯ З'ЄДНАННЯ ДВОХ ТАБЛИЦЬ РАЗОМ.



Малюнок 4.2 – Інфологічна модель бази даних

### 4.1.3 Фізичне проектування бази даних

Фізичне проектування - створення схеми бази даних для конкретної СУБД, яка включає всі необхідні таблиці, стовпці, зв'язки, властивості бази даних для фізичної реалізації. У цьому проекті використовується MySQL. Згідно з типами даних, зв'язками, логічною моделлю та іншими уточненнями з бізнес-процесів була реалізована фізична модель. Фізичне представлення таблиць бази даних наведено на малюнку.



Малюнок 4.3 – Датологічна модель бази даних

## 4.2 Проектування структури програми та паттерни

Проект LibraryDB складається з файлів вихідного коду:

1. Form1.cs – головна форма, на якій відображаються та редагуються дані таблиць, підготовлюються основні елементи управління;
2. Form2.cs — довідкова форма для відображення відомостей таблиць БД;
3. DBExtraReqRequests.cs – створення додаткових довідкових запитів;
4. ViewDBObject.cs – зв'язок функціоналу Form1 з функціоналом нащадків DBIntegration;
5. DBIntegration.cs та його нащадки – використання об'єктів-нащадків IDBObject у необхідних запитах;
6. IDBObject.cs та його нащадки – представлення об'єктів БД;
7. IHuman.cs – класифікація сутності "Людина"; Зв'язок файлів показано на малюнку 4.4.

Для розробки онлайн бібліотеки можна використовувати різні патерни проектування, що допоможуть впоратися з різними завданнями. Ось деякі патерни, які можуть бути корисними у вашому додатку:



**Фабричний** метод (Factory Method): Дозволяє створювати об'єкти різних типів книг чи користувачів залежно від контексту чи вхідних даних.

Ось *приклад* коду на C#, що ілюструє реалізацію фабричного методу для створення об'єктів різних типів книг:

```
using System;

// Абстрактний клас Книга
public abstract class Book
{
    public abstract string Info();
}

// Конкретний клас Роман
public class Novel : Book
{
    public override string Info()
    {
        return "Це роман";
    }
}

// Конкретний клас Довідник
public class Handbook : Book
{
    public override string Info()
    {
        return "Це довідник";
    }
}

// Клас Фабрика книг
public class BookFactory
{
    public Book CreateBook(string bookType)
    {
        switch (bookType)
        {
            case "Роман":
                return new Novel();
            case "Довідник":
                return new Handbook();
            default:
                throw new ArgumentException("Непідтримуваний тип книги");
        }
    }
}
```

**Одинак** (Singleton): Може бути використаний для створення єдиного об'єкту бібліотеки, який буде доступний у всій програмі.

```

csharp Copy code

public class Singleton
{
    private static Singleton instance;

    // Приватний конструктор, щоб заборонити створення екземплярів класу ззовні
    private Singleton() { }

    // Метод для отримання єдиного екземпляру класу
    public static Singleton GetInstance()
    {
        if (instance == null)
        {
            instance = new Singleton();
        }
        return instance;
    }

    // Додаткові методи чи властивості
    public void SomeMethod()
    {
        Console.WriteLine("Виклик методу у Singleton класі.");
    }
}

// Приклад використання Singleton
class Program
{
    static void Main(string[] args)
    {
        Singleton obj1 = Singleton.GetInstance();
        Singleton obj2 = Singleton.GetInstance();

        if (obj1 == obj2)
        {
            Console.WriteLine("obj1 та obj2 вказують на той самий екземпляр");
        }

        obj1.SomeMethod(); // Виклик методу у єдиному екземплярі
    }
}

```

Метод **GetInstance** повертає той самий екземпляр класу **Singleton** кожного разу, коли він викликається. Перевірка **obj1 == obj2** демонструє, що ці об'єкти вказують на один і той же екземпляр класу **Singleton**.

Цей шаблон може бути корисним, коли потрібно мати лише один екземпляр певного класу для всієї програми, наприклад, для збереження глобальних налаштувань або кешування об'єктів.

**Команда** (Command): Дозволяє упаковувати запити на додавання, видалення чи редагування книг у окремі об'єкти, які можна передавати, відкладати чи відміняти.

**Спостерігач** (Observer): Дозволяє створювати систему сповіщень про зміни у стані книг або користувачів.

**Ітератор** (Iterator): Використовується для послідовного перегляду книг у бібліотеці.

**Кешування** (Caching): Дозволяє зберігати книги, які часто запитуються, у кеші для підвищення швидкодії доступу.

**Модель-вид-контролер** (Model-View-Controller, MVC): Допомогає розділити логіку програми, що обробляє бізнес-логіку (Модель), від візуального представлення даних (Вид), та управління взаємодією між ними (Контролер).

#### 4.3 Опис класів та методів застосування

Клас `Form1` містить поля для зберігання ID вибраного елемента в таблиці, таблицю, об'єкти класу `ViewDBObject` та `DBExtraReqRequests`. Клас містить методи, які обробляють всі взаємодії з інтерфейсом та передають дані для запитів. Клас `DBExtraReqRequests` містить методи для виконання відповідних додаткових запитів при натисканні кнопок та виводить їх на внутрішню таблицю на вкладці "Запити". Клас `Form2` містить методи обробки кнопки "Довідка" на вкладці "Запити", які виводять дані по всіх таблицях та шукають дані всередині таблиць. Клас `ViewDBObject` містить поле для обробки вибраної таблиці та множину масивів рядків для заповнення заголовків текстових полів на вкладці "Редагування". Клас містить методи, які виконують різні перетворення для передачі даних до запитів. Абстрактний клас `DBIntegration` містить поле для підключення до БД.

Класи `DBIntegration` та його нащадки містять методи для передачі відповідних нащадків `IDBObject` для виконання запитів до БД. Інтерфейс `IDBObject` містить обов'язкове визначення ID та методи конвертації масиву рядків в об'єкт та назад. Наслідники класу `IDBObject` перевизначають поле ID та доповнюються полями, які співпадають з колонками відповідних таблиць.

#### 4.4 Опис SQL-запитів до бази даних системи

У розробленій системі онлайн бібліотеки активно використовуються SQL-запити для реалізації взаємодії з базою даних. Нижче наведено основні типи запитів та їх функціональне призначення:

- **Отримання списку всіх книг:**

```
SELECT * FROM books;
```

Цей запит використовується для відображення всіх доступних книг на головній сторінці додатку.

- **Пошук книг за автором або назвою:**

```
SELECT * FROM books  
WHERE title LIKE '%ключове_слово%'  
OR author LIKE '%ключове_слово%';
```

Цей запит дозволяє користувачам здійснювати пошук літератури за назвою або ім'ям автора.

- **Додавання нової книги:**

```
INSERT INTO books (title, author, genre, year, description)  
VALUES ('Назва', 'Автор', 'Жанр', 2024, 'Короткий опис');
```

Застосовується адміністраторами бібліотеки для поповнення бази даних новими записами.

- **Формування звіту про боржників:**

```
SELECT users.name, users.email, borrowed_books.due_date  
FROM borrowed_books  
JOIN users ON borrowed_books.user_id = users.id  
WHERE borrowed_books.returned = 0 AND borrowed_books.due_date < CURDATE();
```

Цей запит визначає користувачів, які прострочили термін повернення книг.

- **Оновлення інформації про книгу:**

```
UPDATE books  
SET title = 'Нова назва', year = 2023  
WHERE id = 5;
```

Видалення книги з бази:

```
DELETE FROM books  
WHERE id = 10;
```

Всі SQL-запити реалізовані з урахуванням перевірки введених даних та захисту від SQL-ін'єкцій. Для цього використовується механізм параметризованих запитів, що підтримується у середовищі розробки через ORM або ADO.NET.

SQL запити класу DBInteractionBookLending (повний лістинг файлу в додатку Б1):

Форматування вибірки всієї таблиці Book\_lending:

«CALL SelectBookLending();»

Процедура SelectBookLending:

BEGIN

```
SELECT lending_id AS "ID выдачи", CONCAT_WS(" ", Librarian.surname,
Librarian.name, Librarian.patronymic) AS "Бібліотекар", CONCAT_WS("
", Reader.surname, Reader.name, Reader.patronymic) AS "Читач",
Books.book_name AS "Книга", lend_amount AS "Кількість",
DATE_FORMAT(date_of_issue, '%d.%m.%Y') AS "Дата видачі",
DATE_FORMAT(return_date, '%d.%m.%Y') AS "Дата повернення" FROM
Book_lending
```

```
INNER JOIN Librarian ON Librarian.lib_id = Book_lending.librarian_id
```

```
INNER JOIN Reader ON Reader.reader_id = Book_lending.reader_id
```

```
INNER JOIN Books ON Books.book_id = Book_lending.book_id
```

```
ORDER BY return_date;
```

END

Зформовання вибірка запису таблиці Book\_lending по ID:

"CALL SelectBookLendingByID({ID});":

Процедура SelectBookLendingByID:

BEGIN

```
SELECT lending_id, CONCAT_WS(" ", Librarian.surname, Librarian.name,
Librarian.patronymic), CONCAT_WS(" ", Reader.surname, Reader.name,
Reader.patronymic), Books.book_name, lend_amount,
```

```
DATE_FORMAT(date_of_issue, '%d.%m.%Y'), DATE_FORMAT(return_date,
'%d.%m.%Y') FROM Book_lending
```

```
INNER JOIN Librarian ON Librarian.lib_id = Book_lending.librarian_id
```

```
INNER JOIN Reader ON Reader.reader_id = Book_lending.reader_id
```

```
INNER JOIN Books ON Books.book_id = Book_lending.book_id
```

```
WHERE lending_id = ID;
```

```
END
```

Форматованная ви́бірка запису таблиці Book\_lending по запиту:

```
"CALL BookLendingSearch('{query}')"

```

Процедура BookLendingSearch:

```
SELECT lending_id AS "ID Видачі", CONCAT_WS(" ", Librarian.surname,
Librarian.name, Librarian.patronymic) AS "Бібліотекар", CONCAT_WS("
", Reader.surname, Reader.name, Reader.patronymic) AS "Читач",
Books.book_name AS "Книга", lend_amount AS "Кількість",
DATE_FORMAT(date_of_issue, '%d.%m.%Y') AS "Дата видачі",
DATE_FORMAT(return_date, '%d.%m.%Y') AS "Дата відання" FROM
Book_lending
```

```
INNER JOIN Librarian ON Librarian.lib_id = Book_lending.librarian_id
```

```
INNER JOIN Reader ON Reader.reader_id = Book_lending.reader_id
```

```
INNER JOIN Books ON Books.book_id = Book_lending.book_id
```

```
WHERE LOCATE(searchQuery, CONCAT_WS(" ", lending_id,
Librarian.surname, Librarian.name, Librarian.patronymic, Reader.surname,
Reader.name, Reader.patronymic, Books.book_name, lend_amount, date_of_issue,
DATE_FORMAT(date_of_issue, '%d.%m.%Y'), DATE_FORMAT(return_date,
'%d.%m.%Y')))) >= 1 ORDER BY return_date;
```

Додавання запису в таблицю Book\_lending:

```
"CALL BookLendingInsert('{item.librarian}', '{item.reader}', '{item.book}',
'{item.lendAmount}', '{item.returnDate}')"

```

Процедура BookLendingInsert:

BookLendingInsert:BEGIN

DECLARE librarianID INT;

DECLARE readerID INT;

DECLARE bookID INT;

SELECT lib\_id INTO librarianID FROM Librarian WHERE  
LOCATE(librarianInfo, CONCAT\_WS(" ", lib\_id, surname, name, patronymic)) >=  
1;

SELECT reader\_id INTO readerID FROM Reader WHERE  
LOCATE(readerInfo, CONCAT\_WS(" ", reader\_id, surname, name, patronymic))  
>= 1;

SELECT book\_id INTO bookID FROM Books WHERE  
LOCATE(bookInfo, CONCAT\_WS(" ", book\_name, ISBN)) >= 1;

INSERT INTO Book\_lending (lending\_id, librarian\_id, reader\_id,  
book\_id, lend\_amount, date\_of\_issue, return\_date) VALUES (NULL, librarianID,  
readerID, bookID, lendAmount, NOW(), returnDate);

Змінення запису в таблиці Book\_lending:

"CALL BookLendingChange('{currentID}', '{item.ID}', '{item.librarian}',  
'{item.reader}', '{item.book}', '{item.lendAmount}', '{item.dateOfIssue}',  
'{item.returnDate}')

Процедура BookLendingChange:

BEGIN

DECLARE librarianID INT;

DECLARE readerID INT;

DECLARE bookID INT;

```
SELECT lib_id INTO librarianID FROM Librarian WHERE
LOCATE(librarianInfo, CONCAT_WS(" ", lib_id, surname, name, patronymic)) >=
1;
```

```
SELECT reader_id INTO readerID FROM Reader WHERE
LOCATE(readerInfo, CONCAT_WS(" ", reader_id, surname, name, patronymic))
>= 1;
```

```
SELECT book_id INTO bookID FROM Books WHERE LOCATE(bookInfo,
CONCAT_WS(" ", book_name, ISBN)) >= 1;
```

```
UPDATE Book_lending SET lending_id = lendingID, librarian_id = librarianID,
reader_id = readerID, book_id = bookID, lend_amount = lendingAmount,
date_of_issue = issueDate, return_date = returnDate WHERE lending_id =
currentLendingID;
```

Видалення запису в таблиці Book\_lending:

```
"DELETE FROM Book_lending WHERE Book_lending.lending_id = {ID}"
```

Триггери:

Триггер вичитання кількості виданих книг при додаванні запису видачі:

```
CREATE TRIGGER `lending_after_insert` AFTER INSERT ON
`Book_lending`
FOR EACH ROW UPDATE Books SET Books.amount = (Books.amount -
new.lend_amount) WHERE Books.book_id = NEW.book_id
```

Триггери змінення кількості виданих книг до та після змінення запису видачі соотвественно:

```
CREATE TRIGGER `lending_before_update` BEFORE UPDATE ON
`Book_lending`
```



```
FOR EACH ROW UPDATE Books SET Books.amount = (Books.amount +
OLD.lend_amount)
```

```
WHERE Books.book_id = OLD.book_id
```

```
CREATE TRIGGER `lending_after_update` AFTER UPDATE ON
`Book_lending`
```

```
FOR EACH ROW UPDATE Books SET Books.amount = (Books.amount -
NEW.lend_amount)
```

```
WHERE Books.book_id = NEW.book_id
```

Триггер додавання видачі книг при видаленні запису о видачі:

```
CREATE TRIGGER `lending_before_delete` BEFORE DELETE ON
`Book_lending`
```

```
FOR EACH ROW UPDATE Books SET Books.amount = (Books.amount +
OLD.lend_amount)
```

```
WHERE Books.book_id = OLD.book_id
```

Виборка читачів, які не повернули книги своєчасно:

Процедура SelectDebtors:

```
SELECT      CONCAT_WS("      ",Reader.surname,      Reader.name,
Reader.patronymic) AS      "Должник",      DATE_FORMAT(date_of_issue,
"%d.%m.%Y") AS      "Дата      выдачи",      DATE_FORMAT(return_date,
"%d.%m.%Y") AS "Дата возврата", TIMESTAMPDIFF(DAY, return_date,
NOW()) AS "Задолженность (дн.)" FROM Book_lending
```

```
INNER JOIN Librarian ON Librarian.lib_id = Book_lending.librarian_id
```

```
INNER JOIN Reader ON Reader.reader_id = Book_lending.reader_id
```

```
INNER JOIN Books ON Books.book_id = Book_lending.book_id
```

```
WHERE TIMESTAMPDIFF(DAY, return_date, NOW()) > 0
```

```
ORDER BY TIMESTAMPDIFF(DAY, return_date, NOW()) DESC;
```

## 5. РОЗДІЛ 5. ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

### 5.1 Апаратні та програмні засоби створення та експлуатації програми

Для роботи додатку потрібне наступне апаратне забезпечення:

1. процесор з тактовою частотою 1.0 ГГц;
2. ОЗУ об'ємом не менше 1 Гб;
3. вільне місце на жорсткому диску 300 Мб;
4. відеокарта з об'ємом пам'яті не менше 514 Мб;
5. операційна система Windows 10;
6. наявність засобів введення-виведення: миша, клавіатура, монітор;

Програмні вимоги до додатку. На комп'ютері повинно бути встановлене наступне програмне забезпечення:

1. операційна система - Windows 10;
2. будь-який браузер;
3. платформа Microsoft .NET 6.0;
4. утиліта - Open Server Panel з включеним модулем MySQL;

### 5.2 Інструкція користувача

Для роботи додатку необхідно: Встановити на комп'ютері програмну середу .NET FRAMEWORK 6.0 з офіційного сайту Майкрософт. Встановити утиліту Open Server Panel з офіційного сайту Open Server. Налаштувати утиліту на роботу з СУБД MySQL: налаштування - модулі - MySQL-8.0-Win10. Після цього перезапустити програму. Извлечь из папки файл Library.sql в любую удобную директорию.

X из Open Server Panel PHPMyAdmin.

Войти в СУБД (логин: root, пароль: *пустая строка*).

Натисніть «Імпорт» та вибрати Library.sql. Налаштувати користувача root: облікові записи – root – змінити пароль – встановити пароль «root».

Можна закрити всі непотрібні вікна. Перейти до папки, в якій знаходиться файл LibraryDB.exe, і зробити на ньому подвійний клік мишкою (примітка: Open Server Panel повинен бути в стані «Запустити»). На малюнку 5.1 виділено файл додатку та імпортована база даних.

Имя	Дата изменения	Тип	Размер
BouncyCastle.Crypto.dll	19.10.2021 11:53	Расширение при...	3 241 КБ
Google.Protobuf.dll	28.01.2022 8:49	Расширение при...	394 КБ
K4os.Compression.LZ4.dll	12.09.2020 1:11	Расширение при...	59 КБ
K4os.Compression.LZ4.Streams.dll	12.09.2020 1:11	Расширение при...	37 КБ
K4os.Hash.xxHash.dll	03.07.2019 20:39	Расширение при...	12 КБ
Library.sql	14.12.2022 0:03	Исходный файл S...	18 КБ
LibraryDB.deps.json	13.12.2022 23:34	JSON File	3 КБ
LibraryDB.dll	13.12.2022 23:35	Расширение при...	81 КБ
LibraryDB.exe	13.12.2022 23:35	Приложение	153 КБ
LibraryDB.pdb	13.12.2022 23:35	Program Debug D...	30 КБ
LibraryDB.runtimeconfig.json	11.12.2022 14:38	JSON File	1 КБ
main-ico.ico	21.11.2022 16:27	Файл "ICO"	8 КБ
MySQL.Data.dll	09.09.2022 14:08	Расширение при...	1 492 КБ
MySQL.Data.xml	09.09.2022 14:03	Документ XML	923 КБ
readme.txt	14.12.2022 15:37	Текстовый докум...	1 КБ
ZstdNet.dll	09.09.2022 14:03	Расширение при...	28 КБ

Рисунок 5.1 – Файлова система додатку

### 5.3 Опис контрольних прикладів

Для запуску додатку необхідно натиснути двічі лівою кнопкою миші на файл LibraryDB.exe, після чого відкриється головна сторінка додатку (рис. А1). Для початку роботи з таблицями їх потрібно завантажити. Для цього необхідно натиснути на випадаючий список таблиць та вибрати потрібну таблицю (наприклад, "Видача книг"). Після цього вікно оновиться та завантажить дані (рис. А2). Користувач може скористатися пошуком у вибраній таблиці. Для цього необхідно натиснути на текстове поле "Пошук:" та ввести

натиснути клавішу Enter після введення запиту. У вкладці "Таблиця" відображатимуться результати пошуку (рис. А8). При натисканні на кнопки

"Додати", "Змінити", "Видалити" на вкладці будь-який результат операції оновить таблицю на вкладці "Таблиця", а вкладка "Редагування" очиститься. Для додавання запису необхідно перейти на вкладку "Редагування". Потім можна ввести необхідні дані та натиснути кнопку "Додати" (рис. А3). Після цієї операції таблиця на вкладці "Таблиця" оновиться, а вкладка "Редагування" очиститься. Для зміни або видалення запису необхідно двічі клацнути на відповідну запис та дані будуть додані на вкладку "Редагування" (рис. А5).

У користувача на вкладці "Редагування" є можливість викликати довідкове вікно, що надасть необхідну інформацію для введення з можливістю пошуку (рис. А4). Для копіювання необхідної запису потрібно вибрати її клацанням миші та натиснути комбінацію клавіш Ctrl + C. Для вставки даних - Ctrl + V. Для зміни необхідно оновити відповідні текстові поля та натиснути кнопку "Змінити" (рис. А6). Для видалення необхідно натиснути кнопку "Видалити" (рис. А6). Для виконання додаткових запитів потрібно натиснути на вкладку "Запити" та вибрати необхідний запит, після чого таблиця на вкладці "Запити" покаже результати (рис. А7). Для виходу з додатку потрібно закрити вікно.

## ВИСНОВОК

Проектування програмного забезпечення для онлайн бібліотеки - це складний та багатогранний процес, що вимагає уважного аналізу вимог, ефективного архітектурного планування та гнучкості під час реалізації.

Під час розробки дипломної я виконав основні етапи цієї роботи які включають в себе детальний огляд актуальних аспектів, визначення цілей проєкту, встановлення вимог користувачів та технічного завдання, обґрунтування методів проектування та архітектурних рішень.

Створення програмного продукту, спрямованого на розвиток онлайн бібліотеки, вимагає уваги до деталей та активної участі розробників у процесі роботи над проєктом. Ключовими аспектами успішного проектування є не лише технічні знання, а й уміння ефективно спілкуватися в команді, гнучко адаптуватися до змін та активно враховувати вимоги користувачів.

Також була проведена розробка технічного завдання яке було реалізовано паралельно, визначення архітектурних вирішень, вивчення та застосування методів проектування та управління - усе це необхідно для успішної розробки програмного продукту, який відповідатиме вимогам ринку та задовольнятиме потреби користувачів. Адаптивність, відкритість до змін та постійне удосконалення - ключові аспекти, що дозволять створити ефективну та користувацьки орієнтовану онлайн бібліотеку.

Також у результаті вирішення технічної задачі було спроектовано базу даних та систему, яка автоматизує процес роботи бібліотеки. Проєкт дозволяє зберігати та шукати необхідні дані, а також взаємодіяти з ними через зручний додаток. Додаток дозволяє обробляти дані будь-якого рядка в БД: додавати, змінювати, видаляти та шукати рядки в усіх наданих таблицях. Додаток також може автоматично (з перевіркою) віднімати кількість тих книг, які видавали читачам, та додавати відповідну кількість до повернутих книг, а також знаходити читачів, які молодші за 18 років, читачів, які не повернули

книги вчасно та книги, які закінчились на складі. Додаток LibraryDB може використовуватись приватними та муніципальними бібліотеками малого та середнього рівня для автоматизації ведення обліку діяльності своєї бібліотеки. У майбутньому планується розширення функціоналу додатку та оптимізація його роботи з великими обсягами даних. З урахуванням проведеної роботи можна стверджувати, що поставлені цілі та задачі в проєктувальному та технічному завданні були виконані.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. Затв. Наказом Міністерства соціальної політики України 14.02.2018. Офіційний вісник України від 18.05.2018 – 2018 р., No 38, стор. 121, стаття 1352, код акта 90123/2018 (URL: <https://zakon.rada.gov.ua/laws/show/z0508-18>).
2. Бьерн Страуструп. Язык программирования C++. Специальное издание. Пер. с англ. – М.: Издательство Бином, 2015 г. – 1136 с.: ил. ISBN 978-5-7989-0425-9.
3. Прата, Стивен. Язык программирования C++. Лекции и упражнения, 5-е изд. Пер. с англ. – М.: 000 "И.Д. Вильямс", 2007. – 1184 с.: ил. ISBN 5-8459-1127-3.
4. Седжвик, Роберт. Алгоритмы на C++. – М.: Издательский дом "Вильямс", 2011. ISBN: 978-5-8459-1650-1, 978-0-321-60633-4.
5. Грицюк Ю.І., Рак Т.Є. Об'єктно-орієнтоване програмування мовою C++: навчальний посібник. – Львів: Вид-во Львівського ДУ БЖД, 2011. – 404 с. ISBN 978-966-3466-86-3.
6. Шилдт, Герберт. C++: руководство для начинающих, 2-е издание. Пер. с англ. – М.: Издательский дом "Вильямс", 2005. – 672 с.: ил. ISBN 5-8459-0840-X.
7. C/C++. Программирование на языке высокого уровня / Т.А. Павловская. – СПб.: Питер, 2003. – 461 с: ил. ISBN 5-94723-568-4.
9. Франка П. C++: учебный курс. – СПб.: Питер, 2003. – 521 с.: ил. ISBN 5-314-00136-5.
8. Лаврищева Е.М. , Петрухин В.А. Методы и средства инженерии программного обеспечения. – Учебник, Москва, 2006. – 304 с.
9. С. Орлов. Технологии разработки программного обеспечения: Учебник/— СПб.: Питер, 2002. — 464 с.: ил.

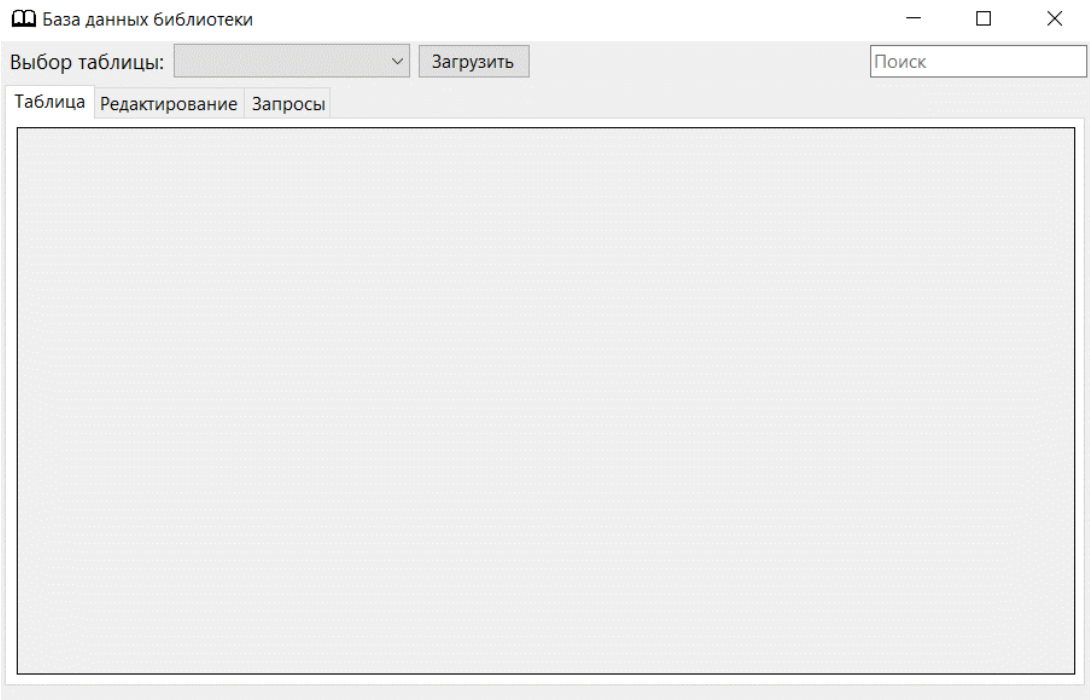


10. Дж.Рамбо, М. Блах. UML 2.0. Объектно-ориентированное моделирование и разработка: Питер, 2007. – 544с.

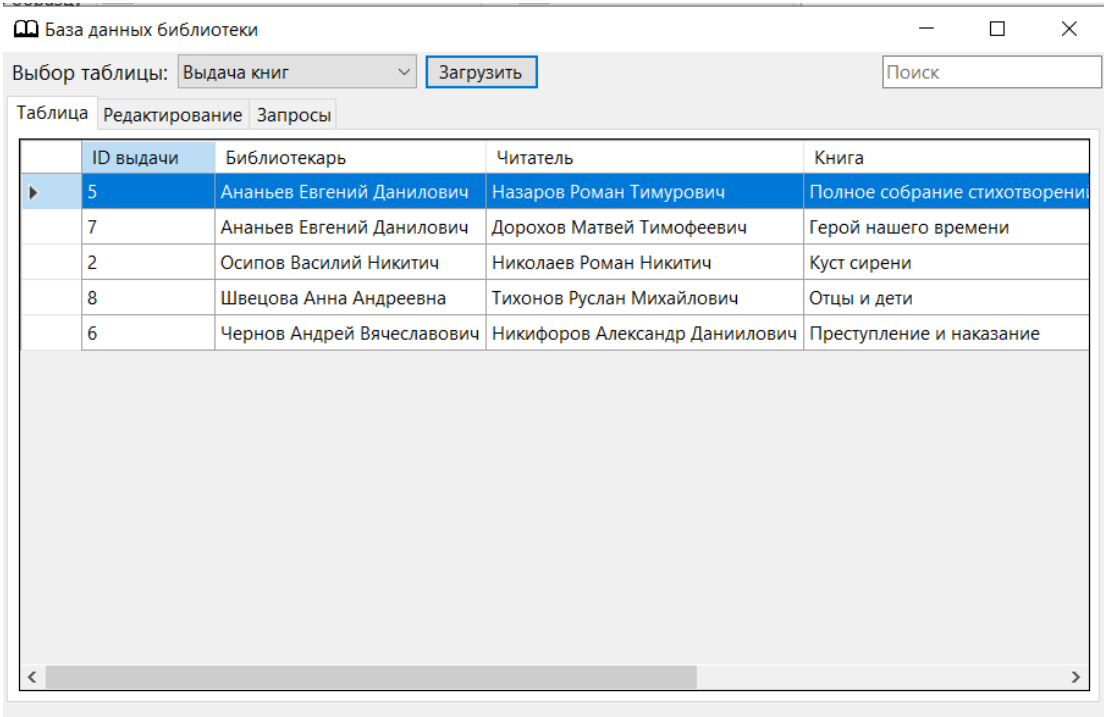
11. Лавріщева К.М. Програмна інженерія.–Київ.– 2008.–319 с

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ ([Окремий файл](#))

ДОДАТОК Б ЕКРАННІ ФОРМИ



Малюнок А1 – Головне вікно додатку



Малюнок А2 – Головне вікно з завантаженою таблицею

База данных библиотеки

Выбор таблицы: **Выдача книг**

Таблица Редактирование Запросы

ID выдачи (не используется в добавлении)

Библиотекарь  
Швецова Анна

Читатель  
Тихонов Руслан

Книга (ISBN/Название)  
Преступление и наказание

Кол-во  
1

Дата выдачи (автоопределение при добавлении)

Дата возврата  
12.12.2023

Запись

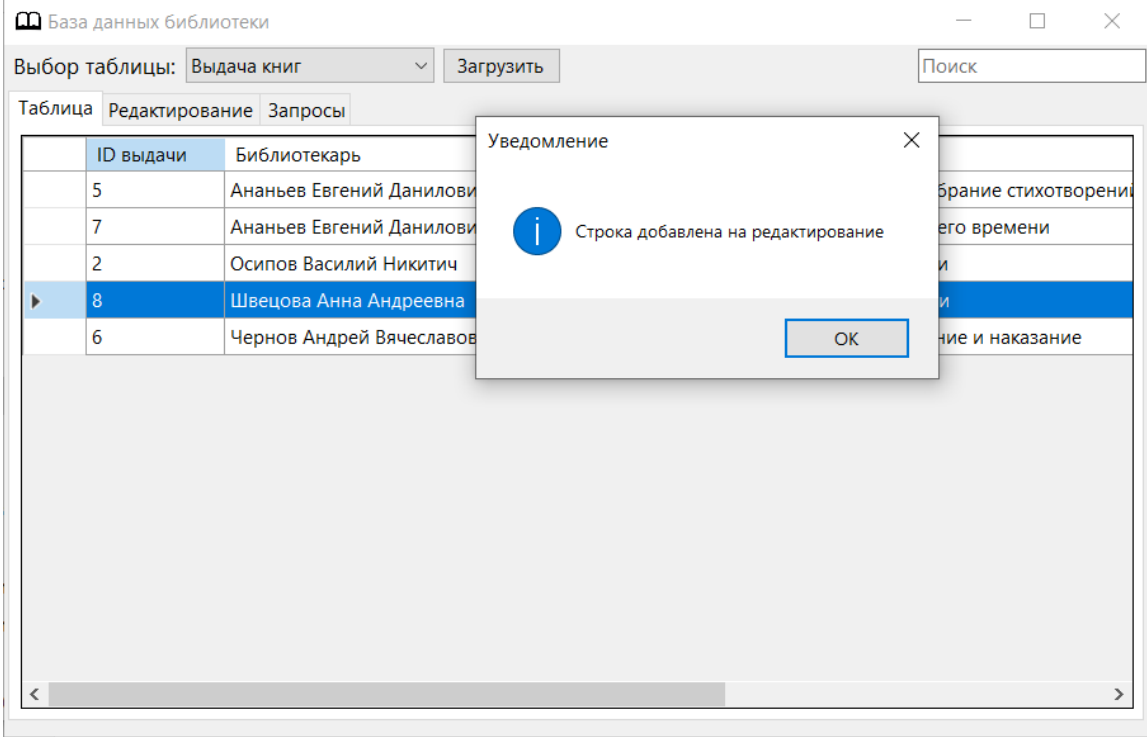
Малюнок А3 – Вікно запису інформації про видачу книги

Справка из БД

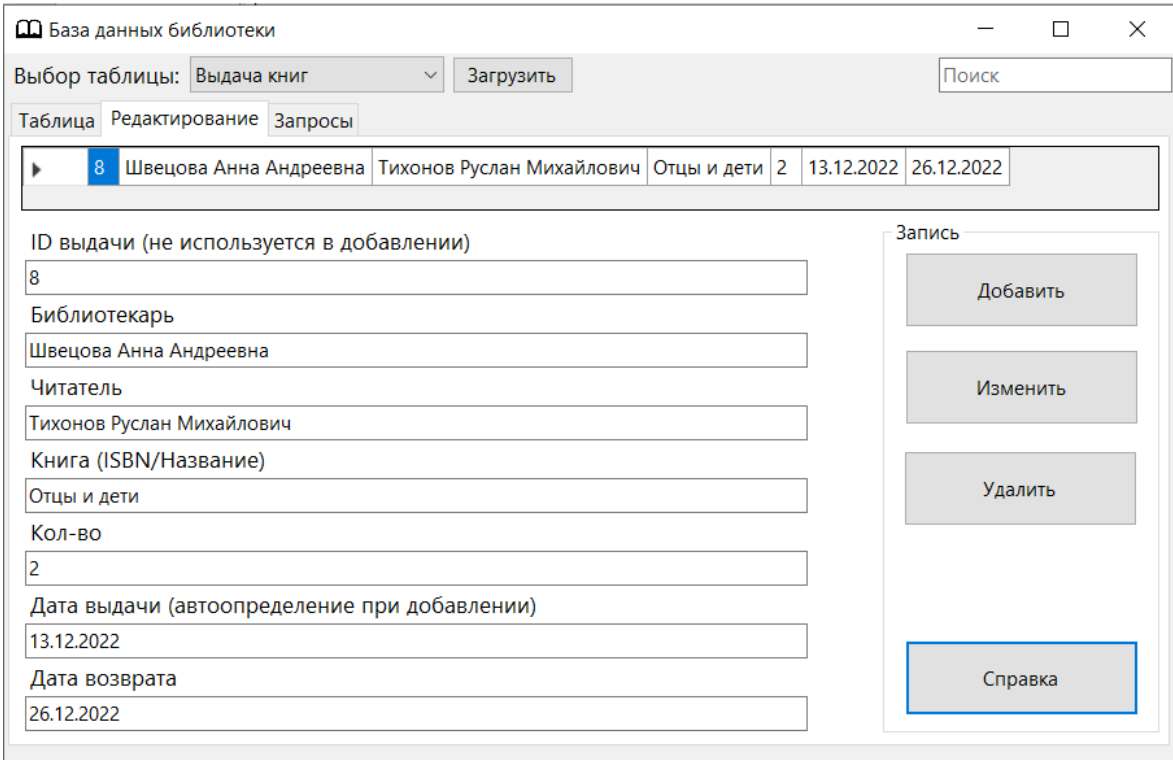
Выбор таблицы: **Читатели**

Выдача книг  
Книги  
**Читатели**  
Библиотекари  
Жанры  
Издания  
Возрастной рейтинг  
Размещение

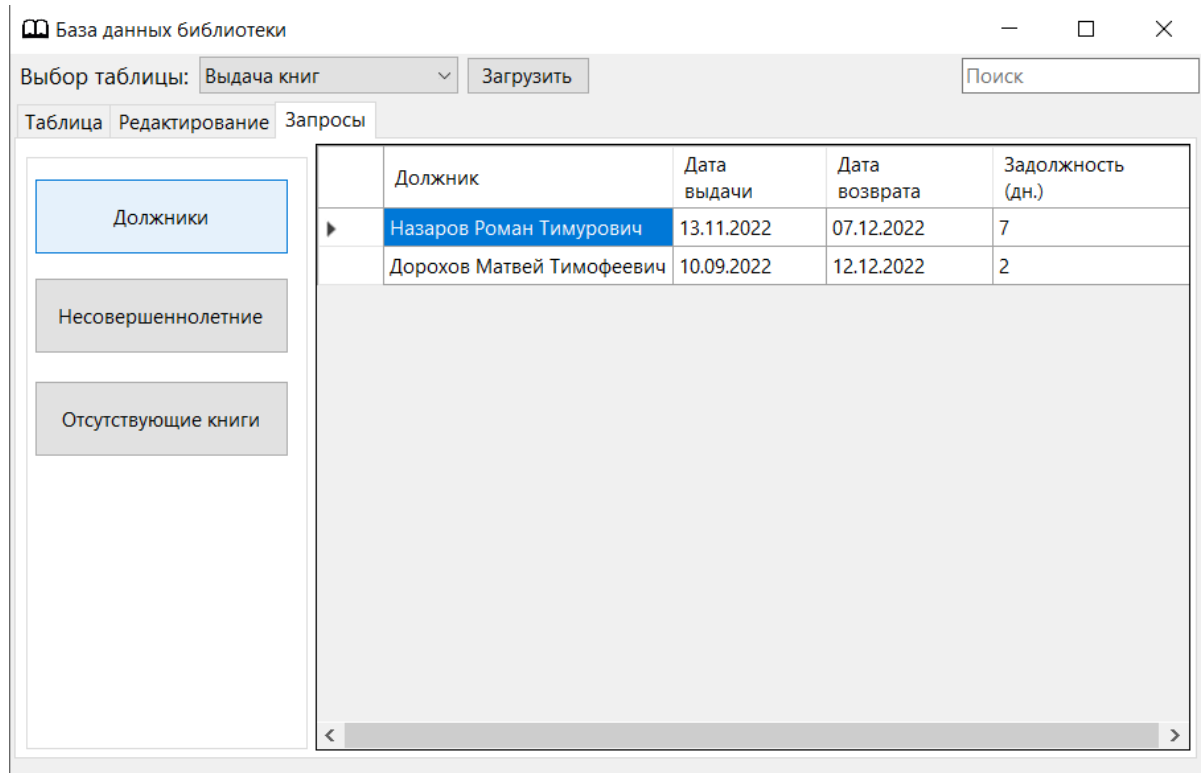
Малюнок А4 – Справочне вікно



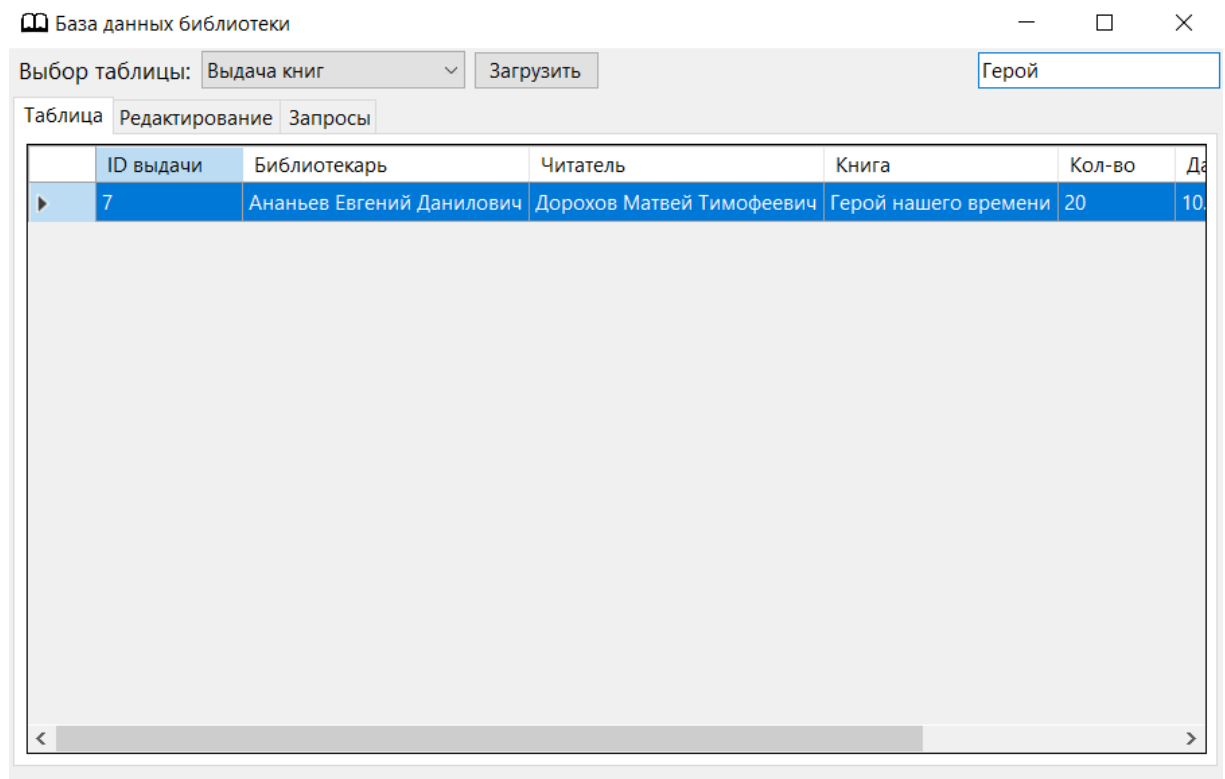
Малюнок А5 – Додавання строки для редагування



Малюнок А6 – Вкладка «Редагування»



Малюнок А7 – Вкладка «Запити»



Малюнок А8 – Результат виконання пошуку

## ДОДАТОК В

### Фрагменти Лістингу

#### Лістинг Б1 – Файл «BookLending.cs»

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LibraryDB.Objects
{
    internal class BookLending : IDBObject<BookLending>
    {
        public int ID { get; set; }
        public string librarian { get; set; }
        public string reader { get; set; }
        public string book { get; set; }
        public int lendAmount { get; set; }
        public string dateOfIssue { get; set; }
        public string returnDate { get; set; }

        public string[] convertToStrArr()
        {
            string[] arr = { ID.ToString(), librarian, reader, book, lendAmount.ToString(), dateOfIssue,
returnDate };
            return arr;
        }

        public BookLending convertStrArrToObj(string[] obj, bool convert)
        {
            try
            {
                BookLending BL = new BookLending();
                BL.ID = Convert.ToInt32(obj[0]);
                BL.librarian = obj[1];
                BL.reader = obj[2];
                BL.book = obj[3];
                BL.lendAmount = Convert.ToInt32(obj[4]);
                if ((DateTime.TryParse(obj[5], out DateTime date5)))
                {
                    if (convert == false)
                    {
                        BL.dateOfIssue = date5.ToString("dd.MM.yyyy");
                    }
                    else
                    {
                        BL.dateOfIssue = date5.ToString("yyyy-MM-dd");
                    }
                }
                if (DateTime.TryParse(obj[6], out DateTime date6))
                {
                    if (convert == false)
                    {

```

```

        BL.returnDate = date6.ToString("dd.MM.yyyy");
    }
    else
    {
        BL.returnDate = date6.ToString("yyyy-MM-dd");
    }
}
return BL;
}
catch (Exception)
{
    throw;
}
}
}
}

```

### Лістниг Б2 – Файл «DBInteractionBookLending.cs»

```

using LibraryDB.Objects;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MySql.Data.MySqlClient;

namespace LibraryDB.DB
{
    internal class DBInteractionBookLending : DBInteraction<BookLending>
    {
        public override DataTable getAll()
        {
            try
            {
                MySqlConnection connection = new MySqlConnection(connString);
                connection.Open();
                string sqlcmdString = "CALL SelectBookLending()";
                MySqlDataAdapter adapter = new MySqlDataAdapter(sqlcmdString, connection);
                DataTable table = new DataTable();
                table.Clear();
                adapter.Fill(table);
                connection.Close();
                return table;
            }
            catch (MySqlException ex)
            {
                throw ex;
            }
        }

        public override BookLending getRow(int ID)
        {
            try
            {
                BookLending DBRow = new BookLending();
                MySqlConnection connection = new MySqlConnection(connString);
                connection.Open();
            }
        }
    }
}

```

```

string sqlcmdString = $"CALL SelectBookLendingByID({ID})";
MySQLCommand sqlcmd = new MySQLCommand(sqlcmdString, connection);
using (MySQLDataReader reader = sqlcmd.ExecuteReader())
{
    if (reader.Read())
    {
        DBRow.ID = reader.GetInt32(0);
        DBRow.librarian = reader.GetString(1);
        DBRow.reader = reader.GetString(2);
        DBRow.book = reader.GetString(3);
        DBRow.lendAmount = reader.GetInt32(4);
        DBRow.dateOfIssue = reader.GetString(5);
        DBRow.returnDate = reader.GetString(6);
    }
    connection.Close();
    return DBRow;
}
}
catch (MySQLException ex)
{
    throw ex;
}
}

public override DataTable search(string query)
{
    try
    {
        MySqlConnection connection = new MySqlConnection(connString);
        connection.Open();
        string sqlcmdString = $"CALL BookLendingSearch('{query}')";
        MySQLDataAdapter adapter = new MySQLDataAdapter(sqlcmdString, connection);
        DataTable table = new DataTable();
        table.Clear();
        adapter.Fill(table);
        connection.Close();
        return table;
    }
    catch (MySQLException ex)
    {
        throw ex;
    }
}

public override void add(BookLending item)
{
    try
    {
        MySqlConnection connection = new MySqlConnection(connString);
        connection.Open();
        string sqlcmdString = $" CALL BookLendingInsert('{item.librarian}', '{item.reader}',
'{item.book}', '{item.lendAmount}', '{item.returnDate}')";
        MySQLCommand sqlcmd = new MySQLCommand(sqlcmdString, connection);
        sqlcmd.ExecuteNonQuery();
        connection.Close();
    }
    catch (MySQLException ex)
    {
        throw ex;
    }
}

```



```

    }
}

public override void update(int currentID, BookLending item)
{
    try
    {
        MySqlConnection connection = new MySqlConnection(connString);
        connection.Open();
        string sqlcmdString = $"CALL BookLendingChange('{currentID}', '{item.ID}', '{item.librarian}',
'{item.reader}', '{item.book}', '{item.lendAmount}', '{item.dateOfIssue}', '{item.returnDate}')";
        MySqlCommand sqlcmd = new MySqlCommand(sqlcmdString, connection);
        sqlcmd.ExecuteNonQuery();
        connection.Close();
    }
    catch (MySqlException ex)
    {
        throw ex;
    }
}

public override void delete(int ID)
{
    try
    {
        MySqlConnection connection = new MySqlConnection(connString);
        connection.Open();
        string sqlcmdString = $"DELETE FROM Book_lending WHERE Book_lending.lending_id =
{ID}";
        MySqlCommand sqlcmd = new MySqlCommand(sqlcmdString, connection);
        sqlcmd.ExecuteNonQuery();
        connection.Close();
    }
    catch (MySqlException ex)
    {
        throw ex;
    }
}
}
}

```

### Лістинг Б3 – Файл «ViewDBObject.cs»

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using LibraryDB.Objects;
using LibraryDB.DB;
using System.Data;
using MySql.Data.MySqlClient;

namespace LibraryDB.View
{
    internal class ViewDBObject
    {
        public int chosenTableID;
    }
}

```

```

    public string[] bookLendingLabels = { "ID выдачи (не используется в добавлении)",
"Библиотекарь", "Читатель", "Книга (ISBN/Название)", "Кол-во",
    "Дата выдачи (автоопределение при добавлении)", "Дата возврата" };
    public string[] bookLabels = { "ID книги (не используется в добавлении)", "Название книги",
"Автор", "Жанр", "Издание", "Возрастной рейтинг",
    "Расположение", "ISBN", "Издатель", "Дата издания", "Кол-во стр.", "Стоимость (руб.)", "Кол-
во" };
    public string[] readerLabels = { "ID читателя", "Фамилия", "Имя", "Отчество", "Дата рождения",
"Адрес проживания", "Номер телефона", "Дата регистрации" };
    public string[] librarianLabels = { "ID библиотекаря", "Фамилия", "Имя", "Отчество" };
    public string[] genreLabels = { "ID записи (не используется в добавлении)", "Жанр" };
    public string[] editionLabels = { "ID записи (не используется в добавлении)", "Тип" };
    public string[] ratingLabels = { "ID записи (не используется в добавлении)", "Возрастной
рейтинг" };
    public string[] storageLabels = { "ID записи (не используется в добавлении)", "Расположение" };

    public enum mode
    {
        add,
        upd,
        del
    }

    public DataTable chooseTable(string table)
    {
        DataTable dt = new DataTable();
        switch (table)
        {
            case "Выдача книг":
                choosenTableID = 1;
                DBInteractionBookLending BookLendingsDB = new DBInteractionBookLending();
                dt = BookLendingsDB.getAll();
                break;
            case "Книги":
                choosenTableID = 2;
                DBInteractionBooks BooksDB = new DBInteractionBooks();
                dt = BooksDB.getAll();
                break;
            case "Читатели":
                choosenTableID = 3;
                DBInteractionReader ReadersDB = new DBInteractionReader();
                dt = ReadersDB.getAll();
                break;
            case "Библиотекари":
                choosenTableID = 4;
                DBInteractionLibrarian LibrariansDB = new DBInteractionLibrarian();
                dt = LibrariansDB.getAll();
                break;
            case "Жанры":
                choosenTableID = 5;
                DBInteractionGenre GenresDB = new DBInteractionGenre();
                dt = GenresDB.getAll();
                break;
            case "Издания":
                choosenTableID = 6;
                DBInteractionEdition EditionsDB = new DBInteractionEdition();
                dt = EditionsDB.getAll();
                break;
            case "Возрастной рейтинг":
                choosenTableID = 7;

```

```

        DBInteractionRating RatingsDB = new DBInteractionRating();
        dt = RatingsDB.getAll();
        break;
    case "Размещение":
        choosenTableID = 8;
        DBIntegrationStorage StorageDB = new DBIntegrationStorage();
        dt = StorageDB.getAll();
        break;
    default:
        MessageBox.Show($"Возникла ошибка загрузки таблицы:\nНе удалось загрузить
таблицу. Возможно, отсутствует подключение к серверу", "Неизвестная ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        break;
    }
    return dt;
}

public DataTable searchRows(string query)
{
    DataTable dt = new DataTable();
    switch (choosenTableID)
    {
        case 1:
            DBInteractionBookLending BookLendingsDB = new DBInteractionBookLending();
            dt = BookLendingsDB.search(query);
            break;
        case 2:
            DBInteractionBooks BooksDB = new DBInteractionBooks();
            dt = BooksDB.search(query);
            break;
        case 3:
            DBInteractionReader ReadersDB = new DBInteractionReader();
            dt = ReadersDB.search(query);
            break;
        case 4:
            DBInteractionLibrarian LibrariansDB = new DBInteractionLibrarian();
            dt = LibrariansDB.search(query);
            break;
        case 5:
            DBInteractionGenre GenresDB = new DBInteractionGenre();
            dt = GenresDB.search(query);
            break;
        case 6:
            DBInteractionEdition EditionsDB = new DBInteractionEdition();
            dt = EditionsDB.search(query);
            break;
        case 7:
            DBInteractionRating RatingsDB = new DBInteractionRating();
            dt = RatingsDB.search(query);
            break;
        case 8:
            DBIntegrationStorage StorageDB = new DBIntegrationStorage();
            dt = StorageDB.search(query);
            break;
        default:
            MessageBox.Show($"Возникла ошибка загрузки таблицы:\nНе удалось найти данные.
Возможно, отсутствует подключение к серверу", "Неизвестная ошибка", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
            break;
    }
}

```

```

    return dt;
}

public string[] getStringRow(int ID)
{
    switch (chosenTableID)
    {
        case 1:
            DBInteractionBookLending BookLendingsDB = new DBInteractionBookLending();
            BookLending BL = BookLendingsDB.getRow(ID);
            string[] strBL = BL.convertToStrArr();
            return strBL;
        case 2:
            DBInteractionBooks BooksDB = new DBInteractionBooks();
            Books BK = BooksDB.getRow(ID);
            string[] strBK = BK.convertToStrArr();
            return strBK;
        case 3:
            DBInteractionReader ReadersDB = new DBInteractionReader();
            Reader RD = ReadersDB.getRow(ID);
            string[] strRD = RD.convertToStrArr();
            return strRD;
        case 4:
            DBInteractionLibrarian LibrariansDB = new DBInteractionLibrarian();
            Librarian LB = LibrariansDB.getRow(ID);
            string[] strLB = LB.convertToStrArr();
            return strLB;
        case 5:
            DBInteractionGenre GenresDB = new DBInteractionGenre();
            Genre GN = GenresDB.getRow(ID);
            string[] strGN = GN.convertToStrArr();
            return strGN;
        case 6:
            DBInteractionEdition EditionsDB = new DBInteractionEdition();
            Edition ED = EditionsDB.getRow(ID);
            string[] strED = ED.convertToStrArr();
            return strED;
        case 7:
            DBInteractionRating RatingsDB = new DBInteractionRating();
            Rating RT = RatingsDB.getRow(ID);
            string[] strRT = RT.convertToStrArr();
            return strRT;
        case 8:
            DBIntegrationStorage StorageDB = new DBIntegrationStorage();
            Storage ST = StorageDB.getRow(ID);
            string[] strST = ST.convertToStrArr();
            return strST;
        default:
            MessageBox.Show($"Возникла ошибка загрузки таблицы:\nНе удалось найти данные. Возможно, отсутствует подключение к серверу", "Неизвестная ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            string[] empty = new string[0];
            return empty;
    }
}

public void viewModeRowDB(string[] obj, int ID, mode mode)
{
    try
    {

```

```

for (int i = 0; i < obj.Length; i++)
{
    obj[i] = System.Text.RegularExpressions.Regex.Replace(obj[i], @"\s+", " ");
}
switch (chosenTableID)
{
    case 1:
        DBInteractionBookLending BookLendingsDB = new DBInteractionBookLending();
        BookLending BL = new BookLending();
        switch (mode)
        {
            case mode.add:
                obj[0] = "-1";
                BL = BL.convertStrArrToObj(obj, true);
                BookLendingsDB.add(BL);
                break;
            case mode.upd:
                BL = BL.convertStrArrToObj(obj, true);
                BookLendingsDB.update(ID, BL);
                break;
            case mode.del:
                BookLendingsDB.delete(ID);
                break;
        }
        break;
    case 2:
        DBInteractionBooks BooksDB = new DBInteractionBooks();
        Books BK = new Books();
        switch (mode)
        {
            case mode.add:
                obj[0] = "-1";
                BK = BK.convertStrArrToObj(obj, true);
                BooksDB.add(BK);
                break;
            case mode.upd:
                BK = BK.convertStrArrToObj(obj, true);
                BooksDB.update(ID, BK);
                break;
            case mode.del:
                BooksDB.delete(ID);
                break;
        }
        break;
    case 3:
        DBInteractionReader ReadersDB = new DBInteractionReader();
        Reader RD = new Reader();
        switch (mode)
        {
            case mode.add:
                RD = RD.convertStrArrToObj(obj, true);
                ReadersDB.add(RD);
                break;
            case mode.upd:
                RD = RD.convertStrArrToObj(obj, true);
                ReadersDB.update(ID, RD);
                break;
            case mode.del:
                ReadersDB.delete(ID);
                break;
        }
    }
}

```

```

    }
    break;
case 4:
    DBInteractionLibrarian LibrariansDB = new DBInteractionLibrarian();
    Librarian LB = new Librarian();
    switch (mode)
    {
        case mode.add:
            LB = LB.convertStrArrToObj(obj, true);
            LibrariansDB.add(LB);
            break;
        case mode.upd:
            LB = LB.convertStrArrToObj(obj, true);
            LibrariansDB.update(ID, LB);
            break;
        case mode.del:
            LibrariansDB.delete(ID);
            break;
    }
    break;
case 5:
    DBInteractionGenre GenresDB = new DBInteractionGenre();
    Genre GN = new Genre();
    switch (mode)
    {
        case mode.add:
            obj[0] = "-1";
            GN = GN.convertStrArrToObj(obj, true);
            GenresDB.add(GN);
            break;
        case mode.upd:
            GN = GN.convertStrArrToObj(obj, true);
            GenresDB.update(ID, GN);
            break;
        case mode.del:
            GenresDB.delete(ID);
            break;
    }
    break;
case 6:
    DBInteractionEdition EditionsDB = new DBInteractionEdition();
    Edition ED = new Edition();
    switch (mode)
    {
        case mode.add:
            obj[0] = "-1";
            ED = ED.convertStrArrToObj(obj, true);
            EditionsDB.add(ED);
            break;
        case mode.upd:
            ED = ED.convertStrArrToObj(obj, true);
            EditionsDB.update(ID, ED);
            break;
        case mode.del:
            EditionsDB.delete(ID);
            break;
    }
    break;
case 7:
    DBInteractionRating RatingsDB = new DBInteractionRating();

```

```

Rating RT = new Rating();
switch (mode)
{
    case mode.add:
        obj[0] = "-1";
        RT = RT.convertStrArrToObj(obj, true);
        RatingsDB.add(RT);
        break;
    case mode.upd:
        RT = RT.convertStrArrToObj(obj, true);
        RatingsDB.update(ID, RT);
        break;
    case mode.del:
        RatingsDB.delete(ID);
        break;
}
break;
case 8:
    DBIntegrationStorage StorageDB = new DBIntegrationStorage();
    Storage ST = new Storage();
    switch (mode)
    {
        case mode.add:
            obj[0] = "-1";
            ST = ST.convertStrArrToObj(obj, true);
            StorageDB.add(ST);
            break;
        case mode.upd:
            ST = ST.convertStrArrToObj(obj, true);
            StorageDB.update(ID, ST);
            break;
        case mode.del:
            StorageDB.delete(ID);
            break;
    }
    break;
}
}
catch (Exception)
{
    MessageBox.Show($"Возникла ошибка интерпретации данных\nПроверьте введённые данные", "Ошибка приложения", MessageBoxButtons.OK, MessageBoxIcon.Error);
    return; //я знаю, что это зло, но это необходимое зло, когда 2 часа третишь, чтобы сделать уведомления
}
    MessageBox.Show("Соответствующий запрос успешно обработан", "Уведомление", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
}
}

```

#### Лістинг Б4 – Файл «Form1.cs»

```

using LibraryDB.Objects;
using LibraryDB.DB;
using LibraryDB.View;
using System.Data;
using MySql.Data.MySqlClient;

namespace LibraryDB

```

```

{
    public partial class Form1 : Form
    {
        int dgvID;
        DataTable dt = new DataTable();
        ViewDBObjects view = new ViewDBObjects();
        DBExtraReqRequests req = new DBExtraReqRequests();

        public Form1()
        {
            InitializeComponent();
        }

        private void selectButton_Click(object sender, EventArgs e)
        {
            try
            {
                dt.Clear();
                dt = view.chooseTable(selectTableCB.Text);
            }
            catch (MySqlException ex)
            {
                MessageBox.Show($"Âîçîèêëà îøéáêà çããðóçêè: {ex}", "Îøéáêà ñãðããðà",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            mainDGV.DataSource = dt;
            objectDGV.Columns.Clear();
            loadEditing(view.chosenTableID, mainDGV.ColumnCount);
        }

        private void searchTextBox_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)13)
            {
                try
                {
                    string query = searchTextBox.Text;
                    query = System.Text.RegularExpressions.Regex.Replace(query, @"\s+", " ");
                    query = query.Trim();
                    dt.Clear();
                    dt = view.searchRows(query);
                }
                catch (MySqlException ex)
                {
                    MessageBox.Show($"Âîçîèêëà îøéáêà çããðóçêè: {ex}", "Îøéáêà ñãðããðà",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                mainDGV.DataSource = dt;
            }
        }

        private void mainDGV_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)
        {
            clearEditing();
            try
            {
                ulong selectedRowCount =
                (ulong)mainDGV.Rows.GetRowCount(DataGridViewElementStates.Selected);
                if (selectedRowCount > 0)
                {

```



```

        dgvID
Convert.ToInt32(mainDGV.Rows[mainDGV.SelectedRows[0].Index].Cells[0].Value);
        string[] dataDGV = new string[mainDGV.ColumnCount];
        for (int i = 0; i < mainDGV.ColumnCount; i++)
        {
            objectDGV.Columns.Add("", "");
        }
        dataDGV = prepareEditing(view.chosenTableID, dgvID, dataDGV.Length);
        objectDGV.Rows.Add(dataDGV);
        MessageBox.Show("Nòðíèà ăíáâêáíà  íà  ðääàèèèðíâíèà", "Óääăîíèáíèà",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
catch (MySqlException ex)
{
    MessageBox.Show($"Âîçíèêêà  îøéáêà  çääðóçêè: {ex}", "Îøéáêà  ñâðââðà",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

private void addButton_Click(object sender, EventArgs e)
{
    try
    {
        string[] obj = new string[mainDGV.ColumnCount];
        int dgvID = 0;
        for (int i = 1; i <= obj.Length; i++)
        {
            string elem = this.Controls.Find($"textBox{i}", true)[0].Text;
            elem = System.Text.RegularExpressions.Regex.Replace(elem, @"\s+", " ");
            obj[i - 1] = elem.Trim();
        }
        view.viewModeRowDB(obj, dgvID, VlewDBObject.mode.add);
        dt.Clear();
        dt = view.chooseTable(selectTableCB.Text);
        mainDGV.DataSource = dt;
    }
    catch (MySqlException ex)
    {
        MessageBox.Show($"Âîçíèêêà  îøéáêà  âíáâêêáíèÿ: {ex}", "Îøéáêà  ñâðââðà",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    clearEditing();
}

private void changeButton_Click(object sender, EventArgs e)
{
    try
    {
        string[] obj = new string[mainDGV.ColumnCount];
        for (int i = 1; i <= obj.Length; i++)
        {
            string elem = this.Controls.Find($"textBox{i}", true)[0].Text;
            elem = System.Text.RegularExpressions.Regex.Replace(elem, @"\s+", " ");
            obj[i - 1] = elem.Trim();
        }
        view.viewModeRowDB(obj, dgvID, VlewDBObject.mode.upd);
    }
}

```

```

        catch (MySqlException ex)
        {
            MessageBox.Show($"Äîçîèëëà îøéáëà ïðè èçîáîáîëè: {ex}", "Îøéáëà ñâðââðà",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        dt.Clear();
        clearEditing();
        dt = view.chooseTable(selectTableCB.Text);
        mainDGV.DataSource = dt;
    }

    private void deleteButton_Click(object sender, EventArgs e)
    {
        try
        {
            string[] obj = new string[mainDGV.ColumnCount];
            for (int i = 1; i <= obj.Length; i++)
            {
                string elem = this.Controls.Find($"textBox{i}", true)[0].Text;
                elem = System.Text.RegularExpressions.Regex.Replace(elem, @"\s+", " ");
                obj[i - 1] = elem.Trim();
            }
            view.viewModeRowDB(obj, dgvID, VlewDBObject.mode.del);
        }
        catch (MySqlException ex)
        {
            MessageBox.Show($"Äîçîèëëà îøéáëà ïðè óääëáîëè: {ex}", "Îøéáëà ñâðââðà",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        dt.Clear();
        clearEditing();
        dt = view.chooseTable(selectTableCB.Text);
        mainDGV.DataSource = dt;
    }

    private void helpButton_Click(object sender, EventArgs e)
    {
        Form2 newForm = new Form2();
        newForm.Show();
    }

    private void req1Button_Click(object sender, EventArgs e)
    {
        try
        {
            reqDGV.DataSource = null;
            reqDGV.DataSource = req.getDebtors();
        }
        catch (MySqlException ex)
        {
            MessageBox.Show($"Äîçîèëëà îøéáëà ïðè çääðóçêâ: {ex}", "Îøéáëà ñâðââðà",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void req2Button_Click(object sender, EventArgs e)
    {
        try
        {
            reqDGV.DataSource = null;

```

```

        reqDGV.DataSource = req.getUnderageReaders();
    }
    catch (MySqlException ex)
    {
        MessageBox.Show($"İşlemler sırasında hata oluştu: {ex}", "İşlemler sırasında hata oluştu",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void req3Button_Click(object sender, EventArgs e)
{
    try
    {
        reqDGV.DataSource = null;
        reqDGV.DataSource = req.getEndedBooks();
    }
    catch (MySqlException ex)
    {
        MessageBox.Show($"İşlemler sırasında hata oluştu: {ex}", "İşlemler sırasında hata oluştu",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void loadEditing(int tableID, int rows)
{
    clearEditing();

    for (int i = 1; i <= rows; i++)
    {
        this.Controls.Find($"label{i}", true)[0].Visible = true;
        this.Controls.Find($"textBox{i}", true)[0].Visible = true;
    }

    for (int i = rows + 1; i <= 13; i++)
    {
        this.Controls.Find($"label{i}", true)[0].Visible = false;
        this.Controls.Find($"textBox{i}", true)[0].Visible = false;
    }

    switch (tableID)
    {
        case 1:
            for (int i = 1; i <= rows; i++)
            {
                this.Controls.Find($"label{i}", true)[0].Text = view.bookLendingLabels[i - 1];
            }
            break;
        case 2:
            for (int i = 1; i <= rows; i++)
            {
                this.Controls.Find($"label{i}", true)[0].Text = view.bookLabels[i - 1];
            }
            break;
        case 3:
            for (int i = 1; i <= rows; i++)
            {
                this.Controls.Find($"label{i}", true)[0].Text = view.readerLabels[i - 1];
            }
            break;
        case 4:
    
```

```

        for (int i = 1; i <= rows; i++)
        {
            this.Controls.Find($"label{i}", true)[0].Text = view.librarianLabels[i - 1];
        }
        break;
    case 5:
        for (int i = 1; i <= rows; i++)
        {
            this.Controls.Find($"label{i}", true)[0].Text = view.genreLabels[i - 1];
        }
        break;
    case 6:
        for (int i = 1; i <= rows; i++)
        {
            this.Controls.Find($"label{i}", true)[0].Text = view.editionLabels[i - 1];
        }
        break;
    case 7:
        for (int i = 1; i <= rows; i++)
        {
            this.Controls.Find($"label{i}", true)[0].Text = view.ratingLabels[i - 1];
        }
        break;
    case 8:
        for (int i = 1; i <= rows; i++)
        {
            this.Controls.Find($"label{i}", true)[0].Text = view.storageLabels[i - 1];
        }
        break;
    default:
        MessageBox.Show($"Άίçíèëëà îøéáëà çããðóçëë ôîðîû:\nÍå óääëîñü èçìáíèòü ààííüå íà ôîðíå  

        \t"Ðääåèèèðîîåíèè", "Íåçãåñîíîíü îøéáëà", MessageBoxButtons.OK, MessageBoxIcon.Error);
        break;
    }
}

private string[] prepareEditing(int tableID, int objID, int length)
{
    try
    {
        string[] obj = view.getStringRow(objID);
        for (int i = 1; i <= length; i++)
        {
            this.Controls.Find($"textBox{i}", true)[0].Text = obj[i - 1];
        }
        return obj;
    }
    catch (Exception)
    {
        MessageBox.Show($"Άίçíèëëà îøéáëà çããðóçëë ñððîëè:\nÍå óääëîñü çããðóçèòü íåäíèíè äëü  

        ïîéå äåíåå å öåååèèèðîîåíèè", "Íåçãåñîíîíü îøéáëà", MessageBoxButtons.OK, MessageBoxIcon.Error);
        string[] empty = new string[0];
        return empty;
    }
}

private void clearEditing()
{
    objectDGV.Columns.Clear();
}

```

```
        for (int i = 1; i <= 13; i++)
        {
            this.Controls.Find($"textBox{i}", true)[0].Text = null;
        }
    }
}
```

**Міністерство освіти і науки України**  
**Державний заклад «Луганський національний університет**  
**імені Тараса Шевченка»**  
Факультет (інститут) Навчально-науковий інститут фізики,  
математики та інформаційних технологій  
(повна назва)  
Кафедра Кафедра інформаційних технологій та систем  
(повна назва)

## **КЕРІВНИЦТВО КОРИСТУВАЧА**

на виконання програмної розробки (ПР):

**«Проектування та розробка пошукового додатку «Бібліотека» з  
системою рекомендацій книг»**

**ПОГОДЖЕНО**  
Керівник кваліфікаційної роботи

Семенов М.А

“ ” 2025р

**ВИКОНАВЕЦЬ**  
Студент групи 4ІПЗ

Залеський О.В

“ ” 2025р

## ПОЛТАВА 2025

### **ЗМІСТ**

- 1.1. Підготовка до роботи з додатком 3
- 1.2. Робота з додатком 4

### 1.1. Підготовка до роботи з додатком

Для запуску додатку «Онлайн бібліотека» необхідно:

- Операційна система: Windows 10 або Windows 11;
- Встановлена програмна платформа: .NET 6.0;
- Локальна система керування базами даних MySQL;
- Завантажена база даних з файлу Library.sql.

Щоб підготувати робоче середовище:

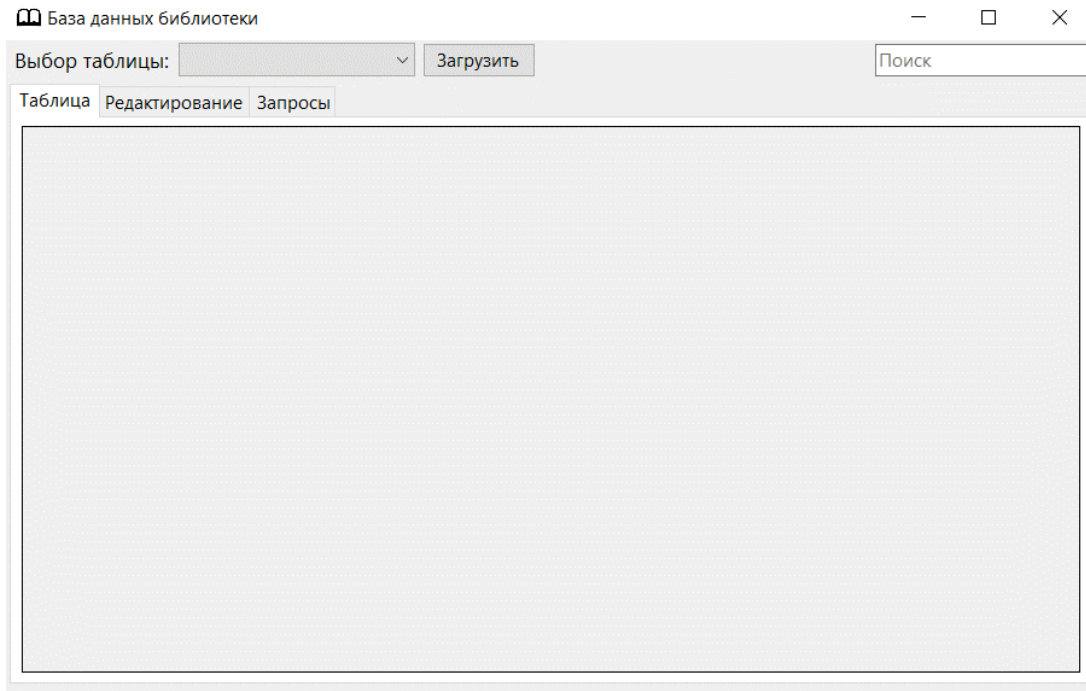
1. Встановіть MySQL Server та створіть базу даних, імпортуючи файл Library.sql;
2. Встановіть .NET 6 SDK з офіційного сайту Microsoft;
3. Завантажте теку з файлами програми;
4. Запустіть виконуваний файл LibraryDB.exe;
5. Переконайтесь, що є з'єднання з базою даних. У разі необхідності перевірте файл конфігурації (appsettings.json або подібний).

### 1.2. Робота з додатком

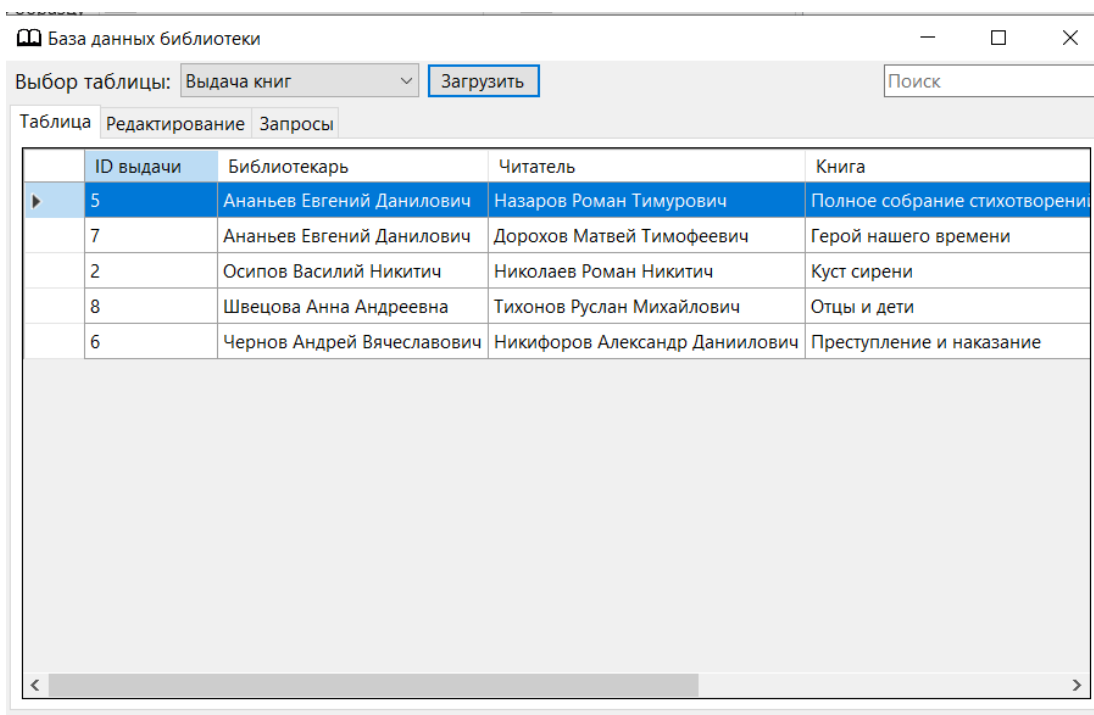
Після запуску додатку користувач потрапляє на екран авторизації. Доступні такі основні модулі:

- Головна сторінка: перегляд списку книг, фільтрація за автором, жанром або назвою;





- Особистий кабінет: інформація про користувача, історія взаємодій з книгами;



- Додавання нової книги: бібліотекар може додавати нові книги, змінювати наявні, видаляти записи;

- Відгуки та оцінки: користувачі можуть залишати коментарі та

оцінювати книги;

- Звіти: система дозволяє формувати звіти, наприклад, про боржників або найпопулярніші книги.

База данных библиотеки

Выбор таблицы: Выдача книг Загрузить Поиск

Таблица Редактирование Запросы

ID выдачи (не используется в добавлении)

Библиотекарь  
Швецова Анна

Читатель  
Тихонов Руслан

Книга (ISBN/Название)  
Преступление и наказание

Кол-во  
1

Дата выдачи (автоопределение при добавлении)

Дата возврата  
12.12.2023

Запись

Добавить

Изменить

Удалить

Справка

Інтерфейс програми адаптивний, розрахований на використання на різних пристроях (настільних ПК, ноутбуках). Усі основні дії супроводжуються підказками та повідомленнями про успішне завершення операцій.

У разі помилки користувач отримає відповідне повідомлення.

Для виходу з програми достатньо натиснути кнопку «Вихід» у верхньому правому куті головного вікна.