НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК 004.8:028

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету інформаційних технологій	Завідувач кафедри комп'ютерних наук
Глазунова О.Г., д.п.н., професор	Голуб Б.Л., к.т.н., доцент
_____	_____
_____ 2022 р.	**_____ 2022 р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інтелектуальна система формування пропозицій читачеві на основі власних вподобань»

Спеціальність: 122 «Комп'ютерні науки»

Освітня програма: Інформаційні управляючі системи та технології

Орієнтація освітньої програми: освітньо-професійна

Гарант освітньої програми

к.е.н., ст.викл. _____ Густера О. М.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

к.т.н., доцент _____ Голуб Б. Л
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Бабін Є. С.
(підпис) (ПІБ студента)

КИЇВ-2022

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) Інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри_____

_____

(науковий ступінь, вчене звання) (підпис) (ПІБ)

"1" Листопада 2021року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Бабіну Євгенію Сергійовичу

(прізвище, ім'я, по батькові)

Спеціальність 122 «Комп'ютерні науки»

(код і назва)

Освітня програма Інформаційні управляючі системи та технології

(назва)

Орієнтація освітньої програми освітньо-професійна

(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи Інтелектуальна система формування пропозицій читачеві на основі власних вподобань

затверджена наказом ректора НУБіП України від "1" Листопада 2021 р. № 1862

Термін подання завершеної роботи на кафедру

(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи: опис методів дистрибуції електронних книжок, опис архітектури з використанням мікро-сервісів, дослідження методів розробки рекомендаційних систем, аналіз досягнутих результатів впровадження рекомендаційних систем в систему дистрибуції електронних книжок

Перелік питань, що підлягають дослідженню:

1. Системний аналіз предметної області
2. Моделювання, проектування та розробка системи
3. Впровадження системи, аналіз отриманих результатів

Перелік графічного матеріалу (за потреби): Постер

Дата видачі завдання "1" Листопада 2021 р.

Керівник магістерської кваліфікаційної роботи _____ к.т.н., доцент, Голуб Б. Л.

(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Бабін Є. С.

(підпис) (прізвище та ініціали студента)

ЗМІСТ

ВСТУП

Актуальність. Однією з основних ланок, на якій було побудовано людський соціум, був процес обміну результатами праці людей в цьому суспільству. Важливим поняттям, що належить до цієї системи є “дистрибуція”, яке охоплює всю множину інструментів, процесів та тощо, що націлені на надання товарів чи послуг їх споживачу.

З появою та розвитком цифрових технологій з’являється новий підхід до організації процесів дистрибуції, а саме цифрова дистрибуція.

Для цифрової дистрибуції характерно:

- товаром виступають виключно цифрові ресурси, тобто, ті, які представлені в цифровій формі;
- процеси дистрибуцію реалізуються за допомогою цифрових технологій;
- збільшення рівня автоматизації процесів;
- передача товарів здійснюється без використання фізичних носіїв інформації.

За останні роки цифрова дистрибуція почала набувати дедалі більшої популярності, що зв’язано з тим, що у порівняні з звичайною дистрибуцією вона дозволяє зменшить вартість та підвищити ефективність процесів дистрибуції. Для споживача це дозволяє зменшити ціну цифрового товару шляхом зменшення його собівартості, збільшити їх доступність, зручність отримання [1].

Окрім оптимізації процесів дистрибуції, використання цифрових технологій та автоматизація процесів дистрибуції дозволяє інтегрувати процес накопичення Big Data, аналізу даних, рішень, які дозволяють допомогти в процесу прийняттю рішень та тощо, що надає додаткові переваги бізнесу та кінцевому користувачу.

Так, система рекомендацій, яка може бути створена на основі цих даних, може дозволити зрозуміти в яких типах товарів зацікавлені користувачі, а користувачам дозволить простіше знаходить товари, які можуть їх зацікавити.

Об'єкт дослідження: Цифрова платформа дистрибуції електронних книжок.

Предмет дослідження: Система рекомендацій на основі вподобань користувача.

Мета дослідження. Метою роботи є підвищити ефективність системи цифрової дистрибуції електронних книжок шляхом інтеграції в неї системи рекомендацій та інших інструментів накопичення та аналізу даних.

Задачі. Для досягнення поставленої мети необхідно виконати наступні кроки:

1. Проведення системного аналізу;
2. Формування переліку вимог;
3. Проведення моделювання;
4. Розробка системи;
5. Аналіз результатів та написання висновку.

Методи. В ході роботи були застосовані наступні методи:

- NET.6 - C#, в якості основної мови програмування для розробки системи;
- ASP.NET CORE для забезпечення веб-складової системи;
- OLTP СУБД для забезпечення операційної інформаційної складової;
- OLAP для накопичення постійних даних, проведення аналізу даних та підтвердження гіпотез, розрахунку ключових показників ефективності;
- Data Mining - Формування нових гіпотез;

- DSE Graph - Для ефективного збереження та взаємодії з даними, для яких важливою характеристикою слугує зв'язок між цими даними.

Новизна. Запропоновано сукупність інформаційних технологій та удосконалення архітектури, що дозволяє підвищити ефективність системи.

Робота складається з наступних розділів:

1. **Аналіз предметної області**, в рамках якого було проведено аналіз системи цифрової дистрибуції електронних книг, як системи електронної комерції. В рамках цього розділу було розглянуто особливості цих систем, розглянуто існуючі рішення й виявлені їх основних недоліків, постановка цілей та вимог до розробленої системи;
2. **Моделювання системи,** в рамках якого було проведено моделювання системи відповідно до аналізу предметної області. Отримані результати були покладені в основі при проектуванні та розробці системи;
3. **Розробка системи,** в рамках якого було описано процеси по проектуванню та розробці системи, продемонстровано архітектуру системи, описано технології, які були використані;
4. **Результати дослідження**, в рамках якого було описано кінцеві параметри системи, включаючи всі її вузли, а також представлено та проаналізовано результати дослідження.

До складу роботи входить 73 сторінок. В роботі використано 24 джерел, 46 рисунків та 11 таблиць. Ця робота була представлена на конференції «XIII Міжнародна науково-практична Інтернет-конференція студентів, аспірантів та молодих вчених "Інформаційні технології: економіка, техніка, освіта"» - «Інтелектуальна система формування пропозицій читачеві на основі власних вподобань».

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Розгляд особливостей цифрової дистрибуції

Цифрова дистрибуція – це тип дистрибуції, яка характеризується тим, що засобами та шляхами розповсюдження контенту виступають цифрові технології без використання фізичних посередників. Відповідно, предметом цієї дистрибуції є виключно цифрові товари та послуги.

Реалізація цифрової дистрибуції вимагає створення системи цифрової дистрибуції, тобто інформаційної системи, яка здатна перевести в цифровий формат основний процес дистрибуції – розповсюдження товарів. Відповідно, для системи цифрової дистрибуції електронних книжок, яка розглядається в цій роботі, таким товаром є електроні книжки.

Хоча, як говорилось раніше, основною функцією системи цифрової дистрибуції є процес реалізація процесу цифрового розповсюдження товарів, тобто можливості отримати користувачем цифрового товару, але досить рідко коли системи обмежується лише нею.

Зазвичай такі системи створюються заради комерційної діяльності [2], що піднімає питання отримання прибутку з цієї системи. Для цифрової дистрибуції поширені такі методи монетизації:

- **збір даних та продаж реклами**, в рамках цієї моделі доступ до цифрового контенту для користувача повністю безкоштовний, але він надає згоду на збір його персональних даних, аналіз та продаж цих даних, перегляд реклами та тощо;
- **придбання**, найбільш звичний для більшості користувачів метод, за яким користувач має придбати повноцінний доступ до книжки без обмежень по часу;

- **оформлення підписки на певний часовий період**, яка дає змогу отримати доступ до контенту. Умови можуть досить сильно відрізняться, так, зазвичай мова йде доступ до всього доступного контенту, але можливі ситуації коли мова йде лише про частину, або навіть оренду конкретного товару. На цей момент цей спосіб менш регламентований законами, що дозволяє укладати більш гнучкі домовленості з користувачами.

Не є рідким явищем коли система цифрової дистрибуції може використовувати декілька з зазначених способів, щоб захопити більшу кількість користувачів. Також досить часто новим користувачам надається пробний період, під час якого користувач може безкоштовно ознайомиться з тим, що пропонує ця система.

Також в системах цифрової дистрибуції зустрічається реалізацію таких задач:

- соціальна – надання можливостей комунікації користувачам, збір думок користувачів, їх інформування про якісь події, зворотній зв’язок та тощо;
- збір даних – реалізація інструментів по збору даних користувача та його взаємодії з системою;
- ідентифікація користувача – авторизацію та аутентифікація користувача [3];
- адміністрування – надання інструментів для адміністрування та ведення обліку;
- рекомендаційна – формування рекомендацій на основі накопичених даних, для допомоги в прийняттю рішення Може буть націлена, як на допомогу користувачу в пошуках актуальних для нього товарів, так і в адмініструванню, в залежності від задач, які були покладені на систему рекомендацій.

1.2 Аналіз існуючих рішень

На цей момент існують надзвичайно розвинуті системи цифрової дистрибуції, такі, як YouTube, Netflix, Spotify [4], Google Store, Apple Store та тощо. Ці системи є представниками, що демонструють найбільш актуальні підходи, але описані системи займаються поширенням іншого контенту, а саме програмних додатків, фото, відео та аудіо контенту.

Якщо говорить про існуючі системи, то варто виділити таких представників:

- Amazon – ця торгівельна платформа містить також функціонал по цифровій дистрибуції електроних книжок, який інтегрований з власними пристроями для перегляду таких книжок. Так, як це лише частина системи, то даний функціонал обмежений, з інших особливостей варто відмітить потужну систему рекомендацій;
- Ebooks.com – реалізація основних задач цифрової дистрибуції, без якихось додаткових особливостей, має застарілий дизайн сайту.

Якщо говорити про рекомендаційні системи, то з наявної інформації можна виділити наступні підходи [5] до реалізації цієї задачі:

- рекомендації на основі абсолютних значень, таких, як середній рейтинг, кількість переглядів, кількість відгуків, кількість придбань та тощо;
- суміжні фільтри - рекомендації, на основі аналізу взаємодії користувачів з предметами, намагаючись передбачити результат нової взаємодії;
- фільтри на основі вмісту - аналіз всієї доступних даних відносно предметів та користувачів з метою знаходження зав’язків між цими даними, що дозволили б створити модель користувача, яка могла б описати його вподобання та дії.

Також, можна зробить висновок, що в багатьох алгоритмах використовується коефіцієнт схожості, який вказує наскільки елементи є схожі між собою. Особливо важливу роль, це займає для суміжних фільтрів.

Часто для розрахунку цього коефіцієнту використовують кореляцію Пірсона, косинус подібності або евклідову дистанцію, хоча вони містять проблеми, які зменшують точність рекомендацій [6].

1.3 Постановка задачі

Розробити платформу цифрової дистрибуції, яка буде здатна виконувати такі бізнес-задачі:

- надання читачу доступу до асортименту системи;
- здійснення електронної оплати;
- система електронного кабінету, яка дозволить ідентифікувати користувача, створювати нові акаунти, можливість відновлення доступу та обмеження доступу на основі поточного профілю;
- інструменти для бібліографа, які б дозволили б йому керувати асортиментом;

Розробити систему рекомендацій для цієї платформи цифрової дистрибуції електронних книжок на основі даних отриманих від діяльності користувачів на цій платформі.

Мають буть створені інструменти рекомендацій, як для користувача цієї платформи, так і працівників, які займаються її адмініструванням.

Проаналізувати процес розробки та результати цієї системи. Дослідити різні підходи до створення цієї системи.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Аналіз даних

Відповідно до проведеного дослідження предметної області було проведено аналіз даних (табл. 2.1), що задіяні в бізнес процесах, або які є наслідком виконання цих бізнес процесів.

Також була проведена категоризація цих даних на:

- постійні – мають зберігатися та завжди бути доступними;
- тимчасові – задіяні лише в ході певного процесу, після його закінчення можуть бути видалені.

Таблиця 2.1

Аналіз даних

Дані	Тип	Опис
Вхідні дані		
Дані про книжковий асортимент	Постійні	Містить дані про весь доступний асортимент платформи
Дані про користувачів платформи	Постійна	Містить повноту доступних даних про користувача, включаючи, як ідентифікаційна, так і персональні
Платіжні дані	Тимчасові	Дані від користувача, які необхідні для здійснення електронної комерції

Закінчення таблиці 2.1

Дані	Тип	Опис
Дані про здійснення грошових транзакцій	Тимчасові	Дані, про статус здійснення платіжних транзакцій
Дані про вподобання користувачів	Постійні	Дані, які дозволяють отримати інформацію про книжкові вподобання користувача
Пошукові дані	Тимчасові	Дані, які дозволяються здійснювати пошук книжок за їх властивостями
Вихідні дані		
Дані про замовлення	Постійні	Дані, які описують придбання книжок користувачем
Дані про авторизацію	Тимчасові	Дані, які містять інформацію на рахунок дозволів користувача в системі
Звітні дані	Тимчасові	Дані, які містять звітну та статистичну інформацію на рахунок діяльності платформи
Рекомендаційні дані	Тимчасові	Дані, що містять рекомендації на основі зібраних та проаналізованих даних

2.2 Моделювання предметної області

На цьому етапі роботи було здійснення моделювання предметної області на основі проаналізованих даних, так, як це дозволяє получить більш глибоке представлення про вигляд кінцевої системи, а результати моделювання можуть бути використані на етапі проектування системи та її розробки.

Для проведення моделювання, в якості основного інструменту було вибрано графічну мову UML (Unified Modeling Language). UML – це графічна мова, яка за допомогою глибокого синтаксису, у вигляді графічних елементів та комплексу стандартів дозволяє провести моделювання чи проектування системи та її процесів у вигляді наглядних діаграм [7].

Для початку було вирішено провести функціональне моделювання системи. Для цього було проаналізовано та сформовано список акторів (табл. 2.2) та прецедентів (табл. 2.3).

Актор – це певне абстрактне розуміння зовнішній сутностей, яке описує собою множину користувачів, що можуть взаємодіяти з системою. Кожен актор охоплює певну множину прецедентів [8].

Таблиця 2.2

Актори

Актор	Опис
Бібліограф	Особа, яка є відповідальною за управління асортиментом платформи
Користувач(Читач)	Ключовий актор, на якого націлена система. Є споживачем послуг електронної дистрибуції
Платіжна система	Зовнішня сутність, яка відповідна за обробку платіжних транзакцій

Закінчення таблиці 2.2

Актор	Опис
Аналітик	Особа, яка займається аналізом даних
Платформа дистрибуції	Інформаційна система, яка займається обробкою транзакцій

Прецедент – це певне абстрактне розуміння, яке описує дію або послідовність дій з боку системи, що веде до отримання конкретного результату.

Таблиця 2.3

Прецеденти

Прецедент	Актор(и)	Опис
Перегляд асортименту	Користувач	Ознайомлення з наявними книжками
Оцінювання книжок	Користувач	Надання оцінки книжці з боку користувача
Формування замовлення	Користувач	Додавання книжок в замовлення
Оплата замовлення	Користувач	Здійснення оплати за замовлення
Перегляд придбаних книжок	Користувач	Ознайомлення з книжками, які вже були придбані
Скачування книжки	Користувач	Скачування копії книжки на власний пристрій
Управління асортиментом платформи	Бібліограф	Видалення, редагування, додавання книжок та тощо
Обробка платежу	Платіжна система	Отримання, обробка платежу та повернення результату

Закінчення таблиці 2.3

Формування звітів	Аналітик	Формування звітів по діяльності платформи
Отримання аналітичних даних	Аналітик	Отримання даних, які необхідні аналітику для здійснення аналізу
Отримання рекомендацій	Користувач	Формування списку книг, що може сподобатися користувачу, на основі наявних даних
Опрацювання операцій	Платформа дистрибуції	Автоматизація процесів

На основі цих акторів та прецедентів була створена діаграма прецедентів (рис. 1), яка дозволяє пов'язати ці сутності між собою.

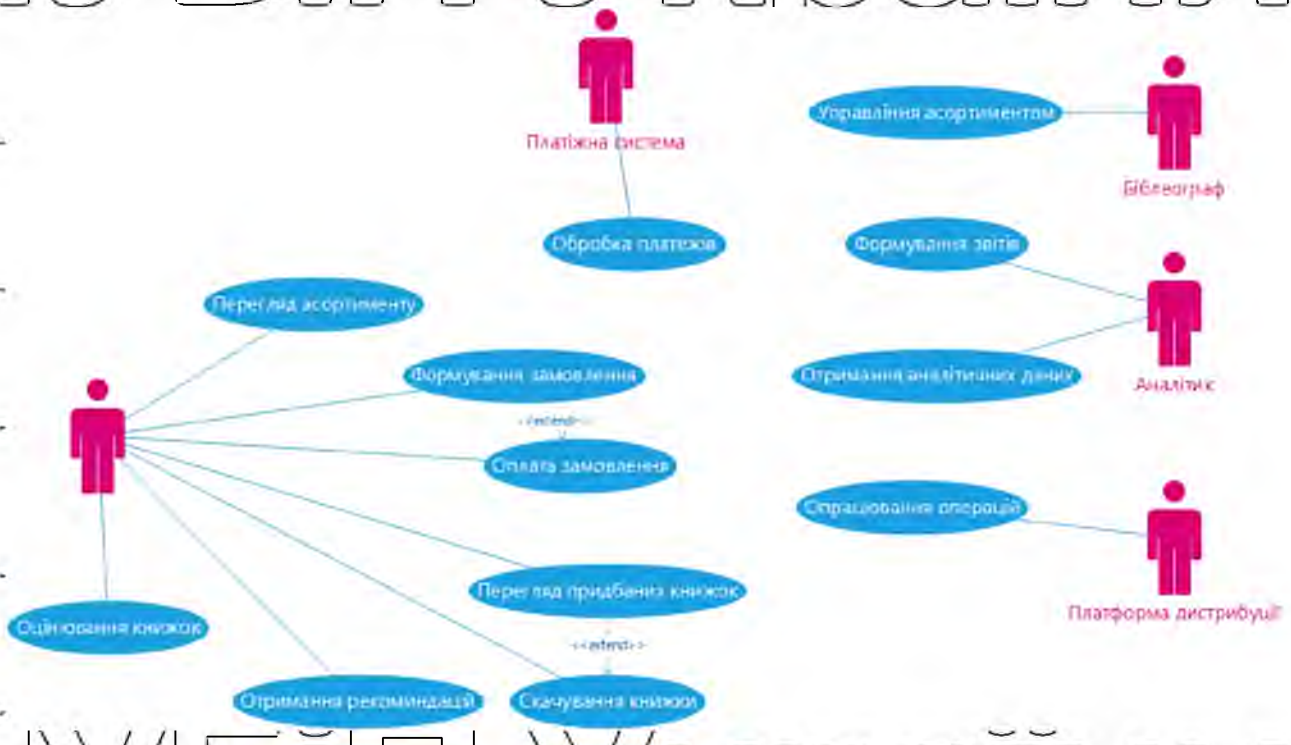


Рисунок 1. Діаграма прецедентів

Завдяки цьому кроку моделювання вдалося виявити групи користувачів та зрозуміти їх функціональні очікування від системи.

Тепер, маючи розуміння процесів в системі, було побудовано діаграму діяльності (рис. 2), яка дозволила пов’язати процеси та показати хід їх протікання в системі [9].

Рисунок 2. Діаграма діяльності

Ця діаграма дає більше розуміння протікання в процесу, включаючи їх зв'язко між собою, загальний хід протікання бізнес процесів, можливі розгилки та тощо.

Далі було вирішено побудувати діаграму послідовностей (рис. 3), яка дозволяє зображати обмін повідомленнями між акторами, тим самим це дозволяє збільшити розуміння протікання даних в системі, що є надзвичайно важливим в контексті розробки інформаційної системи.

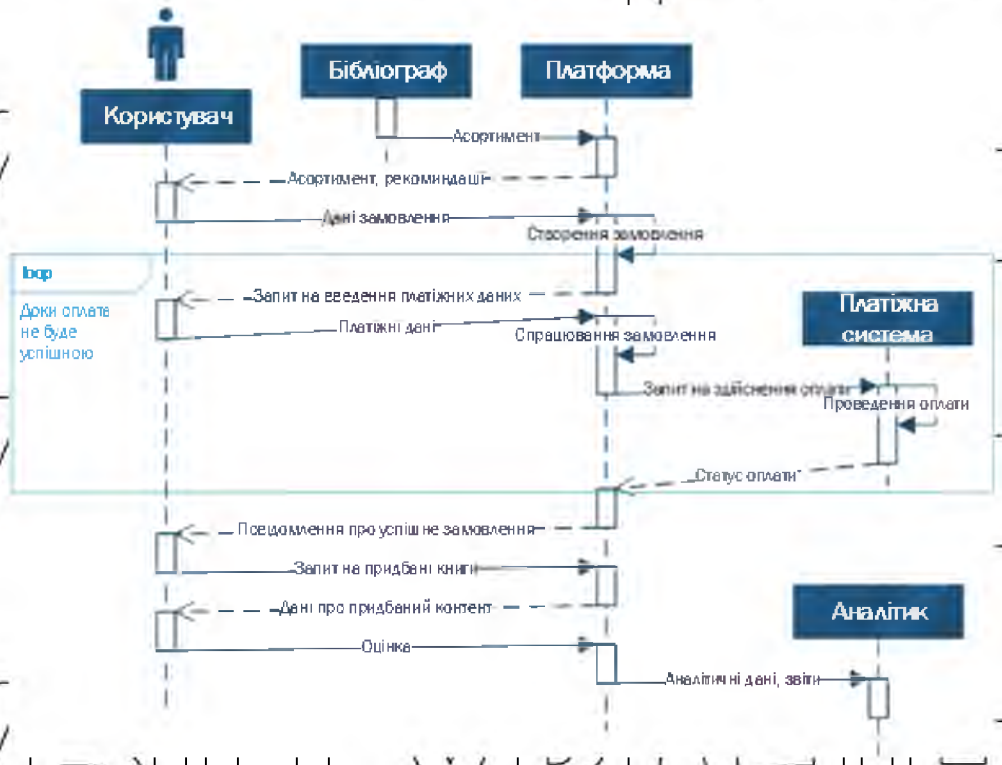


Рисунок 3. Діаграма послідовності

Ця діаграма дозволила продемонструвати обмін повідомленням між об'єктами в системі під час виконання бізнес-процесів, що дозволяє отримати представлена щодо потоків даних в системі [10].

На цій діаграмі можна побачити характерну особливість, про яку йшла мова раніше, а саме те, що центральним елементом є платформа цифрової дистрибуції, або інакше кажучи, розроблена система. З діаграми видно, що саме вона відповідає за логіку управління потоками даних, опрацьовування операцій та тощо.

3 РОЗРОБКА СИСТЕМИ

3.1 Проектування системи

Відповідно до постановки завдання та результатів аналізу предметної області було обрано клієнт-серверну взаємодію, за якою система буде розділена на клієнтський додаток та серверну частину, яка буде відповідати за виконання бізнес-процесів.

Відповідно до постановки задач та аналізу технологій існуючих рішень платформою було обрано веб-систему, у зв'язку з:

- веб-додаток доступний для будь-яких платформ, які підтримують веб-браузер, що дозволяє охопити більшу аудиторію;
- веб-додаток не потребує інсталяція на пристрій користувача та наявності додаткових програмних компонентів окрім браузера;
- у випадку подальшої розробки мобільного додатку існує можливість використовувати існуючу веб-систему, наприклад за допомогою інструментів для роботи з веб-елементами.

На цей момент існують розроблені клієнтські додатки для взаємодії з веб-системи – браузери, які повністю задовольняють вимоги системи, що усуває необхідність розробки власного клієнтського додатка.

При проектуванні системи було досліджено існуючі підходи щодо проектування архітектури програмного забезпечення, а саме монолітну архітектуру та мікрос-сервісну архітектуру [11].

Монолітна архітектура - це традиційна модель програмного забезпечення, яка є єдиним модулем, що може працювати повністю автономно і незалежно від інших додатків. Система, яка розроблена за цим підходом - це

окрема велика обчислювальна мережа з єдиною базою коду, де об'єднані всі бізнес-завдання.

Коли мова заходить про моноліту архітектуру (рис. 4), то часто монолітом називають щось велике і неповоротке, і ці два слова добре описують монолітну архітектуру для проектування ПЗ, що в свою чергу гарно характеризує проблематику цього підходу.

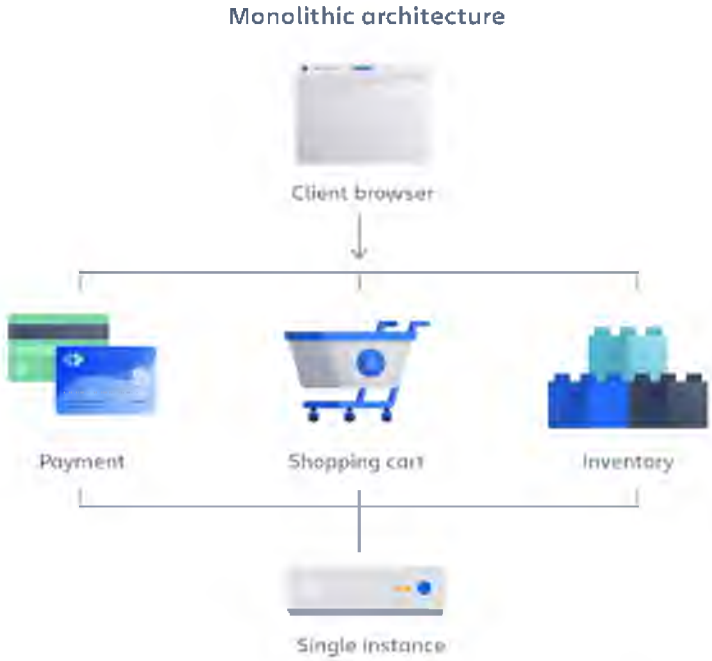


Рисунок 4. Монолітна архітектура

Щоб додати зміни в таку програму, необхідно оновити весь стек через базу коду, а також створити та розгорнути оновлену версію інтерфейсу, що знаходиться на стороні служби. Це обмежує роботу з оновленнями та потребує багато часу.

Іншою проблемою є те, що з часом така система лише накопичує об’єм, складність та обмеження, що обумовлені існуючою логікою, що збільшує складність тривалої підтримки таких систем, особливо для розробників, які раніше не були залучені в розробку цієї системи.

Для подолання цих проблем застосуються різні підходи до проектування моноліту, та патерни програмування, які дозволяють розбивати систему на різні компоненти та зменшувати зв'язність між ними, але це не може повністю вирішити ці проблеми, так, як система залишається монолітною.

Наприклад, одним з етапів прогресу в цьому напрямку була, так, звана багаторівнева архітектура (рис. 5), за якою система поділяється на декілька шарів, які відповідають за різні задачі. Найбільш поширеною була трьохрівнева архітектура, де були такі рівні: рівень презентації(інтерфейс), рівень доступу до даних, та рівень бізнес-логіки.

Ця архітектура відноситься до клієнт-серверних архітектури, яка вводить такі поняття:

- **клієнт** – пристрій на боці користувача, який відповідає за відправку запитів, отримання та обробку відповідей;
- **сервер** – потужний пристрій, який відповідає за отримання та обробку запитів, відправку відповідей.

Модель такої системи полягає в тому, що клієнт відправляє запит на сервер, де він обробляється, і готовий результат відправляється клієнтові. Сервер може обслуговувати кілька клієнтів одночасно.

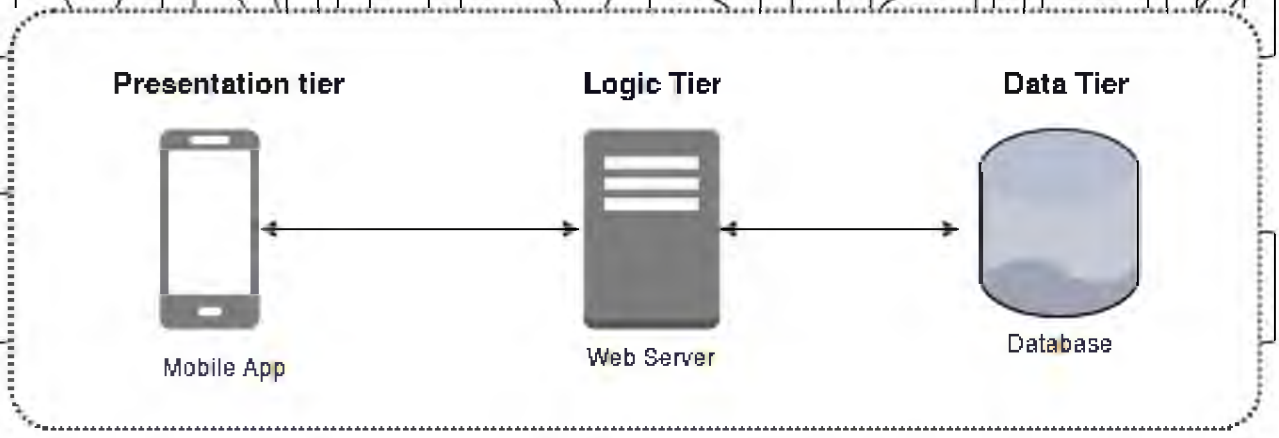


Рисунок 5. Багаторівнева архітектура

Враховуючи те, що з розвитком сфери інформаційних технологій складність та величина систем стрімко зростали, як і розмір команд розробки цих систем, було необхідно знайти спосіб, який би дозволив забезпечити вирішення цих проблем. Таким рішенням стала поява нового підходу, а саме мікро-сервісної архітектури.

Мікро-сервісна архітектура (або просто «мікро-сервіси») являє собою метод організації архітектури, заснований на ряді служб (рис. 6), що незалежно розгортаються. На відміну від розбиття на програмні компоненти, кожна така служба має власну бізнес-логіку та базу даних з конкретною метою. Оновлення, тестування, розгортання та масштабування виконуються всередині кожної служби.

Рисунок 6. Мікро-сервісна архітектура

Мікро-сервіси розбивають великі завдання, характерні для конкретного бізнесу, на кілька незалежних баз коду. Мікро-сервіси не знижують складність, але вони роблять будь-яку складність видимою і більш керованою, поділяючи завдання на дрібніші процеси, які функціонують незалежно один від одного і роблять внесок у загальне ціле.

Перевагами такого підходу є:

- **масштабування**. Кожна служба проектується таким чином, щоб вона відповідала лише за свою область задач та була максимально автономною. Цей підхід дозволяє легко масштабувати систему, так, як існує можливість масштабування саме необхідних компонентів, а не всієї системи цілком, що характерно для монолітної системи;
- **надійність**. У зв'язку з незалежністю кожної служби відмова однієї з них не буде загрожувати працездатності всієї системи, а лише тим функціональним можливостями, які забезпечувала служба, яка вийшла з ладу. Також стійкість системи забезпечується підтримкою гарячої заміни, тобто можливості перезавантажити чи замінити одну з служб, без зупинки всієї системи;
- **повторне використання**. З розгорнутим сервісом можуть працювати всі клієнтські додатки, які існують на цей момент, або будуть розроблені в майбутньому. Так, при розробці нового додатку для отримання доступу до розроблених служб необхідно лише налагодити взаємодію через обмін повідомленнями;
- **гнучкість**. Малий розмір мікро служб дозволяє гнучко ставиться до розгортання системи, наприклад існує можливість розгорнути кожну службу на окремому фізичному вузлу, а принцип спілкування з іншими компонентами дозволяє взаємодіяти з розробленими службами з будь-якої платформи, що підтримує обраний протокол взаємодії, в цьому випадку – http.

Звісно цей підхід не можна назвати ідеальним, й він має свої недоліки [12], а саме:

- **менша теоретична база,** даний підхід є відносно молодим, що означає відсутність такої потужної теоретичної бази, яка була сформована для монолітного підходу за весь час його існування;
- **відсутність стандартизації**, без загальної платформи може виникнути ситуація, де розширюється список мов, стандартів ведення журналів та засобів моніторингу.
- **складність**, мікро-сервіси ускладнюють роботу порівняно з монолітною архітектурою, оскільки у різних місцях виникає дедалі більше служб, створених кількома командами, що відповідно збільшує складність проектування, розробки, тестування, впровадження та підтримки;
- **розростання**, система на основі мікро служб може бути суттєво більшою ніж монолітна у зв'язку з необхідності створення інструментів для спілкування між частинами системи та їх незалежністю. Якщо розростання не контролюється належним чином, воно призводить до уповільнення розробки та зниження операційної ефективності.;
- **експонентне зростання витрат** на інфраструктуру. Кожен новий мікро-сервіс може мати свою вартість комплекту тестів, інструкцій з розгортання, інфраструктури хостингу, інструментів моніторингу тощо.

Таким чином з розгляду цих трьох підходів можна замітить тенденцію розвитку підходу до створення архітектури програмного забезпечення (рис. 7), а саме зменшення зв'язаності програмних компонентів, та їх розділення на відокремлені частини.

Разом з цим також змінювались підходи до розміщення програмного забезпечення на користь збільшення кількості вузлів, та використання готових сервісів, які надають послуги по розміщенню програмного забезпечення. Також

адаптувались і підходи до розробки програмного забезпечення, які мали стать більш гнучкими.

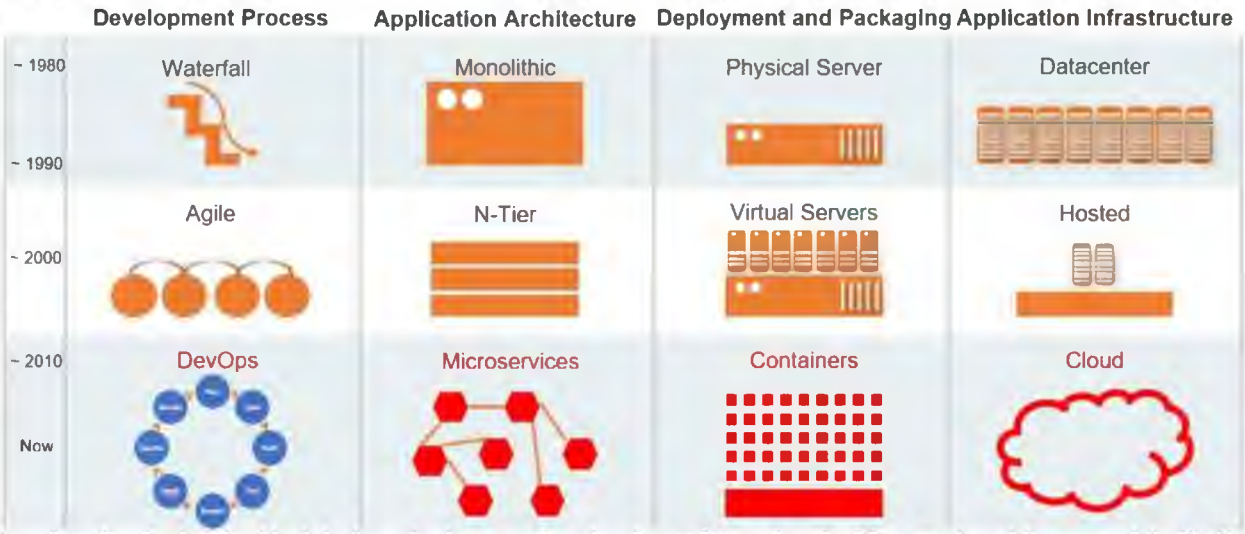


Рисунок 7. Прогрес підходу до розробки програмного забезпечення

Після дослідження існуючих підходів для проектування архітектури програмного забезпечення, аналізу їх переваг та недоліків було вирішено обрати мікро-сервісну архітектуру. Це рішення було прийняте, у зв'язку з тим, що розглянуті переваги цієї архітектури є більш суттєвими ніж недоліками в рамках цього проекту.

Так, було обрано підхід на базі мікро служб тому, що саме він дозволяє створити гнучку, потужну та сучасну систему цифрової дистрибуції, що може задовольнити всі технічні вимоги, та буде здатна до подальшого масштабування, наприклад створення та внесення до інфраструктури системи додатку для мобільних чи настільних ОС.

Відповідно до обраного підходу по розробці архітектури та результатах моделювання предметної області було виділено такі складові системи:

1. клієнтський сервіс, який відповідає за взаємодію з іншими сервісами, та створенню веб-інтерфейсу;
2. сервіс ідентифікації, який відповідає за авторизацію та аутентифікацію користувача;

3. сервіс каталогу, який відповідає за управління асортиментом платформи;
4. сервіс замовлень, який відповідає за процеси створення та оплати замовлень;
5. сервіс рейтингу, який відповідає за процеси надання оцінки книгам та отримання цих відгуків. Також має реалізовувати можливість формування рекомендацій на основі наданих відгуків;
6. система аналізу, яка відповідає за накопичення даних з системи та проведення аналізу цих даних з метою отримання висновків, які можуть бути корисними.

3.2 Розробка клієнтського сервісу

Задача цього сервісу полягає в взаємодії з користувачем, отримання від нього запитів, їх обробку та надання йому відповідей, у зручному для нього форматі. Інтерфейсом взаємодії є графічний веб-інтерфейс.

В якості архітектури цього компоненту було обрано шаблон проектування MVC (рис. 8), або інакше кажучи «Model-View-Controller», який чудово показує себе для веб-орієнтованих додатків [13].

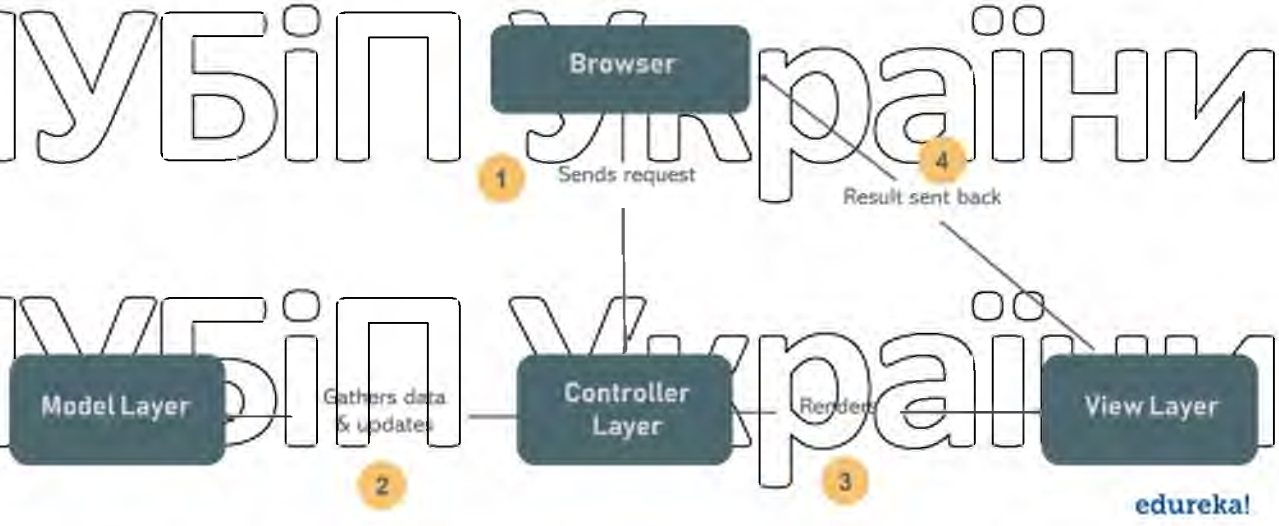


Рисунок 8. Шаблон MVC

Суть цього шаблону побудови додатку полягає поділу даних програми та керуючої логіки на три компоненти, таким чином, що модифікація кожного компонента може здійснюватися незалежно. До цих компонентів відносяться наступні:

- **Model** – компонент модель, який включає дані та поведінку, наприклад правила перевірки коректності даних для форми;
- **View** – компонент представлення, який описує візуальну частину графічного інтерфейсу;
- **Controller** – компонент контролер, який є центральним елементом, який містить логіку по обробці запитів й забезпечує зв'язок між користувачем та системою.

Алгоритм обробки запитів з використанням цього підходу зводиться до таких кроків:

1. отримання від клієнтського додатку (скорінг за все браузера) запиту, сервер аналізує цей запит та обирає до якого контролеру необхідно його передати;
2. на основі запиту контролер вибирає необхідну модель, та ініціює процеси для формування цієї моделі;
3. на основі запиту контролер обирає вибирає необхідне представлення, та на базі сформованої моделі створює її (в рамках даної розробки створюється веб-сторінка, яка включає мову розмітки HTML, каскадну таблицю CSS, та може включати JavaScript код);
4. отримане представляє відправляється на клієнтський додаток, з якого був відправлений запит, як відповідь на цей запит.

3.3 Сервіс ідентифікації

Задачею служби ідентифікації є виконання авторизації та аутентифікації користувача та перевіркою права на доступ до захищених ресурсів.

Ця служба має критичне значення для роботи платформи дистрибуції, так, як вона несе відповідальність за запобігання випадкам неправомірного використання послуг системи

Ця служба була побудована за загально прийнятою технологією делегування авторизації на іншу службу за проколом https - «OAuth2». На цей момент це найбільш поширений стандарт авторизації та автентифікації, за яким працюють такі компанії, як Google, Microsoft, Facebook й підтримка, якої існує в багатьох технологіях.

Додатковою перевагою технології OAuth2 є можливість інтеграціє з іншими сервісами авторизації та аутентифікації, що використовують цю технологію. Так, є можливість використання акаунта, що був створений в сервісу ідентифікації від Google, Facebook та тощо, або навпаки надавати послуги розробленого сервісу стороннім системам [14].

Принцип роботи цієї технології полягає в алгоритму (рис. 9), що включає такі кроки:

1. користувач отримує від сервісу ідентифікації токен доступу, що містить інформацію про цього користувача та його дозволи за допомогою даних авторизації(найбільш поширена практика, це логін або електрона адреса, та пароль). На цьому кроці також можлива двох рівнява авторизація;
2. користувач робить запит до ресурсного сервісу, де до цього запиту кріпиться його токен доступу;
3. ресурсний сервіс отримує запит і робить запит до сервісу ідентифікації для перевірки токена доступу. Якщо перевірка проходить успішно запит від користувача виконується і йому надається відповідна відповідь;

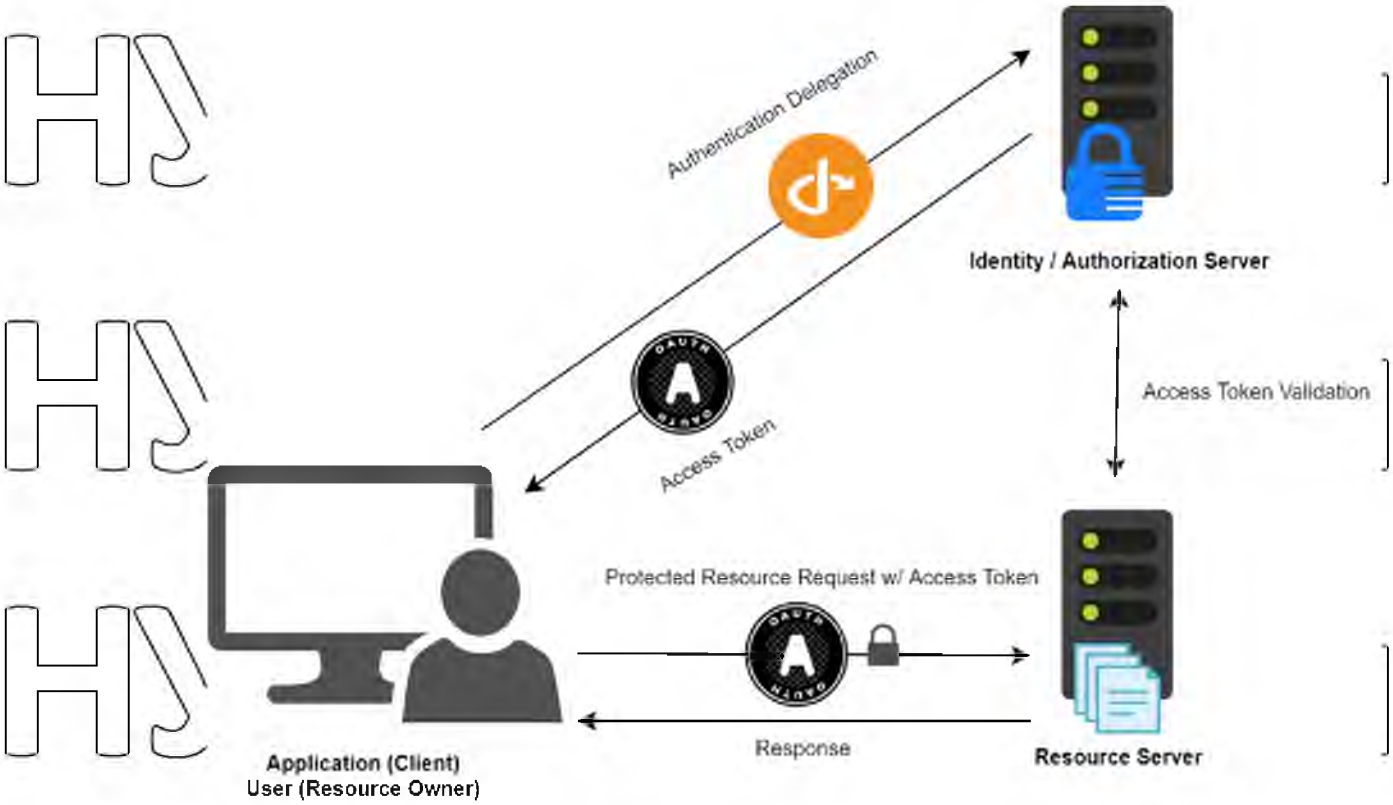


Рисунок 9. Принцип роботи OAuth2

Важливо зауважити, що ідентифікація користувача може відбуватися по різному: користувач може вводить свої дані в клієнтському додатку, який потім передає до служби ідентифікації, або може відбуватись переадресація на службу ідентифікації, де буде надано форма для введення даних.

В розробленій системі було прийнято обрати другий варіант, так, як він є більш захищеним, але це вимагало створення графічного інтерфейсу для цієї служби.

Інформаційна частина. Так, як цей сервіс має зберігати постійну інформацію, то є необхідність в реалізації бази даних та взаємодії з нею.

За основу були взяті сутності (рис. 10), які пропонує компанія Microsoft для створення служб ідентифікації, але трохи розширені відповідно до потреб системи:

- **Користувач** – сутність, яка містить персональні дані та дані про акаунт;
- **Роль** - сутність, яка описує можливі ролі користувача в системі;
- **Дозвіл** - сутність, яка описує окремий дозвіл на якийсь ресурс, може бути прикріпленим, як до ролі, так, і до конкретного користувача.

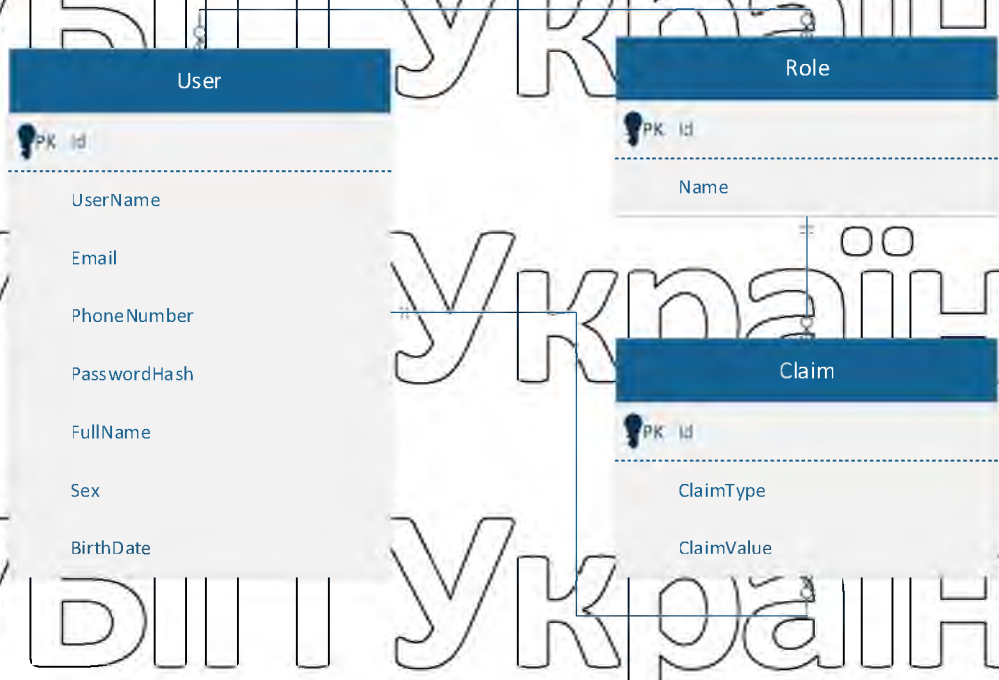


Рисунок 10. Логічна схема

Беручи до уваги структурованість даних в цьому сервісу, в якості бази даних було вирішено зупинить на СУБД. Для збільшення ефективності збереження та роботи з даними було проведена нормалізація даних до 3 нормальної форми (рис. 11), що дозволило збільшити логічну цілісність даних та стійкість до помилок.

У процесі нормалізації було проведено перетворення даних та розбиття таблиць на окремі, що дозволило забезпечити дотримання розробленою моделлю вимог описаних нормальних форм. Також було створено проміжні сутності, які дозволили привести зв'язки типу багато-до-багатьох, які порушують вимоги нормалізації, до зав'язків типу один-до-багатьох.

Рисунок 11. Фізична схема

Рішення зупиниться на третій нормальній формі та відмовиться від приведення до більш високих рівнів нормалізацію, було прийнято у зв'язку з тим, що цілі нормалізації були досягнуті вже на рівні третьої нормальної форми, а подальша нормалізація буде ускладнювати базу даних та негативно впливати на швидкість роботи бази даних [15].

3.4 Сервіс каталогу

Задачею сервісу каталогу є управління асортиментом платформи цифрової дистрибуції, для цього він має надавати відповідні інструменти для маніпулювання з даними. Ця служба була побудована за архітектурним стилем

«REST API». Відповідно до постановки завдання та підходу REST API було спроектовано методи цього сервісу (табл. 3.1).

Таблиця 3.1

Методи сервісу каталогу

REST Метод	Адреса	Опис
GET	Catalog	Асортимент
GET	Catalog/Genre	Перелік жанрів
GET	Catalog/Tag	Перелік тегів
GET	Catalog/Author	Перелік авторів
GET	Catalog/{isbn}	Детальна інформація про книгу
GET	Catalog/{isbn}/Cover	Книжкова обкладинка
GET	Catalog/{isbn}/File	Перелік файлів
GET	Catalog/ File/{id}	Скачати файл
POST	Catalog/Tag	Додати тег
POST	Catalog/Author	Додати автора
POST	Catalog	Додати книгу
PUT	Catalog/{isbn}/File	Додати файл
PUT	Catalog/Tag	Оновити тег
PUT	Catalog/Author	Оновити автора
PUT	Catalog	Оновити книгу
DELETE	Catalog/Tag	Видалить тег
DELETE	Catalog/Author	Видалить автора
DELETE	Catalog/ File/{id}	Видалить файл
DELETE	Catalog/{isbn}	Видалить книгу

Інформаційна частина. Так, як цей сервіс має зберігати постійну інформацію, то є необхідність в реалізації бази даних та взаємодії з нею. Було виділено такі сутності (рис. 12):

- **Книжка** - містить дані про книгу;
- **Жанр** - містить дані про книжковий жанр;
- **Автор** - містить дані про автора книжки;
- **Тег** - містить дані книжкового тегу;
- **Файл** - містить дані про файл, що прикріплений до книжки.

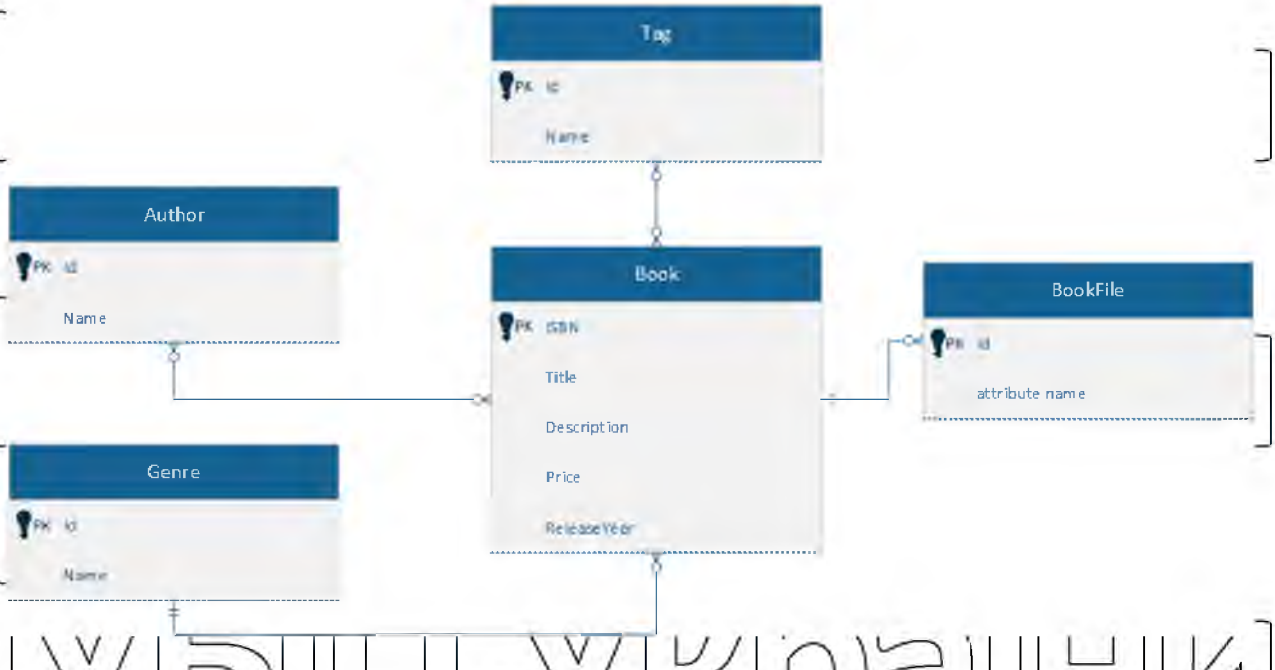


Рисунок 12. Логічна схема каталогу

Беручи до уваги структурованість даних в цьому сервісу, в якості бази даних було вирішено зупинить на СУБД.

Для збільшення ефективності збереження та роботи з даними було проведена нормалізація даних до 3 нормальної форми (рис. 13), що дозволило збільшити логічну цілісність даних та стійкість до помилок.

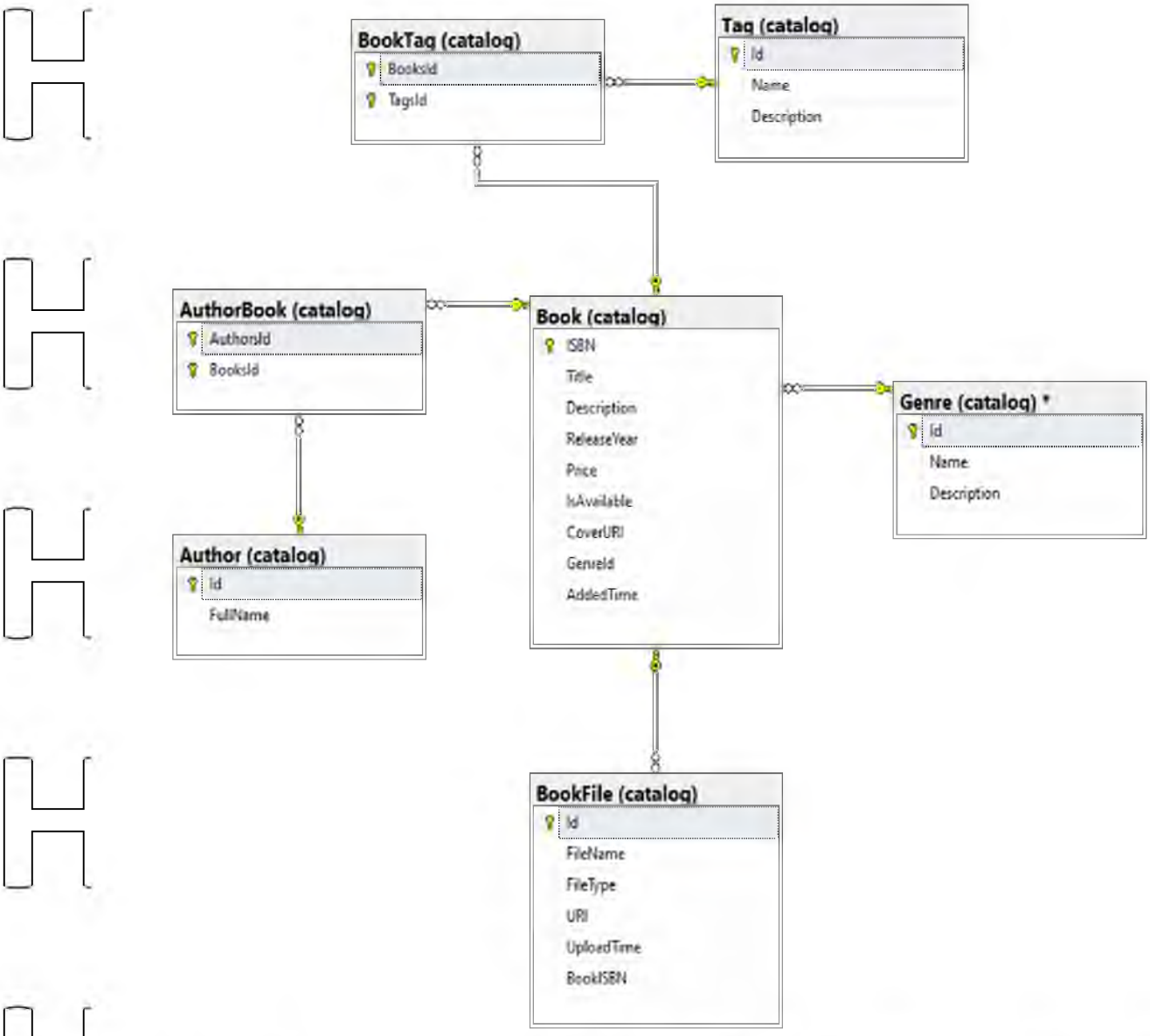


Рисунок 3. Фізична схема каталогу

3.5 Сервіс замовлень

Задачею сервісу замовлень є управління замовленнями в системі, їх формування, оплата та тощо. Для проведення оплати замовлення було здійснено інтеграцію з платіжним сервісом LiqPay.

Ця служба була побудована за архітектурним стилем «REST API». Відповідно до постановки завдання та підходу REST API було спроектовано методи цього сервісу (табл. 3.2).

Таблиця 3.2

Методи сервісу замовлень

REST Метод	Адреса	Опис
GET	Basket	Перелік книг, що зараз знаходиться в корзині
POST	Basket	Додати книгу до корзини
DELETE	Basket	Видалить книгу з корзини
GET	Order	Список існуючих замовлень
POST	Order	Створити нове замовлення
GET	Order/Bought Book	Перелік придбаних книжок

Інформаційна частина. Так, як цей сервіс має зберігати постійну інформацію, то є необхідність в реалізації бази даних та взаємодії з нею.

Було виділено такі сутності (рис. 14):

- **Замовлення** - описує замовлення, що сформувала система;
- **Пункт замовлення** - описує книгу, яка увійшла до складу замовлення;
- **Користувач** - описує користувача;
- **Пункт корзини** - описує книги, які увійшли до корзини користувача.

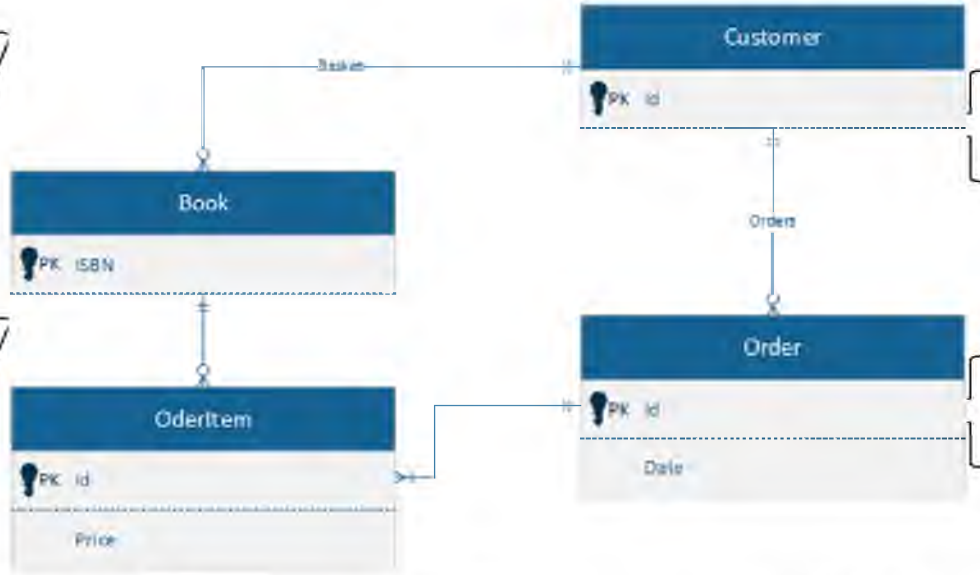


Рисунок 14. Логічна схема

Беручи до уваги структурованість даних в цьому сервісу, в якості бази даних було вирішено зупинить на СУБД. Для збільшення ефективності збереження та роботи з даними було проведена нормалізація даних до 3 нормальної форми (рис. 15), що дозволило збільшити логічну цілісність даних та стійкість до помилок.

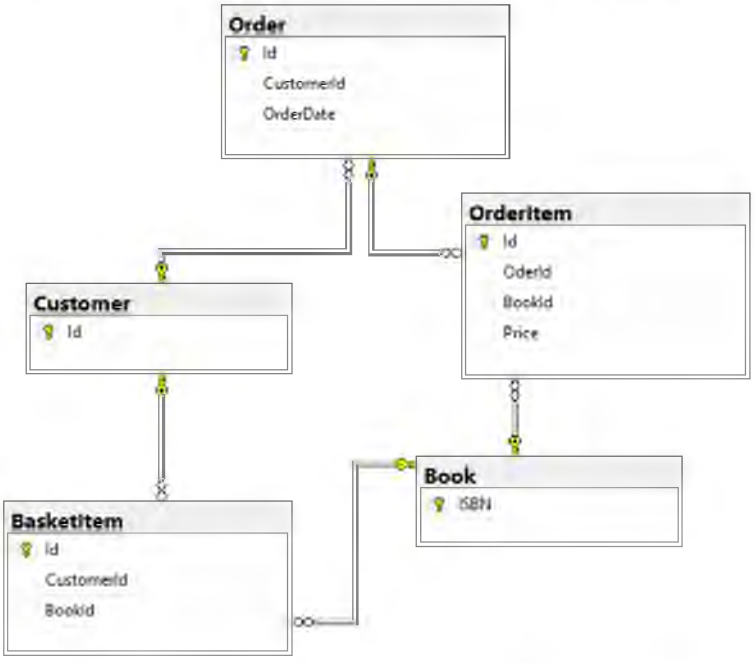


Рисунок 15. Фінальна схема

3.6 Сервіс рейтингу

Заданою сервісу рейтингу є управління надання користувачу можливості оцінити книжку та реалізації інструментів отримання цих відгуків. Також саме цей сервіс має реалізовувати можливість формування оперативних рекомендацій книг на основі наявних відгуків.

Ця служба була побудована за архітектурним стилем «REST API». Відповідно до постановки завдання та підходу REST API було спроектовано методи цього сервісу (табл. 3.3).

Таблиця 3.3

Методи сервісу рейтингу

REST Метод	Адреса	Опис
GET	Review	Перелік наявних відгуків
POST	Review	Створити новий відгук
PUT	Review	Оновити відгук
DELETE	Review	Видалить відгук
GET	CrossedCustomers	Отримати перелік користувачів, які мають суміжні відгуки
GET	CustomerCrossedReviews	Отримати перелік суміжних відгуків
GET	CrossedBooks	Отримати перелік книжок, які мають суміжні відгуки
GET	BookCrossedReviews	Отримати перелік суміжних відгуків
GET	CustomerRecommendation	Отримати рекомендації для коритувача

Алгоритмічна частина. Для реалізації оперативної можливості формування рекомендацій був створений метод.

Мета системи рекомендацій полягає в тому, щоб пропонувати користувачам відповідні елементи. Для вирішення цього завдання існують дві основні категорії методів (рис. 16): методи спільної фільтрації та методи, засновані на вмісті [5].

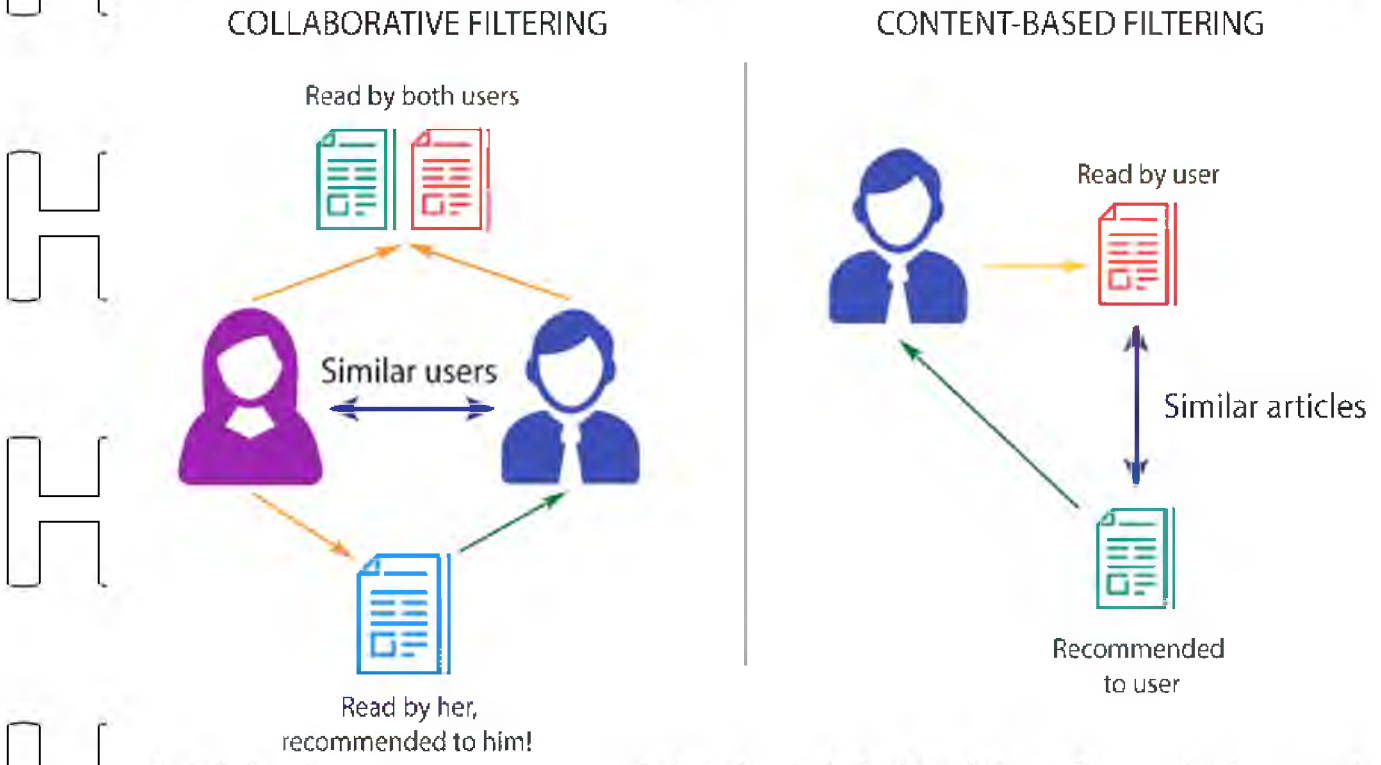


Рисунок 16. Категорії методів

Для розробки методу рекомендацій даного сервісу було вирішено використовувати методи спільної фільтрації, які ґрунтуються виключно на минулих взаємодіях, записаних між користувачами та елементами, для створення нових рекомендацій. Ці взаємодії зберігаються в так званій «матриці взаємодій користувач-елемент».

Головна ідея, яка покладена в основі методів співпраці, полягає в тому, що цих попередніх взаємодій між користувачем і елементом достатньо для

виявлення подібних користувачів та/або схожих елементів і створення прогнозів на основі цих оціночних відстаней.

Основна перевага спільних підходів полягає в тому, що вони не вимагають інформації про користувачів або елементи, тому їх можна використовувати в багатьох ситуаціях. Крім того, чим більше користувачів взаємодіє з елементами, тим точнішими стають нові рекомендації: для фіксованого набору користувачів і елементів нові взаємодії, зареєстровані з часом, приносять нову інформацію та роблять систему дедалі ефективнішою.

Але, оскільки для надання рекомендацій враховуються лише минулі взаємодії, спільне фільтрування страждає від «проблеми холодного запуску», тобто неможливо рекомендувати щось новим користувачам або рекомендувати новий елемент будь-якому користувачеві.

Алгоритми спільної фільтрації можна поділити на дві категорії, а саме: підходи на основі нам'яті та підходи на основі моделі.

Підходи, що опираються на пам'ять, працюють безпосередньо із значеннями, які були записані, та по суті, базуються на пошуку найближчих сусідів.

Підходи, що опираються на моделі, припускають базову «генеративну» модель, яка пояснює взаємодію між користувачем і предметом, і намагаються виявити її, щоб зробити нові прогнози.

Для даного сервісу було вирішено зупиниться на методу, що заснований на пам'яті, по типу «користувач - користувач». Щоб надати нову рекомендацію для користувача, цей приблизно намагається ідентифікувати користувачів із найбільш подібним «профілем взаємодії» (найближчими сусідами), щоб запропонувати елементи, які є найпопулярнішими серед цих сусідів (і цей елемент є новим для користувача).

Цей метод називають «орієнтованим на користувача», оскільки він представляє користувачів на основі їх взаємодії з елементами та оцінює відстані між користувачами.

Описати алгортим (рис. 17) можна наступним чином, кожен користувач може бути представлений своїм вектором взаємодії з різними елементами. Після цього нам необхідно розраховувати схожість між користувачами. Чим більш схожі вектори взаємодії у користувачів, тим більша їх схожість. Після цього нам лише залишається знайти найбільш популярні елементи в найближчих користувачах.

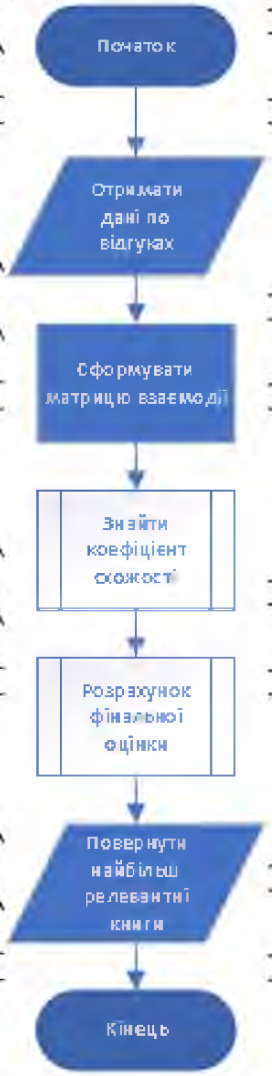


Рисунок 17. Алгоритм формування рейтингу

Ключовою складовою даного алгоритму є методика розрахування коефіцієнту схожості, так, як саме він вирішує наскільки користувачі будуть схожими. При виборі цього методу бажано звернуть увагу на тому, щоб він враховував, не лише самі оцінки, а й кількість схожих елементів.

Як функцію розрахування коефіцієнту схожості було обрано покращену подібність трикутників доповнену налаштуваннями оцінки користувача. Даний метод з'явився досить недавно і являє собою добуток методу покращеної подібності трикутника та уподобання користувача рейтингу URP [16].

Таким чином, покращена схожість трикутників (ITR) не лише зосереджена на загальних оцінках, таких як вимірювання TMJ, але також враховує незагальні оцінки користувачів.

$$sim^{ITR}(u,v) = sim^{Triangle}(u,v) * sim^{URP}(u,v) \quad (1)$$

Де $sim^{Triangle}(u,v)$ та $sim^{URP}(u,v)$ визначені в формулах 2 та 3.

$$sim^{Triangle}(u,v) = 1 - \frac{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}{}^2} + \sqrt{\sum_{i \in I_{uv}} r_{vi}{}^2}} \quad (2)$$

Де I_{uv} позначає набір предметів, оцінених u або v. Відповідно r це рейтинг.

$$sim^{URP}(u,v) = 1 - \frac{1}{1 + \exp(-|\bar{r}_u - \bar{r}_v| * |\sigma_u - \sigma_v|)} \quad (3)$$

Де, $\bar{r}$ означає середнє значення, а σ означає дисперсію.

В якості ж метода підрахунку фінальної оцінки для книги було вирішено взяти добуток коефіцієнту схожості користувача та оцінки цієї книги з боку цього користувача.

$$FinalGrade = koef * grade \quad (4)$$

Де $koef$ це коефіцієнту схожості користувача, а $grade$ це оцінка користувача.

Інформаційна частина. Так, як цей сервіс має зберігати постійну інформацію, то є необхідність в реалізації бази даних та взаємодії з нею. Було виділено такі сутності (рис. 18):

- **Книга** - описує книжку;
- **Користувач** - описує користувача;
- **Відгук** - описує відгук користувача, який він надав книжці;

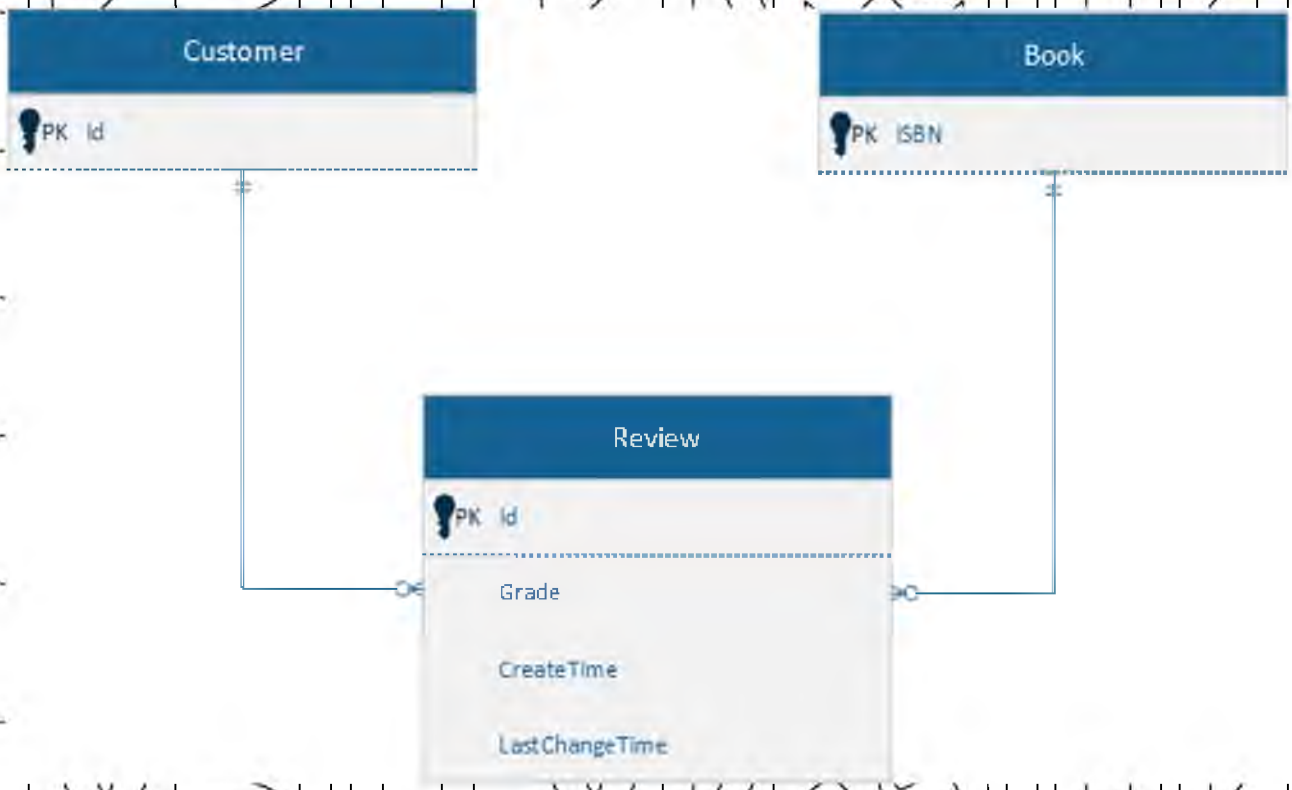


Рисунок 18. Логічна схема

Враховуючи особливості структури даних, а також особливості роботи цього сервісу було вирішено в якості бази даних обрати графову базу даних, так, як її використання здатне підвищити ефективність роботи сервісу.

Особливість графової бази даних полягає в тому, що всі дані в ній представлені у вигляді графа, що складається з вершин та ребр, які служать для опису відносин між вершинами [17].

Така структура дозволяє значено підняти ефективність для операцій, які працюють з зв'язками між сутностями. Наприклад, операції пошуку суміжних

користувачів чи відгуків. В загальному можна говорити, що чим більша глибина запиту, тим більше переваги дає використання графової бази даних.

Відповідно до цього були створено структуру графа, яка складається з компонентів, які представлено в табл. 3.4.

Таблиця 3.4

Елементи графа

Сутність	Тип елемента	Назва
Книжка	Вершина	Book
Користувач	Вершина	Customer
Відгук	Ребро	Grade

Так, за допомогою цих компонентів формується структура графа (рис. 19), яка описує зв'язки між сутностями. Всі ж поля прив'язуються до відповідних компонентів у форматі таблиці «key - value».

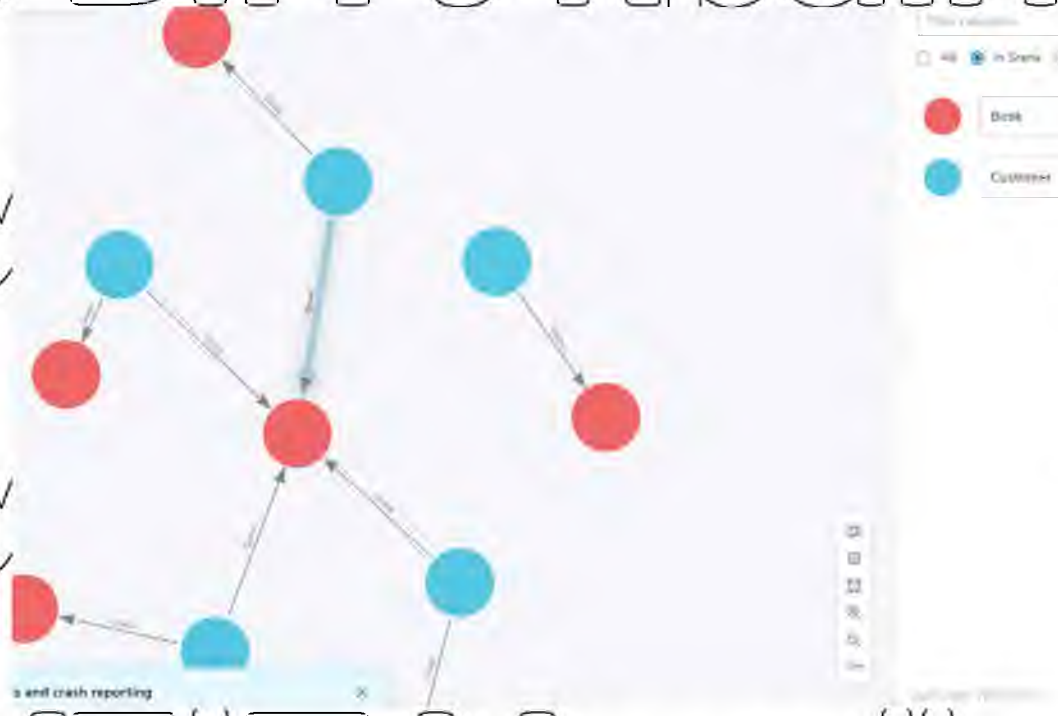


Рисунок 19. Зображення графа

3.7 Система аналізу

Заданою системи аналізу є накопичення даних по діяльності системи з подальшою їх обробкою та аналізом з ціллю отримання нових даних, які мають цінність.

Таким чином ця система буде виконувати 2 задачі:

- надавати аналітичну інформацію, розраховувати показники такі, як KPI та тощо;
- реалізовувати можливість формування рекомендацій.

На цей раз для формування рекомендацій буде застосуватися підхід не на основі суміжних фільтрів, а методи, що засновані на вмісту (рис 20).

На відміну від методів спільної роботи, які покладаються лише на взаємодію між користувачем і елементом, підходи, засновані на вмісті, використовують додаткову інформацію про користувачів та/або елементи. Якщо ми розглянемо приклад системи рекомендацій фільмів, цією додатковою інформацією можуть бути, наприклад, вік, стать, робота або будь-яка інша особиста інформація користувачів, а також категорія, головні актори, тривалість або інші характеристики для фільмів (предмети).

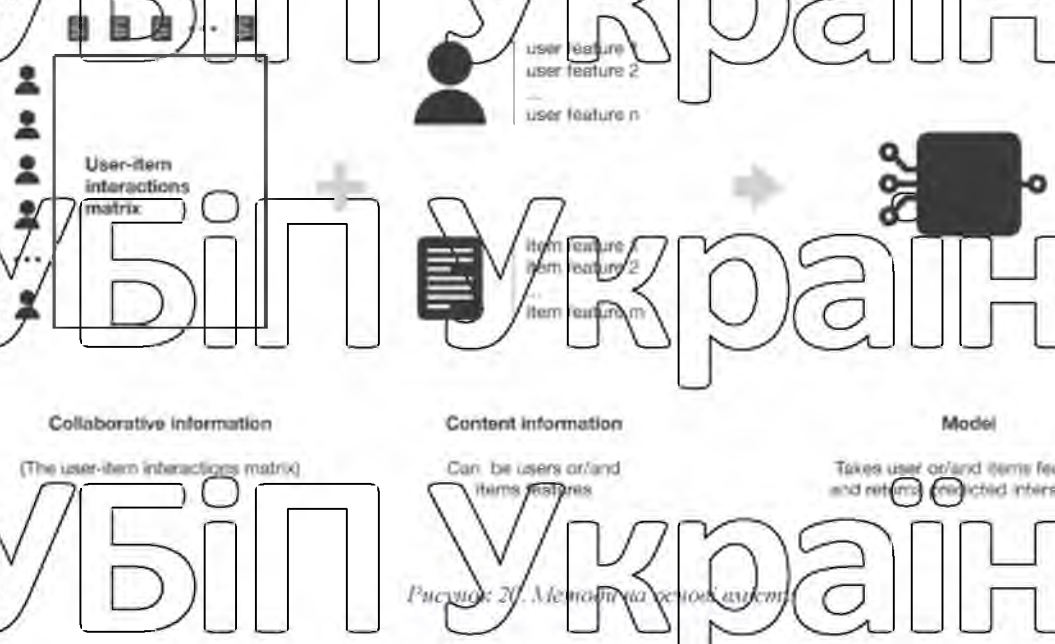


Рисунок 20. Методи на основі вмісту

Тоді ідея методів, заснованих на вмісті, полягає в тому, щоб спробувати побудувати модель на основі доступних «функцій», які пояснюють спостережувану взаємодію між користувачем і елементом. Розглядаючи користувачів і фільми, ми спробуємо, наприклад, змоделювати той факт, що молоді жінки, як правило, краще оцінюють деякі фільми, або те, що молоді чоловіки, як правило, краще оцінюють інші фільми та тощо. Якщо нам вдасться отримати таку модель, то зробити нові прогнози для користувача досить легко: нам просто потрібно подивитися на профіль (вік, стать, …) цього користувача та на основі цієї інформації визначити релевантні фільми для пропонувати.

Методи, засновані на вмісті, страждають від проблеми «холодного запуску» набагато менше, ніж спільні підходи: нові користувачі або елементи можуть бути описані своїми характеристиками (вмістом), і тому для цих нових об’єктів можна зробити відповідні пропозиції. Лише нові користувачі або елементи з функціями, які раніше не бачили, страждатимуть від цього недоліку.

Накопичення даних. Для накопичення даних було вирішено створити сховище даних(рис. 21) на основі реляційної бази даних.

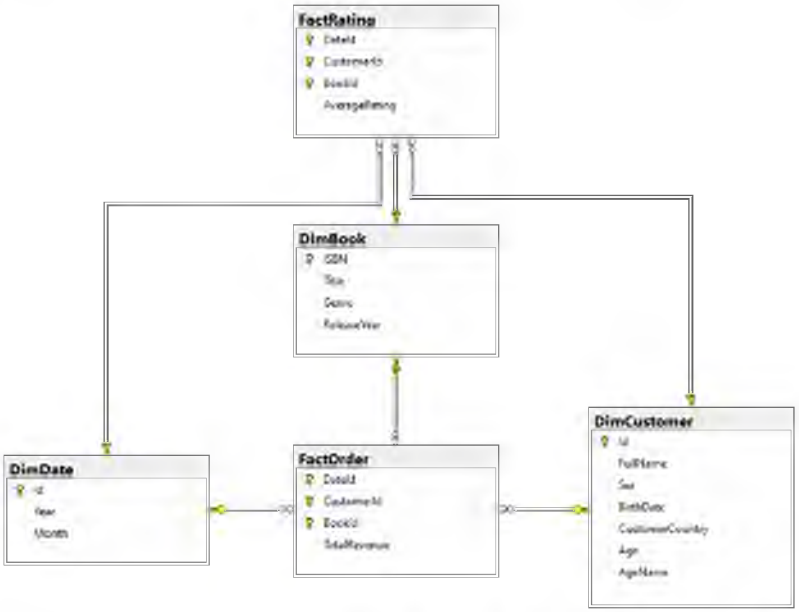


Рисунок 21. Фізична схема сховища даних

Це сховище даних включає містить 3 виміру:

- Dim Date – часовий вимір;
- Dim Book – містить інформацію про книжку;
- Dim Customer – містить інформацію про користувача.

Також до сховища даних входить 2 таблиці фактів:

- Fact Order – містить інформацію про замовлення;
- Fact Rating – містить інформацію про наданні відгуки.

Для реалізації аналізу даних було вибрано підхід на основі OLAP куба, OLAP (рис. 22), який дозволяє побудувати багатомірний куб даних на основі сховища даних. В такому кубу таблиці вимірів стають вимірами, або якщо точніше кожен описовий параметр з таблиць стає окремим виміром, а факти стають точками в місці пересікання цих вимірів [18].

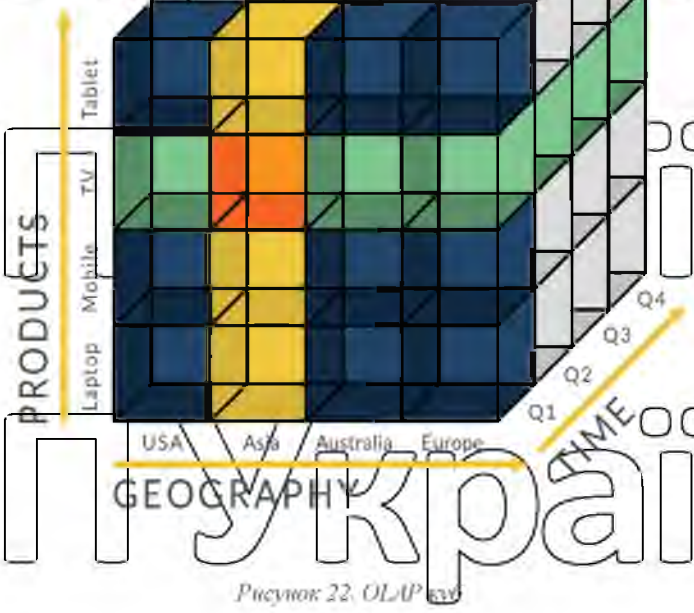


Рисунок 22. OLAP куб

Важливою особливістю цієї технології є те, що куб що підтримує операцію стискання, яка дозволяє нам взяти не точкові значення, а зріз по одному або декількох вимірах знайшовши агреговану інформацію по стиснутих значеннях.

Механізм стискання, а також можливість легко управляти вимірами надає надзвичайно потужний інструмент для аналітика, який може дозволить йому швидко отримувати аналітичні дані та перевіряти гіпотези, не звертаючись до програмісту для розробки нового запиту.

Існує 3 типи OLAP технологій:

- MOLAP (Multidimensional OLAP) – і детальні дані, і агрегати зберігаються у багатовимірній БД. І тут виходить найбільша надмірність, оскільки багатомірні дані повністю містять реляційні.
- ROLAP (Relational OLAP) - детальні дані залишаються у реляційній БД; агрегати зберігаються у тій же БД у спеціально створених службових таблицях.
- HOLAP (Hybrid OLAP) - детальні дані залишаються в реляційної БД, а агрегати зберігаються у багатовимірної БД.

Основним недоліком використання OLAP кубу є те, що він вимагає значного об’єму пам’яті, так, як йому необхідно зберігати всі можливі виміри і їх перерізи. Інший недолік пов’язаний з тим, що розробка OLAP куба пов’язана з певними складнощами.

Для побудови OLAP кубу було вибрано інструмент SQL Server Analysis Services, який дозволяє створювати MOLAP куб. Окрім потужного набору можливостей цей інструмент має ще одну важливу перевагу, а саме можливість інтеграції з іншими розробками від Microsoft, а саме: SQL Service, Power BI, Report Service, Excel та тощо [19].

OLAP куб (рис. 23) був побудований на базі розробленого сховища даних та унаслідував від нього свої виміри та факти без особливих перетворень. Єдиний додатковий елемент, це показник середнього рейтингу, який автоматично розраховується у зв’язку з тим, що по-замовчуванню куп не містить агрегацію типу середнього.

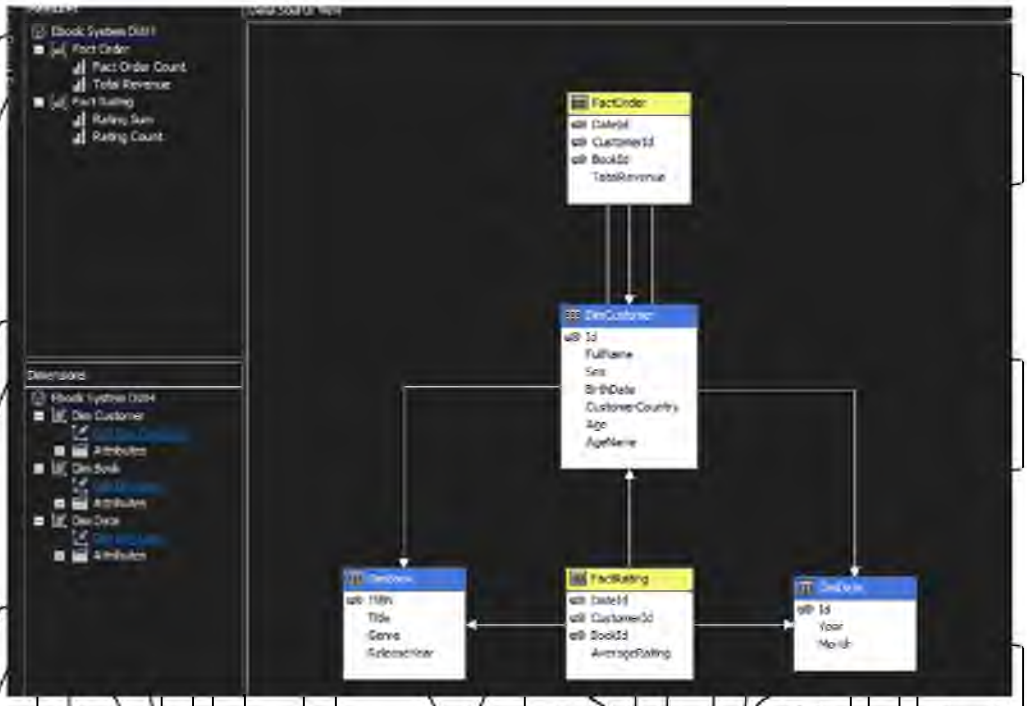


Рисунок 23. Створений куб

Для аналізу діяльності системи був створений KPI (рис. 24) – ключовий показник ефективності [20]

Рисунок 24. Створення KPI

Ціллю цього KPI було обрано перевершувати минулорічний результат, як по місяцю окремо, так і по всьому році. В якості вимірюваної величини служить місячних дохід. Також була реалізована механіка тренду, який вказує на ріст чи падіння прибутку у порівняні з минулим місяцем.

Як вже говорилось Analyses Service дозволяє використовувати методи Data Mining (рис. 25) для пошуку зав'язків в даних для знаходження нових закономірностей, що може дозволить виявить та спрогнозувати вподобання користувача [21].

Рисунок 25. Data Mining методи(моделі)

Для початку був використаний метод кластеризації (рис 26), який дозволяє знайти в набору даних групи, що вміщають схожі елементи. Для цієї моделі були вибрані таки поля: Вікова група, вік, країна, стать та скільки людина витратила на книги. Дана модель має дозволить аналітику зрозуміти, які групи є серед користувачів, кількість людей в цих групах і скільки вони витрачають грошей на книги. Особлива цінність цієї моделі полягає в тому, що методи рекомендацій на основі вмісту базуються не на окремому користувачу, а саме на ось таких спільних параметрів [22].

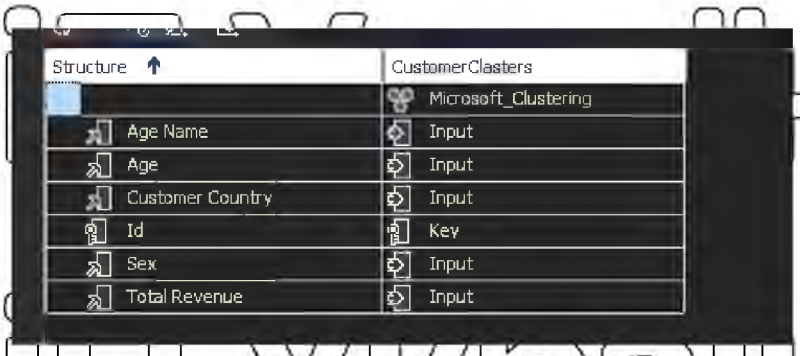


Рисунок 26. Метод кластеризації

Після цього був зроблена модель (рис. 27), яка за допомогою методу асоціативних правил має допомогти аналітику в пошуках правил, які дозволять зрозуміти, чи сподобається книга користувачу. До цієї моделі увійшли такі параметри: вікова група, країна користувача, його стать, жанр книги [23].

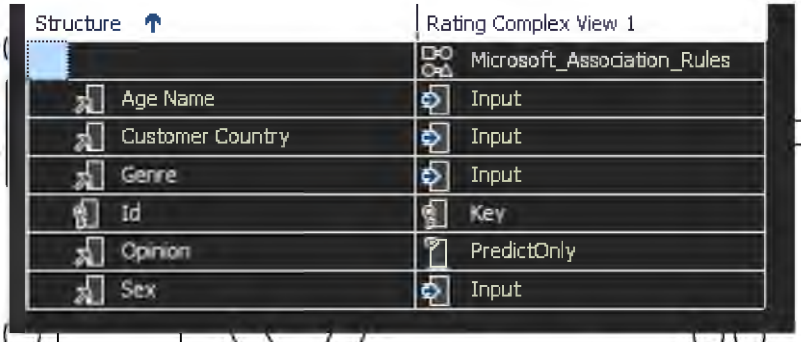


Рисунок 27 Метод асоціативних правил

І-за певних особливостей роботи Data Mining моделей було вирішено створити уявлення (рис. 28), в яке входять параметри цієї моделі(і не тільки).

Рисунок 28. Уявлення з повними даними

Також для цієї же цілі була створена модель (рис. 29) на основі методу наївного Байєса [23].

Рисунок 29. Модель на основі матриці Бакеса

За допомогою інструменту Report Service, який має інтеграцію з Analyses Service було побудовано два звіту для аналітика.

Перший звіт містить інформацію про рейтинг жанрів серед груп користувачів за статей та віковою групою. А другий звіт містить матрицю доходів за ріки та місяці.

3.8 Використання інструментів та технологій

Інструменти. Під час розробки системи застосовались наступні інструменти:

- **Visual Studio 2022** – використовувався в якості IDE;
- **Visual Studio 2019** – використовувався для розробки Analysis Service на базі сховища даних, так, як відповідний функціонал не доступний в Visual Studio 2022;
- **Visual Code** – використовувався в якості IDE;
- **SQL Server Management Studio** – використався для роботи з об'єктами розміщеними на SQL Server;
- **Azure Data Studio** – застосувався для роботи з даними;
- **Docker** – використовувався для розгортання вузлів;
- **Git** – використовувався для управління версіями;
- **Postman** – використовувався для тестування API.

Найбільш важливою складовою є вибір програмної платформи. Беручи до увагу особливості системи в якості платформ розглядались: C#, Python та Java, які на цей момент є найбільш релевантними для систем такого типу.

Після проведеного аналізу, що представлений в таблиці 3.5 було обрано платформу .NET 6, що підтримує синтаксис C#. Окрім розглянутих аспектів також враховувався набір інструментів та технологій, що підтримують цю платформу.

Таблиця 3.5

Аналіз платформи

Платформа	**C# .NET 6**	**JAVA**	**Python**
OS	Windows, Linux	Windows, Linux	Windows, Linux
Web Framework	ASP.NET Core, Blazor	Angular, Apache, Wicket, Grails	Flask, Django, web2Py, Bottle
Швидкодія	Висока	Середня	Низька
Безпека	Висока	Висока	Помірний
Підтримка баз даних	Стабільна	Стабільна	Порівняно слабка
Підтримка	Висока	Середня	Висока
Підтримка технологій	Висока	Висока	Висока
Рівень обізнаності	Високий	Низький	Середній

В якості основного засобу для створення веб-додатків було обрано ASP.Net Core, яка дозволяє створювати, як клієнтські додатки, так і API сервіси. Іншими перевагами цієї технології є: кросплатформеність, висока швидкість та надійність, підтримка Razor Page.

REST API. Це архітектурний стиль (рис. 30), який визначає механізму обміну повідомленнями через http або https з'єднання, саме він був покладений при побудові сервісів.

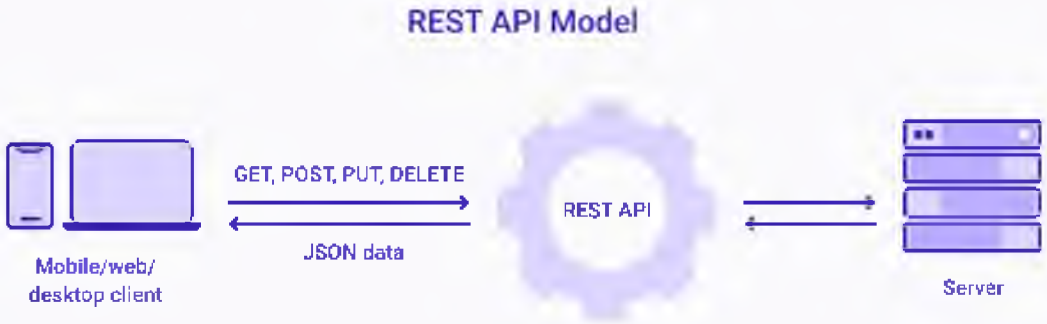


Рисунок 30. Модель Rest API

Відповідно до цього архітектурного стилю служба повинна відповідати таким принципам:

- чітку розмежування клієнта та сервера, де на стороні клієнта має бути інтерфейс користувача, а на стороні серверу логіка доступу до даних та оброблення запиту;
- єдиний та уніфікований набір інтерфейсів (табл. 6), що відповідають CRUD;
- сервер не зберігає стан операцій, що вимагає від клієнта відправлення всіх даних в одному запиту;
- веб-хешування завжди дозволено;
- дозволена багаторівнева архітектура;
- хоча зазвичай відповідь містить json існує можливість відправити виконуваний код.

Кожен запит, який відбувається в рамках цього архітектурного стилю має свій метод. Основні методи перераховані в табл. 3.6, але вони не обмежуються лише цим списком.

Таблиця 3.6

Основні методи REST API

Інтерфейс	Опис
GET	Запит даних
POST	Створення нових даних
PUT	Обновлення даних
DELETE	Видалення даних

Окрема транзакція роботи з такою веб-службою містить наступні компоненти:

- метод HTTP - інтерфейс взаємодії;
- адреса запиту - шлях до ресурсу;
- тіло запиту - дані, що прикріплені до запиту;
- код відповіді, який дозволяє зрозуміти клієнту тип відповіді;
- тіло відповіді – отримані дані, найбільш поширений формат передачі даних це JSON.

Цей спосіб побудови API служби має велику кількість переваг: надійність, масштабування, швидкість, простота та прозорість. Але також є вагомий недолік – не передбачається захист, що вимагає від розробника самостійного вирішення цієї проблеми.

Взаємодія між службами через Event Bus. Кожна з розглянутих служб реалізовує свою задачу і є автономною, але для повноцінної роботи системи існує необхідність в обміну повідомленню між ними.

Хоча служби розділені за предметною областю і є невжаленими, інколи неможливо повністю відділити їх одну від іншої, що допускає стикання їх предметних областей в деяких аспектах. В цьому випадку може існувати

необхідність повідомить одну служба про зміну в іншій, але при цьому необхідно пам'ятати про принципи проектування мікро служб, що забороняють пряму прив'язку однієї служби до іншої.

Цю проблему можна вирішити за допомогою використання технології Event Bus (рис. 31), яка заснована на ООП підходу «Спостерігач».

Рисунок 31. Технологія Event Bus

Принцип цього підходу полягає в публікації повідомлень одними службами, та підпискою на них іншими службами. У випадку коли одна служба публікує повідомлення то у службах, які підписані на цю подію викликається обробник.

Технічного, це відбувається за допомогою спеціалізованого сховища даних, яке може приймати повідомлення від відправника, та поширювати його серед підписників.

Реляційна база даних. Для частина сервісів в якості бази даних використовується СУБД, на роль яких розглядались такі бази даних:

- MySQL;
- Microsoft SQL Server;
- PostgreSQL;
- Oracle;

Щодо інших реляційних баз даних був проведений порівняльний аналіз (табл. 3.7) за важливими особливостями, які є необхідними для розробленої системи, що дозволило визначити найбільш оптимальну базу даних для системи.

За результатами цього аналізу з реляційних СУБД було обрано MS SQL Server 2019. Також, вона містить ряд вагомих переваг, а саме :

- підтримка ряду технологій, які здатні розширити можливості SQL, наприклад власне розширення синтаксису SQL - Transact-SQL;
- підтримка інтелектуальної обробки транзакцій;
- гарно задокументована;
- гарна інтеграція з продуктами Microsoft;
- має підтримку декількох платформ, що полегшує розгортання.

Таблиця 3.7

Порівняння реляційних баз даних

СУБД	Висока швидкість роботи	Високий рівень безпеки	Доступність	Замітка
MySQL	Ні	Ні	Так	Гарний варіант для невеликих проектів
Microsoft SQL Server	Так	Так	Так	Найкращий варіант
Oracle	Так	Так	Ні	Має приблизно такі самі можливості, як MS SQL Server
PostgreSQL	Ні	Ні	Так	Уступає іншим базах даних, але має підтримку об’єктів

Для роботи з базою даних було вибраного ORM рішення - Entity Framework Core, що дозволяє абстрагуватися від обраної бази даних, представити набір даних, як ООП об'єкти.

Цей підхід дозволяє спростити процес розробки й тестування та дотримуватися єдиного стилю написання програми. ORM технологія була обрано у зв'язку з такими перевагами:

- має гарну підтримку реляційної СУБД MS SQL Server;
- має гарну документацію та підтримку спільноти;
- є досить потужною і має такі можливості, як трансляцію виклику методів у SQL запит, що дозволяє не скачувати лишніх даних і є більш ефективним рішенням з точки зору швидкості роботи, так, як в даному випадку основну роботу виконує MS SQL Server, а також підтримку використання транзакцій.

З'єднання з базою даних за допомогою цього підходу складається з двох етапів, що дозволяють спочатку описати дані в сховищу даних, а потім налаштувати підключення.

Спершу необхідно для кожної сутності, з якою система буде працювати створити клас (рис. 32) що буде описувати, яку інформацію включає сутність й якого вона типу.

```
public class Tag : Entity<Guid>
{
    public string Name { get; set; } = null!;
    public string? Description { get; set; }

    [JsonIgnore]
    public ICollection<Book> Books { get; set; } = new HashSet<Book>();

    protected Tag() { }

    public Tag(string name, string? description = null)
    {
        Id = Guid.NewGuid();
        Name = name;
        Description = description;
    }
}
```

Рисунок 32. Модель "Tag"

Після цього необхідно описати спеціальний клас, що називається контекст і відповідає за зв'язок з базою даних. Якщо на попередньому етапі описувались сутності, які входять до бази даних, то на цьому описується вже сама база даних. На рис. 33 представлено, як було описано підключення до бази даних.

```
public class CatalogContext : DbContext
{
    public const string DEFAULT_SCHEME = "catalog";

    internal DbSet<Author> Authors { get; set; } = null!;
    internal DbSet<Genre> Genres { get; set; } = null!;
    internal DbSet<Tag> Tags { get; set; } = null!;
    internal DbSet<Book> Books { get; set; } = null!;
    internal DbSet<BookFile> BookFiles { get; set; } = null!;

    public CatalogContext(DbContextOptions<CatalogContext> options) : base(options)
    {
        //Database.EnsureDeleted();
        Database.EnsureCreated();
    }

    protected override void OnModelCreating(ModelBuilder builder)
    {
        builder.HasDefaultSchema(DEFAULT_SCHEME);

        builder.ApplyConfiguration(new TagEntityConfiguration());
        builder.ApplyConfiguration(new GenreEntityConfiguration());
        builder.ApplyConfiguration(new AuthorEntityConfiguration());
        builder.ApplyConfiguration(new BookFileEntityConfiguration());
        builder.ApplyConfiguration(new BookEntityConfiguration());
    }
}
```

Рисунок 33. Опис бази даних

Після цього за допомогою створеного контексту можливо працювати з базою даних, як з будь-яким іншим класом.

На рис. 34 представлено, як саме реалізоване додавання нової книги в базу даних за допомогою дієї технології. Видно, що взаємодія з базою даних проходить повністю непомітно для розробника. Технологія автоматично перетворює LINQ запити в формат SQL, а результати приводить до необхідного класу.

```
public IEnumerable<BookFile> GetFiles(string bookIsbn)
{
    var book = context
        .Books
        .Include(x => x.BookFiles)
        .FirstOrDefault(x => x.Id == bookIsbn);

    return book?.BookFiles ?? Enumerable.Empty<BookFile>();
}

public Tag? GetTag(Guid id) => context.Tags.FirstOrDefault(x => x.Id == id);
public Genre? GetGenre(int id) => context.Genres.FirstOrDefault(x => x.Id == id);
public Author? GetAuthor(Guid id) => context.Authors.FirstOrDefault(x => x.Id == id);
public BookFile? GetBookFile(Guid id) => context.BookFiles.FirstOrDefault(x => x.Id == id);
public Book? GetBook(string isbn)
{
    var book = context
        .Books
        .Include(x => x.Genre)
        .Include(x => x.Tags)
        .Include(x => x.Authors)
        .Include(x => x.BookFiles)
        .FirstOrDefault(x => x.Id == isbn);

    return book;
}
```

Рисунок 3.4. Методи для отримання даних

Графова база даних. Під час розробки сервісу рейтингу було використано графову базу даних, і-за її особливостей, які допомогли збільшити ефективність цього сервісу.

База даних графів — це система баз даних типу NoSQL, заснована на структурі топографічної мережі. Ідея походить від теорії графів у математиці, де графи представляють набори даних за допомогою вузлів, ребр та властивостей.

Основними перевагами є:

- гнучкість;
- надзвичайно ефективна реалізація складних відношень між сутностями;
- масштабованість запитів.

В якості можливих баз даних розглядались:

- Neo4j
- Cosmos Db;
- SQL Server (підтримка лише на нових версіях).

- Amazon Neptune;
- Arango DB.

В кінці було вирішено зупинитися на Neo4j у зв'язку з наступними перевагами:

- це є повноцінною графовою базою даних, а не реалізацією поверх іншої бази даних, яка імітує графи, як наприклад робить SQL Server;
- дана база даних має свою власну мову запитів – Cypher, яка є дуже лаконічною і дозволяє легко працювати з нею;
- дана база даних містить можливості, які не мають інші, наприклад створення індексів та тригерів;
- ефективне виконання запитів;
- існує можливість розгорнути безкоштовно.

Для взаємодії з цією базою даних було використано офіційний драйвер, який дозволяє виконувати операції за допомогою мови запитів Cypher (рис. 35).

```
public async IAsyncEnumerable<Customer> GetCrossCustomers(string customerId)
{
    var query = @"
    MATCH (c1:Customer) - [:Grade] -> (:Book) <-[:Grade] - (c2:Customer)
    WHERE c1.CustomerId = $customerId
    RETURN c2";

    var parameters = new
    {
        customerId
    };

    var ssession = _driver.AsyncSession();

    try
    {
        var readTransaction = await ssession.ExecuteReadAsync(async rx =>
        {
            var result = await rx.RunAsync(query, parameters);
            return await result.ToListAsync();
        });

        return readTransaction.Select(item =>
        {
            var customerNode = item["c2"].As<INode>();

            return new Customer
            {
                Id = customerNode.Properties["CustomerId"].As<string>()
            };
        });
    }
    catch(Neo4jException exception)
    {
        throw exception;
    }
```

Рисунок 35. Пошук схожих користувачів

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Вузли системи

Було побудовано діаграму розгортання (рис. 36), що показує вузли, які потрібні для повноцінної роботи системи [24]:

- клієнтський пристрій;
- пристрій аналітика;
- SQL Server 2019: OLTP та Сховище даних;
- RabbitMQ Database;
- Neo4j Database;
- Сервіси ідентифікації, каталогу, замовлень рейтингу та клієнтський.

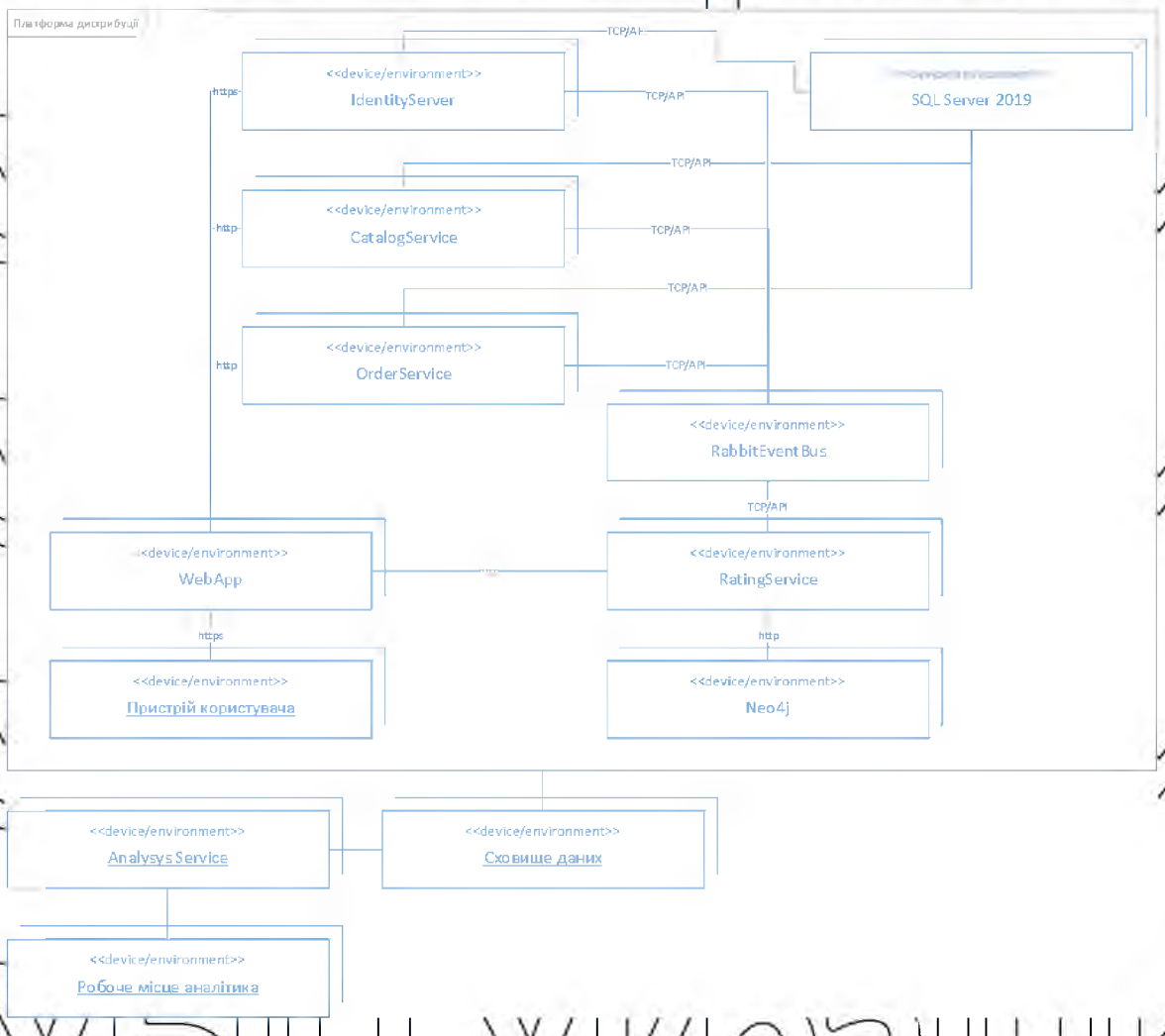


Рисунок 36. Діаграма розгортання

4.2 Результати дослідження

Для створення системи рекомендацій (рис. 37) на основі спільних фільтрах було запропоновано два рішення для покращення ефективності, а саме: використання бази даних на основі графів для описує взаємозв'язків та більш досконалий метод пошуку коефіцієнта суміжності для спільного фільтру.

Рисунок 37. Отримання рекомендації

В рамках розробки складової сервісу рейтингу, а саме спільного фільтру, який відповідає за формування рекомендацій користувачу на основі його відгуків. Для цієї розробки було запропоновано використання формулу

покращеної подібності трикутників доповнену налаштуваннями оцінки користувача, як функції суміжності.

Для визначення ефективності цього рішення було проведено дослідження, в якому було порівняно цей метод з іншими функціями, які найбільш часто використовуються, а саме: евклідова відстань, кореляція Пірсона, косинус подібності. Результати дослідження представлена в таблиці 4.1.

Перевірка відбувалась за допомогою тесту на:

- RMSE – кореневе середньоквадратичне відхилення;
- MAE – середня абсолютна похибка;
- R^2 - коефіцієнт детермінації.

Таблиця 4.1

Порівняння функцій суміжності

Метод	Min RMSE	MAE	Max R^2
Евклідова відстань	1.1529	0.8787	0.0069
Кореляція Пірсона	1.0217	0.8121	0.1780
Косинус подібності	1.0568	0.8028	0.1759
ITR	**0.9891**	**0.7753**	**0.2371**

З результатів порівняння можна побачити, що обраний метод дійсно дозволяє помітно покращити ефективність спільного фільтру у порівняні з іншими розглянутими методами.

Якщо говорить про ефективність впровадження бази даних на основі графів замість реляційної бази даних, то покращення є, але не великі, що зв’язано

з тим, з невеликою максимальною глибиною операцій. Хоча приріст невеликий, але можна говорить, що при збільшені кількості зав'язків та глибини операцій приріст буде збільшуватися.

Якщо говорить про систему аналітики, то було реалізовано рішення, яке надає широкі можливості для аналітика та функцію пошуку рекомендації на основі вмісту.

Була реалізована можливість за допомогою графічного інтерфейсу OLAP системи формувати звіт (рис. 38), який формується дуже швидко, що дозволяє аналітику легко отримувати необхідну йому аналітичну інформацію. Підтримує режим роботи в Excel.

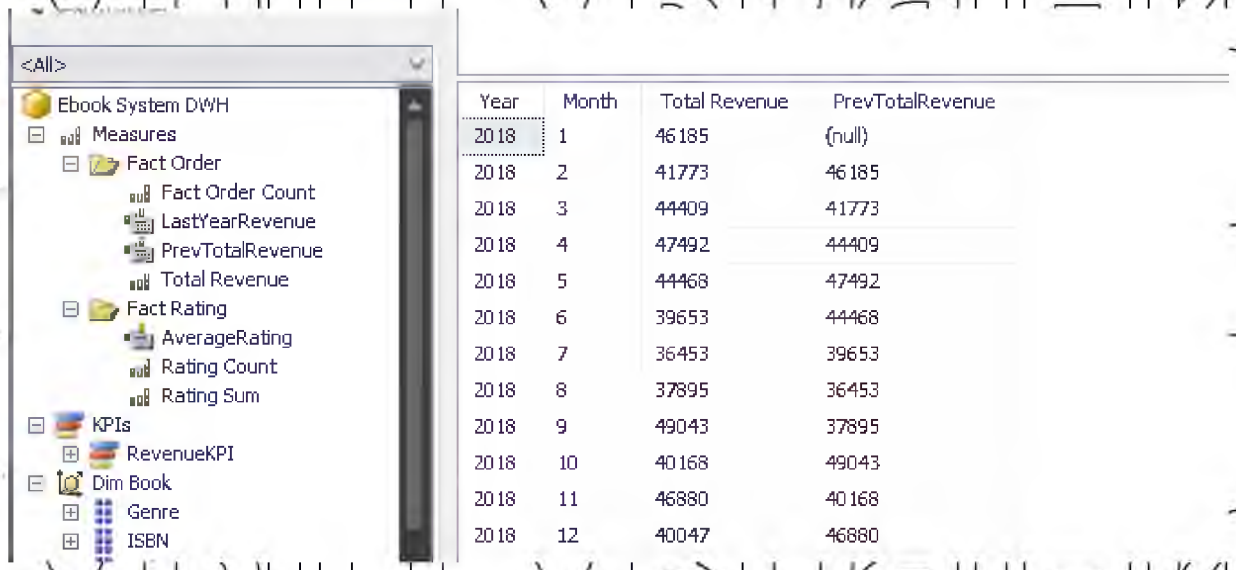


Рисунок 38. OLAP куб - Звіт

Була розроблена можливість перевірки успішності діяльності через розробку KPI та формування відповідного звіту (рис. 39).

KPI означає ключовий показник ефективності, кількісну міру продуктивності протягом певного часу для певної цілі. Ключові показники ефективності містять цілі, до яких мають прагнути команди, віхи для оцінки прогресу та інформацію, яка допомагає людям у всій організації приймати кращі рішення.

Row Labels	RevenueKPI	RevenueKPI Goal	RevenueKPI Status	RevenueKPI Trend
2018				
1	46185			
2	41773			
3	44409			
4	47492			
5	44468			
6	39653			
7	36453			
8	37895			
9	49043			
10	40168			
11	46880			
12	40047			
2019				
1	50758	46185		
2	38818	41773		
3	40683	44409		
4	38600	47492		
5	35473	44468		
6	39597	39653		
7	42329	36453		
8	28991	37895		
9	46663	49043		
10	35431	40168		
11	35307	46880		
12	42548	40047		
2020	483693	475198		
2021	497937	483693		
2022	516166	497937		
Grand Total	2487460			

Рисунок 39. KPI

За допомогою моделі кластеризації було створено інструмент, який дозволяє знаходить кластери користувачів (рис. 40).

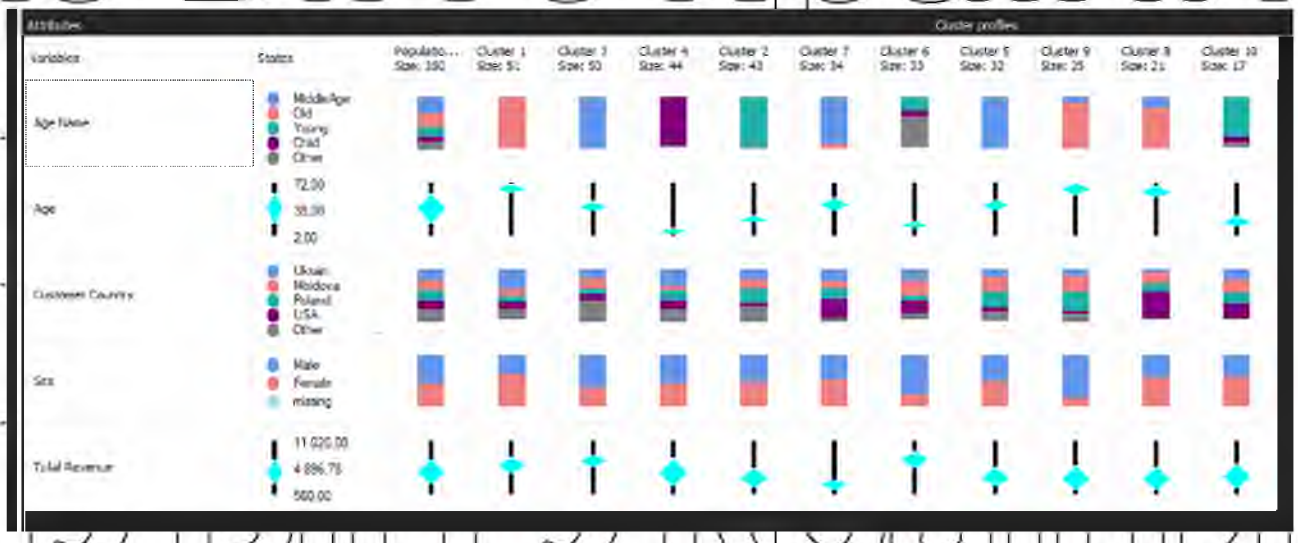


Рисунок 40. Модель кластеризації - Результат

За допомогою моделі пошуку асоціативних правил було реалізовано інструмент (рис. 41), який дозволяє зрозуміти, яка книга сподобається користувачу.

Pr...	Importance	Rule
1,000	0,292	Genre = Detective, Age Name = Child -> Opinion = Not liked
1,000	0,238	Genre = Education, Age Name = Teenager -> Opinion = Not liked
1,000	0,301	Genre = Education, Age Name = Child -> Opinion = Not liked
1,000	0,303	Genre = Drama, Age Name = Child -> Opinion = Not liked
1,000	0,345	Genre = Drama, Sex = Male -> Opinion = Not liked
1,000	0,306	Genre = Crime, Age Name = Child -> Opinion = Not liked
0,952	0,282	Genre = Roman, Age Name = Child -> Opinion = Not liked
0,938	0,269	Genre = Poem, Age Name = Child -> Opinion = Not liked
0,929	0,233	Genre = Adventure, Age Name = Child -> Opinion = Liked
0,929	0,233	Age Name = Teenager, Genre = Fantasy -> Opinion = Liked
0,923	0,257	Genre = Education, Age Name = Young -> Opinion = Not liked
0,889	0,221	Genre = Fantastic, Age Name = Child -> Opinion = Liked
0,889	0,258	Genre = Drama, Customer Country = Moldova -> Opinion = Not liked
0,889	0,221	Genre = Fantasy, Age Name = Child -> Opinion = Liked
0,870	0,216	Genre = Fantastic, Customer Country = England -> Opinion = Liked
0,857	0,218	Genre = Classic, Age Name = Old -> Opinion = Liked
0,851	0,257	Genre = Crime, Sex = Female -> Opinion = Not liked
0,846	0,193	Age Name = Teenager, Genre = Fantastic -> Opinion = Liked

Рисунок 41. Модель асоціативних правил – Результати

За допомогою вбудовано інструменту для перевірки точності було перевірено точність цієї моделі. З результатів цієї перевірки (рис. 42), можна побачить, що хоча точність не ідеальна, але достатньо непогана. Загальна точність становить близько 67 відсотків.

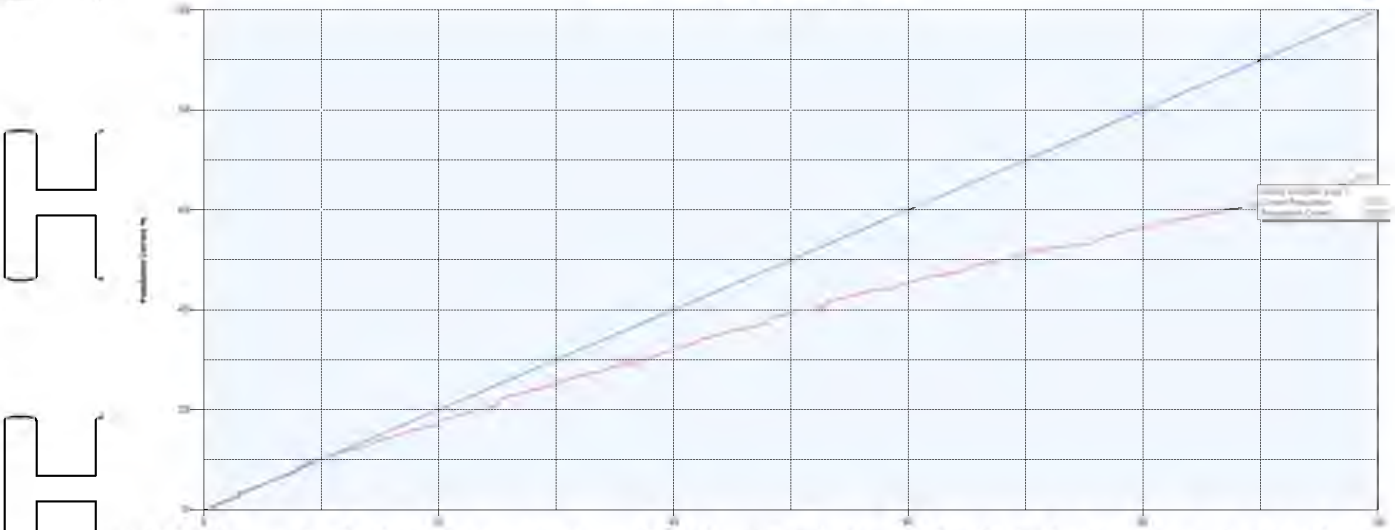
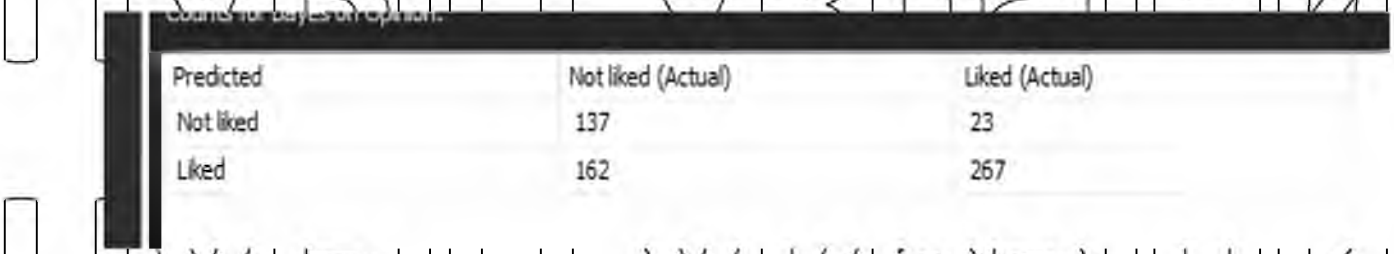


Рисунок 42. Модель асоціативних правил – Точність

За допомогою моделі наївного байєсу(рис. 43) було реалізовано інструмент, який дозволяє зрозуміти, яка книга сподобається користувачу.

Рисунок 43. Модель наївного байєсу – Результат

За допомогою вбудовано інструменту для перевірки точності було перевірено точність цієї моделі. З результатів цієї перевірки (рис. 42), можна побачить, що було даний правильний прогноз для 404 з 589 тестів, що становить приблизно 68.5 відсотка. Даний результат також є не ідеальним, але досить гарним і на 1.5 відсотка кращий ніж дає метод асоціативних правил.



Predicted	Not liked (Actual)	Liked (Actual)
Not liked	137	23
Liked	162	267

Рисунок 44. Модель наївного байєсу – Перевірка

Також було створено звіт по популярності жанрів(рис. 45). Цей звіт охоплює середній бал кожного жанру з груп користувачів, які згруповані за віковою групою та статтю.

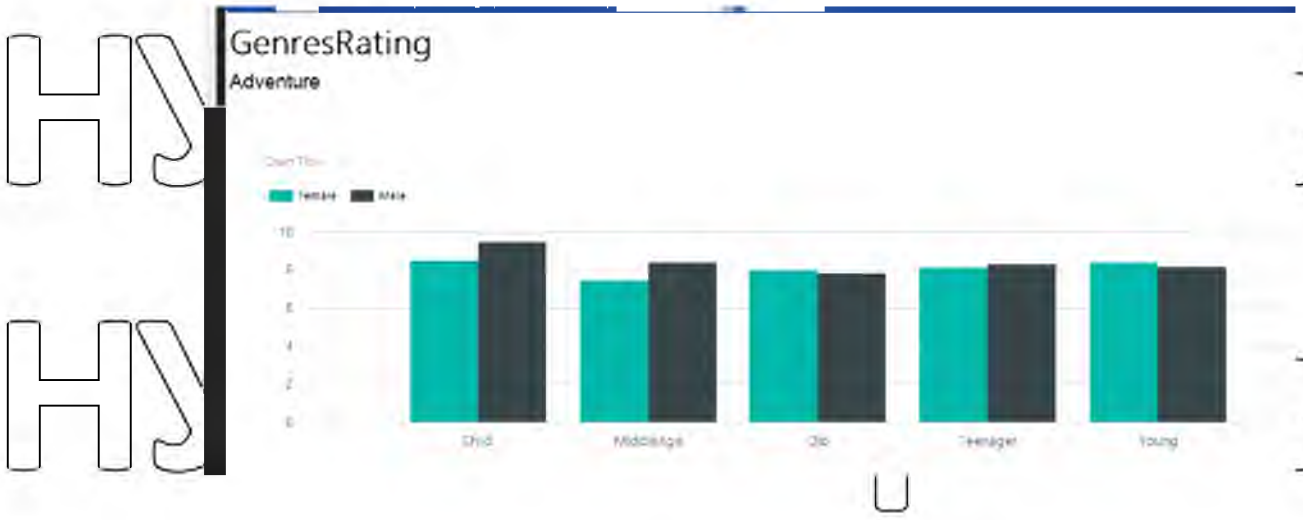


Рисунок 45. Звіт по популярності жанрів

Також було створено звіт по прибутку в проміжку часу(рис. 46). Даний звіт відображає у формату матриці доходи отримані від продажу електронних копій книжок в проміжку часу рік та місяць.

RevenueReport

	2018	2019	2020	2021	2022
1	46185	50758	37279	39492	35970
10	40168	35431	41915	30076	44984
11	46880	35307	41254	38082	35525
12	40047	42548	45482	42176	46376
2	41773	38818	36182	30850	52918
3	44409	40683	32417	44097	41314
4	47492	38600	44420	40725	41060
5	44468	35473	41370	46094	49603
6	39653	39597	38299	49140	37935
7	36453	42329	42473	42009	43330
8	37895	28991	33383	50400	47506
9	49043	46663	49219	44796	39645

Рисунок 46. Звіт на доходами у проміжку часу

ВИСНОВКИ

В ході роботи та дослідження інтелектуальної системи формування пропозицій читачеві на основі власних вподобань було проведено такі роботи:

1. проведено аналіз предметної області цифрової дистрибуції книжок, проаналізовано наявні рішення та поставлені подальші цілі;
2. проведення моделювання системи;
3. проведено проектування та розробку системи відповідно до поставлених цілей та етапу моделювання, описано використані технології та інструменти;
4. надано остаточний опис фізичних вузлів системи, що необхідні для розгортання системи;
5. продемонстровано результати та перевірено їх ефективність за допомогою тестових методів та інструментів.

Для досягнення поставленої мети було використано два підходи:

Перший, це підхід на основі сумісної фільтрації, який дозволяє створювати персональні рекомендацій на основі історії взаємодій. В рамках цієї розробки було запропоновано більш досконалу формулу схожості, яка є ключовою для цього методу, а саме ITR, а також заміну бази реляційної бази даних на базу даних на основі графів. Обидва рішення показали свою ефективність та доцільність.

Ця система рекомендації дозволила пропонувати користувачу книги на основі його минулих книжних відгуків, а також на основі взаємодій користувачів, що мають схожий досвід взаємодії.

Другий, основується на пошуках зав'язків між даними користувачів та націлена на створення моделі, яка охоплює кластер користувачів і може роботи читачам з цього кластера пропозиції.

Для цього, було розроблено систему аналізу на основі сховища даних та OLAP кубу, яка надає широкі можливості аналітику по роботі з аналітичними даними, а також реалізовує систему рекомендацій на основі вмісту за допомогою моделей Data Mining. Розроблена система аналізу показала досить непогані результати.

Розробка цієї системи аналізу окрім надання аналітику потужного інструменту для отримання аналітичних даних дозволила виділити кластери серед користувачів за їх схожими рисами, а також за допомогою методів Data Mining(метод асоціативних правил та метод наївного байєса) було розроблено інструмент, який дозволяє з певною точністю говорити, чи може сподобатися книжка користувачу на основі такої інформацію про нього, як стать, вік та тощо.

Провівши дослідження цих методів було виявлено не лише, що вони є ефективними для досягнення поставлених цілей, а й взаємодоповнюють друг друга. Так, метод на основі вмісту дозволяє вирішити проблему холодного запуску, коли спільна фільтрація і-за відсутності взаємодій нічого не може запропонувати, а в майбутньому коли ці дані накопичиться буде можливість формувати пропозиції, які будуть будувати на персональних вподобаннях конкретного користувача, а не всього його кластера.

Отже, підводячи підсумки можна говорити, що вдалось досягнути поставленних задач та мети, але ще залишається досить місця для полішшення, а саме: накопичення більшої кількості даних, збільшення точності методів, збільшення їх інтеграції, створення нових гібридних методів та тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] S. Ross, «Benefits of digital distribution,» CCN, 12 09 2019. [Онлайновий]. Available: https://www.climatecontrolnews.com.au/opinion/benefits-of-digital-distribution. [Дата звернення: 24 04 2021].

[2] S. Kilpatrick, «Ecommerce Payment Processing 101: A Beginner's Guide for Small Businesses,» BigCommerce, 2021. [Онлайновий]. Available: https://num.org/16K. [Дата звернення: 20 03 2021].

[3] L. Rosencrance, «authentication,» SearchSecurity, [Онлайновий]. Available: https://searchsecurity.techtarget.com/definition/authentication. [Дата звернення: 5 02 2021].

[4] P. Vonderau, «The Spotify Effect: Digital Distribution and Financial Growth,» 01 01 2019. [Онлайновий]. Available: https://journals.sagepub.com/doi/full/10.1177/1527476417741200. [Дата звернення: 13 04 2021].

[5] M. B. Rajshree та S. Malik, «Recommendation System: Techniques and Issues,» 10 9 2019. [Онлайновий]. Available: https://www.researchgate.net/publication/338547981_Recommendation_System_Techniques_and_Issues.

[6] H. Liu, Z. Hu, A. Mian, H. Tian та X. Zhu, «A new user similarity model to improve the accuracy of collaborative filtering,» *Knowledge-Based Systems*, т. 56, pp. 156-166, 2014.

[7] UML, «WHAT IS UML,» Object Managment Group, 5 07 2005. [Онлайновий]. Available: https://www.uml.org/what-is-uml.htm. [Дата звернення: 23 03 2021].

[8] Visual Paradigm, «Use Case Analysis: How to Identify Actors?,» Visual Paradigm, [Онлайновий]. Available: https://numl.org/I6M. [Дата звернення: 13 03 2021].

[9] Visual Paradigm, «What is Activity Diagram?,» Visual Paradigm, [Онлайновий]. Available: https://numl.org/I6O. [Дата звернення: 10 02 2021].

[10] O. Elgabry, «Object-Oriented Analysis And Design — Interaction Models (Part 4),» Medium, 19 03 2017. [Онлайновий]. Available: https://numl.org/I6N. [Дата звернення: 02 03 2021].

[11] N-iX, «Microservices vs Monolith: which architecture is the best choice for your business?,» Romana Gnaty, 03 10 2018. [Онлайновий]. Available: https://numl.org/I6T. [Дата звернення: 14 04 2021].

[12] IBM, «Challenges and benefits of the microservice architectural style,» IBM Developer, 30 01 2019. [Онлайновий]. Available: https://cutt.ly/zvYmRcO. [Дата звернення: 02 03 2021].

[13] R. D. Hernandez, «The Model View Controller Pattern – MVC Architecture and Frameworks Explained,» freeCodeCamp, 19 04 2021. [Онлайновий]. Available: https://numl.org/I6Y. [Дата звернення: 01 05 2021].

[14] OAuth 2.0, «OAuth 2.0,» OAuth 2.0, [Онлайновий]. Available: https://oauth.net/2/. [Дата звернення: 23 01 2021].

[15] Microsoft, «Description of the database normalization basics,» Microsoft, 17 05 2021. [Онлайновий]. Available: https://numl.org/I6P. [Дата звернення: 10 02 2021].

[16] F. Fkih, «Similarity measures for Collaborative Filtering-based Recommender Systems: Review and experimental comparison,» *Journal of King Saud University - Computer and Information Sciences*, т. 34, № 9, pp. 7645-7669, 2022.

[17] Oracle, «Graph Database Defined,» Oracle, [Онлайновий]. Available: https://www.oracle.com/autonomous-database/what-is-graph-database/. [Дата звернення: 15 10 2022].

[18] Microsoft, «Overview of Service Manger OLAP cubes for advanced analytics,» Microsoft, 21 08 2020. [Онлайновий]. Available: shorturl.at/ntxKQ. [Дата звернення: 15 12 2021].

[19] Microsoft, «Analysis Services documentation,» Microsoft, [Онлайновий]. Available: shorturl.at/hwMS1. [Дата звернення: 15 12 2021].

[20] Microsoft, «Key Performance Indicators (KPIs) in Multidimensional Models,» Microsoft, 02 09 2021. [Онлайновий]. Available: shorturl.at/gloPV. [Дата звернення: 15 12 2021].

[21] Microsoft, «Data Mining (SSAS),» Microsoft, 04 01 2022. [Онлайновий]. Available: https://cutt.ly/UGU014X. [Дата звернення: 20 04 2022].

[22] R. Sharma, «Cluster Analysis in Data Mining: Applications, Methods & Requirements,» UpGrad, 20 06 2022. [Онлайновий]. Available:

https://www.upgrad.com/blog/cluster-analysis-data-mining/. [Дата звернення: 20 04 2022].

[23] R. Sharma, «Classification in Data Mining Explained: Types, Classifiers & Applications,» UpGrad, 4 06 2021. [Онлайновий]. Available: https://www.upgrad.com/blog/classification-in-data-mining/. [Дата звернення: 20 04 2022].

[24] Visual Paradigm, «What is Deployment Diagram?,» Visual Paradigm, [Онлайновий]. Available: https://numl.org/I7a. [Дата звернення: 08 03 2021].