МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ:

«Інтелектуальний модуль рекомендаційної системи вибору художніх творів»

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма «**Комп'ютерні науки»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи КН- 42

Кузнєцов Владислав Володимирович
(прізвище та ініціали)

Керівник Іларіонов О.Є.
(прізвище та ініціали)
к.н.т, доцент
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 13 від 05.06.2023 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ - 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ

Завідувач кафедри
інтелектуальних технологій

Іларіонов О.Є.

"___"_____2023 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Кузнєцову Владиславу Володимировичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)

Інтелектуальний модуль рекомендаційної системи вибору художніх творів
затверджена протоколом засідання кафедри від « 11 » листопада 2022 р. протокол № 4

2. Термін здачі студентом закінченого проекту (роботи) травня 2023 року

3. Вихідні дані до проекту (роботи) прочитані книги користувачів системи

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

4.1 Аналіз існуючих бібліотек та алгоритмів рекомендації.
4.2. Розробка архітектури онлайн бібліотеки.
4.3. Практична реалізація рекомендаційної системи вибору художніх творів

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)

5.1. Вивчення і формалізація предметної області: 3 слайди
5.2. Вибір варіантів реалізації, проектування складових інформаційних систем і технологій та засобів їх реалізації: 4 слайди
5.3. Розробка програмних засобів: 4 слайди
5.4. Висновки по роботі: 1 - 2 слайди

6. Консультанти з випускної кваліфікаційної роботи із зазначенням її розділів, що їх стосуються і

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 15 лютого 2023 року

Керівник _____ /Іларіонов О.Є. /
(підпис) (ініціали та прізвище)

Завдання прийняв до виконання _____ /Кузнєцов В.В. /
(підпис) (ініціали та прізвище)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1.	Обговорення з керівником постановки завдання, підбір літератури, огляд існуючих рішень	15.02.2023 – 01.03.2023	
2.	Аналіз постановки задачі, формалізація задачі, аналіз літературних джерел, написання розділу 1	02.03.2023 – 20.03.2023	
3.	Проектно-технологічна реалізація інтелектуального модуля рекомендаційної системи вибору художніх творів	21.03.2023 – 11.04.2023	
4.	Розробка та тестування інтелектуального модуля рекомендаційної системи вибору художніх творів	12.04.2023 – 15.05.2023	
5.	Робота над оформленням пояснювальної записки та презентаційних матеріалів	16.05.2023 – 29.05.2023	

Студент _____ /Кузнєцов В.В. /
(підпис) (ініціали та прізвище)

Керівник випускної кваліфікаційної роботи _____ / Іларіонов О.Є. /
(підпис) (ініціали та прізвище)

Анотація

Кузнєцов Владислав Володимирович виконав випускну кваліфікаційну роботу на тему «Інтелектуальний модуль рекомендаційної системи вибору художніх творів» за спеціальністю 122 – «Комп'ютерні науки».

У випускній кваліфікаційній роботі проведено аналіз актуальності та розробку інтелектуального модулю рекомендаційної системи вибору художніх творів. У рамках дослідження було створено модуль рекомендацій, який дозволяє користувачам отримувати персоналізовані пропозиції щодо нових книжок для читання, з урахуванням їхніх інтересів.

Ключові слова: інтелектуальний модуль, рекомендаційна система, вибір художніх творів, модуль рекомендацій, персоналізовані пропозиції, нові книжки, читання, інтереси

Summary

The degree project: «Intelligent module of the recommendation system for the selection of works of art» has completed by **Vladyslav Kuznietsov** specialty 122 – «Computer Scienses».

In the graduation thesis, the relevance analysis and development of the intellectual module of the recommendation system for the selection of works of art was carried out. The research created a recommendation module that allows users to receive personalized suggestions for new books to read based on their interests.

Keywords: intelligent module, recommendation system, selection of works of art, recommendation module, personalized suggestions, new books, reading, interests

Зміст

Вступ 2
РОЗДІЛ 1. АНАЛІЗ РОЗПОДІЛЕНИХ ІС ДЛЯ ЗБЕРІГАННЯ РІЗНОРОДНИХ ЕЛЕКТРОННИХ ДОКУМЕНТІВ 4
1.1 Аналіз існуючих онлайн бібліотек 4
1.2 Аналіз проблеми інформаційного перевантаження користувачів послуг Інтернет-бібліотеки 7
1.3 Актуальність методів вилучення знань для покращення систем рекомендацій 8
1.4 Аналіз онлайн бібліотек 10
1.5 Опис цільової аудиторії, якої потребує система рекомендацій. 16
1.6 Постановка задачі на розробку рекомендаційної системи онлайн бібліотеки 18
1.7 Висновки до першого розділу 22
РОЗДІЛ 2. ПРОЕКТУВАННЯ ІС “ОНЛАЙН БІБЛІОТЕКА” 24
2.1 Розробка архітектури 24
2.2 Аналіз для практичної реалізації 35
2.3 Висновки до другого розділу 43
РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ 45
3.1 Обґрунтування вибору програмних засобів 45
3.2 Структура RESTful API 48
3.3 Специфікація OpenAPI 49
3.4 Тестування API 50
3.5 Інструкція користувача 55
3.3 Висновки до третього розділу 60
Висновки 62
Використана література 63
Додатки 66

Вступ

На сьогоднішній день, зі станом массового карантину, воєнного положення та сучасних технологій - необхідно мати можливість відвідувати зручну бібліотеку біля твого будинку на бібліотеку в твоєму смартфоні або комп’ютері. Це сприятиме для кожної людини зручність використання та збереження екології планети.

Мета дипломної роботи: Розробка та реалізація інтелектуального модуля рекомендаційної системи для вибору художніх творів з метою поліпшення користувацького досвіду та забезпечення персоналізованої рекомендації з урахуванням індивідуальних смаків і вподобань користувачів. Формування рекомендацій літератури у бібліотеці є об’єктом дослідження.

Предметом дослідження є методика надання рекомендацій щодо відбору документів в електронних бібліотеках.

Бібліотека – це місце або будівля, де зберігаються такі речі, як книги, газети, відео та музика, для читання, використання чи позичання.

Рекомендаційна система – це програмний комплекс, який визначає інтереси та вподобання користувача та формує контентні пропозиції на основі цих інтересів та уподобань. Такі системи змінюють спосіб взаємодії програмних систем зі своїми користувачами. Система не надає статистичну інформацію, а змінюється, підлаштовується під користувача, підвищує ступінь інтерактивності для розширення сервісу для користувача. [9]

В даній системі відсутні люди які видають книжки, тільки технічне обслуговування та збільшення каталогу книг. Також система має можливість заохочувати читати книжки саме по тим напрямкам які цікавлять користувача.

Основні моделі бібліотеки які існують на даний момент:

1. Текстовий варіант книги;
2. Відео матеріал;
3. Аудіо книжки;
4. Тести на перевірку заохочень користувачів;

Основним мотивом додатку - це мати найновіші та найпопулярніші книги сучасності в одному місці та легко знаходити нові за його допомогою.

РОЗДІЛ 1. АНАЛІЗ РОЗПОДІЛЕНИХ ІС ДЛЯ ЗБЕРІГАННЯ РІЗНОРОДНИХ ЕЛЕКТРОННИХ ДОКУМЕНТІВ

Більшість систем рекомендацій книг мають функцію відображення персональних рекомендацій користувачам. Система рекомендацій передбачає вподобання певного користувача щодо книги на основі даних, явно вказаних користувачем або зібраних з історії його взаємодії з певним сайтом чи різними іншими ресурсами.

Системи рекомендацій використовуються в багатьох комерційних проектах для визначення вподобань клієнтів на основі їхньої поведінки та даних профілю: Amazon (використовує кілька типів систем рекомендацій, одна з яких застосовується у вигляді додаткових пропозицій щодо подібних або споріднених продуктів), Last.fm (Використання механізму рекомендацій на основі тегів), Netflix (функція перемішування надає користувачам відповідні фільми та серіали). За останні роки якість таких алгоритмів значно покращилася. Тому актуальною на сьогодні є проблема розробки та дослідження методів побудови рекомендаційних систем.

1.1 Аналіз існуючих онлайн бібліотек

На даний момент, альтернативні сайти онлайн бібліотек, такі як: “Національна бібліотека України імені В.І.Вернадського”(далі “НБУВ”), “УкрЛіб”, “Libruk”, “e-bookua” та інші. Всі платформи бібліотек мають одну й ту саму мету, а саме онлайн бібліотека з книгами, які розподілені за жанрами, та зручний застосунок перегляду сторінок.

Саме перегляд книжок є головною задачею кожного онлайн застосунку, це вибір варіантів тексту - шрифт, розмір, колір тексту та колір сторінки які може змінювати сам користувач для перегляду сторінок під свій стан очей та навколишнього світла.

Всі онлайн бібліотеки мають свою бібліотеку жанрів та пошукову система за необхідними параметрами яка допоможе знайти необхідне. Жанри - історія,

оповідання, біографія, повість та велика кількість інших, які необхідні та цікаві для користувача.

Аудіокниги, які колись були дуже нішевими та з обмеженою пропозицією назв, тепер є основною діяльністю для багатьох із мільйонами доступних назв. Замінивши громіздкі касети та DVD-диски, вони пропонують доступний спосіб читання, якщо вам важко тримати книгу, якщо у вас проблеми із зором, або якщо у вас брак часу чи неспокійність, щоб сидіти й читати книгу.

З аналізу артельнативних бібліотек виявив необхідні критерії для користування сайту бібліотеки, а саме:

1. Зручний пошуковий застосунок для пошуку ;
2. Можливість змінювати стиль сторінки;
3. Алгоритм підбору книг за захопленнями користувача.

Метою мого дослідження альтернативних сайтів є перевірка актуальності моєї розробки сайту, а саме варіанти передачі інформації користувачу.

На сайтах застосовують всі необхідні на даний момент технології. Такі як:

1. Текстовий варіант книг;
2. Аудіо книги;
3. Рекомендаційна система;
4. Зручний перегляд текстового варіанту;
5. Вибір мови запису електронної книги;

З даного аналізу існуючих систем онлайн бібліотек можно виділити основні фактори:

1. Різноманітний вибір книг за жанрами
2. Актуальність книг
3. Алгоритм пропонування книг
4. Зручний редактор сторінки
5. Сучасні технології для засвоєння інформації

Далі порівняємо існуючі бібліотеки у таблиці 1.1.

Таблиця 1.1 Порівняння існуючих бібліотек.

	“НБУВ”	“ УкрЛіб”	“ Libruk”	e-bookua
Різноманітний вибір за жанрами	+	+	+	+
Сучасні технології для перегляду книги	-	-	+	-
Редагування сторінки	-	-	+	-
Актуальність книг	+	+	+	+
Алгоритм рекомендацій	-	+	+	-
Зручний інтерфейс	-	+	+	-
Адаптивність	+	+	+	+
Кроссбраузерний сайт	+	+	+	+

Висновок про актуальність розробки: Особливо важливою є актуальність розробки сайту бібліотеки в епоху цифрової трансформації, коли люди все більше переходять до онлайн-формату. Сайт бібліотеки зможе надати користувачам можливість швидко та зручно знайти необхідну літературу, дізнатися про новинки та інші актуальні події, зв’язані зі світом книг. Крім того, рекомендаційна система, що може бути впроваджена на сайті бібліотеки, є актуальним інструментом для покращення користувацького досвіду. Це дозволить пропонувати користувачам книги, які їм можуть сподобатися, на основі їхнього історичного читання та вибору книг, що рекомендуються.

Рекомендаційна система може стати важливим кроком у підвищенні задоволення користувачів від використання сайту бібліотеки, що в свою чергу може призвести до збільшення популярності та залучення нових користувачів.

1.2 Аналіз проблеми інформаційного перевантаження користувачів послуг Інтернет-бібліотеки

Сучасна проблема полягає в розробці ефективних методів вилучення знань для створення рекомендаційних систем. Це завдання важливе для багатьох областей, таких як інтернет-магазини, пошукові системи, музичні та відеосервіси та бібліотеки. Системи рекомендацій допомагають персоналізувати контент, пристосовуючи його до інтересів та вподобань конкретного користувача. Хоча такі системи існують вже більше 20 років, наразі існує дуже мало успішних прикладів, і ця проблема все ще не вирішена через недосконаліть наявних методів.

Один із шляхів підвищення точності рекомендацій - розширення переліку даних, що використовуються для формування рекомендацій, наприклад, за рахунок використання контексту. Рекомендаційні системи відрізняються від традиційних програмних систем тим, що вони визначають інтереси та вподобання користувачів та створюють персоналізовані рекомендації щодо контенту, підлаштовуючись під кожного конкретного користувача.

Типово рекомендаційні системи мають справу з двома основними сутностями: користувачами та об'єктами. Користувачі є отримувачами рекомендацій і джерелом даних про їх вподобання, тоді як об'єкти можуть бути продуктами, фільмами, музичними треками, книгами, новинами, веб-сайтами і т.д. Основне завдання рекомендаційної системи - визначити цілі, які користувач не знає про або не використовував певний час, але які можуть бути корисними або цікавими йому в поточний момент.

Отже, після визначення цілей користувача, рекомендаційна система повинна знайти відповідні предмети, що задовольнять його потреби. Це може бути зроблено за допомогою аналізу даних про користувача, таких як його

історія покупок, переглядів веб-сторінок, або демографічних даних, таких як вік, стать, місце проживання та інші.

Після отримання цих даних, система може використовувати алгоритми рекомендацій, такі як колаборативний фільтр, факторизація матриць, або глибокі нейронні мережі, щоб знайти відповідні товари або послуги, які можуть бути цікавими користувачеві.

Наприклад, якщо користувач купує книги в жанрі наукової фантастики, система може рекомендувати йому інші книги в тому ж жанрі. Якщо користувач зазвичай замовляє піцу, система може рекомендувати йому інші види їжі, які можуть сподобатися йому.

Крім того, система може враховувати також контекст, наприклад, місце знаходження користувача або час доби, щоб зробити більш точні рекомендації. Наприклад, якщо користувач знаходиться в кінотеатрі, система може рекомендувати фільми, які доступні для перегляду на поточний час і в цьому кінотеатрі.

Нарешті, рекомендаційна система може враховувати також інші фактори, такі як ціна, наявність товарів на складі або рейтинги товарів, щоб зробити більш придатні рекомендації. Загалом, метою рекомендаційної системи є забезпечення користувачів персоналізова ними рекомендаціями, які допоможуть їм знайти товари або послуги, які вони шукають, або навіть ті, про які вони навіть не здогадуються. Це забезпечує більш зручне та ефективне шопінг-досвід, що зазвичай призводить до задоволення користувачів та збільшення продажів. Однак, важливо зазначити, що рекомендаційна система повинна бути побудована з урахуванням приватності користувачів, захищати їхні дані та уникати використання їх в несанкціонованих цілях.

1.3 Актуальність методів вилучення знань для покращення систем рекомендацій

Проблема побудови методів вилучення знань для рекомендацій в даний час актуальна для багатьох областей ii є частиною таких завдань, як пропозиція товарів в інтернет-магазинах, ранжування результатів виведення в пошукових

системах, пошук релевантного контенту в музичних, відеосервісах і ЗМІ, а також книг у бібліотеках.

Загалом, системи рекомендацій в Інтернеті використовуються для персоналізації контенту, тобто для автоматичного пристосування його до потреб конкретних користувачів. Хоча персоналізовані рекомендації у веб-системах існують вже понад 20 років, успішних прикладів лише кілька десятків. Дослідження в цій галузі все ще тривають, головним чином через існування невирішених проблем з існуючими методами. Один із способів підвищити точність рекомендаційних систем - розширити перелік інформації, яка використовується для формування рекомендацій, наприклад, за рахунок використання контексту.

Рекомендаційні системи - це програмні пакети, які визначають інтереси та вподобання користувачів і створюють відповідні рекомендації щодо контенту. Ці системи змінили спосіб взаємодії користувачів з програмними системами. Замість того, щоб надавати статистичну інформацію, система змінюється, підлаштовується під користувача, підвищує інтерактивність і розширює послуги для користувачів.

Традиційні системи рекомендацій мають справу з двома типами сутностей: користувачами та об'єктами. Тут користувачі є одержувачами рекомендацій і джерелом даних про вподобання. Об'єкти - це, залежно від домену, продукти, фільми, музичні треки, книги, новини, веб-сайти, тобто те, що надається користувачеві як рекомендації. Загалом, завдання рекомендаційної системи можна сформулювати як "визначення цілей, які незнайомі (або не використовувалися протягом певного часу) користувачеві, але є корисними або цікавими в поточному контексті". Існує кілька підходів до розробки рекомендаційних систем, які можуть бути застосовані, наприклад:

- Наявні дані про користувачів та рекомендовані сутності ;
- Типи явноготанеявногозворотного зв'язку відкористувачів.;
- області домену .

Створюється ефективна система рекомендацій з використанням новітніх методів зберігання та аналізу даних для залучення клієнтів.

1.4 Аналіз онлайн бібліотек

Створення інформаційної системи для служби онлайн бібліотеки з метою полегшення життя читачів під час карантину.

Користувач переходить за посиланням на сторінку бібліотеки, за своїм бажанням реєструється - якщо новий користувач, або переглядає книжки в вільному форматі, або входить до особистого кабінету і переглядаю особисту інформацію.

Система в свою чергу додає до бази даних інформацію про користувача і реєструє на сайті(рисунок 1.1).

Основні цілі системи:

- Зручний перегляд книг;
- Вибір стилю перегляду сторінки;
- Рекомендації;
- Особистий кабінет

Складові елементи:

- Підсистема авторизації
- База даних книг
- База даних читачів
- Модуль рекомендацій

Опис профілів зацікавлених сторін:

1. Внутрішні зацікавлені сторони:

- Власники офлайн бібліотек

2. Зовнішні зацікавлені сторони:

- Користувачі
- Школа
- Університет

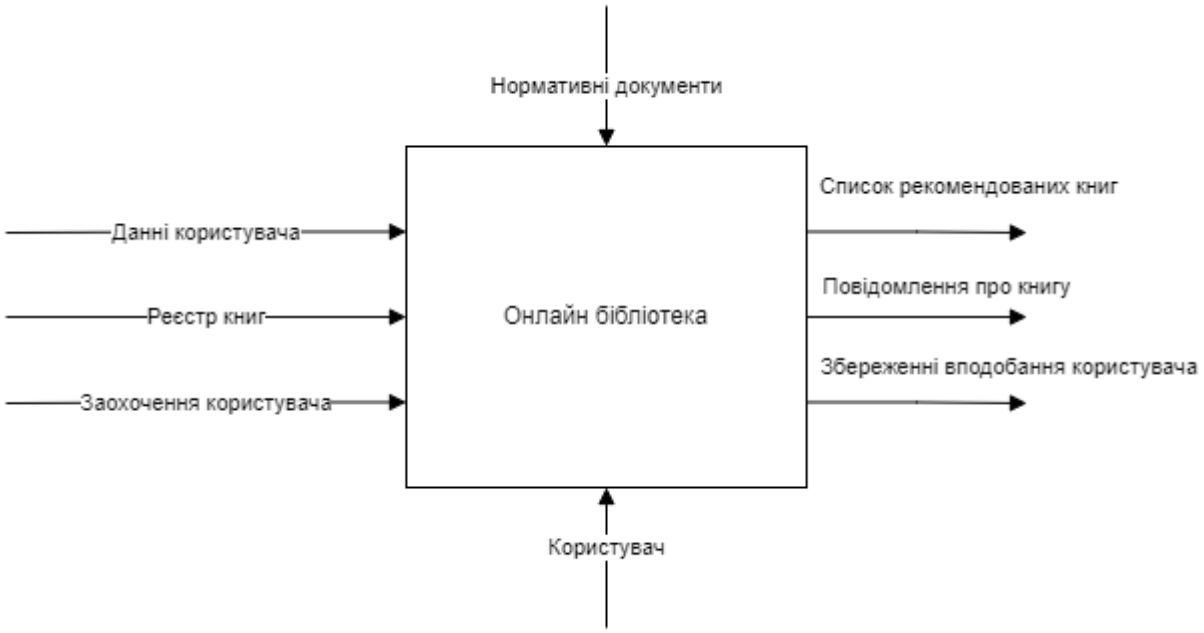

Рисунок 1.1 IDEF0 діаграма інформаційної системи для служби онлайн бібліотеки

“Онлайн бібліотека” - отримує всю необхідну інформацію про користувача, а саме його дані та заохочення(перегляд минулих книг та побажання читача по жанрам). Завдяки цим даним, рекомендаційному механізму та підтримкою редактора самого сайту, який збільшує бібліотеку, читач отримує список книг які йому можуть бути цікавими, а сайт отримує вибір користувача для подальшої рекомендації для читача.(рисунок 1.2)

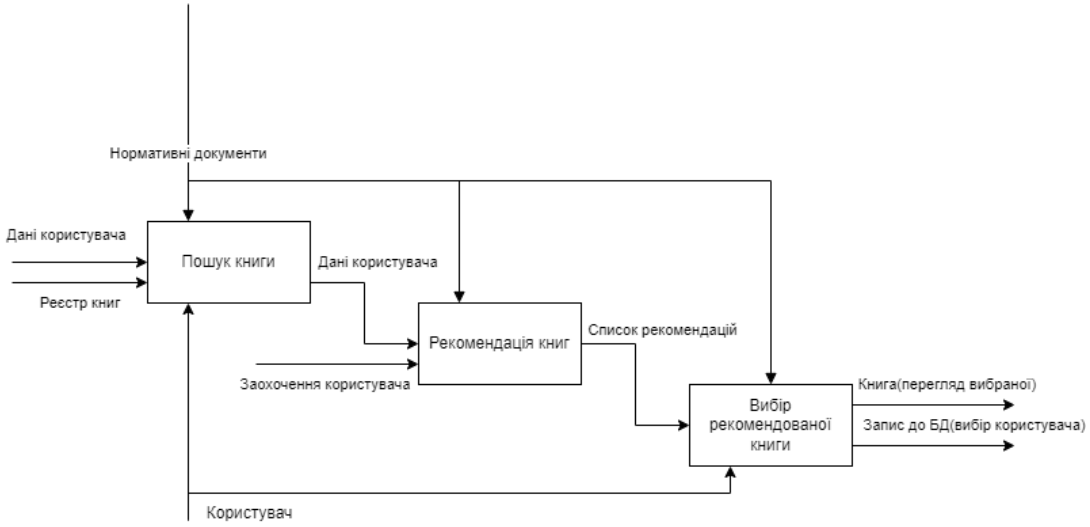

Рисунок 1.2 Деталізована діаграма IDEF0 інформаційної системи для служби онлайн бібліотеки

Деталізована діаграма перегляду рекомендації, читач хоче знайти необхідну книгу, система отримує дані, проводить пошук з БД для читача і видає необхідну книгу та рекомендаційний список книг - сам список підтримує редактор енциклопедичних та довідкових знань та поповнює базу даних книг, а сайт обробляє інформацію, завдяки цій роботі читач отримує книгу, а ми його заохочення.

1.4.1 Аналіз способів віртуалізації та контейнеризації для спрощення розгортання програмного забезпечення онлайн бібліотеки

Програмний застосунок може бути складним для перенесення з одного середовища до іншого через різні фактори, такі як апаратна та програмна інфраструктура, залежності, налаштування та інші чинники. Це може бути особливо проблематичним у випадку, коли застосунок має багато компонентів або відповідає за складні бізнес-процеси.

Для того, щоб спростити процес перенесення програмного забезпечення, можна використовувати віртуалізацію

У цьому розділі розглянемо питання розгортання застосунку та його залежностей. У розробці програм можуть виникнути різноманітні проблеми, пов'язані з відстеженням залежностей, масштабуванням та оновленням окремих компонентів. Для розв'язання цих проблем ми розглянемо віртуальні машини (VM) та контейнери. Віртуальні машини були створені для емуляції певної апаратної системи та запуску програмного забезпечення на фізичних серверах. Гіпервізор або монітор віртуальної машини є програмним забезпеченням, яке створює та запускає віртуальні машини. У кожній віртуальній машині працює унікальна гостьова операційна система та власні двійкові файли, бібліотеки та програми. Контейнери, з іншого боку, забезпечують ізольоване середовище для виконання додатків та мають менше накладних витрат, ніж віртуальні машини. В цьому розділі ми розглянемо, як вибрати оптимальний підхід до розгортання застосунку та які переваги та недоліки мають віртуальні машини та контейнери.(рисунок 1.3)[18]

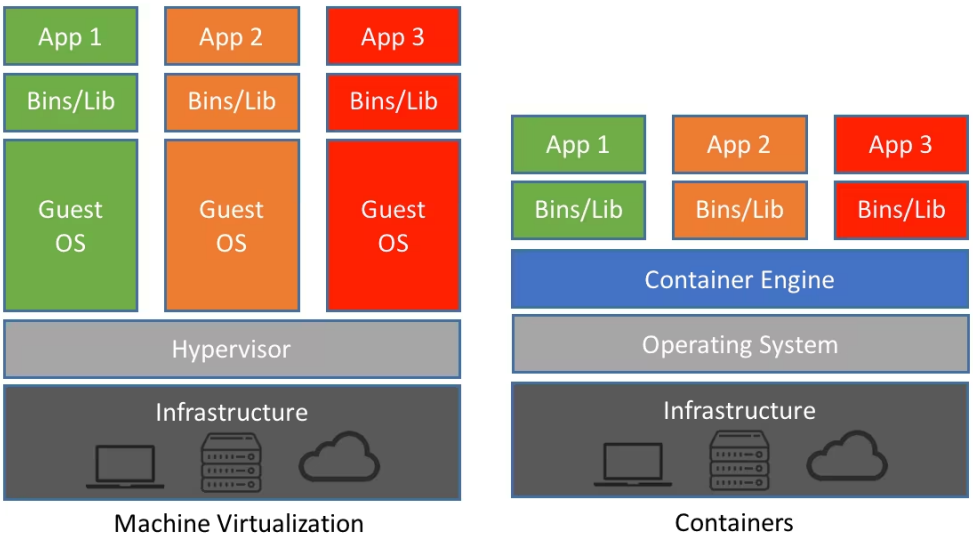

Рисунок 1.3 Представлення віртуалізації та контейнерів

Віртуалізація серверів забезпечила низку переваг, однією з найбільших була можливість консолідувати програми в одній системі. Часи, коли одна програма працювала на одному сервері, минули. Віртуалізація започаткувала економію коштів завдяки зменшенню займаної площі, швидшому забезпеченню сервера та покращеному аварійному відновленню (DR), оскільки апаратне забезпечення сайту DR більше не повинно було віддзеркалювати основний центр обробки даних.

Ця фізична консолідація також виграла від цієї фізичної консолідації, оскільки більше використання більших і швидших серверів звільняло згодом невикористані сервери для перепрофілювання для забезпечення якості, розробки або лабораторного обладнання. Але цей підхід мав свої недоліки. Кожна віртуальна машина містить окремий образ операційної системи, що збільшує обсяг пам’яті та сховища. Як виявилося, ця проблема ускладнює всі етапи життєвого циклу розробки програмного забезпечення — від розробки та тестування до виробництва та аварійного відновлення. Цей підхід також серйозно обмежує переносимість програм між публічними хмарами, приватними хмарами та традиційними центрами обробки даних. Віртуалізація операційної системи (ОС) набула популярності за останнє десятиліття, щоб забезпечити

передбачувану та ефективну роботу програмного забезпечення під час переміщення з одного серверного середовища в інше. Але контейнери забезпечують спосіб запуску цих ізольованих систем на одному сервері або хост-ОС.

Контейнери - це ізольовані віртуальні середовища, які дозволяють запускати додатки в умовах, які можна відтворити на будь-якому сервері, а не тільки на конкретній машині з відповідною конфігурацією.

Одна з ключових відмінностей контейнерів від віртуальних машин (VM) полягає у тому, що контейнери не включають повні операційні системи. Кожен контейнер використовує спільну операційну систему, яка встановлена на хост-машині. Це означає, що контейнери мають дуже малий розмір порівняно з VM і вимагають набагато менше ресурсів для запуску.

Крім того, контейнери дозволяють спільне використання компонентів між контейнерами, тому що кожен контейнер містить лише необхідний набір компонентів та залежностей. Це дозволяє зменшити зайвий об'єм даних та витрати на їх збереження.

Однак, на відміну від VM, контейнери не забезпечують повної ізоляції між додатками. Кожен контейнер може використовувати спільний ресурс, такий як ядро хост-машина, інтерфейс мережі тощо. Тому контейнери не підходять для всіх сценаріїв розробки, особливо для додатків, що містять важливі конфіденційні дані.

Контейнеризація Docker та сервіс-орієнтований дизайн намагаються вирішити багато з цих проблем. Додатки можна розбити на керовані функціональні компоненти, упакувати окремо, включаючи всі залежності, і легко розгортати на нестандартних архітектурах. Це також полегшує розширення та оновлення компонентів.[17]

Докер зазвичай складається з трьох основних компонентів

- Зображення Docker зазвичай є шаблоном, доступним лише для читання. Наприклад, операційна система Mint OS з додатками React всередині може бути включена в образ. Образи зазвичай використовуються для створення нових

контейнерів з сервісами всередині, і Docker був створений для того, щоб полегшити створення нових образів, редагування існуючих образів та їх видалення. Таким чином, розробники можуть завантажувати зображення, створені іншими, і базувати їх на них. Зображення є основним будівельним блоком Docker.

- Докер-реєстр зберігає існуючі зображення. Існують публічні та приватні реєстри. Ці реєстри дозволяють шукати і завантажувати корисні зображення, щоб уникнути дублювання. Публічний реєстр докерів називається Docker Hub. У ньому зберігається величезна кількість зображень. Ви можете створювати власні зображення або використовувати вже існуючі, створені іншими. Реєстр називається компонентом доставки зображень.

- Контейнери, що нагадують каталоги операційної системи. Контейнер містить все необхідне для запуску програми. Щоб отримати контейнер, його потрібно створити з зображення. Контейнери можна створювати, виконувати, зупиняти та видаляти. Кожен контейнер є унікальним і безпечним для всієї програми. Контейнери називаються робочими компонентами.(рисунок 1.4)

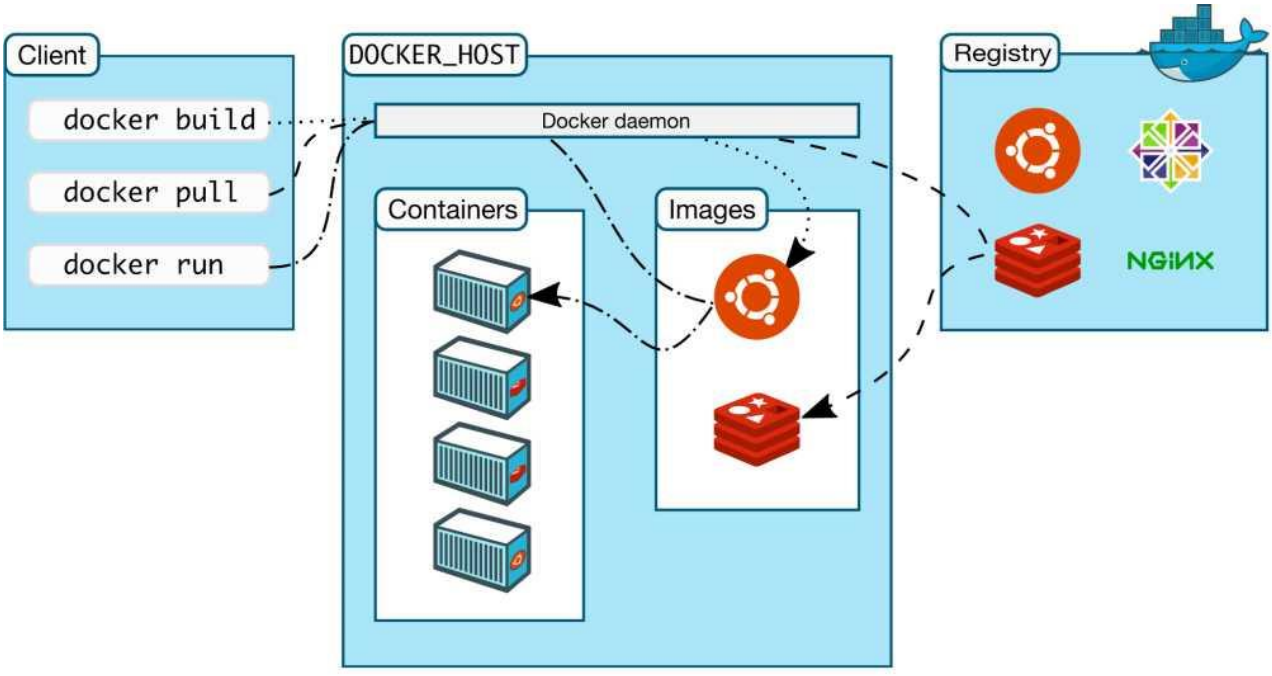

Рисунок 1.4 Архітектура Docker

Далі буде описано переваги контейнеризації і те, як Docker може допомогти вирішити багато з цих проблем. Docker є ключовим компонентом для розподіленого розгортання контейнерів, що забезпечує легке масштабування та управління.[19]

При створенні додатків для розгортання в контейнерах, важливо спочатку визначити архітектуру додатку. Зокрема, контейнерні додатки найкраще працюють з сервіс-орієнтованою архітектурою. Це означає, що функціональність системи розбивається на окремі компоненти, які взаємодіють між собою через чітко визначені інтерфейси. Контейнерна технологія стимулює такий дизайн, оскільки кожен компонент може бути розширений і оновлений незалежно.

Для того, щоб додаток був пристосований до контейнерів, він повинен відповідати наступним характеристикам. Він не повинен залежати від функціональності хост-системи, а кожен сервіс повинен враховувати змінні навколишнього середовища під час початкової конфігурації. Кожен компонент також повинен надавати сумісний інтерфейс, який користувачі можуть використовувати для доступу до сервісів. Крім того, дані програми повинні зберігатися у змонтованому томі поза контейнером або в окремому контейнері даних.

Ця стратегія дозволяє замінювати або модифікувати кожен компонент незалежно (якщо підтримується API) та зосереджується на горизонтальній масштабованості, оскільки кожен компонент може бути розширений. Загалом, сервіс-орієнтовані додатки мають перевагу для розгортання в контейнерах, і контейнерна технологія допомагає реалізувати цей підхід до проектування додатків.

1.5 Опис цільової аудиторії, якої потребує система рекомендацій.

Рекомендаційна система, яку ми описуємо, призначена для веб-сайту бібліотеки, який є електронним сервісом для користувачів. Оскільки в бібліотеці

існує різна функціональність для різних ролей, то цільовою аудиторією системи є три типи користувачів: незареєстрований користувач, клієнт та адміністратор.

Незареєстрований користувач - це ті, хто вперше відвідують сайт бібліотеки і не зареєстровані у системі. Клієнти - це користувачі, які зареєструвалися в системі та здійснюють пошук та оформлення замовлення книг, переглядають свій історію взяття книг на читання і т. д. Адміністратори - це співробітники бібліотеки, які мають розширені можливості в системі, включаючи можливість керування даними користувачів та книгами, редагування даних та багато іншого.

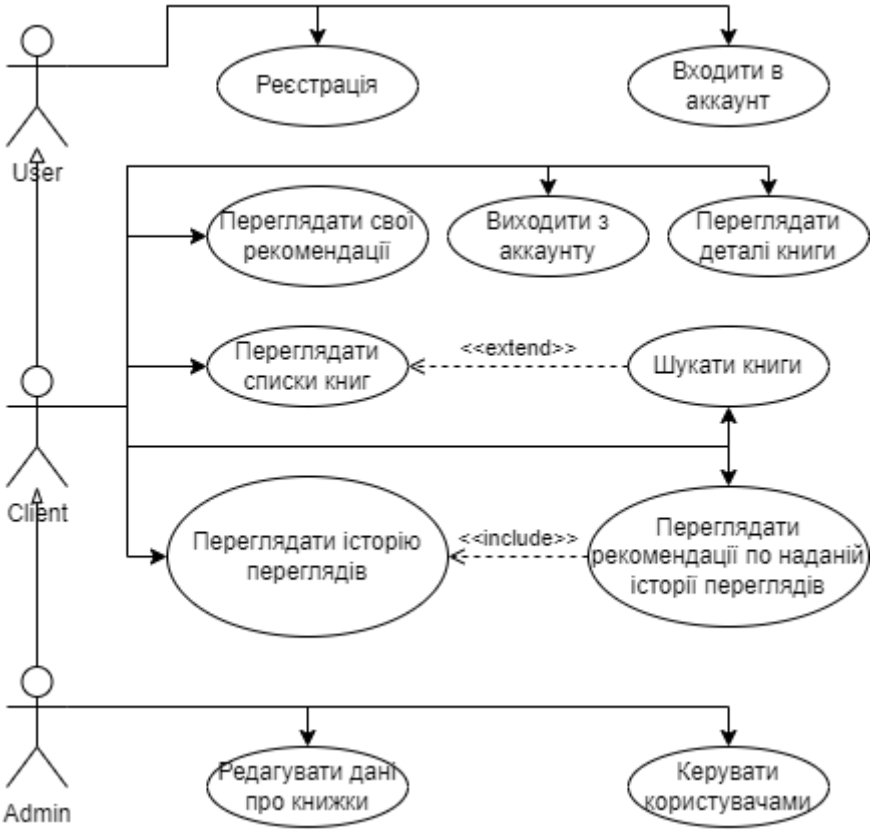

Рисунок 1.5 – Діаграма варіантів використання для онлайн бібліотеки

Для кожного з цих типів користувачів рекомендаційна система повинна пропонувати різні типи книг, що відповідають їхнім потребам та інтересам. Наприклад, незареєстрованим користувачам можуть бути запропоновані найбільш популярні книги, а клієнти можуть отримати персоналізовані рекомендації, враховуючи їх історію взяття книг на читання та рейтинги. Для

адміністраторів, можуть бути запропоновані рекомендації щодо книг, які можуть бути важливі для бібліотеки та покращити її колекцію. (рисунок 1.5)[14]

Можливості незареєстрованого користувача:

- реєстрація на сайті;.
- входити в аккаунт;

Можливості клієнта:

- авторизація на сайті;
- перегляд переглянутих книг;
- перегляд свого профілю;
- фільтрація усіх книжок за параметрами;
- перегляд кожної книги докладніше;
- перегляд автора книги;
- переглядати рекомендації по наданій історії переглядів;
- переглядати історії переглядів;
- перегляд індивідуальної рекомендації;

Можливості адміністратора:

- управління книгами (редагування, додавання, видалення);
- управління користувачами (редагування, додавання, перегляд, зміна ролі);

Для створення рекомендаційної системи важливо мати список усіх книг у бібліотеці, профіль користувача та базу даних усіх прочитаних книг. Надані рекомендації будуть корисними не лише для відвідувачів, але й для адміністраторів та сервіс-менеджерів.

1.6 Постановка задачі на розробку рекомендаційної системи онлайн бібліотеки

Метою цього дослідження є аналіз існуючих методів створення рекомендаційних систем та визначення найкращого методу обробки даних бібліотечних електронних сервісів для створення унікальної методології вилучення знань.

Таким чином, питання дослідження можуть бути обмежені наступними пунктами:

- Проаналізувати, як створити систему рекомендацій.

- На основі проаналізованих методів розробити метод інтелектуального аналізу знань для виявлення вподобань користувачів електронної бібліотеки.

Задача вилучення знань використовує наступну інформацію як вхідні дані:

- Список користувачів з роллю "клієнт" в системі (дані користувача системи включають в себе особистість, стать і дату народження).

- Список рекомендованих книг, доступних у бібліотеці.

- Список попереднього перегляду клієнтів.

- Кількість книг, які потрібно обробити.

Початковою інформацією для завдання є знання у вигляді моделі продукту знань. Виробнича модель - це модель, заснована на правилах, в якій знання можуть бути виражені у вигляді операторів типу "ЯКЩО умова, ТО дія". У нашому випадку умова включає:

- дані про книги.

- дані профілю клієнта.

Основна задача для сайту - розробка адаптивного та кроссбраузерного сайту онлайн бібліотеки. Головні критерії якого є сучасні матеріали які можливо замінити, додати або редагувати. Використовувати всі необхідні вимоги для сучасного ознайомлення книг.

Функціональні вимоги для сайту бібліотеки:

1. Реєстрація та авторизація на сайті

Для зручності системи є не обов'язкова реєстрація для кожного читача, читач може читати книгу як без реєстрації, або пройти швидку реєстрацію для особистого кабінету, в якому може переглядати історію прочитаних книг та редагувати свої пріоритети по жанрам

2. Збереження інформації користувача

Для більш коректної праці рекомендаційного списку, система буде записувати всі прочитані книги користувача, переглянуті жанри та переглянуті книги які користувач дивився раніше

3. Особисті оцінки користувача

В особистому кабінеті користувач зможе ставити оцінки всім прочитаним книгам, для зручного перегляду його книг та рекомендації друзям, а для сайту зручний рейтинг задоволення користувачів від читання книги

4. Редактор енциклопедичних та довідкових знань

Редактора книжок потрібен мати можливості для додавання, редагування або видалення книги, перегляд користувачів та їх заохочення - рейтинг популярних книжок, та додавання нових жанрів або піджанрів.

Нефункціональні вимоги до сайту онлайн бібліотеки можуть бути наступними:

1. Ефективність та швидкість: Сайт повинен завантажуватися швидко, щоб користувачі не чекали занадто довго. Додатково, сайт повинен мати ефективну роботу для забезпечення більшої кількості запитів в одиницю часу та ефективного використання ресурсів сервера.

2. Безпека: Сайт повинен забезпечувати безпеку користувачів, запобігаючи шахрайству, крадіжці особистої інформації та іншим формам кібератак.

3. Сумісність з браузерами та пристроями: Сайт повинен бути доступний для користувачів на різних пристроях та платформах (наприклад, мобільні телефони, планшети, настільні комп'ютери) та повинен працювати на різних браузерах.

4. Доступність: Сайт повинен бути доступним для людей з різними фізичними та розумовими обмеженнями. Наприклад, сайт повинен мати адаптивний дизайн для слабозорих та слабочутливих користувачів, а також повинен мати можливість відтворення звуку та тексту.

5. Надійність: Сайт повинен бути надійним та стійким до відмов, щоб запобігти втраті даних користувачів та забезпечити доступ до сайту в будь-який час.

6. Якість контенту: Контент, який розміщується на сайті, повинен бути високої якості та повинен бути чітким та легким для сприйняття.

7. Зручність використання: Сайт повинен бути простим у використанні та інтуїтивно зрозумілим для користувачів. Навігаційне меню та інтерфейс повинні бути зрозумілими та простими для використання, щоб забезпечити зручну та швидку навігацію.

8. Гнучкість та розширюваність: Сайт повинен бути гнучким та здатним до розширення, щоб забезпечити зміну функціональності та можливостей у майбутньому.

9. Зручний пошук: Сайт повинен мати зручний та ефективний пошук, щоб користувачі могли знайти необхідний контент швидко та ефективно.

10. Підтримка різних форматів: Сайт повинен підтримувати різні формати контенту, такі як електронні книги, аудіокниги, відео та інші, щоб забезпечити більшу гнучкість та доступність для користувачів.

11. Підтримка мультиплатформеності: Сайт повинен бути доступним на різних пристроях та платформах, включаючи комп'ютери, планшети та мобільні телефони, забезпечуючи зручний та швидкий доступ до контенту незалежно від типу пристрою та розміру екрану.

12. Захист персональних даних: Сайт повинен гарантувати захист персональних даних користувачів, включаючи особисті дані, інформацію про кредитні картки та іншу конфіденційну інформацію. Сайт повинен використовувати безпечні протоколи та механізми шифрування, щоб забезпечити захист персональних даних користувачів від несанкціонованого доступу.

13. Доступність для людей з обмеженими можливостями: Сайт повинен бути доступним для людей з обмеженими можливостями, включаючи особливі потреби у візуальному, слуховому та фізичному доступі до контенту.

14. Підтримка різних браузерів: Сайт повинен бути сумісним з різними веб-браузерами, щоб забезпечити доступність для користувачів з різних пристроїв та браузерів.

15. Забезпечення продуктивності та швидкості: Сайт повинен бути швидким та продуктивним, забезпечуючи користувачам швидкий та ефективний доступ до контенту та інших функціональних можливостей.

16. Підтримка високоякісного контенту: Сайт повинен підтримувати високоякісний контент, включаючи високу якість зображень, відео та звуку, щоб забезпечити користувачам максимальний комфорт та задоволення від користування сайтом.

17. Розгортання за допомогою контейнерів: Сайт повинен бути розгорнутий за допомогою контейнерів, що забезпечує швидку та надійну інсталяцію на різних серверах та платформах, зменшує час віддачі відомостей та забезпечує простір для розвитку та масштабування.

18. Зручний інтерфейс сайту

Бібліотека повинна мати зручний та приємний дизайн, завдяки якому користувачу буде зручно читати улюблені книги. Та змінювати поле для перегляду сторінки

19. Зручна сторінка контенту

Сторінка повинна бути зручною, має бути варіанти перегляду книги, його редагування шрифтів та кольору сторінки.

1.7 Висновки до першого розділу

У даному розділі дипломної роботи проведено аналіз існуючих онлайн бібліотек та визначено їхні недоліки, зокрема проблеми інформаційного перевантаження користувачів. Також розглянуто актуальність розробки методу вилучення знань для вирішення цих проблем. Описано цільову аудиторію, яка потребує рекомендаційної системи в онлайн бібліотеці, та надано детальний аналіз існуючих онлайн бібліотек з їхніми функціональними можливостями. Визначено характеристики цільової аудиторії, які повинні враховуватися при

розробці рекомендаційної системи. У розділі також описано процес розгортання застосунку та сформульовано конкретну мету дипломної роботи з постановкою задачі на розробку рекомендаційної системи онлайн бібліотеки. Завдяки проведеному аналізу вдалося визначити напрямки подальшої розробки та покращення онлайн бібліотек та їхніх рекомендаційних систем, що забезпечить більш ефективне та зручне використання цих послуг користувачами.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ІС “ОНЛАЙН БІБЛІОТЕКА”

2.1 Розробка архітектури

2.1.1 Функціональний аналіз предметної області

Бізнес-процеси (рисунок 2.1) допомагають отримати повну картину початку розробки системи.

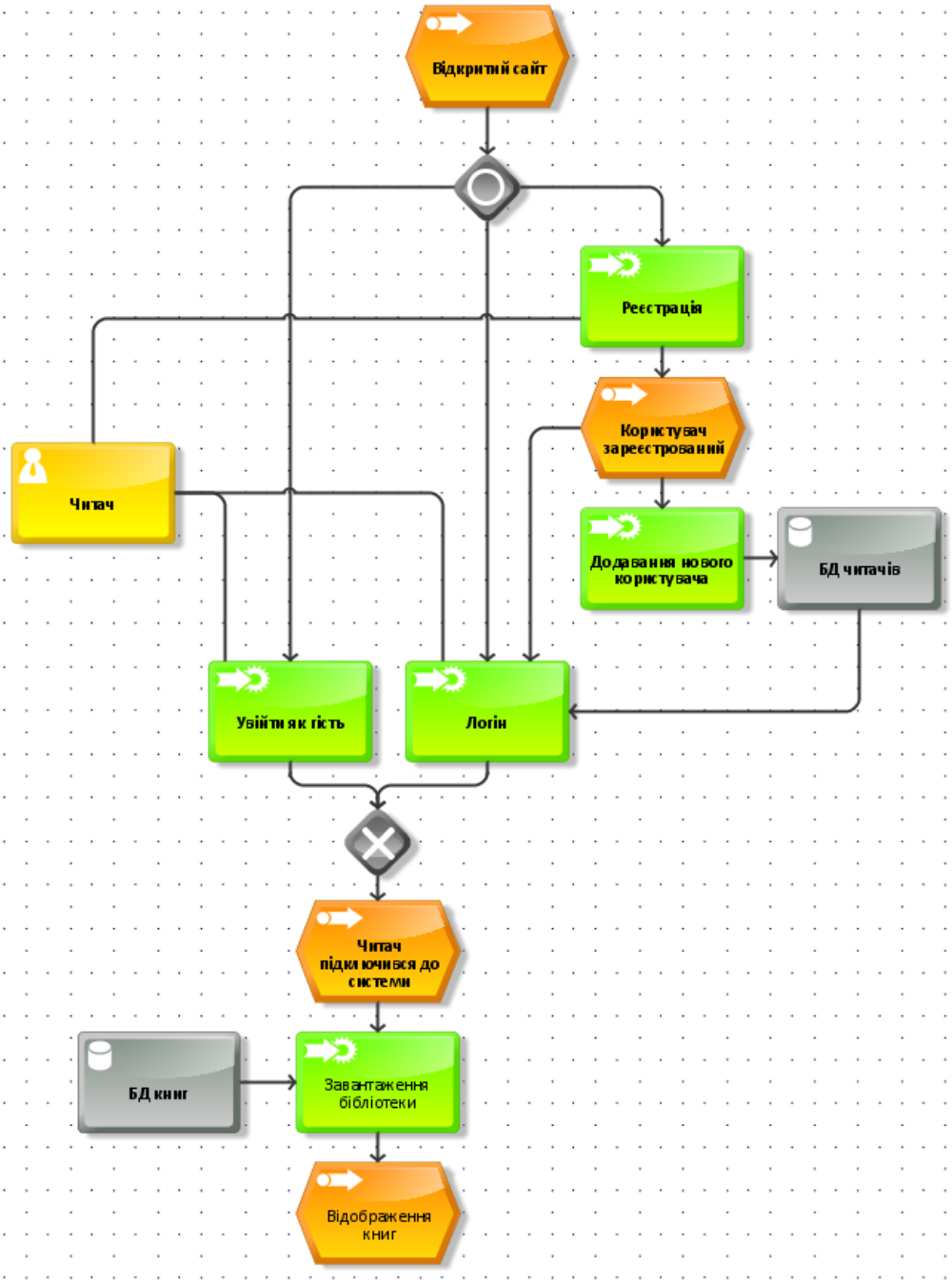

Рисунок 2.1 Діаграма Event-Driven Process Chain для функціонування сайту

Вершина дерева функцій є функція модулю бібліотеки - головний модуль додатку бібліотеки. Функції даного застосунку діляться на два модулі: клієнтський та адміністративний модуль.

До адміністративного застосунку належить основна функція - робота з контентом, а саме додавання нових книг до бібліотеки сайту для доповнення основного контенту та перегляду сторінок для користувачів, та остання функція - редагування існуючих книг, головна функція для самого сайту, в нинішній ситуації можливі обмеження для вільного доступу для книг які заборонені законом України.

Клієнтський функціонал(рисунок 2.2) вміщує основні модулі для користувача: реєстрація, пошук книг та ведення акаунту.

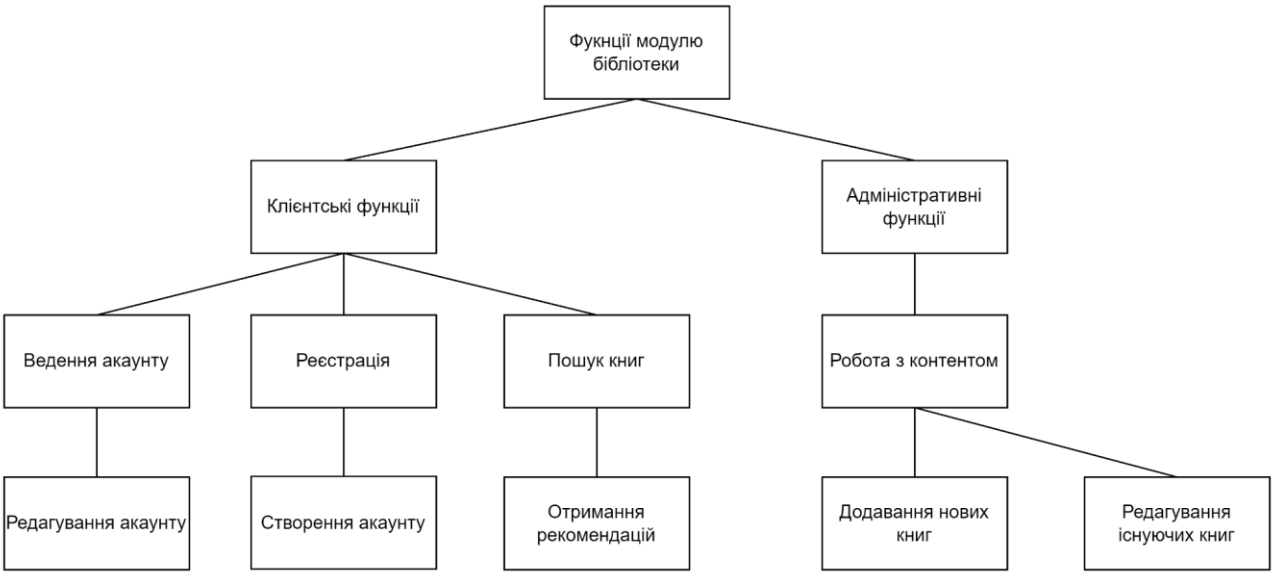

Рисунок 2.2 Дерево функцій модулю бібліотеки

Реєстрація допомагає відрізняти нового користувача та вже зареєстрованого, для нових користувачів доступний модуль реєстрації для доступу наступної функції клієнтського модулю - ведення акаунту в якому читач може редагувати особисті дані та добавляти нові прочитані книги. Пошук книг головна функція всього сайту, отримання книг за назвою та отримання рекомендацій по жанру книг.

Таким чином працює сайт онлайн бібліотеки: початком запуску системи є відкритий сайт користувачем в своєму браузер; Перед користувачем відкривається можливість трьох функцій: реєстрація, логін та акаунт гостя, реєстрація отримує дані від користувача та додає до бази даних нового читача. Логін вже для зареєстрованих користувачів які вже реєструвались раніше. Гість це та людина яка не бажає зареєструвалась на сайті, а просто бажає читати книжки. Після авторизації читачу відкривається можливість почати пошук книг та отримати необхідні книги.

Ланцюжок процесів(рис. 2.3), що відбуваються під час функціонування сайту: основою є процес “Робота з бібліотекою” з двома підпроцесами “Реєстрація на сайті” та “Користування бібліотекою”. В реєстрацію входить підпроцес введення основних даних, а саме введення логіну та паролю для акаунту читача. Користування бібліотекою включає можливість відреагувати особистий кабінет, особистих полів та пріоритетів пошуку книг та пошук книг за назвою або жанрами.

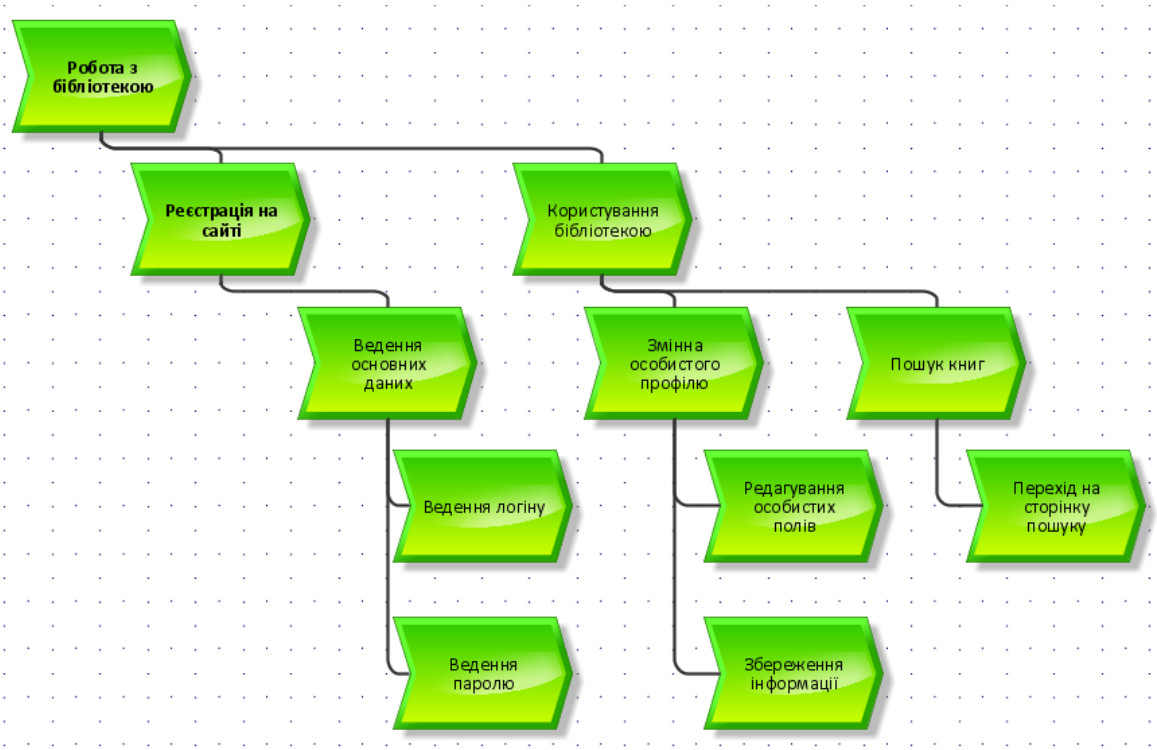

Рисунок 2.3 Діаграма Value Added Chain Diagram для функціонування мобільного додатку

2.1.2 IDEF0 процесу видачі рекомендацій за допомогою інформаційної системи

Представимо виконувані процеси у предметному середовищі за допомогою діаграм «ЯК БУДЕ» у нотації IDEF0[20], тобто вже з використанням сайту бібліотеки.

Першим етапом проектування є побудова контекстної діаграми (рисунок 2.4). На вході система отримує дані користувача, а саме його логін, для входу у систему, можливо реєстрація користувача, а можливо це звичайний гість сайту який просто дає системі дані для отримання на виході системи - книги.

Елементом керування є закон України про «Захист персональних даних», тому що на кожному етапі зберігаються особисті дані користувача. Та закон України про “Авторські права” для можливості розповсюдження авторських книг та вільного перегляду в інтернеті на території України. Механізмом системи виступає користувач.

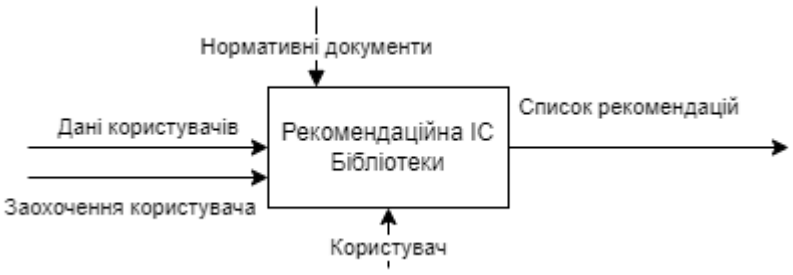

Рисунок 2.4 Контекстна діаграма ЯК БУДЕ

На наступному етапі проведемо декомпозицію(рисунок 2.5) основних процесів діаграми ЯК БУДЕ (рисунок 2.4).

Першим процесом є авторизація завдяки якій система зможе отримати необхідні дані від користувача та розуміти - це гість або авторизований читач. Далі системі необхідно відібрати інформацію за предметами користувача які цікавлять користувача та проаналізувати їх, для створення списку книг які необхідні читачу для вибору необхідної книги для перегляду.

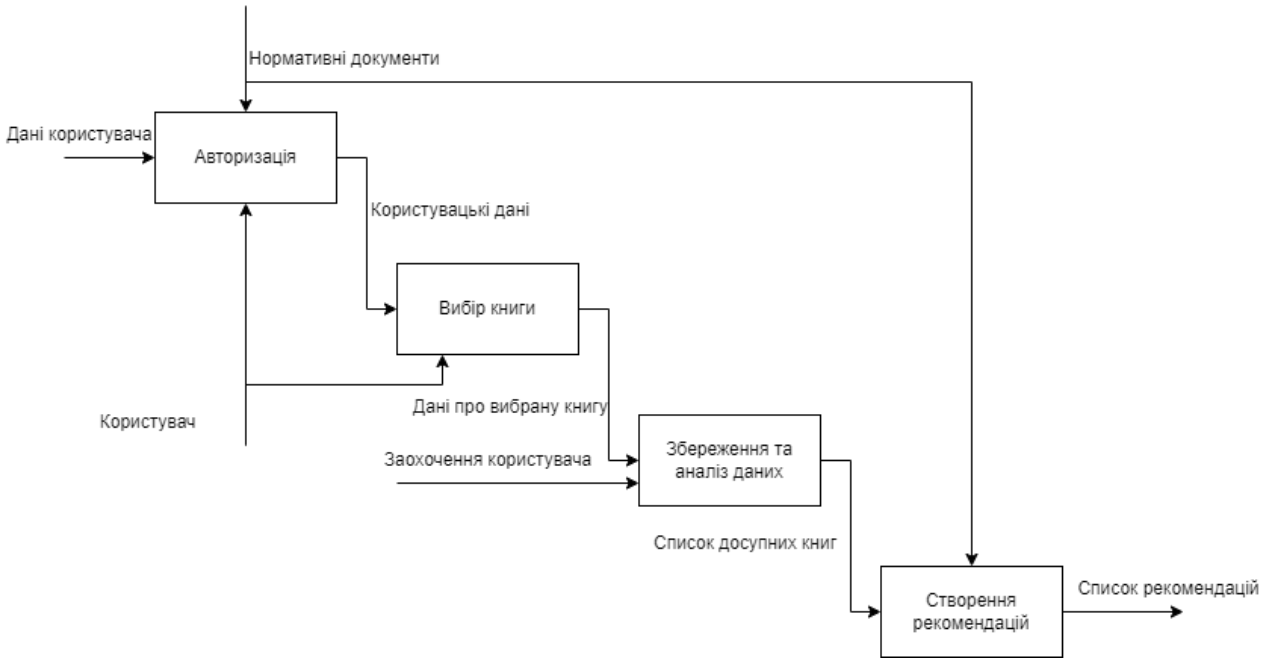

Рисунок 2.5 Декомпозиція основних процесів діаграми ЯК БУДЕ

2.1.3 Розробка архітектура інформаційної системи

Новий етап - побудова архітектури розроблюваної інформаційної системи. Дана схема відображає модель, структуру, виконувані функції та взаємозв'язок компонентів.

Клієнтський модуль вміщує в собі 3 модулі, робота з профілем, модуль реєстрації та модуль пошуку книг. Робота з профілем має функціонал змінювати особистий кабінет, змінити пароль, логін або інші особисті дані користувача. Модуль реєстрації має спільне з модулем роботи з профілем - тільки реєстрація це міняти пусті поля для запису даних читача до системи. Модуль пошуку книг вміщує в собі 3 пункти: пошук за предметом, пошук за назвою та пошук за автором, ці всі модулі дозволяють швидко та зручно знайти необхідну книгу для користувача.

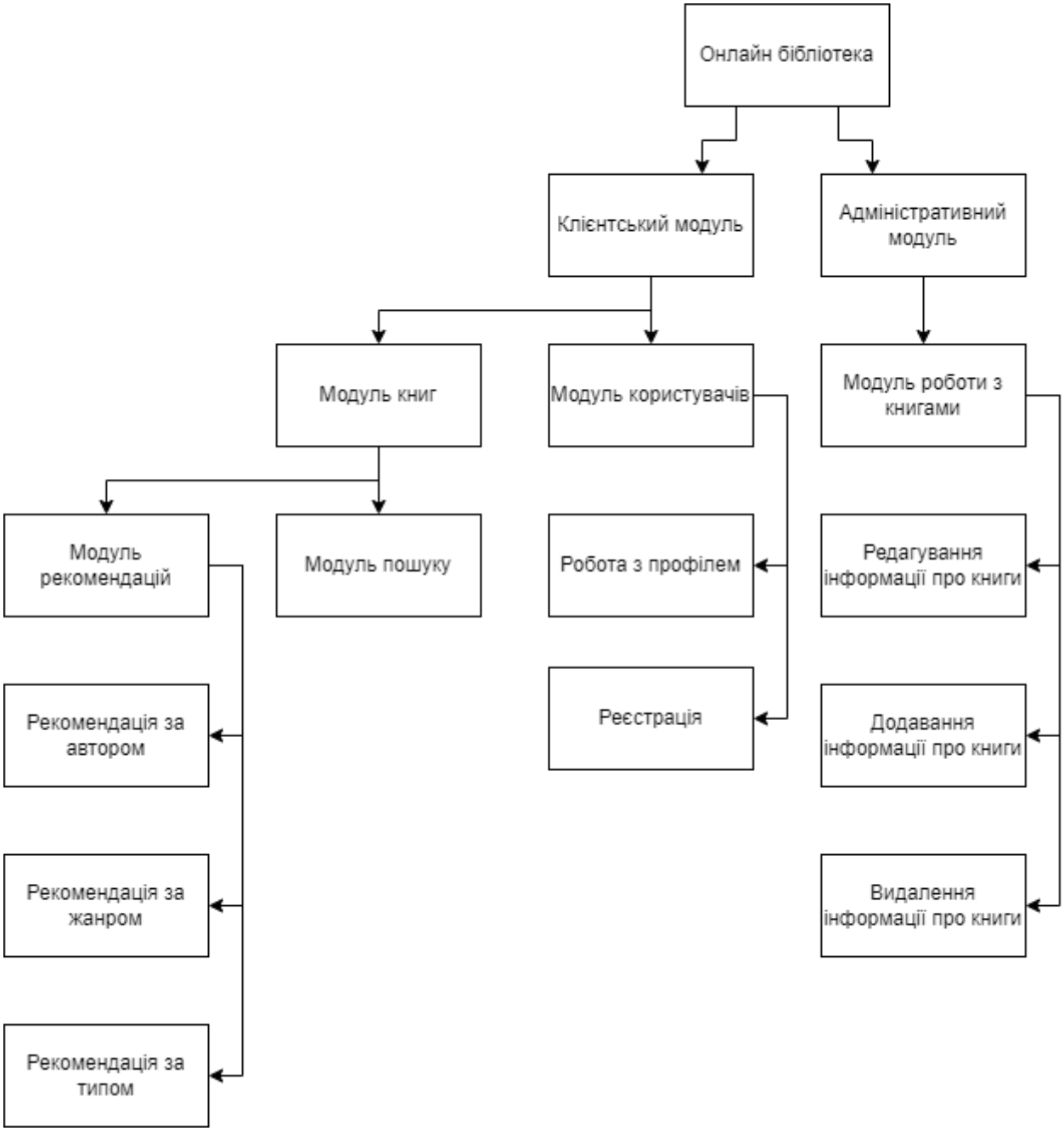

Рисунок 2.6 Архітектура модулів онлайн бібліотеки

Адміністративний модуль включає в собі тільки один модуль - роботу з книгами. Адміністратор має можливість змінювати існуючі книги, для редагування людского фактору помилки. Адміністратор може добавити нові книги для оновлення бібліотеки та буди в сучасним сайтом з новими книгами та як додавати так може і видаляти книги які можливо були заборонені на території України.(рисунок 2.6)

2.1.4 Моделювання розгортання інтелектуального модуля

Для данного програмного застосунку було виділено основні контейнери(рисунок 2.7)[21]:

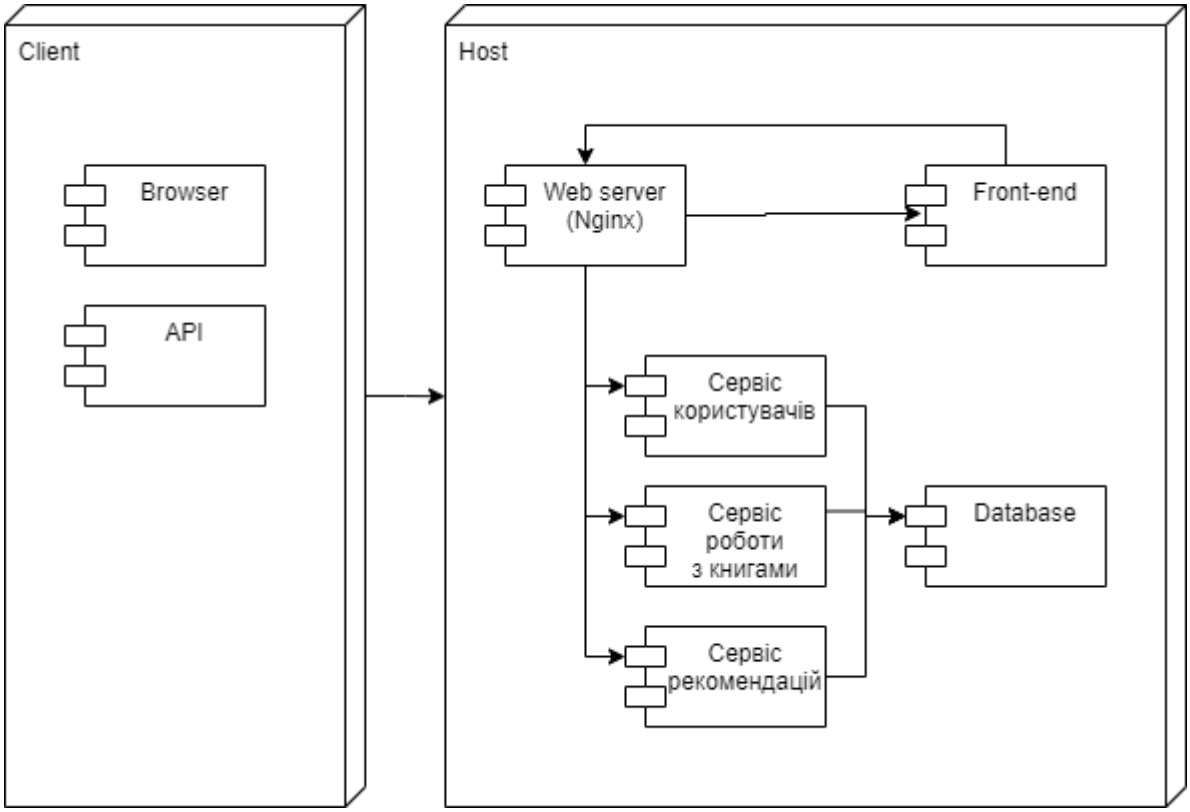

Рисунок 2.7 Діаграми розгортання інтелектуального модуля

- Веб сервер - сервіс, що відповідає за налаштування комунікації між серверною частиною застосунку і клієнтами-користувачами;
- Сервіс користувацького інтерфейсу – сервіс, що відповідає за комунікацію між користувацьким інтерфейсом і користувачем;
- Сервіс користувачів - сервіс, що відповідає за користувачів застосунку;
- Сервіс роботи з книгами- сервіс, що відповідає за книги застосунку;
- Сервіс рекомендацій- сервіс, що відповідає за рекомендації для користувачів застосунку;

2.1.5 Веб сервер

На данний момент існує багато полулярних та широко-застосованих вебе серверів, але я зупинив свій вибір на Nginx.

Саме з веб серверу починається взаємодія з сервісами всередині програми.

Nginx - це служба зворотного проксі, яка приймає клієнтські запити на вхід і пересилає їх на один або декілька серверів. Після обробки запиту клієнта проксі-сервер надсилає відповідь сервера назад.

У той час як більшість сучасних додатків працюють самостійно як веб-сервери, веб-сервер Nginx надає розширені можливості, такі як:

- Балансування навантаження між серверами - зважене балансування та можливість розподіляти трафік між сервісами.

- Функціональність та прискорення TLS/SSL, яка відсутня у більшості користувацьких додатків.

У нашому випадку програма повинна реалізовувати конфігурацію для сервера, яка описує наступну логіку:

- Запити приймаються на порти 80 і 443, які відповідають за протоколи HTTP і HTTPS.

- Якщо користувачі не знаходять HTTPS-адресу для веб-служби Nginx, вони автоматично перенаправляються на безпечний протокол.

- SSL налаштовується за допомогою HTTPS-сертифіката.

- Кожен вхідний запит до веб-проксі отримує стандартний проксі-заголовок, в якому вказується, звідки походить запит і до якого сервісу його слід направити. Таке розташування полегшує навігацію мережею послуг.

- Нарешті, команда UPSTREAM може бути використана для перенаправлення трафіку на докер-сервіс, який обробляє запит користувача. В даному випадку - "сервер".

2.1.6 Моделі бази даних

Можлива концептуальна модель рекомендаційного модуля може мати наступний вигляд(рисунок 2.8):

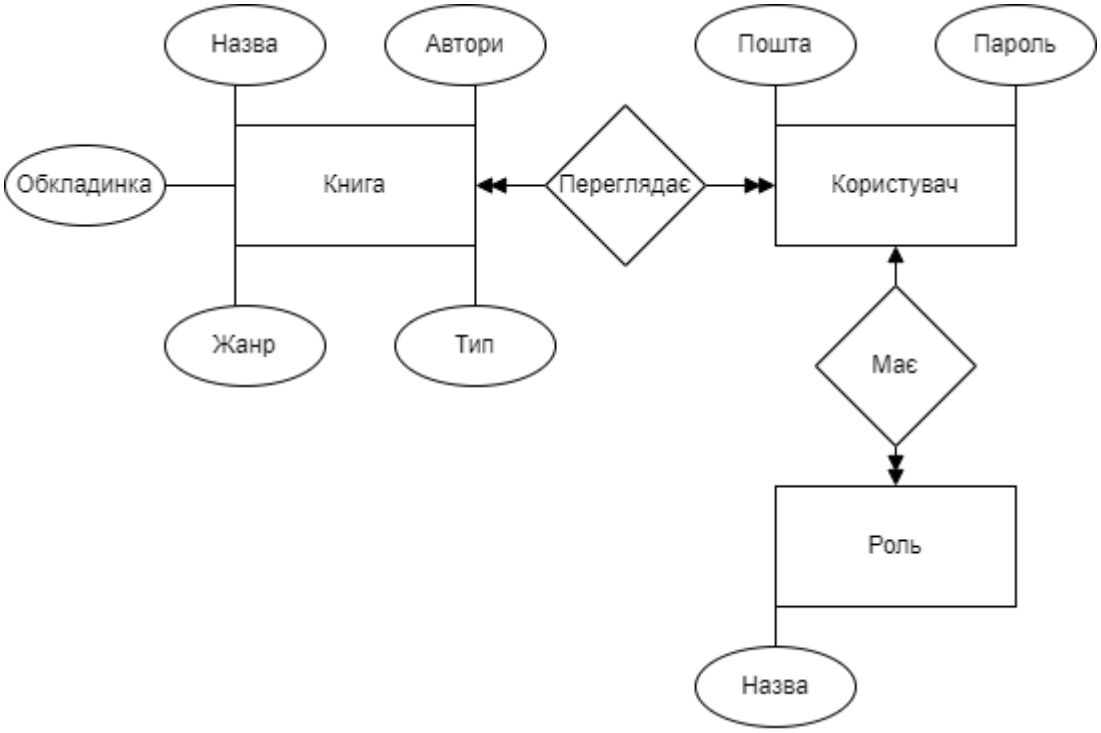

Рисунок 2.8 Концептуальна модель БД

Елементи концептуальної моделі БД:

1. Збір даних користувача: коли ви входите в систему або переглядаєте книгу, наш модуль може збирати інформацію про вас, таку як ідентифікатор користувача, ім’я, адреса електронної пошти, роль, прочитані книги та інші характеристики.[26]

2. Збір даних про книги: модуль може збирати інформацію про книги (таку як ідентифікатор, назва, автор, жанр, тип та інші характеристики).

3. Збір даних перегляду: модуль може збирати інформацію про читання книг користувачем, таку як ідентифікатор користувача, ідентифікатор книги, дата читання.

4. Аналіз даних: на основі зібраних даних модуль може виконувати аналіз, щоб зрозуміти, які книги користувачі читають частіше, а які можуть зацікавити їх у майбутньому.

5. Створення рекомендацій: на основі результатів аналізу модуль може генерувати персоналізовані рекомендації для користувача. Наприклад, якщо

користувач переглядає багато книг певного жанру, модуль може рекомендувати книги того самого жанру.

6. Відображення рекомендацій: Рекомендації можуть відображатися користувачам на сторінках книг або інших сторінках сайту.

Отже, така концептуальна модель може допомогти підвищити задоволення користувачів від використання системи, роблячи її більш персоналізованою і зацікавлюючою.

Далі опишемо у таблиці 2.1 зв’язки між сутностями.

Таблиця 2.1 Зв’язки між сутностями

Сутності, що утворюють зв’язок	**Тип зв’язку**	**Пояснення**
User - View	one-to-many	Кожен користувач може переглядати багато книжок.
Book - View	one-to-many	Кожна книжка може бути переглянута багатьма користувачами.
User - Role	one-to-many	Кожен користувач може мати тільки одну роль, але кожна роль може належати до багатьох користувачів.(ролі представлені у розділі 1.5)

Опишемо у таблиці 2.2 обмеження та типи даних атрибутів:

Таблиця 2.2 Обмеження та типи даних атрибутів

Модель	**Атрибут**	**Тип даних**	**Обмеження**
User	id	Integer	primary key, index
User	email	String	unique, index
User	hashed_password	String	
User	role_id	Integer	foreign key ("roles.id")
Role	id	Integer	primary key, index
Role	name	String	unique, index
Book	id	Integer	primary key, index
Book	title	String	
Book	authors	String	
Book	genre	String	
Book	type	String	
Book	img_url	String	
View	user_id	Integer	foreign key ("users.id"), primary key
View	book_id	Integer	foreign key ("books.id"), primary key

У таблиці 2.2 наведено типи даних та обмеження для кожного атрибуту моделі. Також зазначено зовнішні ключі ("foreign key") для атрибутів, що посилаються на інші таблиці, та ключі, що визначають первинний ключ ("primary key") та індекси ("index").

Розробимо фізичну модель бази даних(рисунок 2.9).

У фізичній моделі бази даних кожна модель відображається у відповідній таблиці. Для зв'язків many-to-many були створені додаткові таблиці.

Також для кожного атрибуту було визначено його тип даних та обмеження (такі як primary key, foreign key, unique та index). Типи даних char(256) використовуються для збереження текстових даних, а тип int для цілих чисел. Обмеження unique та index використовуються для підвищення ефективності пошуку даних.

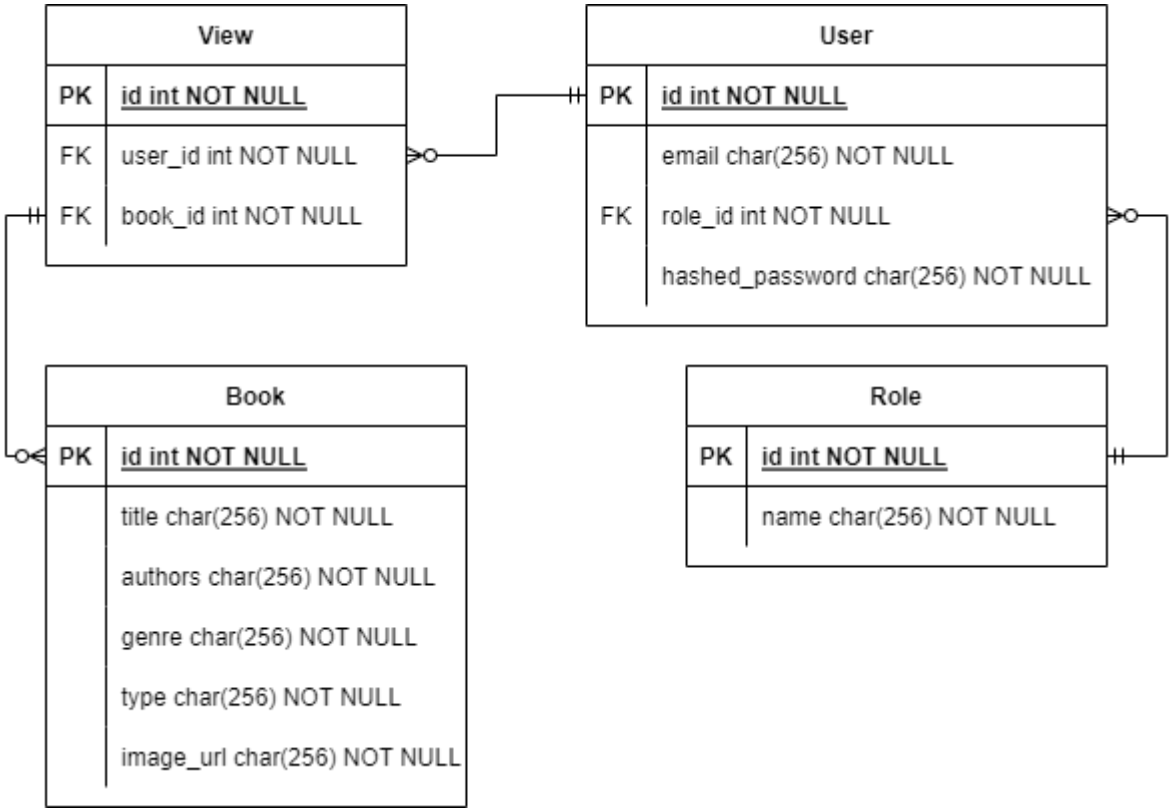

Рисунок 2.9 Фізична модель бази даних

2.2 Аналіз для практичної реалізації

Початком розробки рекомендаційної системи онлайн бібліотеки, починається з вибору саме алгоритму на якому буде побудований сам додаток – зупинилися на методі колаборативної фільтрації.

У системах рекомендацій часто використовуються методи колаборативної фільтрації, яка полягає у фільтрації даних або шаблонів з використанням співпраці між різними агентами, точками зору, джерелами даних тощо. Існує два типи спільної фільтрації: вузький та більш широкий. Ця технологія застосовується для обробки великих обсягів інформації в різних галузях, таких як зондування та моніторинг, фінансові установи, електронна комерція тощо. У даному дослідженні буде розглянуто колаборативну фільтрацію користувацьких даних, хоча деякі методи та підходи можуть застосовуватися і в інших випадках.[1]

У новішому, вужчому розумінні, спільна фільтрація — це метод прогнозування в системах рекомендацій, який використовує відомі переваги

(рейтинги) набору користувачів, щоб передбачити невідомі переваги іншого користувача. Головне припущення спільної фільтрації лежить в наступному: користувачі, хто оцінював будь-який елемент таким же чином у минулому, мають тенденцію оцінювати інші елементи подібним чином у майбутньому. Наприклад, при допомозі спільної фільтрації застосунок може передбачити, яка книга сподобається користувачеві, враховуючи неповний список уподобань користувача (лайків і антипатій). Хоча використовувана інформація збирається від багатьох учасників, кожен користувач робить прогнози індивідуально. Це відрізняє спільну фільтрацію від простішого підходу, який дає кожному об’єкту інтересу середній бал, наприклад, на основі кількості голосів за нього. Колаборативна фільтрація відрізняється від більш простих підходів тим, що надає усереднену оцінку для кожного об'єкта інтересу, що базується на кількості голосів, які були подані за нього. У цій області активно проводяться дослідження навіть у наш час, оскільки існують невирішені проблеми в методі колаборативної фільтрації. Це може бути корисно для включення до дипломної роботи.[8]

У нашому столітті інформаційного вибуху методи, такі як колаборативна фільтрація, є надзвичайно корисними для створення персоналізованих рекомендацій. Кількість об'єктів, що належать до однієї категорії, наприклад, фільмів, музики, книг, новин або веб-сайтів, стала настільки великою, що окрема людина не здатна ознайомитись зі всіма ними і відібрати відповідні. [10]

Системи колаборативної фільтрації зазвичай застосовують двоступеневу схему :

Знаходять тих, хто поділяє оціночні судження «активного» (прогнозованого) користувача.[2]

У столітті інформаційного розквіту методи колаборативної фільтрації є важливим інструментом для створення персоналізованих рекомендацій, оскільки кількість об'єктів у категоріях, таких як фільми, музика, книги, новини та вебсайти, стала настільки великою, що неможливо оглянути їх усіх, щоб зробити відповідний вибір. Колаборативна фільтрація може бути побудована відносно користувачів системи або предметів, що є в системі. Алгоритм, що ґрунтується

на предметах, будує матрицю, що відображає відносини між предметами, для пошуку схожих елементів, а потім використовує цю матрицю та інформацію про користувача для створення прогнозу оцінок. Існує також інша форма колаборативної фільтрації, яка базується на неявній поведінці користувача, де система аналізує, як користувачі взаємодіють з продуктами, і використовує ці дані для передбачення майбутньої поведінки користувача.[3]

2.2.1 Типи колаборативної фільтрації

- Заснований на пам'яті

Цей метод використовує рейтинг користувача як основу для порівняння між користувачами або предметами з метою надання рекомендацій. Цей підхід був одним з перших, що застосовувався в багатьох системах електронної комерції, тому що він є ефективним та простим у реалізації. Типові приклади такого методу включають CF та рекомендації на основі топ-N товарів або користувачів. Наприклад, у підходах, заснованих на користувачеві, вартість оцінки, яку користувач u дає виробу «i» розрахована як сукупність схожих оцінок виробу іншими користувачами(формула 2.1):

$$r_{u,i} = aggr_{u\prime \in U} r_{u\prime,i}, \tag{2.1}$$

де «U» позначає сукупність N «найкращих» користувачів, які найбільш близькі до користувача u, що оцінює виріб «i».

Деякі приклади функцій агрегації(формули 2.2-2.4):

$$r_{u,i} = \frac{1}{N} \sum_{u\prime \in U} \quad r_{u\prime,i}, \tag{2.2}$$

$$r_{u,i} = \frac{1}{N} \sum_{u\prime \in U} \quad simil(u, u') r_{u\prime,i}, \tag{2.3}$$

$$r_{u,i} = \frac{1}{N} \sum_{u\prime \in U} \quad simil(u, u')(r_{u\prime,i} - r_{\underline{u\prime}}) , \tag{2.4}$$

де k - нормуючий множник, задається як $k = 1/\sum_{u' \in U}$ |simil(u,u')|i$r_{\underline{u}}$ є середня оцінка користувача u для всіх виробів, оцінених ним.

- Заснований на сусідстві

Алгоритм, заснований на сусідстві, обчислює подібність двох користувачів або виробів, виробляє прогноз для користувача, приймаючи середнє зважене всіх оцінок. Обчислення схожості між виробами або користувачами є важливою частиною цього підходу. Багаторазові заходи, такі як кореляції Пірсона і схожість, заснована на скалярному добутку, використовується для цього.[11]

Схожість двох користувачів X, Y через кореляцію Пірсона визначається як(формула 2.5)

$$simil(x,y) = \frac{\sum_{i \in I_{xy}} (r_{x.i} - r_{\underline{x}})(r_{y.i} - r_{\underline{y}})}{\sqrt{\sum_{i \in I_{xy}} (r_{x.i} - r_{\underline{x}})^2 \sum_{i \in I} (r_{y.i} - r_{\underline{y}})^2}}, \quad (2.5)$$

де Іху - це набір елементів, оцінених як користувачем x, так і користувачем y.

Підхід, заснований на скалярному добутку визначає скалярний добуток між двома користувачами x і y, як(формула 2.6):

$$simil(x,y) = cos(\underline{x}, \underline{y}) = \frac{\underline{x} * \underline{y}}{||x|| * ||y||} = \frac{\sum_{i \in I_{xy}} r_{x.i} r_{y.i}}{\sqrt{\sum_{i \in I_x} r_{x.i}^2} \sqrt{\sum_{i \in I_y} r_{y.i}^2}}, \quad (2.6)$$

Заснований на користувачеві алгоритм топ-H рекомендації використовує засновану на подібності векторну модель для визначення K — більшості подібних користувачів до активного користувача. Після того, як знайдені найбільш схожі користувачі, їх відповідні матриці агрегуються для визначення рекомендованого набору елементів. Популярний метод, знаходження схожих користувачів — Locality-sensitive hashing, який реалізує механізм пошуку найближчих сусідів у лінійному часі.[4]

Цей підхід має кілька переваг, зокрема, очікуваність результатів, яка є важливою складовою рекомендаційних систем; легкість створення і використання; простота оновлення даних; та гарна масштабованість зі співавторами рейтингових пунктів. Однак, при такому підході є кілька недоліків. Наприклад, його продуктивність знижується, коли дані стають розрідженими, що є частим для виробів, пов'язаних з мережею. Це ускладнює масштабованість такого підходу та створює проблеми з обробкою великих об'ємів даних. Хоча цей підхід може ефективно обробляти нових користувачів, додавання нових елементів стає складнішим, оскільки потрібно переробляти всі елементи структури векторного простору. При додаванні нового елементу необхідно оновлювати структуру векторного простору, включаючи всі елементи.

- Заснований на моделі

Цей підхід пропонує рекомендації, що ґрунтуються на статистичних моделях, які оцінюються за допомогою таких методів, як байєсовські мережі, кластеризація та латентно-семантичні моделі, такі як сингулярний розклад, імовірнісний латентно-семантичний аналіз, прихований розподіл Діріхле та марковський процес вирішення на основі моделей. Ці моделі розробляються з використанням алгоритмів машинного навчання та інтелектуального аналізу даних з метою знаходження закономірностей на основі навчальних даних. Кількість параметрів у моделі може бути зменшена залежно від типу за допомогою методу головних компонент. [12]

У порівнянні з методом, що базується на сусідстві, цей підхід є більш складним і дозволяє отримувати більш точні передбачення, оскільки він допомагає виявити приховані фактори, що пояснюють спостережувані оцінки. Крім того, цей підхід краще обробляє розріджені матриці, що дозволяє ефективно використовувати його для масштабних даних. Ці переваги роблять його відмінним вибором для досліджень, що включають великі обсяги даних.

Недоліки цього підходу полягають в «дорогому» створенні моделі. Необхідний компроміс між точністю і розміром моделі, тому що можна втратити корисну інформацію у зв'язку із скороченням моделей.

- Гібридний підхід

У розробці рекомендаційних систем для комерційних сайтів, найбільш поширеним підходом є гібридний підхід, що об'єднує в собі підхід на основі сусідства та модельний підхід. Цей підхід допомагає вирішити обмеження початкового підходу на основі сусідства та покращити якість прогнозів, а також подолати проблему розрідженості даних та втрати інформації. Однак, реалізація та застосування даного підходу є складним та дорогим.[5]

2.2.2 Приклад розрахунку колаборативної фільтрації

Для проведення дослідження щодо перегляду книжок 3-ма користувачами, доступними до перегляду, була створена таблиця 2.3, що складається з 3 рядків (відповідно до кількості користувачів) та 5 стовпців (відповідно до кількості книжок). Кожна комірка таблиці містить значення "1", якщо відповідний користувач переглядав відповідну книгу.

Таблиця 2.3 Перегляд книжок кожним з користувачів

	Користувач 1	Користувач 2	Користувач 3
Книга 1 Автор 1 Тип 1 Жанр 1	1		
Книга 2 Автор 2 Тип 2 Жанр 3		1	
Книга 3 Автор 2 Тип 3 Жанр 3		1	
Книга 4 Автор 3 Тип 2 Жанр 3	1		1
Книга 5 Автор 2 Тип 3 Жанр 2		1	

Ця таблиця дозволяє нам візуалізувати перегляд книжок кожним з користувачів. Наприклад, з таблиці видно, що Користувач 1 переглядав Книгу 1, та Книгу 4, а Користувач 2 переглядав Книгу 2, Книгу 4 та Книгу 5.

На основі наданої таблиці та формул 2.1 – 2.4 можна отримати наступні рекомендації для користувачів:

Незалежні рекомендації на основі переглянутих книг іншими для Користувача 1 включатимуть Книгу 2, Книгу 3 та Книгу 5, оскільки Користувач 1 вже переглянув Книгу 1 та Книгу 4. Отже, Книга 4 не буде прорекомендована. Візуалізація цих рекомендацій представлена на рисунку 2.10.

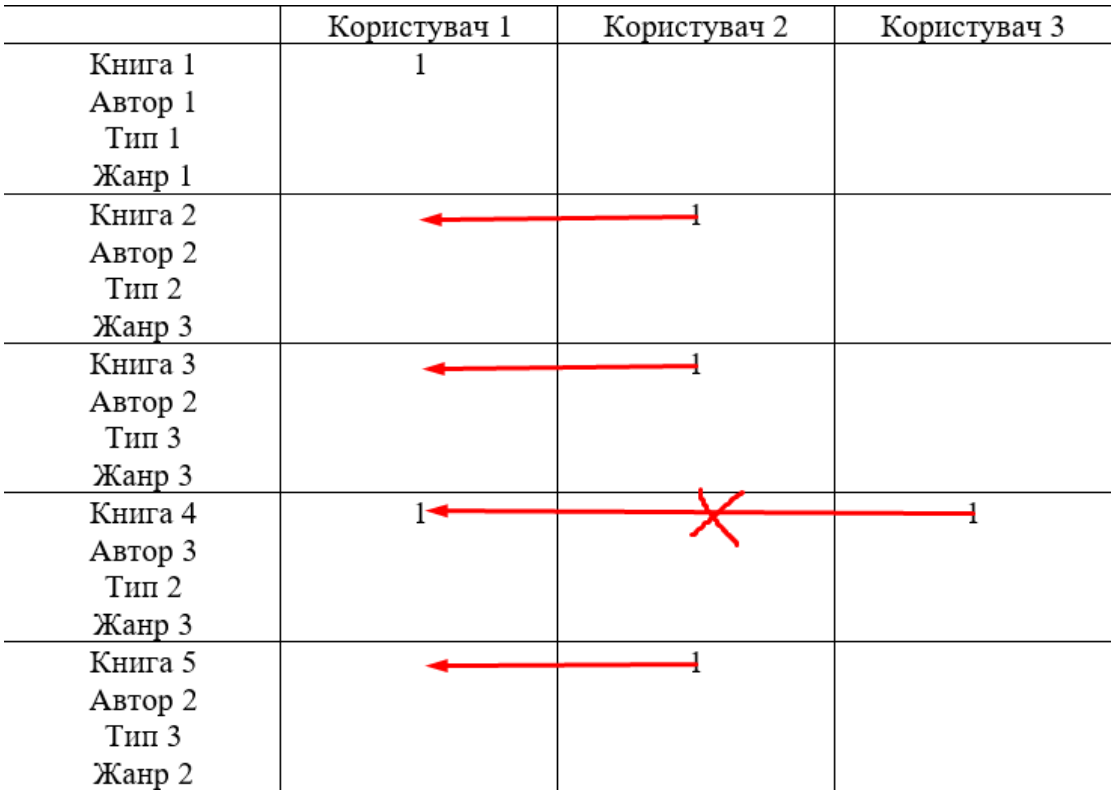

Рисунок 2.10 Результати незалежної рекомендації для першого користувача

Рекомендації за типом на основі переглянутих книг іншими для Користувача 3 включатимуть лише Книгу 2, оскільки Користувач 1 або вже переглянув інші книги, або не цікавиться типом. Інші книги не будуть прорекомендовані. Візуалізація цих рекомендацій представлена на Малюнку 2.11.

Рисунок 2.11 Результати рекомендації за типом для третього користувача

Аналогічні рекомендації будуть відбуватися із врахуванням жанру та автора книг.

2.2.3 Обгрунтування вибору типу алгоритму колаборативної фільтрації

На основі проведених експериментів і аналізу отриманих результатів було встановлено, що алгоритм, базований на пам'яті, виявився найефективнішим з точки зору точності рекомендацій. Цей тип колаборативної фільтрації використовує пам'ять системи для збереження інформації про взаємодії користувачів з об'єктами та рекомендаційні зв'язки між ними.

Основними перевагами алгоритму, базованого на пам'яті, є його простота та можливість враховувати складні взаємодії між користувачами та об'єктами. Він може працювати з різноманітними типами даних та добре справляється з проблемою холодного старту, коли недостатньо інформації про нових користувачів або об'єкти.

Проте, варто відзначити, що алгоритм, базований на пам'яті, може вимагати значних обчислювальних ресурсів при роботі з великими обсягами даних, оскільки він оперує повним набором взаємодій. Також, в разі зміни взаємодій або додавання нових користувачів та об'єктів, алгоритм може потребувати перерахунку рекомендаційної моделі.

Отже, на основі проведених досліджень, алгоритм, базований на пам'яті, є рекомендованим вибором для реалізації колаборативної фільтрації у рекомендаційних системах, оскільки він забезпечує високу точність рекомендацій та легко адаптується до різних типів даних та сценаріїв використання. При цьому, необхідно враховувати обчислювальні вимоги та потребу періодичного оновлення рекомендаційної моделі при змінах в даних.

2.3 Висновки до другого розділу

Критично важливим етапом в розробці будь-якої інформаційної системи є проектування її архітектури. У цьому розділі дипломної роботи було розглянуто процес проектування архітектури інтелектуального модулю.

У результаті функціонального аналізу предметної області було визначено, що основним завданням інформаційної системи є надання користувачам можливості зручного та швидкого пошуку, вибору та отримання доступу до необхідної літератури. За допомогою IDEF0 процесу було розроблено модель процесу видачі рекомендацій користувачам.

Початок шляху - розробка архітектуру системи яка описана через дерево функцій, діаграму Event-Driven Process Chain, діаграму Value Added Chain Diagram та детальний опис архітектури системи.

У даному розділі були дані визначення для зв'язків між моделями бази даних, описано обмеження та типи даних атрибутів, а також була надана фізична модель бази даних.

Крім цього, було запропоновано використання різних типів зв'язку між деякими моделями, а саме many-to-many, з метою покращення функціональності та гнучкості бази даних.

Було визначено структуру програмного застосунку, побудовано її у вигляді схеми та описано специфікацію програмних модулів та детальний опис методу рекомендацій. Також, на основі проведених досліджень, алгоритм, базований на пам'яті, є вибором для реалізації колаборативної фільтрації для рекомендаційного модулю.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Обґрунтування вибору програмних засобів

3.1.1 Програмні засоби розробки інтерфейсу онлайн бібліотеки

HTML, CSS та JavaScript - це стандартні технології, що використовуються для розробки інтерфейс-частини веб-додатків. Вони дозволяють розробникам створювати різноманітні веб-сторінки та додатки з високою функціональністю та дизайном. HTML використовується для створення структури та контенту веб-сторінки,[6] CSS використовується для оформлення та стилізації цього контенту[7], а JavaScript дозволяє створювати динамічний та інтерактивний дизайн.

React - це JavaScript-бібліотека для створення користувацьких інтерфейсів. Вона дозволяє розробникам створювати високоякісні та ефективні веб-додатки з використанням компонентного підходу. React дає можливість розробникам швидко створювати, тестувати та підтримувати користувацький інтерфейс, що є важливим для розробки системи рекомендації книжок.

Materialize - це CSS-фреймворк, який дозволяє створювати веб-додатки з використанням готових компонентів та стилів. Він забезпечує швидку та легку розробку веб-додатків з готовим дизайном та компонентами, що сприяє швидкій та ефективній розробці інтерфейс-частини системи рекомендації книжок.

Таким чином, використання HTML[7], CSS, JavaScript та React дозволяє створити високоякісну та ефективну інтерфейс-частину системи рекомендації книжок з використанням компонентного підходу та динамічним дизайном. Використання Materialize дозволяє швидко та легко створювати веб-додатки з готовим дизайном та компонентами, що сприяє швидкій та ефектив ній розробці інтерфейс-частини системи. Materialize містить набір готових компонентів, таких як кнопки, поля вводу, таблиці, каруселі та багато інших, що дозволяє значно скоротити час розробки і зосередитись на функціональності додатку та взаємодії з користувачем.

Крім того, React є однією з найпопулярніших JavaScript-бібліотек та має велику спільноту розробників, що забезпечує швидкий розвиток та підтримку. Використання React дозволяє створювати компоненти з власним станом та реагувати на зміни в ньому, що дозволяє забезпечити швидкий та динамічний інтерфейс.

Також, React дозволяє розробникам використовувати JSX, що є розширенням синтаксису JavaScript та дозволяє використовувати HTML-подібні теги для створення компонентів. Це забезпечує зручну та просту розробку інтерфейсу та зменшує кількість коду, що необхідний для його створення.[13]

Отже, використання HTML, CSS, JavaScript та React забезпечує ефективну та швидку розробку інтерфейс-частини системи рекомендації книжок з використанням компонентного підходу та динамічного дизайну. Використання Materialize дозволяє зосередитись на функціональності додатку та швидко створити готовий дизайн з використанням готових компонентів.

3.1.2 Програмні засоби розробки серверної частини онлайн бібліотеки

Python є однією з найпопулярніших мов програмування в світі завдяки своїй простоті, ефективності та розширюваності. Використання Python дозволяє розробити серверну частину системи рекомендації книжок швидко та ефективно.[22]

FastAPI є мінімалістичним веб-фреймворком для Python, який забезпечує швидкий розрідження та високу продуктивність. FastAPI має вбудовану підтримку асинхронного програмування та автоматичну генерацію документації API з OpenAPI та JSON Schema.

SQLAlchemy є потужним та гнучким ORM (Object-Relational Mapping) для Python, який дозволяє взаємодіяти з базами даних за допомогою об'єктів Python замість роботи з SQL запитами безпосередньо. SQLAlchemy підтримує різні типи баз даних, включаючи SQLite, який є простим у використанні та може бути добрим вибором для невеликих проектів.[24]

SQLite є легким та простим у використанні СУБД, який підтримує багато операцій з базами даних та підходить для використання в невеликих проектах. SQLite має невисокі вимоги до ресурсів та не вимагає окремого сервера баз даних, тому його можна використовувати на різних платформах без особливих налаштувань.[28]

OAuth2 та JWT токени - це стандартні протоколи для аутентифікації та авторизації користувачів у веб-додатках. Використання цих протоколів дозволяє забезпечити безпеку та захист від несанкціонованого доступу до системи.

Таким чином, вибір Python, FastAPI, SQLAlchemy та SQLite для розробки серверної частини системи рекомендації книжок є логічним, оскільки ці технології дозволяють швидко та ефективно р озробляти систему рекомендації книжок. За допомогою Python можна швидко розробити потрібні функції та алгоритми для системи рекомендації книжок, які потім можуть бути легко інтегровані в FastAPI,[25] який забезпечує високу продуктивність та можливість асинхронного програмування.

SQLAlchemy надає зручні інструменти для роботи з базами даних, що дозволяє взаємодіяти з ними за допомогою об'єктів Python замість написання складних SQL запитів. SQLite відмінно підходить для невеликих проектів, оскільки має невисокі вимоги до ресурсів та не потребує окремого сервера баз даних.

За допомогою протоколів OAuth2 та JWT токенів можна забезпечити безпеку та захист від несанкціонованого доступу до системи. Це дозволяє забезпечити аутентифікацію користувачів та рольовий доступ до різних функцій системи залежно від їхніх прав.

Загалом, вибір Python, FastAPI, SQLAlchemy та SQLite для розробки серверної частини системи рекомендації книжок дозволяє розробити ефективну та безпечну систему з високою продуктивністю та гнучкістю, що може бути легко розширена у майбутньому.

3.2 Структура RESTful API

Одним з найважливіших аспектів при розробці веб-додатків є структура ендпоінтів.[16] Вона визначає, яким чином клієнти можуть взаємодіяти з різними ресурсами в системі. У даному додатку використовується структура ендпоінтів, яка є досить поширеною в сучасному програмуванні.[15]

Згідно з дослідженням, виконаним у процесі розробки веб-додатків, доцільно використовувати наступну структуру ендпоінтів:

- GET /<resource>/
- GET /<resource>/<resource_id>/
- POST /<resource>/
- DELETE /<resource>/<resource_id>/

У цих ендпоінтах, "ресурс" може мати різні значення в залежності від призначення системи. В цьому дослідженні використовуються такі ресурси, як:

- Users
- Roles
- Books
- Views

GET /<resource>/ дозволяє отримати список всіх ресурсів даного типу. GET /<resource>/<resource_id>/ дозволяє отримати конкретний ресурс за його ідентифікатором. POST /<resource>/ дозволяє створити новий ресурс, а DELETE /<resource>/<resource_id>/ дозволяє видалити конкретний ресурс за його ідентифікатором.

Ця структура ендпоінтів є досить простою та зручною у використанні. Вона забезпечує зручну взаємодію з різними типами ресурсів у системі, та дозволяє з легкістю виконувати операції з ними. Розробка веб-додатків з використанням такої структури ендпоінтів може сприяти покращенню їх якості та забезпеченню кращої користувацької зручності.

Наступний ендпоінт, є GET ендпоінтом з шляхом "/books/recommendations/{user_id}", який повертає список рекомендованих книг

для конкретного користувача. Він містить параметри запиту, такі як user_id та field, які можуть використовуватися для отримання більш точних результатів. Ендпоінт використовує залежності для отримання даних користувача та сесії бази даних.

3.3 Специфікація OpenAPI

OpenAPI специфікація є стандартом опису API, що дозволяє створювати і документувати RESTful API.[23][16] Для даного застосунку було розроблено OpenAPI специфікацію(рисунок 3.1), що дозволяє документувати ендпоінти та методи, що використовуються в системі.

Рисунок 3.1 Інтерактивний інтерфейс OpenAPI

OpenAPI специфікація містить повний опис ендпоінтів(рисунок 3.2), методів, параметрів запиту, параметрів відповіді та їх форматів. Вона також визначає правила взаємодії між клієнтом та сервером, у тому числі правила валідації параметрів запиту, визначення форматів відповіді та кодів статусу HTTP.

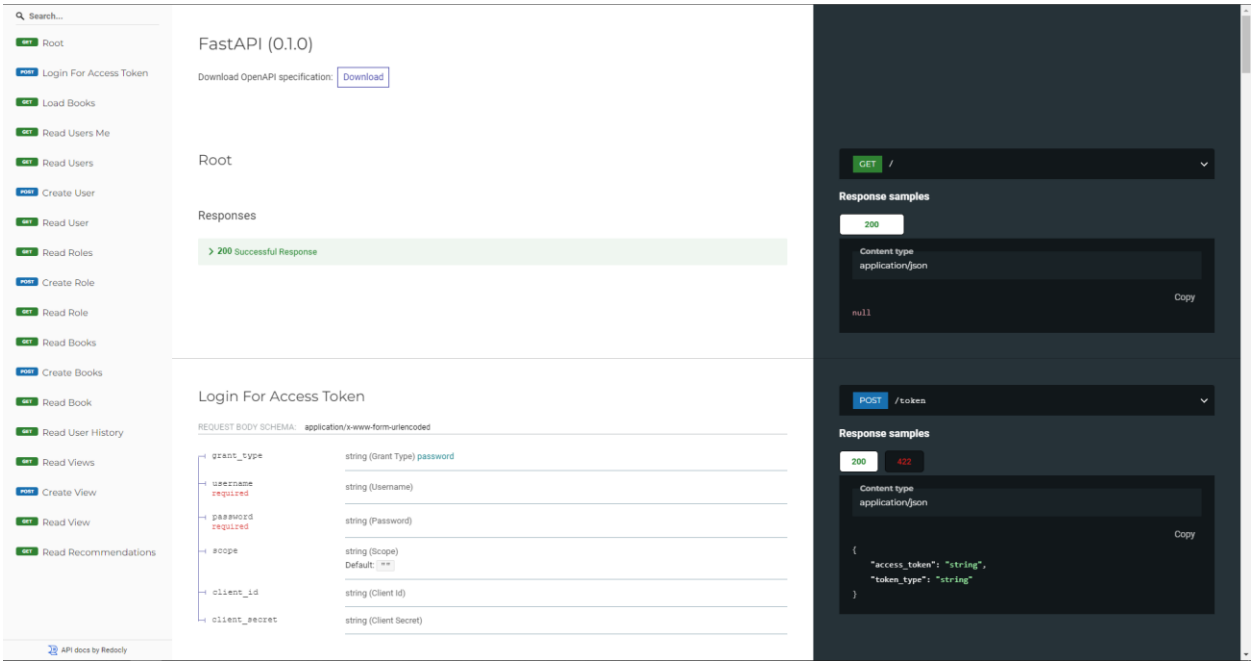

Рисунок 3.2 Документація OpenAPI

Застосування OpenAPI специфікації дозволяє забезпечити якість API, створення ефективної документації, а також допомагає збільшити швидкість розробки та забезпечує стабільність застосунку. Крім того, специфікація може використовуватись для автоматичної генерації клієнтського коду для різних мов програмування, що дозволяє зменшити час на розробку клієнтських додатків.

У даному застосунку, OpenAPI специфікація була розроблена з використанням JSON-формату та була включена до документації системи.(специфікацію подано у додатку 1) Це дозволяє розробникам та іншим користувачам з легкістю знайти необхідну інформацію про ендпоінти, методи, параметри запиту та формати відповіді. Більш того, OpenAPI специфікація може бути використана для автоматичної генерації клієнтського коду для додатків, що спрощує розробку та зменшує час на розробку.

3.4 Тестування API

Під час розробки веб-додатків, одним з найважливіших аспектів є тестування. Тестування ендпоінтів допомагає впевнитися в правильному функціонуванні системи, зменшує кількість помилок та підвищує якість

продукту. Специфікація тестів для даних ендпоінтів визначає можливі сценарії взаємодії з системою.[27]

Для ендпоінта GET /<resource>/ тестування може включати перевірку правильності повернення списку всіх ресурсів даного типу, включаючи перевірку кількості та формату повернутих даних.

Наприклад для ресурсу users були складені наступні тест кейси:

1. Перевірка, що запит на ендпоінт повертає статус код 200 та список користувачів
2. Перевірка, що користувач без прав адміністратора отримує статус код 403
3. Перевірка, що можливо пропустити певну кількість записів та отримати обмежену кількість записів
4. Перевірка, що при пропуску записів, список повертає записи після пропущених

Для ендпоінта GET /<resource>/<resource_id>/ тестування може включати перевірку правильності повернення конкретного ресурсу за його ідентифікатором, включаючи перевірку відповідності повернутих даних запиту.

Наприклад для ресурсу users були складені наступні тест кейси:

1. Тест на успішне отримання користувача за допомогою дійсного ідентифікатора користувача:
   - Вхідні дані: дійсний ідентифікатор користувача
   - Очікуваний результат: статус код 200 та відповідь, яка містить дані користувача, включаючи його **id**, **email**, **role_id** та **role**
2. Тест на отримання помилки 404, якщо користувач не знайдений в базі даних:
   - Вхідні дані: недійсний ідентифікатор користувача
   - Очікуваний результат: статус код 404 та повідомлення про помилку "User not found"
3. Тест на отримання помилки 401, якщо користувач не має адміністративної ролі:

- Вхідні дані: дійсний ідентифікатор користувача, користувач без адміністративної ролі
- Очікуваний результат: статус код 401 та повідомлення про помилку "Not authenticated as admin"

4. Тест на отримання даних лише для адміністраторів:

- Вхідні дані: дійсний ідентифікатор користувача, користувач з адміністративною роллю
- Очікуваний результат: статус код 200 та відповідь, яка містить дані користувача, включаючи його **id**, **email**, **role_id** та **role**

5. Тест на правильність форматування відповіді:

- Вхідні дані: дійсний ідентифікатор користувача
- Очікуваний результат: статус код 200 та відповідь, яка містить дані користувача у форматі, визначеному моделлю відповіді **schemas.User**

Для ендпоінта POST /<resource>/ тестування може включати перевірку правильності створення нового ресурсу, включаючи перевірку відповідності створених даних запиту та коректності збереження нового ресурсу в базі даних.

Наприклад для ресурсу users були складені наступні тест кейси:

1. Тест на успішне створення користувача

- Опис: Виконати POST запит на **/users/** з коректними даними користувача
- Очікуваний результат: Статус код 200 та повернення об'єкта користувача з бази даних з відповідними полями

2. Тест на спробу створити користувача з існуючою електронною поштою

- Опис: Виконати POST запит на **/users/** з користувачем, що має електронну пошту, яка вже зареєстрована в базі даних
- Очікуваний результат: Статус код 400 та повідомлення "Email already registered"

3. Тест на відправлення некоректного запиту (відсутність обов'язкового поля)

- Опис: Виконати POST запит на **/users/** з користувачем, що не містить обов'язкове поле **email**
- Очікуваний результат: Статус код 422 та повідомлення про невідповідність обов'язкового поля.

Для ендпоінта DELETE /<resource>/<resource_id>/ тестування може включати перевірку правильності видалення конкретного ресурсу за його ідентифікатором, включаючи перевірку відсутності ресурсу у системі після виконання запиту.

Наприклад для ресурсу users були складені наступні тест кейси:

1. Тест на успішне видалення користувача

- Опис: Виконати DELETE запит на **/users/{user_id}**, де **{user_id}** - існуючий ідентифікатор користувача
- Очікуваний результат: Статус код 204 та відсутність тіла відповіді

2. Тест на спробу видалити неіснуючого користувача

- Опис: Виконати DELETE запит на **/users/{user_id}**, де **{user_id}** - невірний ідентифікатор користувача
- Очікуваний результат: Статус код 404 та повідомлення "User not found"

3. Тест на спробу видалити користувача без необхідних привілеїв

- Опис: Виконати DELETE запит на **/users/{user_id}**, де **{user_id}** - існуючий ідентифікатор користувача, але запит виконується користувачем без необхідних привілеїв
- Очікуваний результат: Статус код 403 та повідомлення про недостатній рівень привілеїв.

Для інших ресурсів були розроблені аналогічні тести.

Тести для ендпоінту **GET /books/recommendations/{user_id}**:

1. Тест на успішне отримання рекомендаційних книг для користувача

- Опис: Виконати GET запит на **/books/recommendations/{user_id}**, де **{user_id}** - існуючий ідентифікатор користувача
- Очікуваний результат: Статус код 200 та список книг у форматі **schemas.Book**

2. Тест на спробу отримати рекомендації для неіснуючого користувача

- Опис: Виконати GET запит на **/books/recommendations/{user_id}**, де **{user_id}** - невірний ідентифікатор користувача
- Очікуваний результат: Статус код 404 та повідомлення "User not found"

3. Тест на отримання рекомендаційних книг з певним полем

- Опис: Виконати GET запит на **/books/recommendations/{user_id}?field={field}**, де **{user_id}** - існуючий ідентифікатор користувача, **{field}** - існуюче поле у форматі str
- Очікуваний результат: Статус код 200 та список книг у форматі **schemas.Book**, які мають вказане поле зі значенням, яке рекомендується для даного користувача

4. Тест на отримання рекомендаційних книг без вказання поля

- Опис: Виконати GET запит на **/books/recommendations/{user_id}**, де **{user_id}** - існуючий ідентифікатор користувача, **{field}** - не вказане
- Очікуваний результат: Статус код 200 та список книг у форматі **schemas.Book**, що є рекомендаційними для даного користувача без урахування поля.

В залежності від призначення системи та її функціональності, можуть бути визначені й інші можливі тестові сценарії для кожного ендпоінту. Розробка тестових випадків та їх виконання є важливим етапом в розробці веб-додатків та допомагає забезпечити стабільну та якісну роботу системи.

3.5 Інструкція користувача

Перше що бачить перед собою користувач коли заходить на сайт – це сторінка реєстрації(рисунок 3.3):

1. Відкрийте домашню сторінку сайту електронної бібліотеки.
2. Заповніть всі обов'язкові поля форми реєстрації, такі як електронна пошта, пароль.
3. Після того, як ви заповните всі поля, натисніть кнопку "Зареєструватися" або "Створити аккаунт".

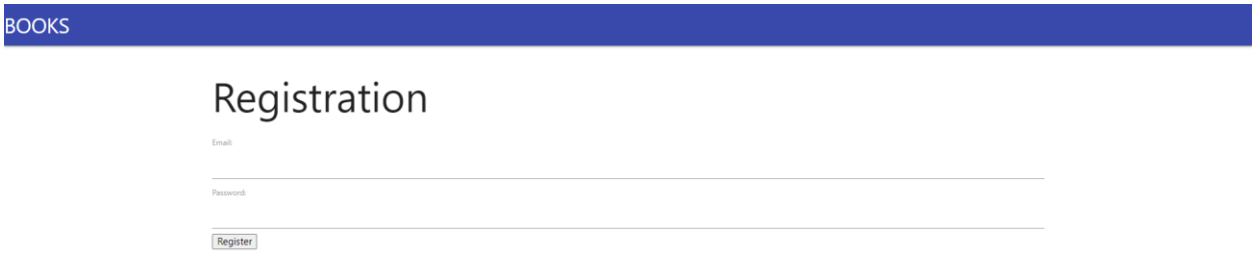

Рисунок 3.3 Сторінка регестрації

Далі користувач може зайти на сайт(рисунок 3.4):

1. Відкрийте домашню сторінку сайту електронної бібліотеки.
2. Натисніть на кнопку "Увійти" або "Вхід".
3. Введіть свою електронну пошту та пароль, які ви використовували при реєстрації.
4. Натисніть кнопку "Увійти" або "Вхід".
5. Якщо ви ввели правильну інформацію, ви будете перенаправлені на свій обліковий запис на сайті електронної бібліотеки.

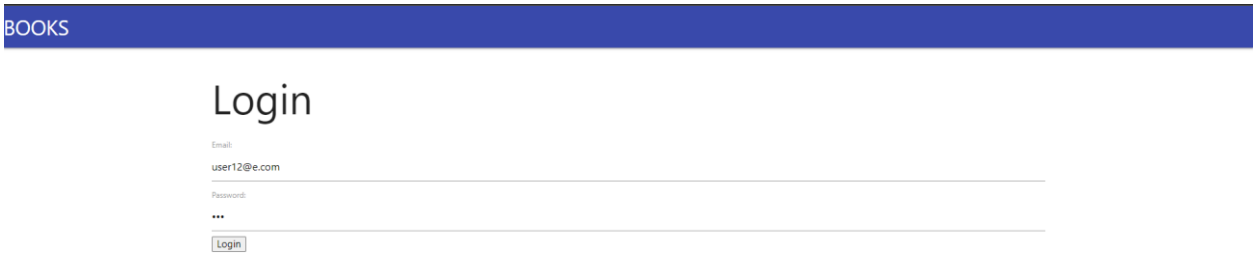

Рисунок 3.4 Сторінка логіну

Після того, як користувач успішно зайшов на сайт, він побачить головну сторінку з наступною структурою(рисунок 3.5).

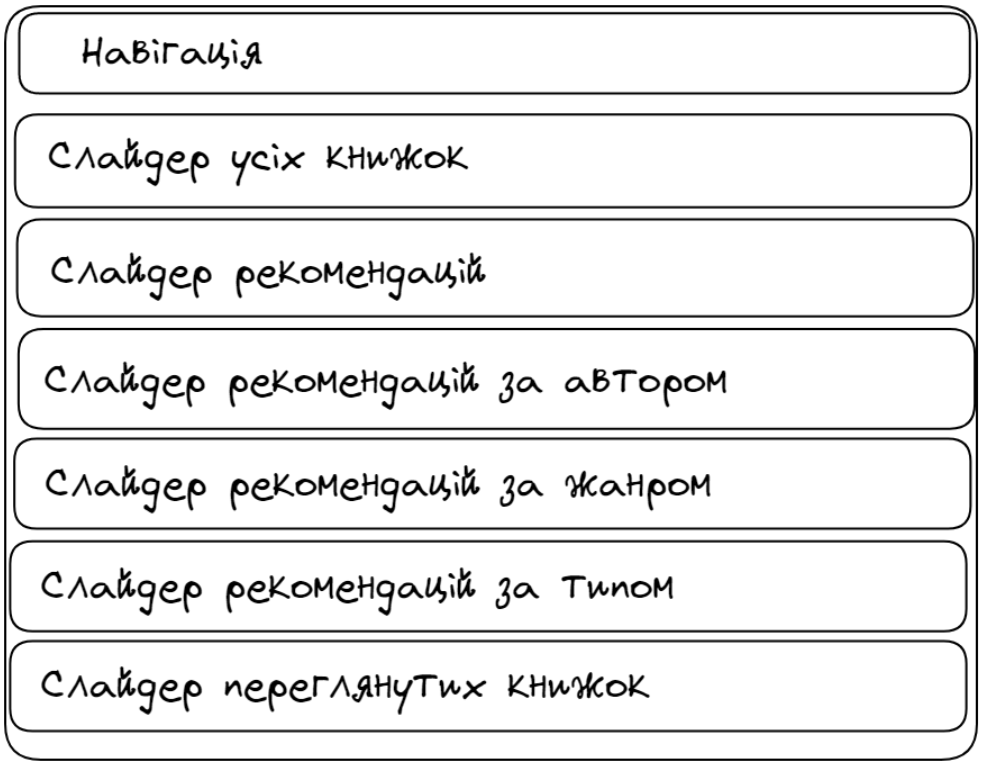

Рисунок 3.5 Схема головної сторінки

Початок користування веб-додатку починається з перегляду головної сторінки зі слайдером всіх можливих книг на сайті.(рисунок 3.6)

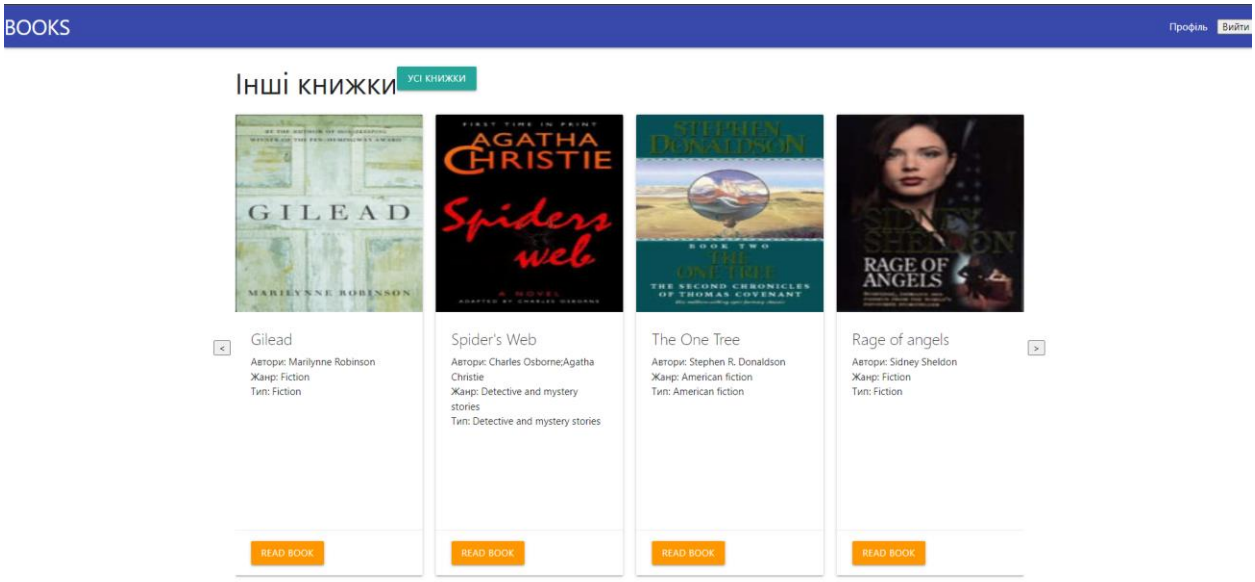

Рисунок 3.6 Сторінка початкового списоку

Користувачу відкривається можливість переглянути через слайдер всі можливі книжки, або перейти на сторінку повний списоку книг(рисунок 3.8), яка має наступну схему(рисунок 3.7).

Рисунок 3.7 Схема сторінки пошуку

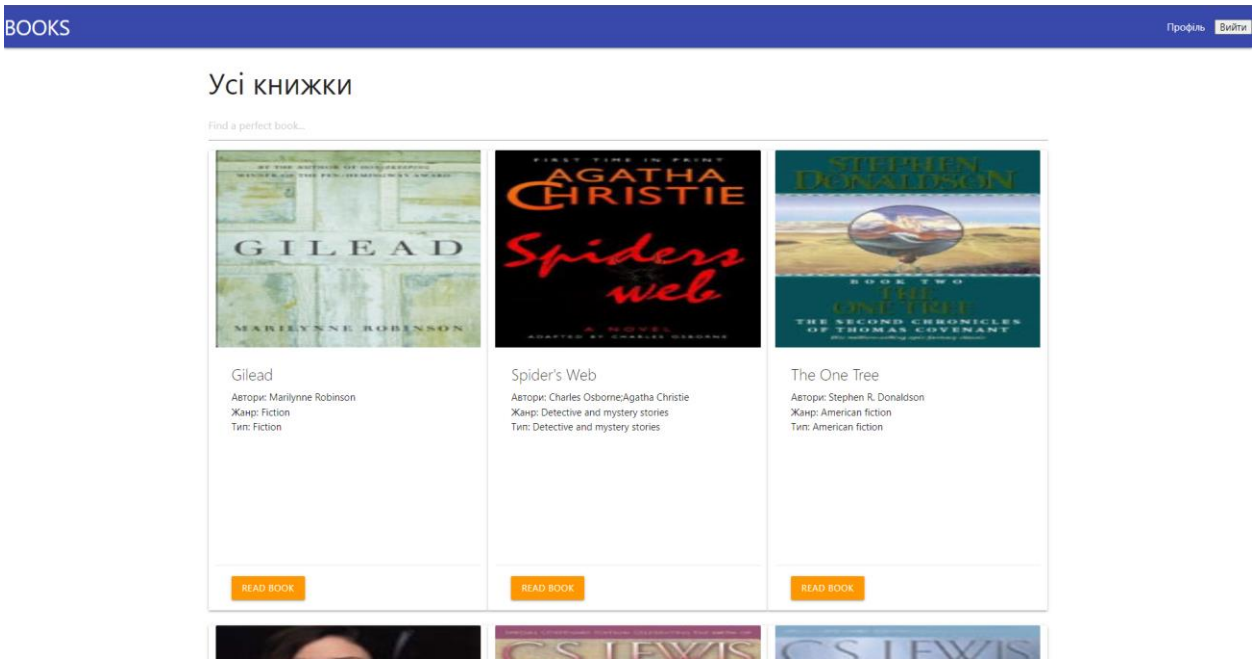

Рисунок 3.8 Сторінка повного списоку книг та пошуку

Якщо користувач знайде цікаву книжку, він зможе переглянути розгорнуту повну інформацію про книгу. (рисунок 3.9)

Рисунок 3.9 Сторінка повної інформації про книгу

Після перегляду книжки яка зацікавила користувача. Система заповнює інформацію про історію перегляду. (рисунок 3.10)

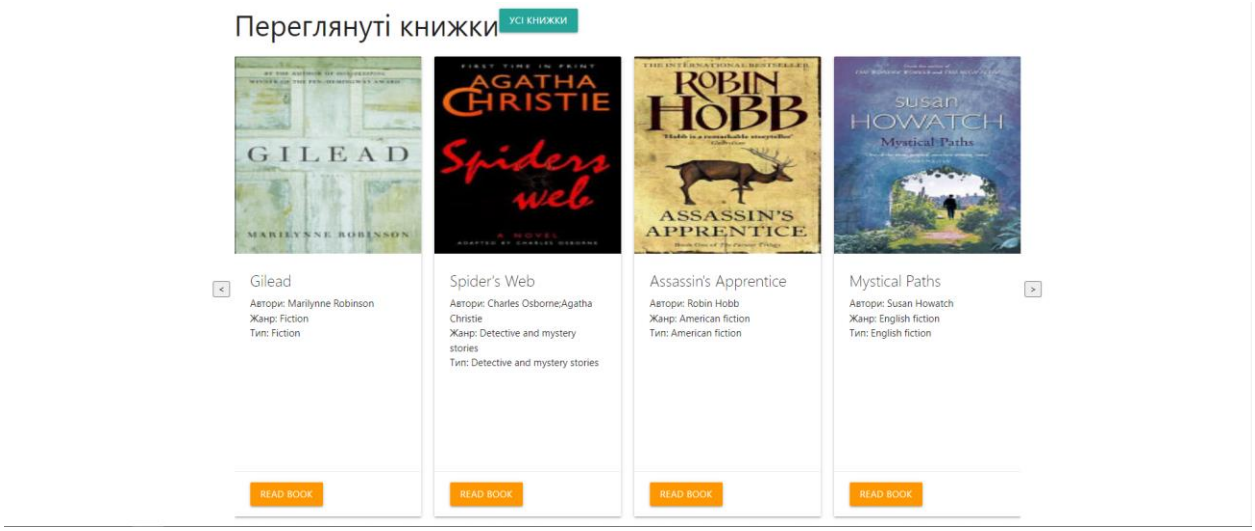

Рисунок 3.10 Сторінка історії переглядів

Наступним етапом є реалізація рекомендацій для користувача за автором та жанром який цікавив користувача в минулому, та можливо цікавлять наступні книжки. (рисунок 3.11)

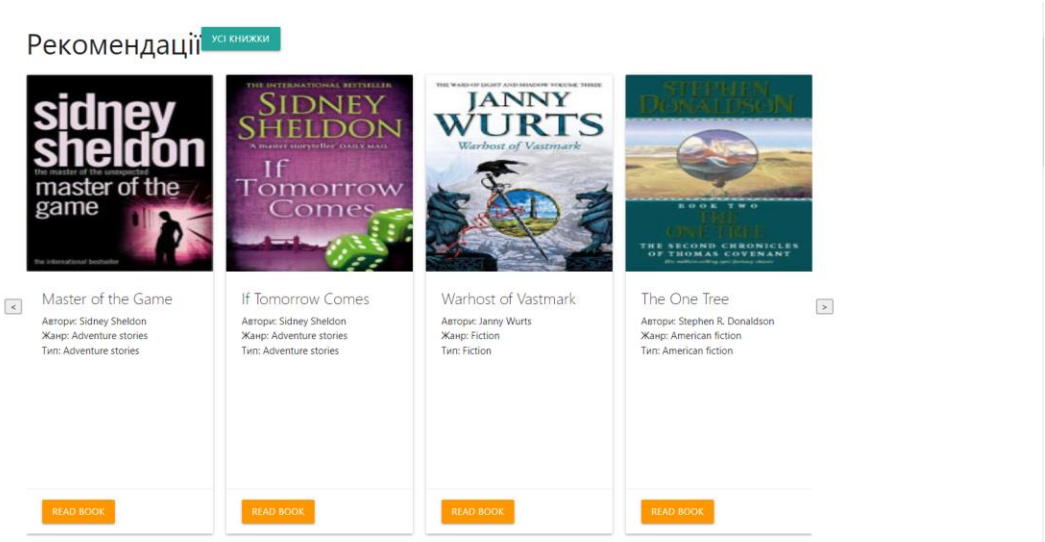

Рисунок 3.11 Елемент рекомендацій

Також користувач може переглянути окремі рекомендації за автором(рисунок 3.12), що знаходяться у відповідному блоці рекомендацій

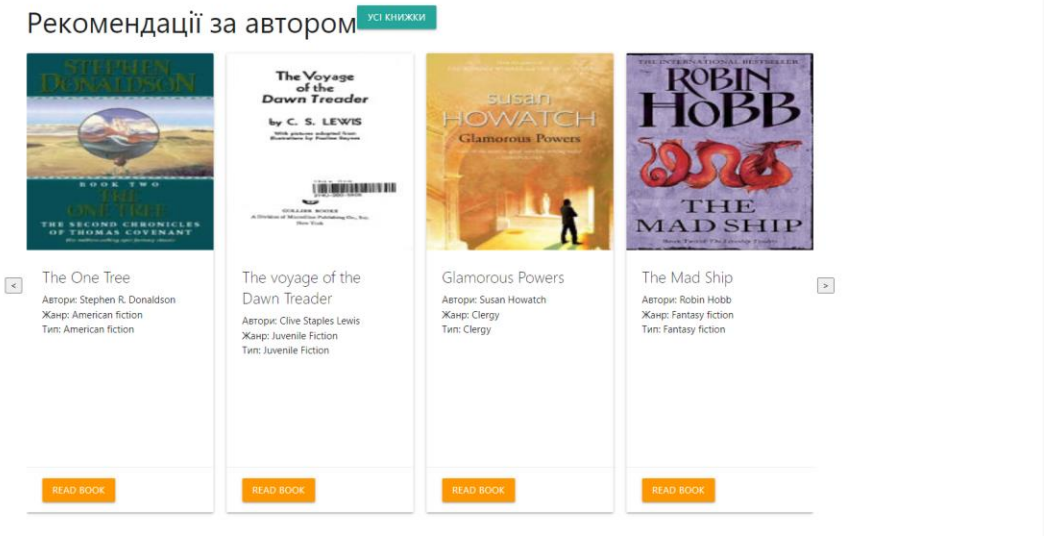

Рисунок 3.12 Елемент рекомендації за автором

Аналогічними чином користувачу доступні рекомендації за жанром(рисунок 3.13)

Рисунок 3.13 Елемент рекомендації за жанром

Функціоналом застосунку передбачено видачу рекомендації за типом творів(рисунок 3.14)

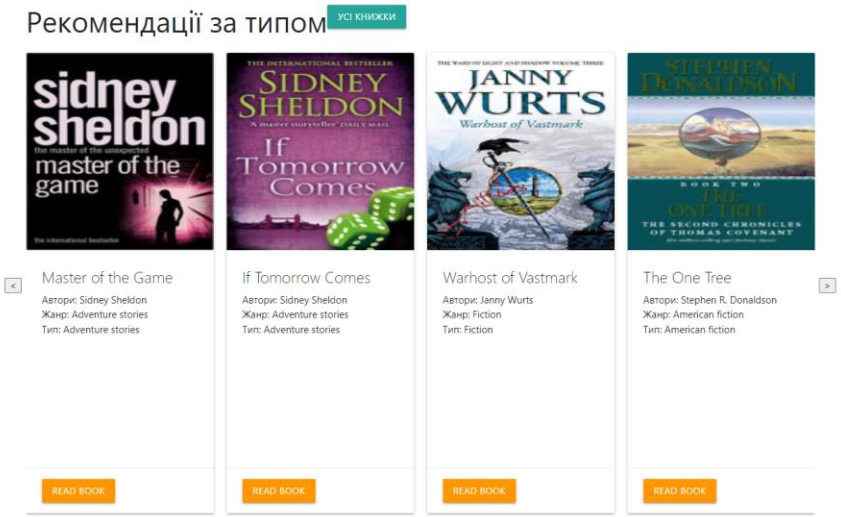

Рисунок 3.14 Елемент рекомендації за типом

3.3 Висновки до третього розділу

У розділі була проведена практична реалізація онлайн бібліотеки з використанням різних програмних засобів, зокрема засобів розробки інтерфейсу та серверної частини. Була описана структура RESTful API та здійснене тестування цього API. Також в розділі була надана інструкція користувача, яка допоможе користувачам зрозуміти, як використовувати розроблену онлайн бібліотеку.

В результаті проведеної роботи було доведено ефективність використання програмних засобів розробки інтерфейсу та серверної частини для створення функціональної та зручної в користуванні онлайн бібліотеки. Також було продемонстровано важливість правильного побудови RESTful API та проведення його тестування для забезпечення якості та надійності роботи системи.

Отримана інструкція користувача дозволить користувачам легко та швидко знайти необхідну інформацію та виконати потрібні дії у системі. В цілому, практична реалізація проекту підтвердила доцільність використання розглянутих програмних засобів для розробки онлайн бібліотеки та забезпечення її функціональності та надійності.

Висновки

У даній дипломній роботі було проведено аналіз розподілених інформаційних систем для зберігання різнорідних електронних документів та проблеми інформаційного перевантаження користувачів послуг Інтернет-бібліотеки. Було проведено дослідження онлайн бібліотек, їхнього розгортання та розроблено метод вилучення знань, сформовано функціональні та нефункціональні вимоги до застосунку у результаті досліджень.

У другому розділі було розроблено архітектуру інформаційної системи "Онлайн бібліотека", включаючи функціональний аналіз предметної області, процес видачі рекомендацій, архітектуру системи, моделювання у вигляді контейнерів, веб-сервер та моделі бази даних. Було також проведено аналіз типів колаборативної фільтрації.

У третьому розділі була описана практична реалізація проекту, включаючи вибір програмних засобів, структуру RESTful API, OpenAPI, тестування API та інструкцію користувача.

Загальний висновок полягає в тому, що було розроблено рекомендаційну систему для онлайн бібліотеки, яка дозволяє користувачам знайти потрібну літературу з мінімальними зусиллями. Реалізація проекту дозволяє зменшити інформаційне перевантаження та підвищити ефективність використання бібліотеки.

Використана література

1. Fleder D., Hosanagar K. Blockbuster Culture's Next Rise or Fall: The Impact of Recommender Systems on Sales Diversity (англ.) // Management Science, Vol. 55, No. 5, May 2009, pp. 697-712

2. C. C. Aggarwal, Recommender Systems: The Textbook. Springer.- 2016 – 837 p.

3. Xiaoyuan Su, Taghi M. Khoshgoftaar A Survey of Collaborative Filtering Techniques. SuXiaoyuan, M. Khoshgoftaar Taghi [Електронний ресурс] // Режим доступу: http://www.hindawi.com/journals/aai/2009/421425/

4. Melville P.,Mooney R., Nagarajan R. Content-Boosted Collaborative Filtering for Improved Recommendations // University of Texas, USA. — 2002. — P. 187-192.

5. Z. Huang, D. Zeng, and H. Chen. A comparison of collaborative-_ltering recommendation algorithms for e-commerce. IEEE Intelligent Systems, - 2007. P. 68-78,

6. What Is HTML5? by Brett McLaughlin. Released July 2011. Publisher(s): O'Reilly Media, Inc. ISBN: 9781449308148.

7. Connolly D., Masinter L. The 'text/html' Media Type — IETF, 2000. — 8 p.

8. ISSN 1999-9941, “ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА КОМП’ЮТЕРНА ІНЖЕНЕРІЯ”, 2021, № 3– Режим доступу до ресурсу: https://itce.vntu.edu.ua/index.php/itce/article/download/831/547 – Назва з екрану

9. Oliver Theobald. Machine Learning: Make Your Own Recommender System (MachineLearningFromScratchBook3). [Електронний ресурс] // Режим доступу: https://www.amazon.com/Machine-Learning-Recommender-System-Beginners-ebook/dp/B07DWS346Y

10. P. Lops, M. Gemmis, G. Semeraro Recommender Systems Handbook Springer-Verlag, 2011. [Електронний ресурс] // Режим доступу: https://www.cse.iitk.ac.in/users/nsrivast/HCC/Recommender_systems_handbook.pdf

11. D. Bugaychenko, A. Dzuba, «Musical recommendations and personalization in a social network», RecSys '13: Proceedings of the 7th ACM conference on Recommender systems, October 2013, p. 367– 370.

12. Melville P., Sindhwani V. Recommender systems; Encyclopedia of Machine Learning, 2010 [Електронний ресурс] // Режим доступу: https://vikas.sindhwani.org/recommender.pdf

13. React Cookbook: Recipes for Mastering the React Framework. 1st Ed. David Griffiths, Dawn Griffiths (english) 9781492085843 [Електронний ресурс] // Режим доступу: https://bit.ly/45FFYkk

14. Файлер М., Скотт К. «UML. Основы». –М.: Вильяме, – 2002. – 192 с.

15. Філдінг Р. Architectural Styles and the Design of Network-based Software Architectures. / Р. Філдінг. – К.: «Саміт», 2018. – 520 с.

16. «What is Rest?». 2018– [Електронний ресурс] // Режим доступу: https://restfulapi.net/

17. Моуэт Э.Использование Docker / пер. с англ. А. В. Снастина; науч. ред. А. А. Маркелов. –М.: ДМК Пресс, 2017. –354 с.: ил.

18. About images, containers, and storage drivers – [Електронний ресурс] // Режим доступу: https://bit.ly/3qj1HhZ

19. Экосистема Docker: сетевое взаимодействие – [Електронний ресурс] // Режим доступу: https://bit.ly/42dvXb5

20. Силич М. П. Моделирование бизнеса — IDEF, UML, ARIS [Електронний ресурс] / М. П. Силич // Business Anaiysis. – 2020. – Режим доступу: https://analytics.infozone.pro/business-modeling-idef-uml-aris/

21. Марголін О. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс] / О. Марголін // Evergreen. – 2021. – Режим доступу: https://evergreens.com.ua/ua/articles/umldiagrams.html

22. Мова програмуваня Python [Електронний ресурс] // Режим доступу: https://www.python.org/

23. Специфікація OpenAPI [Електронний ресурс] // Режим доступу: https://spec.openapis.org/oas/latest.html

24. Object Relational Mapper SQLAlchemy [Електронний ресурс] // Режим доступу: https://docs.sqlalchemy.org/en/20/

25. FastAPI as web framework for building RESTful APIs in Python [Електронний ресурс] // Режим доступу: https://fastapi.tiangolo.com/lo/

26. Stefan nadschlager, Analysis of GoF Design Patterns used in Knowledge Processing Systems, Institute for Application Oriented Knowledge Processing. [Текст] / Stefan nadschlager, Josef kung. - Linz: Johannes Kepler University, 2017 - 22 с.

27. IEEE Standard for Software and System Test Documentation // IEEE Std 829- 2008. — P. 150.

28. Thomas M. Connolly, Carolyn E. Begg. Database Systems: A Practical Approach to Design, Implementation and Management (5th Edition). — Pearson, 2009. — ISBN: 0321523067.

Додатки

Додаток 1

Специфікація OpenAPI


```
{
    "openapi": "3.0.2",
    "info": {
        "title": "FastAPI",
        "version": "0.1.0"
    },
    "paths": {
        "/": {
            "get": {
                "summary": "Root",
                "operationId": "root__get",
                "responses": {
                    "200": {
                        "description": "Successful Response",
                        "content": {
                            "application/json": {
                                "schema": {}
                            }
                        }
                    }
                }
            }
        },
        "/token": {
            "post": {
                "summary": "Login For Access Token",
                "operationId": "login_for_access_token_token_post",
                "requestBody": {
                    "content": {
                        "application/x-www-form-urlencoded": {
                            "schema": {
                                "$ref": "#/components/schemas/Body_login_for_access_token_token_post"
                            }
                        }
                    },
                    "required": true
                },
                "responses": {
                    "200": {
                        "description": "Successful Response",
                        "content": {
                            "application/json": {
                                "schema": {
                                    "$ref": "#/components/schemas/Token"
                                }
                            }
                        }
                    },
                    "422": {
                        "description": "Validation Error",
                        "content": {
```