Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Кафедра інформатики факультету інформатики

Кваліфікаційна робота

освітній ступінь - бакалавр

на тему: **«Розробка рекомендаційної системи для веб-застосунку (маркетплейсу) надання послуг»**

за спеціальністю «Інженерія програмного забезпечення» - 121

Керівник кваліфікаційної роботи

Доцент, кандидат наук

Олецький Олексій Віталійович

_____/підпис/

“_____” _____ 2023 р.

Виконав студент ІПЗ-4:

Ковриженко Владислав Євгенійович

_____/підпис/

“_____” _____ 2023 р.

Київ 2023

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

_____

_____

(підпис) “____”

_____ 2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студенту Ковриженку Владиславу Євгенійовичу факультету інформатики 4-го курсу ТЕМА: Розробка рекомендаційних систем для веб-застосунку (маркетплейсу) надання послуг. Вихідні дані: Аналіз особливостей реалізації маркетплейсів та рекомендаційних систем. Розробка веб-застосунку(маркетплейс) та рекомендаційної системи до нього.
Зміст ТЧ до кваліфікаційної роботи:
Індивідуальне завдання

Вступ

Розділ 1: Визначення маркетплейсу та його типи

Розділ 2: Розробка маркетплейсу

Розділ 3: Вступ до рекомендаційних систем

Розділ 4: Практична частина розробки маркетплейсу

Розділ 5: Практична частина розробки рекомендаційної системи

Розділ 6: Приклад роботи маркетплейсу та рекомендаційної системи

Висновки

Список літератури

Дата видачі „___” _____ 2023 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Тема: Розробка рекомендаційної системи для веб-застосунку (маркетплейсу) надання послуг.

Календарний план виконання роботи:

№	Назва етапу	Термін виконання	Примітка
1.	Отримання теми кваліфікаційної роботи	15.07.2022	
2.	Пошук літератури за темою роботи	20.01.2023	
3.	Ознайомлення з науковою літературою	21.02.2023	
4.	Ознайомлення з особливостями побудови рекомендаційних систем	10.03.2023	
5.	Проектування структури маркетплейсу	13.03.2023	
6.	Розробка маркетплейсу	05.04.2023	
7.	Написання першої частини кваліфікаційної роботи	7.04.2023	
8.	Написання другої частини кваліфікаційної роботи	10.04.2023	
9.	Написання третьої частини кваліфікаційної роботи	11.04.2023	
10.	Написання четвертої частини кваліфікаційної роботи	15.04.2023	
11.	Написання п’ятої частини кваліфікаційної роботи	16.04.2023	
12.	Внесення змін до роботи відповідно до коментарів керівника	01.05.2023	
13.	Створення презентації	08.05.2023	
14.	Подача кваліфікаційної роботи на перевірку	24.05.2023	

Зміст

Вступ

Заголовок: Розробка рекомендаційної системи для веб-застосунку (маркетплейсу) надання послуг.

Актуальність теми: Маркетплейси здатні забезпечити широкий вибір товарів та послуг, але зростаюча кількість товарів може створювати складнощі для користувачів при пошуку необхідного контенту, тому рекомендаційні системи є важливою частиною будь якого сервісу. Вони надають персоналізовані рекомендації користувачам, що робить маркетплейс більш зручним для користувачів, а маркетплейси здійснюють більше продажів.

Мета дослідження: Метою дослідження, є розробка рекомендаційних систем для маркетплейсів з метою покращення користувацького досвіду, збільшення залученості та зростання продажів.

Об'єкт, предмет: Об'єкта дослідження - маркетплейс, та предмет дослідження - рекомендаційні системи. Рекомендаційні системи напряму залежать від сервісу в якому вони працюють, тому при розробці використання різних типів рекомендаційних систем залежатиме від типу маркетплейсу.

Структура роботи: Розділ 1 та 2 - класифікація та виділення базового функціоналу маркетплейсу, розділ 3 - огляд рекомендаційних систем, 4 та 5 розділ опис практичної частини кваліфікаційної роботи, 6 розділ приклади використання застосунку.

Методологія дослідження: Для полегшення подальшої розробки як маркетплейсів, так і рекомендаційних систем використовувалися різні методи дослідження. Метод класифікації, за допомогою якого я розподілив маркетплейси та рекомендаційні системи за певними критеріями, що в свою чергу допомагає з визначенням функціональних вимог для рекомендаційних систем та маркетплейсів. Аналіз і аналітика – було проаналізовано вже існуючі реалізації маркетплейсів та супутніх їх рекомендаційних систем.

Практична цінність: Моє дослідження спрямоване на вирішення нагальної проблеми в індустрії маркетплейсів. І хоча ця тема не є науковою інновацією, але розробка рекомендаційної системи, яка поєднує різні способи аналізу даних користувачів для надання персоналізованих рекомендацій, є важливим елементом будь якого новітнього сервісу. А в рамках роботи маркетплейсів покращує користувацький досвід та призводить до кращої конверсії продажів. Також робота має практичну цінність, адже в ній розписано різноманітні види рекомендаційних систем та типи рекомендацій, що вони генерують. Також описано методологію створення маркетплейсу як веб-застосунку, його можливі модифікації та зміни в залежності від предметної області.

Очікувані результати: Створення опису структурованих знань про маркетплейси та рекомендаційні системи що, може стати відправною точкою для створення нових маркетплейсів.

1. Визначення маркетплейсу та його типи

Маркетплейс (англ. marketplace) - це електронна платформа, на якій продавці можуть представляти свої товари або послуги, а покупці можуть здійснювати покупки. У маркетплейсі можуть бути присутні як великі роздрібні мережі, так і невеликі інтернет-магазини, а також приватні продавці. З точки зору електронної комерції маркетплейси можна розділити на категорії в залежності від того, які користувачі взаємодіють між собою, та які категорії товарів чи(та) послуг пропонують маркетплейси[1]:

- Вертикальний маркетплейс – це платформа, яка спеціалізується на продажу товарів або послуг у певній ніші чи категорії, наприклад, електроніка, мода, косметика тощо. Вертикальний маркетплейс призначений для задоволення конкретних потреб аудиторії та пропонує великий вибір товарів у певній категорії.
- Горизонтальний маркетплейс – це платформа, яка пропонує широкий вибір товарів і послуг у різних категоріях, таких як електроніка, мода, спорт, здоров’я, косметика тощо. Горизонтальні ринки покликані максимально задовольнити потреби різних категорій аудиторії.
- Глобальний маркетплейс – це онлайн-платформа, яка об'єднує продавців і покупців з різних країн світу. Це дозволяє продавцям розширити свій бізнес на міжнародні ринки та залучити нових клієнтів з інших країн, а також покупців - для різноманітних товарів і послуг з усього світу.
- C2C – у цій моделі одна людина продає товари безпосередньо іншій, хоча сама їх не виробляє.
- B2C – у цій моделі компанія надає свої товари, послуги та послуги безпосередньо людині.
- B2B – у цій моделі одна компанія надає свої послуги або товари іншій фірмі, а не рядовим споживачам. Тут важливо, щоб відносини між сторонами були строго регламентовані договором.

Приклади маркетплейсів:

- Продуктові маркетплейси: Це маркетплейси, де продавці продають різні товари, від одягу та електроніки до продуктів харчування та косметики. Найвідомішими продуктовими маркетплейсами є Amazon, eBay та Alibaba, вони є горизонтальними маркетплейсами, з використанням моделі C2C та B2C, а Alibaba також має модель B2B, адже пропонує великі обсяги товарів за оптовими цінами та можливість створення договору про постачання.
- Маркетплейси для послуг: Це маркетплейси, де продавці пропонують різні послуги, від дизайну та розробки програмного забезпечення до послуг таких як ремонт пральних машин або прибирання в будинках. Найвідомішими маркетплейсами для послуг є Upwork, Freelancer та TaskRabbit, вони є вертикальними маркетплейсами, з використанням моделі C2C та B2C.
- Маркетплейси для туризму: Це маркетплейси, які дозволяють забронювати готелі, квартири, авіаквитки та інші подорожні послуги. Найвідомішими маркетплейсами для туризму є Booking.com, Airbnb та Expedia, вони є вертикальними маркетплейсами, з використанням моделі C2C та B2C.

Перший маркетплейс, який з'явився в інтернеті, - це компанія eBay. Вона була запущена в 1995 році, коли її засновник П'єр Омідьяр створив веб-сайт для продажу і купівлі товарів, який відомий своєю аукціонною моделлю продажу. Початково сайт називався AuctionWeb, але пізніше було перейменовано на eBay. Засновник компанії мав велике бажання знайти місце, де люди могли б купувати і продавати різноманітні товари в інтернеті, що зробило eBay першим масштабним маркетплейсом в інтернеті. Сьогодні eBay є однією з найбільших і найдавніших онлайн-торгових платформ у світі.

Також, варто виділити особливості які відрізняють маркетплейси від онлайн магазинів:

- Кількість продавців та товарів: У маркетплейса можуть бути десятки тисяч продавців та мільйони товарів, тоді як у онлайн-магазині зазвичай обмежена кількість товарів та продавців.

- Бізнес-модель: Інтернет-магазини продають лише свої товари чи послуги, тоді як маркетплейси дозволяють продавцям розміщувати свої товари на платформі. Торгові майданчики зазвичай отримують комісійні за продаж продуктів або послуг, тоді як онлайн-магазини отримують комісійні за продаж власних продуктів.
- Взаємодія з користувачем: ринкові майданчики часто пропонують широкі можливості пошуку та фільтрації продуктів, тоді як онлайн-магазини можуть запропонувати простіший та інтуїтивно зрозуміліший інтерфейс користувача.
- Логістика: на ринку логістикою та доставкою зазвичай займається продавець, тоді як в інтернет-магазині ними зазвичай займається сам магазин.
- Відповідальність за товар: на маркетплейсі продавець зазвичай відповідає за продукт, тоді як в інтернет-магазині власне сам інтернет-магазин.

При порівнянні маркетплейсу та онлайн магазинів, виникає питання в доцільності та вигідності створення маркетплейсу для власника, адже при створенні свого онлайн магазину, компанія-власник отримує список переваг таких як: розширення аудиторії споживачів, зменшення витрат за рахунок того, що не витрачаються ресурси на відкриття фізичних магазинів, збільшення можливостей для маркетингу за рахунок різних інструментів для просування своїх товарів та послуг, такі як SEO, соціальні медіа та рекламні кампанії. Але маркетплейс може мати ряд вигід для його власника:

В першу чергу це прибуток від комісії, маркетплейс може отримувати прибуток від продажу товарів або послуг на платформі, що дозволяє отримувати стабільний дохід при мінімальних витратах на інфраструктуру та рекламу. Також маркетплейси більш гнучкіші в створенні джерел прибутку, наприклад маркетплейс Allyouneed, створений логістичним оператором DHL, здійснюючи доставку замовлень з маркетплейсу, задіює свої потужності основного напрямку бізнесу – логістику. Також з урахування великої конкуренції продавців, маркетплейси пропонують платне просування у топ окремих товарних позицій:

розміщення банерів на видному місці, у товарному розділі, просування товару в пошуку тощо. [2]

З точки зору постачальника є показовим опитування яке в 2021 році провела Hubber(IT-платформа для роботи по дропшипінгу) серед учасників своєї платформи - постачальників та онлайн-вітрин.[3] Опитування стосувалося розвитку e-commerce-ринку, а саме товарного бізнесу на маркетплейсах. Цікавим є те, що 82,6% з опитаних постачальників вважають, що маркетплейси - найперспективніший канал, і тільки 40,6% проголосували за власний інтернет-магазин. А в перспективність офлайну вірить лише 8,7%. Маркетплейси зазвичай забезпечують продавцям доступ до великої аудиторії клієнтів, що дозволяє їм збільшувати обсяги продажів. Тому компанії такі як Hubber популярні, адже надають інструментарій для створення оголошень на як можливо більшій кількості площадок, для охоплення максимальної аудиторії клієнтів.
А з точки зору рядового продавця маркетплейс ледь не єдина можливість продати свій товар чи послугу.

Покупці ж віддають перевагу маркетплейсам, бо ті в свою чергу пропонують широкий асортимент товарів у різних категоріях, що допомагає відразу знайти потрібний товар, нижчі ціни, оскільки продавці конкурують один з одним тому роблять знижки та спеціальні пропозиції, також маркетплейси дають змогу легко знайти інформацію про товари, їхні характеристики та відгуки минулих покупців, що дає можливість зробити більш усвідомлений вибір.

2. Розробка маркетплейсу

З точки зору маркетингу, створення нового продукту має починатися з оцінки доцільності створення цього продукту та аналізу потенційних конкурентів. В першому розділі було розглянуто те, що користувачі, як продавці, так і покупці - зацікавленні в появі нових маркетплейсів, адже для продавця це можливість розширити аудиторію покупців, а для покупця це нова, зручна платформа для здійснення покупок або пошуку чогось нового. Аналіз конкурентів може допомогти у визначенні мінімально-необхідних функціональних вимог. Для аналізу достатньо використати всього 3 маркетплейси: Rozetka, OLX, Booking. Вибір зумовлений тим, що група цих маркетплейсів повністю перекриває всі з перерахованих категорій, так:
Rozetka - горизонтальний маркетплейс з використанням моделі B2C та B2B,
OLX - горизонтальний маркетплейс з використанням моделі B2C, B2B та C2C,
Booking - вертикальний та глобальний з використанням моделі B2C та C2C.
Тематики та відмінності в реалізації бізнес моделей кожного маркетплейсу допоможуть виділити, особливості при створенні нового маркетплейсу.

2.1 Виділення ролей у користувачів

Почати варто з того як бачать користувача на різних маркетплейсах, тобто які існують ролі, та чим вони відрізняються. Почнемо з продавця та покупця, на Rozetka продавати товари можуть тільки зареєстровані ФОП та ТОВ, тому продавець це окрема сутність з своїм функціоналом, а покупець це окрема зі своїм. На Booking система схожа до Rozetka, користувач, який надає послуги оренди (продавець), це окрема сутність та має окремий аккаунт. На OLX продавець та покупець це одна сутність, лишень для отримання можливості публікації оголошень, як продавець, потрібно пройти верифікацію. Також на кожній платформі є окремий вид користувачів - модератори, вони мають право блокувати/видаляти користувачів, оголошення, відгуки тощо. Як зазначає сам OLX більшість підозрілих оголошень та акаунтів видаляється за допомоги

штучного інтелекту, але досі в день модератори блокують близько трьох тисяч оголошень “власноруч”[4], а на Rozetka можемо помітити відповіді на коментарі до товарів від модераторів. Отже можемо виділити три групи користувачів - покупець, продавець, модератор. А також відмітити те, що продавець та покупець в залежності від реалізації є однією сутністю або двома різними.

2.2 Створення реляційної схеми БД.

Звернувшись до базового визначення маркетплейсу можемо винести мінімальний функціонал для маркетплейсу в залежності від користувача. Кожен тип користувачів взаємодіє з товаром/послугою, наприклад, покупець - переглядає або додає в обрані, продавець - додає нові, редагує чи видаляє вже створені ним раніше, модератор - видаляє ті, які не пройшли модерацію. В подальшому доцільно товар або послугу називати оголошенням, бо як сутності вони відрізняються лише деякими полями. Оголошення є невід’ємною частиною сутностей і в залежності від тематики маркетплейсу мають різне наповнення. Отже в результаті маємо дві сутності: користувач та оголошення, та зв’язок один до багатьох, що демонструє, що користувач(продавець) може мати декілька оголошень, а оголошення належить одному користувачу. Графічне зображення реляційної бази даних зображено на рисунку 2.1.

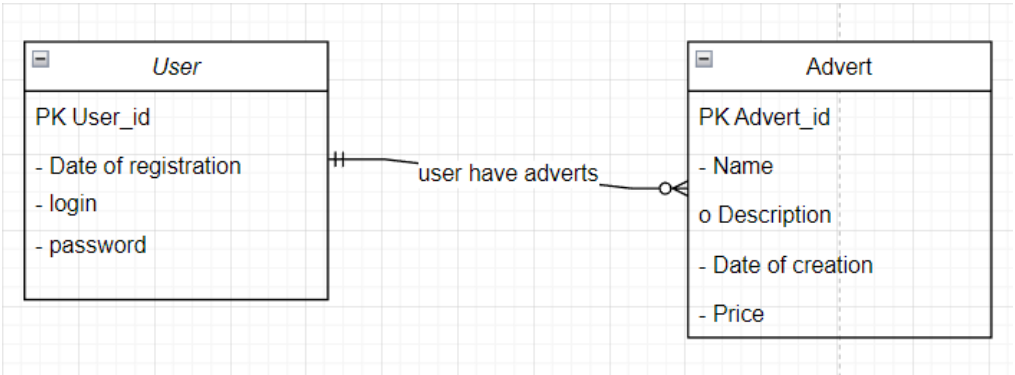

Рис. 2.1 Реляційна схема користувач та оголошення

Виділяючи спільні елементи, які має кожен новітній маркетплейс, варто виділити наявність категорій, вони допомагають організувати оголошення в зручному для

покупців вигляді та зробити пошук більш ефективним і швидким, а продавцям відображати свої товари в правильному контексті та привертати увагу зацікавлених покупців до своєї продукції. Крім того, категорії можна використовувати для збору статистики продажів, що дає змогу маркетплейсам зрозуміти, які категорії користуються більшою популярністю і які товари слід додати в асортимент, щоб задовольнити потреби покупців.

Наступним елементом є списки побажань - це потужний інструмент для залучення та утримання користувачів, а також підвищення ефективності маркетплейсу завдяки вдосконаленню рекомендаційної системи і збору даних про популярність товарів і тенденції покупок. Списки побажань можна використовувати для підвищення залученості клієнтів, збільшення продажів і поліпшення загального користувацького досвіду. Графічне зображення реляційної бази даних з додавання нових сутностей зображено на рисунку 2.2.

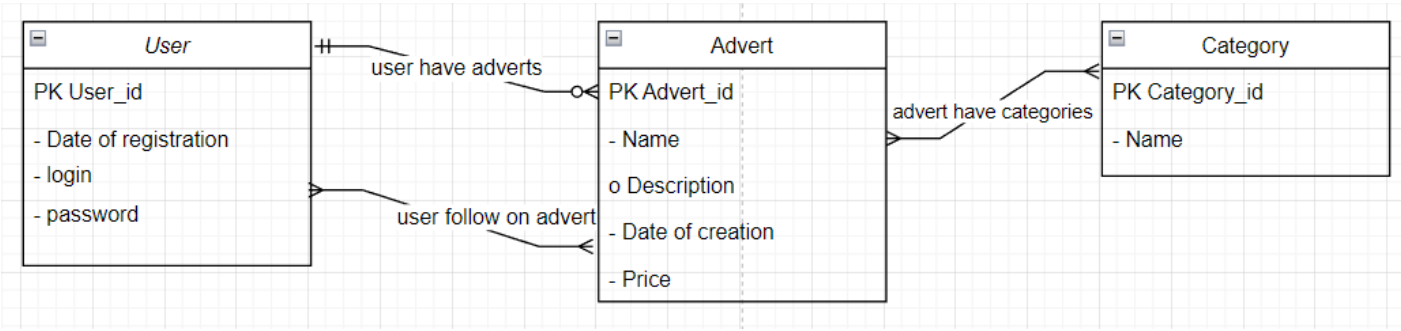

Рис. 2.2 Реляційна схема з доданням категорій та списків бажаного

3. Вступ до рекомендаційних систем

3.1 Термінологія

Аби уникнути неточностей при розгляді рекомендаційних систем, варто спочатку виділити термінологію:

- Контент - це товари або послуги, які залежать від специфіки застосунку чи бізнесу, наприклад контентом онлайн кінотеатру будуть фільми, серіали чи мультфільми, а для магазину електронної техніки - холодильники, телевізори, смартфони і тп.
- Прогноз - це припущення про те, наскільки користувачі будуть зацікавлені в запропонованому контенті.
- Релевантність - це порядок розміщення контенту залежно від того, що якнайкраще для користувача підходить в цей момент.
- Рекомендація - результат роботи рекомендаційної системи, лідери в списку релевантності.
- Профіль уподобань - це набір даних, що описують уподобання, поведінку та інтереси конкретного користувача.
- Персоналізація - надання користувачам рекомендацій на основі їхніх профілю уподобань, що дає змогу поліпшити користувацький досвід і підвищує ймовірність того, що користувачі приймуть рекомендації та виконають супутні їм дії (наприклад, здійснять покупку, переглянуть фільм або прочитають статтю).

Рекомендаційна система - це система, яка аналізує дані про користувачів та контент, щоб за потреби підбирати та запропонувати найбільш релевантний контент користувачам на основі їх уподобань, інтересів, поведінки.

3.2 Класифікація рекомендаційних систем

Перед побудовою рекомендаційної системи, варто спочатку класифікувати, яку саме рекомендаційну систему ми хочемо бачити в якості кінцевого результату. З цим може допомогти таксономія професора Джозефа А. Констана та Майкла Д. Екстранда.[6] Вони описали рекомендаційну систему як набір параметрів: спеціалізація, задача, контекст, ступінь персоналізації, експертність, конфіденційність та надійність, інтерфейс, алгоритм, де:

- Спеціалізація залежить від типу контенту який надає сервіс. Залежно від контенту приймаються відповідні рішення, чи прийнято пропонувати користувачу один і той самий контент декілька разів та яка ціна помилки при невдалій роботі рекомендаційної системи.
- Задача кожної рекомендаційної системи зробити життя кінцевого користувача простішим, шляхом фільтрації та підбору найбільш релевантного контенту, що спрощує вибір та підштовхує користувача до виконання певних дій, наприклад придбання певного товару, які в свою чергу вигідні власнику сервісу який має рекомендаційну систему.
- Контекст - це час, місце розташування, соціальний контекст, настрій користувача, тип або(та) модель пристрою, погода за місцем розташування користувача, рівень шуму навколо, тощо. Будь яка корисна, додаткова інформація, що враховуються під час генерації рекомендацій, може бути застосована для більш релевантних результатів саме в момент коли робиться запит.
- Ступінь персоналізації може бути різною і в рамках однієї рекомендаційної системи, рекомендації можуть бути:

    1. Не персональні рекомендації формуються без врахування даних профілю уподобання окремого користувача. Найчастіше такі рекомендації формуються за рахунок узагальнення певних поведінкових патернів людини. Наприклад, людина вранці приходить в ресторан, найбільш логічним для офіціанта, є рекомендувати людині

саме сніданки, адже зазвичай, вранці люди снідають. І ця рекомендація, прогнозовано, буде мати високу релевантність.

2. Частково персональні рекомендації також формуються без врахування даних профілю уподобання, але на основі контексту додають користувача в певну групу й надають спільні рекомендації для всієї групи користувачів. Повернемось до прикладу з рестораном, ви обрали страви і офіціант рекомендує напої, візуально визначаючи або запитуючи, дізнається чи є ви повнолітніми, з урахуванням цього відносить вас до певної групи людей: повнолітні або неповнолітні. І вже в залежності до того до якої групи ви належите - пропонує різний набір напоїв.
3. Персональні рекомендації формуються за урахування профілю уподобання. В прикладі з рестораном уявимо, що ви відвідуєте ресторан регулярно протягам певного часу, у вас з’явилися улюблені страви і звички, наприклад пити каву з тістечком вранці, тоді якщо ви прийдете в ранці до ресторану офіціант з великою ймовірністю запропонує вам каву з тістечком або новою смачною випічкою з начинкою.

- Експертність здебільшого стосується експертних рекомендаційних систем, які на сьогоднішній день втрачають свою популярність. Ці системи формують рекомендацій на основі рекомендацій прописаних вручну спеціалістами в певній сфері.

- Конфіденційність і надійність є основними факторами, що впливають на довіру користувачів до сервісу, вони будуть неохоче користуватися послугами, якщо вони відчувають, що їхні дані не захищені, а сервіс не заслуговує на довіру. Факт ненадійності може бути викликаним явним налаштуванням рекомендаційної системи для видачі не оптимального контенту користувачу, а вигідного для її власника. Показником надійності рекомендаційної системи є те, наскільки користувач приймає рекомендації, замість того щоб розцінювати їх як рекламу.

- При розгляді *інтерфейсів* рекомендаційної системи можемо розділити на інтерфейс:

    1. Вводу - це те в який спосіб система збирає дані про вподобання користувача, це може бути ***явним*** як виставлення оцінки з відгуком, додавання контенту в обране, або взагалі заповнення користувачем власного профілю вподобань. Так і ***неявним*** - аналіз рекомендаційної системи відвіданих вами сторінок контенту за останні декілька днів, чи аналіз зроблених вами покупок або те скільки ви переглядали певну сторінку, все це робиться для того, щоб система краще розуміла користувача. Використання обох видів вводу є досить ефективним, адже в реальності часто речі які людина заповнила в профілі уподобань відрізняється від її захоплень.
    2. Виводу - це те в який спосіб користувачу виводяться рекомендацій. Вивід може мати неорганічний вигляд - це означає, що текстом вказується “це рекомендація”. І органічний вигляд - рекомендація завуальована і сприймається користувачем як звичайна частина інтерфейсу. Також вивід може застосовуватися за принципом “білого ящика” - пояснювати на основі чого були зроблені ці рекомендації, “чорного ящика” - відповідно не пояснювати на основі чого були зроблені ці рекомендації.

- Алгоритми можна розділити на два типи в залежності від того, якими даними оперує рекомендаційна система для створення рекомендації.

    - Сумісна (колоборативна) фільтрація використовує дані користувача про контент з яким він вже взаємодіяв до моменту створення запиту до рекомендаційної системи, аби додати його до певної групи користувачів, які відвідували/користувалися/придбали той самий контент, що і цей користувач. Алгоритм припускає, що якщо користувачу подобається певний контент то з високою ймовірністю його зацікавить контент, який зацікавив групу користувачів з

схожими смаками. Отже, можливими рекомендаціями буде жовта фігура на рисунку 3.1.

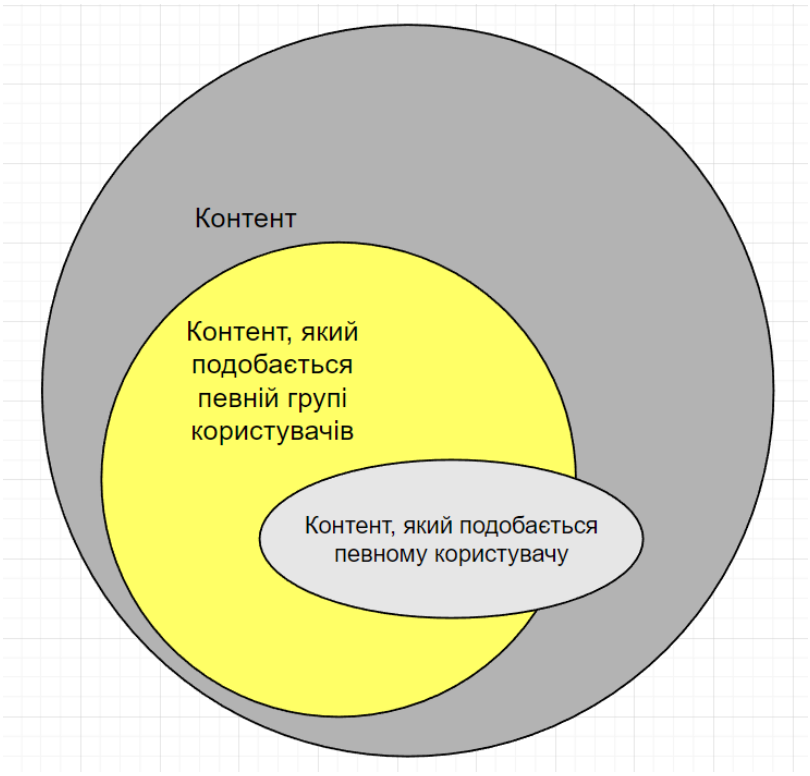

Рис.3.1 Схема сумісної фільтрації

- Фільтрація за вмістом(Контентна фільтрація) - це алгоритм створення рекомендації з використанням характеристик і властивостей контенту (наприклад, ключових слів, жанрів, авторів і тп…) з метою визначення його відповідності потребам і вподобанням користувача. Система рекомендацій обирає контент, що відповідає інтересам користувача, беручи до уваги інформацію про користувача з профілю уподобань, та його історії його активності. Фільтрація контенту може бути виконана з використанням різних методів аналізу контенту, як-от аналіз ключових слів, семантичний аналіз.
- Гібридна фільтрація - це алгоритм комбінації фільтрації за контентом та сумісної фільтрації для створення більш точних рекомендацій. Даний алгоритм є ідеальним рішенням майже для будь якої рекомендаційної системи, бо фільтрація за контентом не буде функціонувати для системи де контент не структурований та не має

достатнього опису, а колоборативна фільтрація не буде функціонувати, якщо немає достатньої кількості взаємодії користувачів з контентом та немає системи оцінок на сервісі.

3.3 Збір інформації для рекомендаційної системи

Рекомендаційні системи збирають інформацію про користувачів з різних джерел, залежно від конкретної системи та її функцій. Основними джерелами збору інформації є:

- Поведінкові дані: рекомендаційні системи аналізують поведінку користувачів, таку як перегляд, кліки, покупки, оцінки та коментарі. Ці дані допомагають зрозуміти вподобання та інтереси користувачів.
- Демографічні дані: інформація про користувача, така як вік, стать, місце проживання та освіта, що допомагає створити більш точний профіль користувача.
- Соціальні дані: рекомендаційні системи можуть отримувати інформацію з соціальних мереж. Соціальні мережі містять дані про друзів, підписників, групи, інтереси та інші дані, корисні для рекомендацій.
- Інтереси та особисті уподобання: користувачі часто власноруч надають інформацію про свої інтереси, уподобання та особисті налаштування, щоб отримувати більш точні та персоналізовані рекомендації.

Інформація, зібрана від користувачів, використовується рекомендаційною системою для створення персоналізованих пропозицій, щоб відображати лише той, що відповідає інтересам користувача або для прогнозування майбутніх уподобань і адаптації рекомендацій для підвищення довгострокової задоволеності користувачів. Важливо, щоб рекомендаційні системи були прозорими і дотримувалися принципів конфіденційності та безпеки даних. Користувачі повинні бути проінформовані про типи даних, які збираються, і про те, з якою метою вони будуть використовуватися. Крім того, користувачам повинна бути надана можливість контролювати свої дані і давати згоду на збір і використання їхньої інформації. Також важливо, щоб інформація про користувачів

використовувалася в рамках етичних стандартів і законодавства. Підсумовуючи, рекомендаційні системи збирають різноманітну інформацію про користувачів, включаючи поведінкові дані, демографічну інформацію, соціальні дані та інтереси. Ця інформація використовується для створення персоналізованих рекомендацій, фільтрації контенту, підвищення рівня задоволеності користувачів і надання підтримки у прийнятті рішень. Однак важливо, щоб дані збиралися і використовувалися відповідно до етичних стандартів і з повагою до приватності та конфіденційності користувачів.

3.4 Оцінки

Персоналізовані рекомендації формуються на смаках користувача, а для визначення його смаків потрібно мати певну інформацію, про вже відому його взаємодію з контентом. Для роботи рекомендаційної системи найкраще підходить система оцінок, яка формується з поведінкових даних користувача. Найпростішим є оцінка, яку користувач сам виставить певному контенту, але не завжди користувачі це роблять та й не завжди даний функціонал реалізований в веб-застосунку. Іншим варіантом створення оцінок є розрахунок оцінки з взаємодії користувача з контентом, це може бути як перегляд сторінки, відкриття окремих її елементів, або додавання в обране. Кожна дія має свою вагу яку розробник сам обирає в залежності від спеціалізації рекомендаційної системи. Далі з враховуючи вагу та взаємодію користувача з контентом, можемо створити оцінку, яка охарактеризовує ступінь зацікавленості користувача з певним контентом.

3.5 Неперсоналізовані рекомендації

Найпростіші в реалізації - не персоналізовані рекомендації. Для їх генерації рекомендаційній системі не потрібно знати який саме користувач запитує рекомендацію. Створення таких рекомендацій ґрунтується на припущенні, що з великою ймовірністю людині подобається контент, який подобається більшості. До таких рекомендацій може входити контент відсортований за: новизною, ціною, кількістю доданих в обране, або проданих. За такими фільтрами можна

створювати рекомендації з заголовками на кшталт: “Новинки”, “Найдорожчі пропозиції” або “Найдешевші пропозиції”, “Популярні”, “Бестселери”. Додавши фільтрацію за певним проміжком часу до попередні рекомендацій, можемо створити сезонні неперсональні рекомендації, які будуть відображати більш точні дані про популярність контенту в певний проміжок часу. FBT (з англ. frequently buy together - часто купують разом) - один з найпопулярніших та ефективних видів не персоналізованої рекомендації, метою якої є запропонувати користувачу переглянути контент, який часто користувачі купують разом з тим, що переглядає користувач. Для реалізації такого методу потрібно використати дані про здійснених покупок користувачами, створити список контенту і обчислити його *підтримку* (кількість покупок де присутній цей контент, поділену на кількість всіх покупок) і встановіть *впевненість* на одиницю. Створити список наборів елементів, що містить більше одного елементу контенту, розрахувати *підтримку для набору* (кількість покупок де присутній набір елементів контенту, поділену на кількість всіх транзакцій) і *впевненість* (кількість покупок де присутній набір елементів контенту, поділити на кількість покупок де присутній контент для якого ми шукаємо FBT), для набору елементів. В кінці відфільтрувати ті, які мають низьку *впевненість* та *підтримку*. Іншим варіантом є не відкидати товари, які мають низьку *підтримку*, це означатиме, що до рекомендацій додадуться абсолютно всі продукти які траплялися разом із заданою в одній транзакції. Впорядкувавши за *впевненість* отримаємо список з тими, що трапляються найчастіше, до тих що трапляються рідше.

3.6 Персоналізовані рекомендації

Для даного типу рекомендацій потрібна вже більш чітка взаємодія користувача з контентом, найкращим варіантом є система оцінок.

Колаборативна фільтрація - це метод рекомендаційних систем, який заснований на схожості користувачів. Якщо два користувачі мають схожі вподобання і взаємодію з контентом, вони вважаються схожими, і контент, якому віддає перевагу один користувач, з більшою ймовірністю буде рекомендований іншому

користувачеві з схожими смаками. Першим кроком роботи фільтру є збір даних про взаємодію між користувачами та контентом - оцінки. На основі зібраних даних будується модель, яка відображає взаємодію між користувачем і контентом. Це може бути матриця взаємодії, де кожен рядок відповідає користувачеві, кожен стовпець - контенту, а значення в матриці відображають взаємодію - оцінку. За допомогою метрик подібності, таких як косинусна подібність, евклідова відстань та кореляція визначаємо схожість користувачів або контенту. Після виявлення схожих користувачів або контенту можна генерувати рекомендації для конкретних користувачів. Наприклад, вибрати контент, який був позитивно оцінений схожими користувачами, але ще не були спожиті користувачем якому формуються рекомендації. Якщо користувачі оцінюють рекомендовані контент або взаємодіють з ними, ця інформація може бути використана для оновлення моделі з урахуванням нових уподобань користувачів. Основними перевагами колаборативної фільтрації є те, що метод базується на реальній взаємодії між користувачем і контентом, це в свою чергу дозволяє отримувати більш природні та персоналізовані рекомендації. Також колаборативна фільтрація може виявляти складні взаємозв'язки між користувачами та контентом, які важко формалізувати або визначити. Але даний метод має ряд недоліків: у великих системах спільної фільтрації брак даних може бути проблемою, оскільки багато користувачів не взаємодіють з великою кількістю контенту. Це може ускладнити пошук достатньої кількості схожих користувачів і контенту для створення рекомендацій. Також фільтр має проблему “холодного старту”, коли система стикається з новим користувачем або новим об'єктом, для яких немає достатньої кількості взаємодій або інформації. Спільна фільтрація вимагає спільного використання даних про взаємодію між користувачами. Це може призвести до проблем конфіденційності та довіри, а користувачі можуть не бажати розкривати свої особисті уподобання та дані.

Фільтрація на основі вмісту - це метод рекомендаційних систем, який аналізує властивості контенту такі як текст, ключові слова, типи, теми, атрибути тощо, рекомендує користувачам схожий контент. Основна ідея полягає в тому, що якщо

користувачеві подобаються певні властивості контенту, то йому, швидше за все, сподобаються й інший контент зі схожими властивостями. Першим кроком в роботі фільтру є збір властивостей контенту, який рекомендується. Властивості можуть бути різних типів, включаючи текстові описи, заголовки, ключові слова, зображення, аудіо та відео. Інформація може надходити з веб-сторінок, баз даних, соціальних мереж тощо. Наступним кроком є створення профілів контенту, які включають важливі властивості та атрибути. Далі створюється векторне представлення контенту - вміст перетворюється на числове векторне представлення. Це можна зробити за допомогою методів векторизації, таких як мішок слів(bag-of-words), TF-IDF, нейромережевих моделей. Наступним етапом є створення профілів користувача, які описують взаємодію його з контентом (наприклад, виставлення оцінок, переглянутого контенту, доданого контенту в обране тощо). В кінці застосовуються методи для порівняння профілів користувача та профілю контенту і визначення ступеня відповідності. Це можна зробити, наприклад, шляхом обчислення косинуса подібності між векторами профілів або застосування інших метрик подібності, контент найбільш близький до профілю користувача, ранжуються в порядку спадання релевантності. Найвищий рейтинг отримує контент, який найбільше відповідає уподобанням та інтересам користувача. На основі цього ранжування система надає користувачеві рекомендації та пропонує набір контенту, який користувачеві може бути цікаво споживати. Метод фільтрації на основі контенту має ряд переваг. Перше це незалежність від взаємодії з користувачем, вона не вимагає взаємодії або інформації від інших користувачів, адже ґрунтується на аналізі власного вмісту контенту та вподобаннях окремого користувача. Відносно просте додавання нового контенту, що дуже корисно в середовищі, де з'являється новий контент. Незважаючи на ці переваги, методи фільтрації на основі вмісту також мають певні обмеження. Перше це проблема недостатньої варіативності, якщо система враховує лише той контент, який користувач вже споживав, рекомендуватися буде щось дуже схоже на те, що користувач вже споживав, не враховуючи можливості виходу в нові області або відкриття нового контенту. Фільтрації на

основі контенту можуть бути обмежені у своїй здатності розуміти складні аспекти вподобань і контексту користувача. Фільтрація за вмістом також має проблему “проблему холодного старту”, але потребує набагато менше взаємодії від користувачів з контентом для старту.

3.7 Аналіз наявних реалізацій

Перед початком розробки власної рекомендаційної системи варто проаналізувати вже наявні рішення, для аналізу використаємо Rozetka.

Спеціалізація: Rozetka спеціалізується на продажу товарів з різноманітних категорій.

Задача: Основна задача рекомендаційної системи Rozetka полягає у забезпеченні користувачам релевантних і цікавих товарних рекомендацій на основі їхніх інтересів, задля збільшення кількості продажів та отримання більшого прибутку.

Контекст: Як зазначає сама Rozetka в “Положення про обробку і захист персональних даних”[5], вона отримує та зберігає дані про: тип пристрою, тип браузера, операційну систему пристрою. Ця інформація не є введена користувачем, а формується при запиті користувача до сервісу. Також платформа зберігає, будь-яку інформацію, законно отриману від третіх осіб або інформацію, яка доступна з профілю користувача в соціальних мережах (якщо користувач зареєструвався на Rozetka з використанням соціальних мережах).

Рівень персоналізації: Rozetka має два типи рекомендацій. Не персоналізовані, коли користувач тільки зареєструвався та потрапив на головну сторінку, будуть присутні лише рекомендації які з високою ймовірністю не пов'язані з користувачем, але можуть його зацікавити, бо зацікавили більшість. Ці категорії мають заголовки: “Акційні пропозиції”, “Гарячі новинки”, “Зараз користується попитом в категорії *назва категорії* ”,“Найчастіше додають в лист бажань”, “Найбільш очікувані”, “Найбільш обговорювані товари”, “Зараз користуються попитом”. В ході користуванням сервісом, це може бути пошук та перегляд товарів або заповнення профілю уподобань, який знаходиться в особистому

кабінеті (інтерфейс налаштування профілю уподобань зображено на рис 4.1), на головній сторінці з’являться нові категорії рекомендацій.

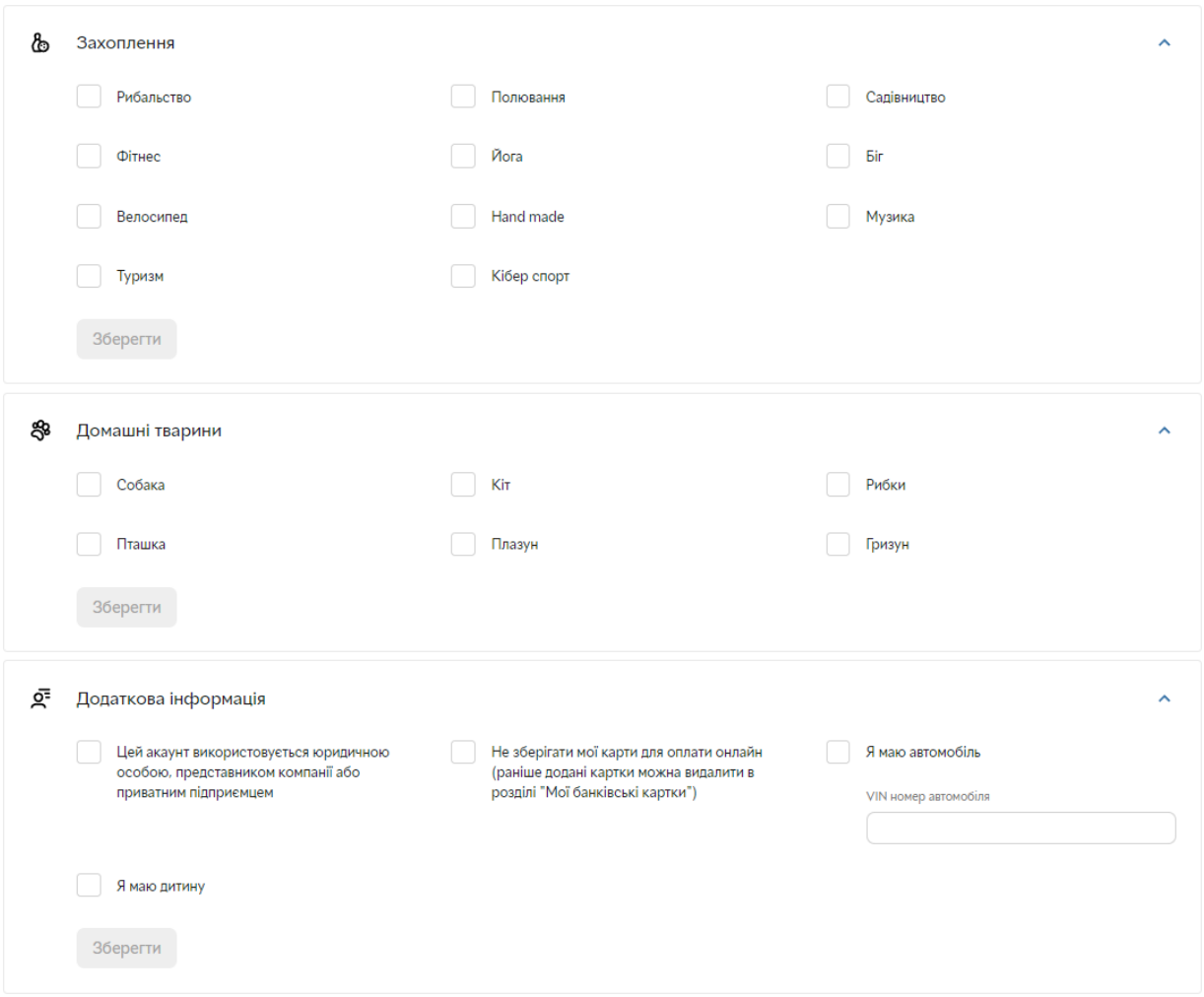

Рис. 4.1 Профіль уподобань на Rozetka

Це будуть товари, з категорій які ви переглядали за останній час, так якщо ви переглядали товари з категорії велосипеди та корм для собак, то на головній сторінці з'являються категорії, “Більше товарів для вибору”, “Рекомендації на основі ваших переглядів”, “Бестселери в категорії Корм для собак”, “Бестселери в категорії Велосипеди”. Ці рекомендації вже персоналізовані, адже ґрунтуються на вашій взаємодії з сервісом.

Експертність: Rozetka не використовує експертну рекомендаційну систему, але частково до експертних рекомендацій можна виділити “Rozetka рекомендує”.

Конфіденційність і надійність: Як зазначає сама Rozetka в “Положення про

обробку і захист персональних даних”[5] персональні дані користувачів обробляються та зберігаються на захищених серверах.

Інтерфейс вводу: Явним інтерфейсом вводу, є додавання товарів до списку бажаного, а також відгуків до них. Неявними є відстеження відвідуваних вами сторінок товару, здійснених покупок, та ще ряд даних з переліку в “Положення про обробку і захист персональних даних”.

Інтерфейс виводу: Кожна рекомендація має заголовок який описує на базі чого була зроблена ця рекомендація. Більшість рекомендації мають неорганічний вивід, адже в заголовку мають слово “Рекомендація”.

Алгоритми: Найпростіше розпізнати “фільтрацію за контентом”, для цього варто лише обрати будь-який товар та прогорнути сторінку до пункту “Також вас можуть зацікавити”, список буде містити товари схожі до того який ви відкрили, також товар буде впорядкований на початку списку найбільше схожі, в кінці найменш. Сумісна фільтрацію можна помітити, якщо прогортати сторінку товару, ще нижче до пункту “Разом з цим товаром купують”, тут будуть товари які найчастіше зустрічаються разом в одній покупці із заданим товаром.

Переконатися в тому, що використовуються саме ці алгоритми досить просто, потрібно обрати будь-який товар, який є дуже унікальним або новим та не користується популярністю, в такому випадку за відсутності наявних покупок з даним товаром рекомендація “Разом з цим товаром купують” буде відсутня. А якщо товар дуже унікальний або його опис недостатній то буде відсутній пункт “Також вас можуть зацікавити”. Приклад товару який не має жодної рекомендації зображено на рисунку 4.2.

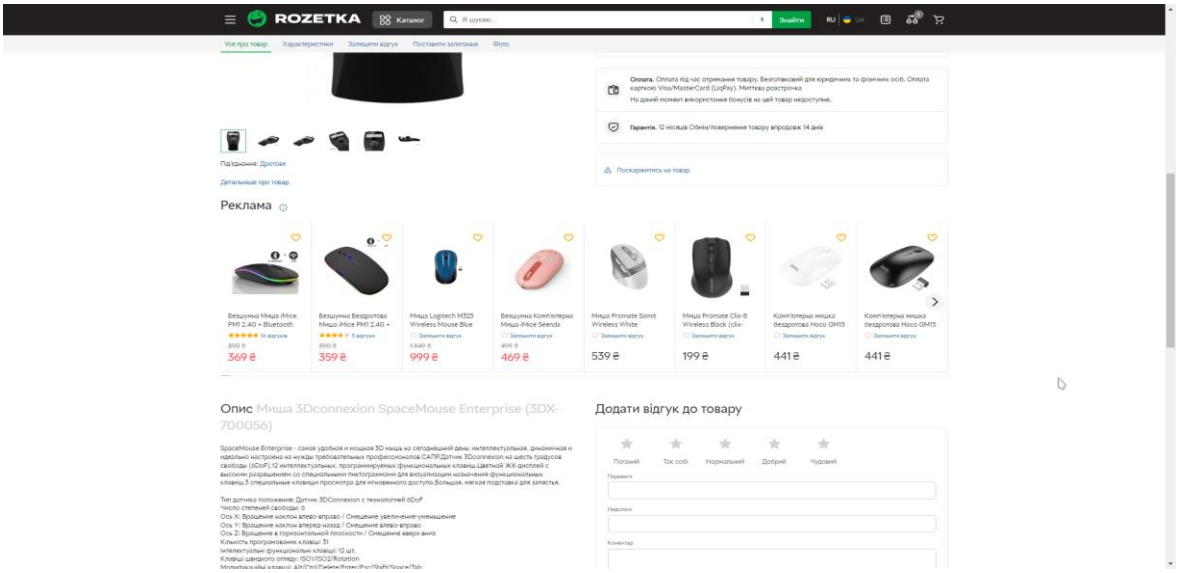

Рис. 4.2 Приклад відсутності рекомендацій

4. Практична частина розробки маркетплейсу

Розробка маркетплейсу здійснювалася за допомоги фреймворку Spring Boot. Ідеєю для практичної частини кваліфікаційної роботи є створення backend частини маркетплейсу, яка надає можливість користувачам публікувати оголошення та рекомендаційної системи до цього застосунку.

4.1 Визначення функціональних вимог та структури бази даних

Розробку варто почати з виділення функціональних вимог до маркетплейсу, як вже було розглянуто в розділі 2, будь який маркетплейс має мінімально-потрібний функціонал, до такого можемо винести: можливість реєстрації нових користувачів з обов'язковими полями і авторизації для доступу до особистих облікових записів, додавання нового контенту до маркетплейсу з вказанням назви, опису, ціни, фотографій, категорій до яких належить тощо, редагування та видалення існуючого, пошук за різними критеріями. Також важливим є створення користувачів з різним набором прав, адже ми надаємо можливість викладання у вільний доступ контенту, який може бути неприйнятним для більшості користувачів. Тому варто додати роль модератора(адміністратора), надавши йому розширений список прав. Для реалізації ролей, а також реєстрації, автентифікації, авторизації використовувалася Spring Security - це потужна бібліотека, яка надає різноманітні механізми для забезпечення безпеки в додатках, розроблених з використанням фреймворка Spring. За допомоги конфігураційних файлів, та додавання JWT токенів користувачам реалізована автентифікації користувачів. Після успішної аутентифікації, сервер генерує JWT, який містить інформацію про користувача та його ролі. Цей токен передається назад клієнту і використовується для ідентифікації та авторизації користувача при подальших запитах до сервера. Використовуючи реляційну діаграму Рис. 2.2, додав невистачаючої сутності Token, яка відповідає за JWT-токени, які належать певним користувачам.

4.2 Розробка схеми бази даних

В кінцевому результаті отримано ER-діаграму на рисунку 4.1, що використовується для візуалізації і моделювання структури даних в базі даних мого проекту.

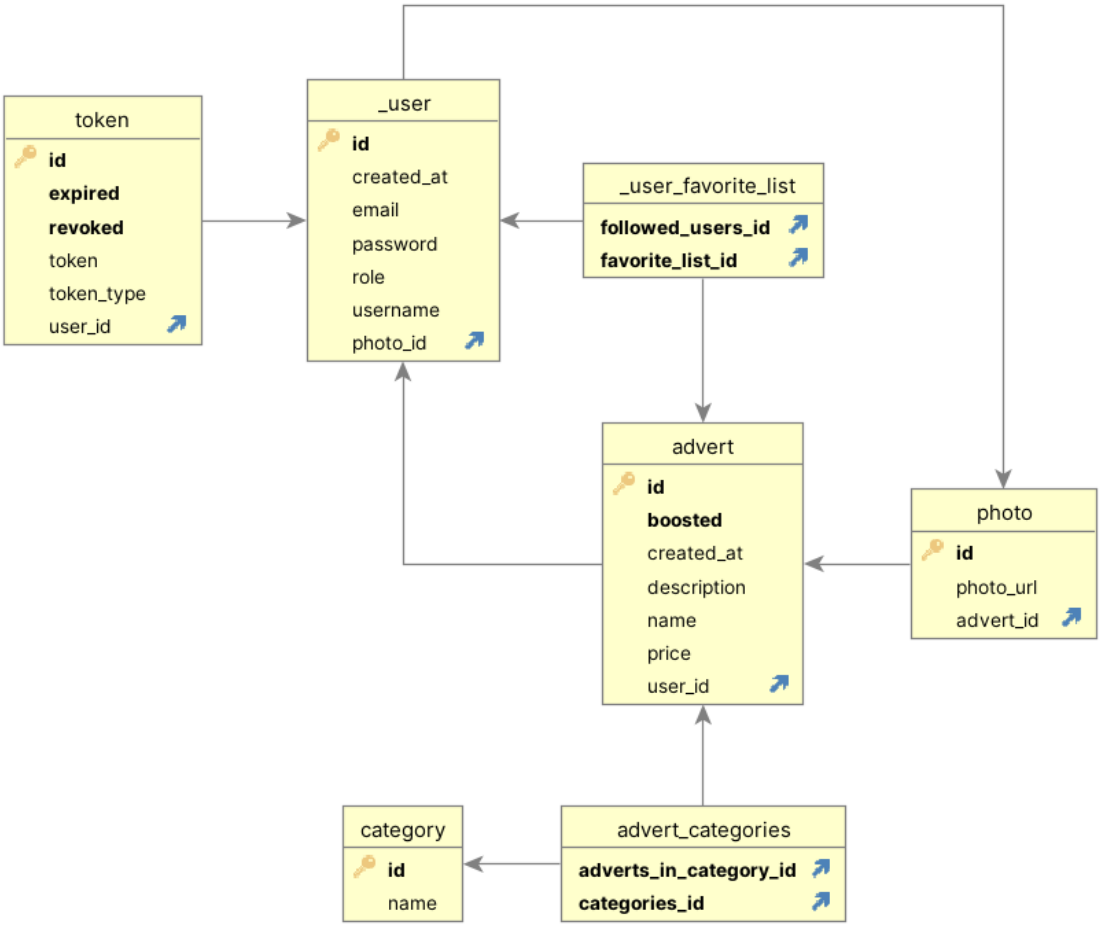

Рис. 4.1, ER-діаграма маркетплейсу

4.3 Сутності

Реалізація частини для взаємодії застосунку використано Spring JPA та Hibernate це фреймворки, які спрощують роботу з базами даних у Java-додатках. Spring JPA надає абстракцію репозиторіїв та інші зручні функції для роботи з даними, в той час як Hibernate забезпечує ORM-функціональність для збереження та отримання об'єктів з бази даних. Hibernate дозволяє мапити об'єкти Java на таблиці бази даних та автоматично вирішувати питання розбиття, кешування та управління транзакціями.[9] Для взаємодії між Hibernate та Java використовують моделі які

описують сутності (Entity). Розглянемо приклад написання сутності, на прикладі сутності Категорія(Category), рис. 4.2.

```java
package com.example.marketplace.model;

import ...

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@JsonIgnoreProperties({"advertsInCategory"})
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    private String name;

    @ManyToMany(mappedBy = "categories")
    private List<Advert> advertsInCategory;
}
```

рис. 4.2 Приклад опису сутності

В даному прикладі продемонстроване використання бібліотеки Lombok, яка вміє генерувати методи в процесі компіляції. Анотація @Data автоматично генерує стандартні методи доступу (getters та setters) для всіх полів класу, що спрощує роботу з класами Java, які містять поля даних, @NoArgsConstructor - генерує конструктор без аргументів, а @AllArgsConstructor - конструктор, який приймає аргументи для всіх полів класу. Анотація @Entity, є частиною Java Persistence API (JPA) і використовується для вказівки, що клас є сутністю (Entity), яка може бути збережена в базі даних. @JsonIgnoreProperties вказує, які властивості об'єкта мають бути проігноровані при конвертації об'єкта у JSON-рядок, що є досить зручним у випадку коли ми хочемо передавати об'єкт користувачу, але хочемо приховати певні поля від користувача. Анотація @Id вказує, що поле класу, до

якого вона застосовується, є ідентифікатором (primary key) сутності. Анотація @GeneratedValue(strategy = GenerationType.IDENTITY) вказує, стратегію за якою генеруються значення ідентифікатора. Анотація @ManyToMany вказує на наявність зв'язку "багато-до-багатьох" між двома сутностями. Цей тип зв'язку означає, що одна сутність може мати багато екземплярів іншої сутності, і навпаки. Окрім анотації @ManyToMany, існує кілька інших анотацій зв'язку, які використовуються в JPA: @ManyToOne - вказує зв'язок "багато-до-одного" між сутностями це означає, що багато екземплярів поточної сутності відносяться до одного екземпляра іншої сутності, @OneToMany - вказує зв'язок "один-до-багатьох" це означає, що один екземпляр поточної сутності відноситься до багатьох екземплярів іншої сутності. @OneToOne - вказує зв'язок "один-до-одного" між сутностями, один екземпляр поточної сутності відноситься до одного екземпляра іншої сутності [10].

На рис. 4.1, зображені таблиці _user_favorite_list та advert_categories є проміжними таблицями створеними за рахунок зв'язку багато-до-багатьох. Тому при описі сутностей можемо використовувати зв’язок багато-до-багатьох або описати окремим класом сутність проміжної таблиці, додавши зв'язки багато-до-одного до сутностей які були з'єднані зв'язком багато-до-багатьох. В файлі налаштувань застосунку(application.properties), можемо встановити spring.jpa.hibernate.ddl-auto=create, що дозволяє hibernate автоматично створити базу даних та таблиці в заданій СУБД. Після чого буде створено відповідні таблиці для кожної сутності, де стовпці таблиці відповідають за поля класу, який описує сутність, а рядки за кожен окремий екземпляр класу збережений в БД.

4.4 Репозиторії

За доступ до даних і взаємодію з базою даних відповідають репозиторії. Вони надають абстракцію над зберіганням та отриманням даних з моделей. Репозиторії виконують операції збереження, оновлення, видалення та отримання даних з моделей. Вони дозволяють ізолювати логіку доступу до даних від решти додатку і

забезпечувати її переносимість між різними джерелами даних. JPA-репозиторії автоматично генерує запити на основі назв методів. Наприклад, метод з назвою "findByName(String name)" знайде всі записи з вказаним значенням поля name. Але для більш складніших запитів пошуку я використовував анотацію @Query, щоб вказати власний JPQL запит. Це дозволило створювати складніші запити з використанням виразів, об'єднань, умов, параметрів тощо. JPA-репозиторій варто створити для кожної сутності в проекті, адже вони надають базові можливості взаємодії з рядками таблиць такі як додавання нових, редагування або видалення вже наявних. Повертаючись до прикладу з Категоріями, приклад реалізації РА-репозиторію на рис. 4.3, даний інтерфейс вже надає можливості для більшості можливих взаємодій з сутністю.

```
package com.example.marketplace.repository;

import ...

@Repository
public interface CategoryRepository extends JpaRepository<Category, Integer> {
    Optional<Category> findByName(String name);
}
```

Рис. 4.3 Реалізація репозиторію для Категорій

4.5 Сервіси

Більш складну бізнес логіку я переніс в сервіси, вони виконує більш складні розрахунки та дії, пов'язані з координацією між моделями та іншими компонентами системи. Сервіси включають такі операції, як обробка бізнес-правил, виконання транзакцій і взаємодію з іншими сервісами та різними репозиторіями. В результаті вони надають інтерфейс для зовнішніх компонентів, як-от контролери та моделі, для взаємодії з функціональністю програми. Також на рівні сервісів варто виконувати валідацію, та обробляти помилки при некоректній взаємодії користувача з API застосунку.

```
public ResponseEntity<?> addUsersAdvert(AddAdvertRequest advertRequest, Integer id) {
    User user = getById(id);
    Advert advert = new Advert();
    advert.setName(advertRequest.getName());
    advert.setDescription(advertRequest.getDescription());
    advert.setPrice(advertRequest.getPrice());
    advert.setSeller(user);
    advert.setCategories(categoryService.findAllByIds(advertRequest.getCategoryIds()))
    advertService.save(advert);
    return ResponseEntity.ok( body: "Advert added to system!");
}

public User getById(Integer id) {
    Optional<User> user = userRepository.findById(id);
    if (user.isPresent())return user.get();
    else throw new NoSuchElementException("No user found");
}

public ResponseEntity<?> deleteUsersAdvert(Integer advertId, Integer userId) {
    User user = getById(userId);
    Advert advert = advertService.getById(advertId);
    if(user.isHaveAdvert(advert)) {
        user.deleteAdvert(advert);
        userRepository.save(user);
        advertService.deleteById(advertId);
    }
    else return ResponseEntity.badRequest().body("User don`t own this advert");
    return ResponseEntity.ok( body: "Advert deleted!");
}
```

Рис 4.4 Приклади методів сервісу

Отже, головною задачею сервісів є створення методів, які взаємодіють з сутностями, та формують функціонал застосунку, на рисунку 4.4 зображено декілька методів, які створюють взаємодію між користувачем та оголошенням, в методі deleteUsersAdvert, виконується перевірка, чи виставляв оголошення користувач який подав запит на видалення, і якщо користувач володіє цим оголошенням, використовуємо JPA-репозиторій користувача аби видалити зв'язок між оголошенням та цим користувачем, а далі використовуємо сервіс оголошення, для видалення оголошення з таблиці оголошень. Було написано по одному сервісу для кожної сутності, а також 4 сервіси які відповідають за функції Spring Security а саме: аутентифікацію, реєстрацію, логін, логаут та взаєможію з JWT-токенами.

4.6 Контролери

Контролери використовуються в архітектурі веб-додатків для обробки та відповіді на HTTP-запити. Вони відповідають за прийняття HTTP-запиту від клієнта, прослуховуючи і відповідає на певні шляхи (URL) і типи запитів (GET, POST, PUT, DELETE), виконання необхідних дій(в сервісі) і формування відповіді. Контролер виступає посередником між користувачем і різними компонентами системи, такими як сервіси і база даних. Контролери забезпечують відокремлення логіки обробки запитів від бізнес-логіки додатку, що забезпечує чіткий рівень абстракції між веб-інтерфейсом та іншими компонентами програми. Це дозволяє легко розширювати і змінювати функціональність, не впливаючи на інші компоненти веб-застосунку. В рамках мого застосунку, я розділив контроллери на user-controller - відповідає за функціонал користувача(додавання нових оголошень, редагування, слідкування за іншими оголошеннями, видалення і тд.), authentication-controller - відповідає за реєстрацію, логін та логаут, admin-controller - функціонал пов'язаний з модерацією та роботою рекомендаційної системи , free-controller - набір функціоналу доступний будь-якому користувачу.

5. Практична частина розробки рекомендаційної системи

Отже як і зазначалося в розділі 3.2, розробку рекомендаційної системи варто почати з її класифікації та розгляду параметрів за якими можемо її описати.
Спеціалізація - оголошення, на мою думку не створюють особливих умов для рекомендаційної системи, адже якщо певне оголошення трапиться декілька разів користувачу, тільки означатиме те, що це оголошення має досить високу релевантність, та не здає позицію іншим товарам. Це в свою чергу означає, що товар з високою долею ймовірності зацікавить користувача. А повертаючись до важливості надання точних рекомендацій, на мою думку не є надто важливим, адже якщо оголошення в підбірці рекомендацій будуть мати не надто високу релевантність з смаками користувача, це зменшить ефект бульбашки фільтрів, коли користувачу пропонується контент виключно ідентичний його смакам і він практично не має змоги побачити інший, окрім як цілenапрямлено його шукати[7].
Задача - як будь яка рекомендаційної системи, збільшити залученість користувачів, до сервісу.
Контекст - в рамках цього проекту вирішено не використовувати контекстні данні.
Ступінь персоналізації - основною задачею є демонстрація роботи рекомендаційної системи, тому рекомендації формуються як персональні так і не персональні, також використання обох варіантів зменшує ефект бульбашки фільтрів.
Експертність - вже давно застарілий метод фільтрації, а в рамках даного проекту взагалі немає доцільності використання експертних систем.
Конфіденційність та надійність - Доступу до даних взаємодії користувачів з контентом, має лише рекомендаційна система.
Інтерфейси - вводу є неявним, збирається інформація користувачів про відвідані сторінки, виводу - явні, чітко прописано на основі чого сформовані рекомендації.
Алгоритми - використано фільтрацію за контентом. На мою думку використання

колаборативної фільтрації, не є надто доцільним з трьох причин. Перша, для того щоб сформувати групи користувачів, потрібно мати реальну активність великої кількості людей. Друга, на мою думку, формування груп не буде надто ефективним, адже в подібних сервісах люди шукають дуже різноманітні товари, і сформовані групи користувачів за смаками не будуть чітко описувати певну категорію покупців. І третя, найголовніше те, що для формування груп підходять сервіси де чітко можливо виділити факт взаємодії користувача з контентом (наприклад придбання/перегляду фільму в онлайн кінотеатрі або придбання товару в інтернет магазині), тоді як на маркетплейсі де тільки створюються та переглядаються оголошення це неможливо.

5.1 Неперсоналізовані рекомендації

Почнемо з не персоналізованих рекомендацій. Я вирішив виділити 3 типи рекомендацій в рамках проекту:

Найновіші пропозицій - формування рекомендації з списку найновіших оголошень. Розрахунок в цих рекомендацій йде на те, що користувачу випадково попадеться оголошення яке його зацікавить.

Найулюбленіші пропозицій - формування рекомендації з списку оголошень які мають найбільшу кількість користувачів які додали це оголошення до обраного. На мою думку ці товари є дуже цікавими, адже додавання товару до улюбленого свідчить про високу зацікавленість цим оголошенням, а велика кількість зацікавлених свідчить про певну унікальність цього оголошення. Також ці оголошення мають зникати(видалятись), після того як знайдуть свого користувача(покупця). Тому список буде постійно оновлюватись і просувати в рекомендаціях цікаві оголошення.

Оголошення з високою активністю користувачів - цей тип рекомендацій, схожий до попереднього, але буде оновлюватися частіше, і матиме однакові оголошення в підбірці протягом меншого часу.

5.2 Персоналізовані рекомендації

В якості персоналізованих рекомендації використано алгоритм фільтрації за контентом(вмістом), це означає те, що система має знаходити схожий контент, а схожість визначатиметься за вмістом оголошення таким як назва, опис тощо. Для реалізації алгоритму схожості використано косинусну подібність і TF-IDF (частота терміна - зворотна частота документа) - це дві широко використовувані концепції в обробці текстів і рекомендаційних системах. TF-IDF використовується для оцінки важливості терміна в документі, розглядаючи частоту терміна в одному документі - TF

$$TF = \frac{n_i}{\sum_k n_k} \tag{5.1}$$

де n_i - число входжень слова в документ,

$\sum_k n_k$ - загальна кількість слів в документі.

як обернену до його частоти в інших документах - IDF

$$IDF = \frac{|D|}{|d_i \supset t_i|} \tag{5.2}$$

де $|D|$ - кількість документів колекції,

$|d_i \supset t_i|$ - кількість документів d_i, в яких зустрічається слово t_i

та обчислюється за формулою

$$TF - IDF(t, d) = TF(t, d) \times IDF(t)\,. \tag{5.3}$$

де TF - відображає частоту терміна (слова) t в документі d (5.1),

IDF(t) - відображає зворотну частоту документа для терміна t (5.2).

Нам це дозволяє вибрати важливі терміни, що характеризують оголошення, і зважити їх відповідно до їхньої важливості. TF-IDF також може бути використаний для векторного представлення контенту, де кожне оголошення представлене у вигляді вектора, а значення вектора відображають важливість терміна. Наприклад оголошення номер 1 має терміни та їх TF-IDF: "term1":0.1, "term2":0.2, "term3":0.3, а друге оголошення має терміни та їх TF-IDF: "term2":0.2,

"term3":0.3, "term4":0.4. При представленні оголошень у вигляді векторів отримуємо для оголошення номер 1 - v1(0.1, 0.2, 0.3, 0) та відповідно для оголошення номер 2 - v1(0, 0.2, 0.3, 0.4). Представлення кожного оголошення у векторному вигляді дає змогу обчислити подібність, шляхом порівняння векторів за допомогою косинусної подібності

$$S_C = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2}\sqrt{\sum_{i=1}^{n} {B_i}^2}} \quad (5.4)$$

де A_i та B_i - координати вектору A та B відповідно.

На рисунку 5.1 зображено лістинг програми з обрахуванням косинусної подібності, де в першому for обраховується $\sum_{i=1}^{n} A_i B_i$ та $\sqrt{\sum_{i=1}^{n} {A_i}^2}$, в другому $\sqrt{\sum_{i=1}^{n} {B_i}^2}$, а в return значення S_C.

```
public static double calculateCosineSimilarity(Map<String, Double> vector1, Map<String, Double> vector2) {
    double dotProduct = 0.0;
    double magnitude1 = 0.0;
    double magnitude2 = 0.0;
    for (String term : vector1.keySet()) {
        if (vector2.containsKey(term)) {
            dotProduct += vector1.get(term) * vector2.get(term);
        }
        magnitude1 += Math.pow(vector1.get(term), 2);
    }
    for (Double value : vector2.values()) {
        magnitude2 += Math.pow(value, 2);
    }
    magnitude1 = Math.sqrt(magnitude1);
    magnitude2 = Math.sqrt(magnitude2);
    if (magnitude1 != 0.0 && magnitude2 != 0.0) {
        return dotProduct / (magnitude1 * magnitude2);
    } else {
        return 0.0;
    }
}
```

Рис. 5.1 Лістинг обрахунку подібності

Чим вище значення косинусної подібності, тим вища подібність між оголошеннями. В якості персоналізованих рекомендацій, пропонуються схожі товари до тих які найбільше зацікавили користувача, користувач має високу неявну оцінку до товару, з цього можемо сформувати рекомендацію - “Вас також може зацікавити”.

5.3 Необхідні доповнення до маркетплейсу

Неперсоналізовані рекомендації - це рекомендації створений шляхом аналізу великого об'єму даних, таких як історія взаємодії користувача, підрахунок релевантності та сортування. Враховуючи великий обсяг даних та довгий час виконання неперсоналізованих рекомендацій, передчасне формування та зберігання рекомендацій є ефективним рішенням. Це дозволяє забезпечити швидкий доступ до рекомендацій для користувачів і поліпшити загальний досвід використання системи рекомендацій. Для цього можливо створити сутність яка формується з типу рекомендації, оголошень які ми рекомендуємо, а також змінної яка буде відповідати за те чи придатні ці рекомендації до використання.

Для формування персоналізованих рекомендацій потрібен збір оцінок. Неявні оцінки взаємодії користувача з контентом ідеально підходять, для даного типу сервісу. Варто створити окремий контролер, методи якого будуть викликатися при певній взаємодії користувача з оголошеннями. А також ці активності користувача, варто зберігати, аби потім можливо було сформувати оцінку, яка охарактеризовує зацікавленість користувача в певному контенті. Оцінки також можливо зберігати окремо, для зручності формування рекомендацій. Для реалізації алгоритму схожості я додав сутність словника та терміну. Терміни зберігають IDF коефіцієнт, що зручно коли ми намагаємося порахувати схожість для нового оголошення. Словник містить всі терміни, і створений для того, щоб можна було мати певну версію набору термінів та їх коефіцієнтів, та формувати новий, наприклад коли сервіс наповнився новими категоріями та унікальними оголошеннями. Збереження попередніх версій дає можливість повернутися до використання попереднього словника в разі якщо новий сформований не відповідає критеріям або не такий ефективний як попередній. Також можемо використати сутність для збереження схожості між оголошеннями. Отже після додавання нових сутностей маємо наступну ER-діаграму(рисунок 5.2).

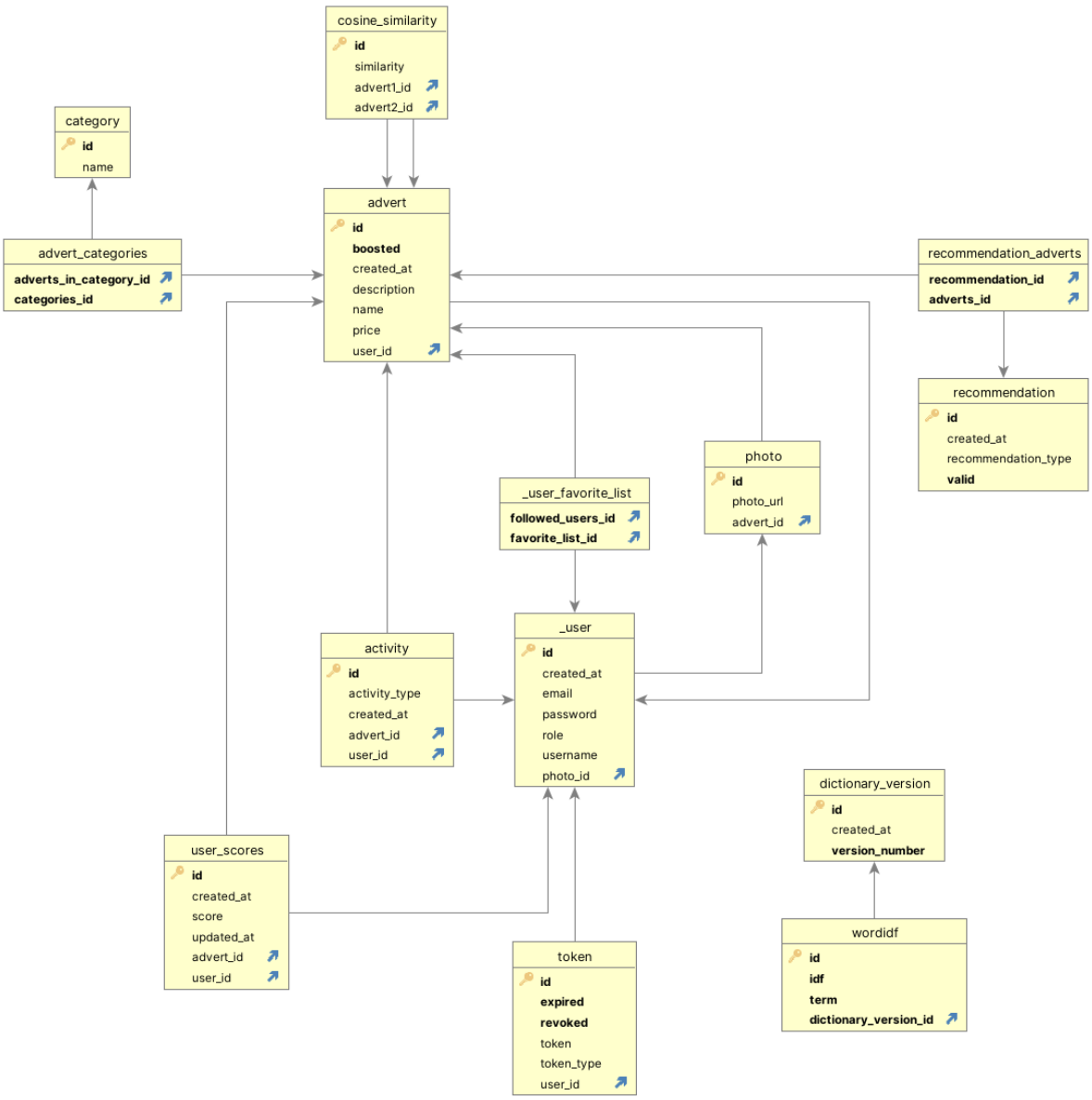

Рис. 5.2 Кінцева ER-Діаграма

Для кожної нової сутності додано JPA-репозиторій. Створено сервіси які відповідають за певну частину рекомендаційної системи: ActivityService - відповідає за бізнес логіку активностей користувача, таких як додавання взаємодії користувача з оголошеннями та його окремими елементами, та пошук певних активностей, та їх видалення. UserScoreService - відповідає за формування неявних оцінок взаємодії користувача з оголошеннями. TFIDFService - відповідає за створення словника та аналіз тексту. CosineSimilarityService - відповідає за обрахування схожості між оголошеннями. RecommendationSystemService - для видачі рекомендацій користувачам.

Розберемо ключові методи з кожного сервісу, TFIDFService:

- analyzeText - формує з тексту, список термінів, в якості аналізатору використовується морфологічний український аналізатор з бібліотеки apache.lucene.
- generateIDF - формує словник термінів на основі оголошень наявних в БД.
- saveDictionary - зберігає словник сформований функцією generateIDF.

CosineSimilarityService:

- findIDF - знаходить значення IDF в словнику.
- calculateTF - вираховує TF для кожного терміну в оголошенні.
- calculateTFIDF - за допомоги функції findIDF та calculateTF обраховує значення TF-IDF.
- calculateCosineSimilarity - обраховує косинусну подібність між двома векторами або оголошеннями .
- cosSimOneToOther - обраховує косинусну подібність між заданим оголошенням та та іншими оголошеннями які знаходяться в спільній категорії.
- cosSimAllToOther - заповнення подібності для всіх оголошень.

ActivityService:

- userOpenAdvertPage - додає активність відкриття сторінки оголошення користувачем.
- userOpenAdvertDescription - додає активність перегляду опису оголошення користувачем.
- userScrollAdvertPhotos - додає активність перегляду фотографій оголошення користувачем.
- findUserActivitiesByAdvert - пошук активностей за оголошенням.
- findAdvertsByUser - пошук активностей за користувачем.

UserScoreService:

- refreshScore - додає або оновлює оцінку між користувачем та товаром. Як видно в лістингу 5.1 кожна активність має свої коефіцієнти які в сумі дають 10 балів, а також кожна активність має властивість зістарюватися. Допоміжний метод calculateAttenuation, вираховує коефіцієнт на який ділиться значення яке припадає на певну активність. Таким чином оновлючи оцінки користувача кожного дня, або при кожній авторизації, ми матимемо актуальну представлення смаків користувачів.
- refreshAllUserScores - оновлення всіх оцінок певного користувача.
- refreshScoresAllForUser - оновлення всіх оцінок в БД.

RecommendationSystemService:

- mostFollowed - видає в якості результату список неперсональних рекомендацій, оголошення які додали до обраного найбільша кількість користувачів.
- anyLatest - видає в якості результату список неперсональних рекомендацій, а саме останні створені оголошення.
- mostScored - видає в якості результату список рекомендацій з найбільшою активністю.
- personalRecommendation - видає в якості результату список персональних рекомендацій, вигляду ID оголошення яке зацікавило користувача та список оголошень які також можуть зацікавити користувача.
- similarAdvert - видає в якості результату список схожих оголошень до заданого.

```
public void refreshScore(Integer user_id, Advert ad) {
    Date currentDate = new Date();
    User user = userService.getById(user id);
    Integer advert_id = ad.getId();
    int scoreOpenPage = 0;
    int scoreOpenDescription = 0;
    int scoreScrollingPhoto = 0;
    UserScores userScore;
    Optional<UserScores> userScoreOp = userScoresRepository.findByAdvert_IdAndUser_Id(advert_id, user_id);
    if (userScoreOp.isEmpty()) {
        userScore = new UserScores();
        userScore.setUser(user);
        userScore.setAdvert(ad);
    } else {
        userScore = userScoreOp.get();
    }
    for (Activity ac : activityService.findUserActivitiesByAdvert(advert id, user id)) {
        switch (ac.getActivityType()) {
            case OPEN_PAGE -> {
                if (scoreOpenPage == 0)
                    scoreOpenPage = (int) (OPEN_PAGE_C / calculateAttenuation(currentDate, ac.getCreatedAt()));
            }
            case OPEN_DESCRIPTION -> {
                if (scoreOpenDescription == 0)
                    scoreOpenDescription = (int) (OPEN_DESCRIPTION_C / calculateAttenuation(currentDate, ac.getCreatedAt()));
            }
            case SCROLLING_PHOTOS -> {
                if (scoreScrollingPhoto == 0)
                    scoreScrollingPhoto = (int) (SCROLL_PHOTO_C / calculateAttenuation(currentDate, ac.getCreatedAt()));
            }
            default -> throw new IllegalStateException("Unexpected value: " + ac.getActivityType());
        }
    }
    userScore.setScore(scoreOpenPage+scoreOpenDescription+scoreScrollingPhoto);
    userScoresRepository.save(userScore);
}
private int calculateAttenuation(Date date1, Date date2){
    return (int) (1 + TimeUnit.DAYS.convert(date1.getTime() - date2.getTime(), TimeUnit.MILLISECONDS));}
```

Лістинг 5.1 Метод оновлення або створення оцінки

6. Приклад роботи маркетплейсу та рекомендаційної системи

Для тестування системи було створено сервіс GenerateUsersAndAdverts в якому напряму через підключення до БД додаються довільні користувачі, заготовлений датасет оголошень та генерується активність користувачів. В якості візуального інтерфейсу використано Swagger UI це інтерактивна документація, яка автоматично генерується зі специфікації OpenAPI та надає веб-інтерфейс, де можна переглядати та тестувати доступні ресурси та операції API.

Завантаживши дані до БД, почнемо тестування:

1) Спробуємо авторизуватися як Адмін. (на рис. 6.1)

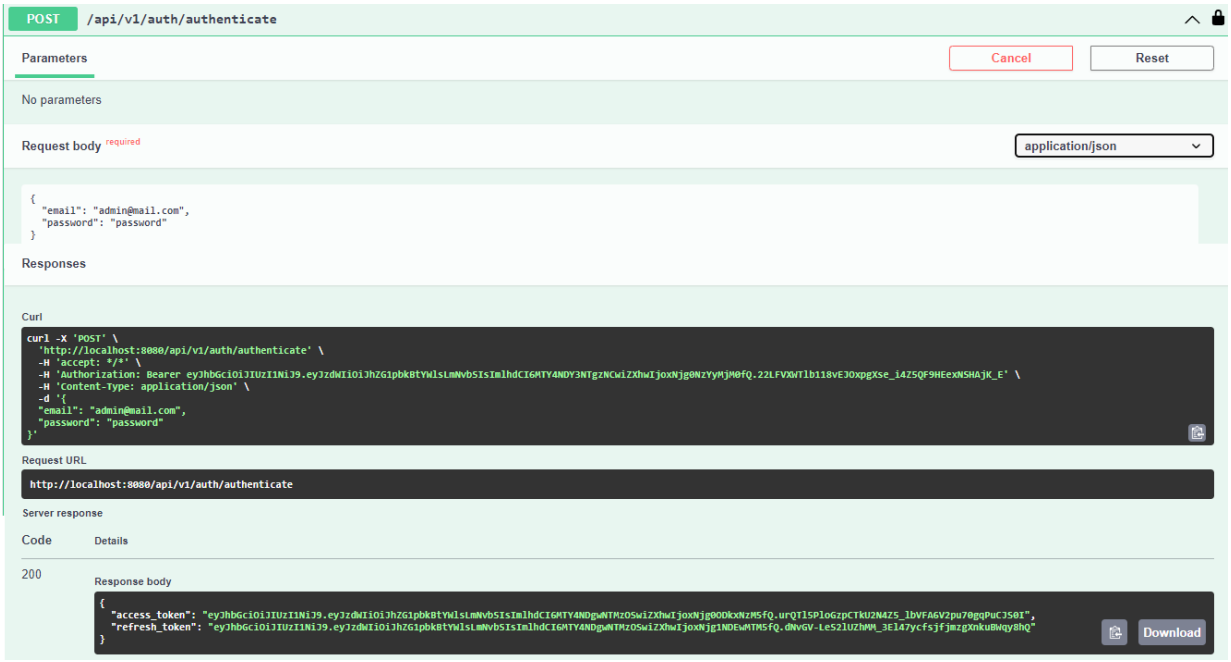

Рис. 6.1 інтерфейс open API з викликом методу авторизації

2) Встановимо згенерований токен (на рис. 6.2)

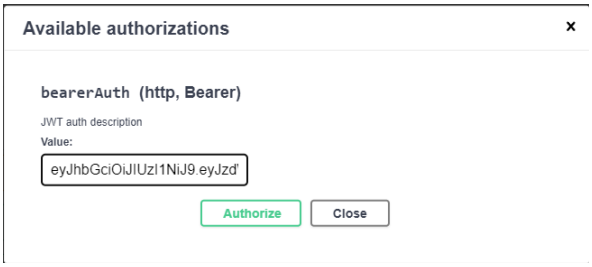

Рис. 6.2 інтерфейс open API з вікном встановлення JWT-токену

3) Згенеруємо підбірку рекомендацій на основі додавання оголошень в обране. (на рис. 6.3)

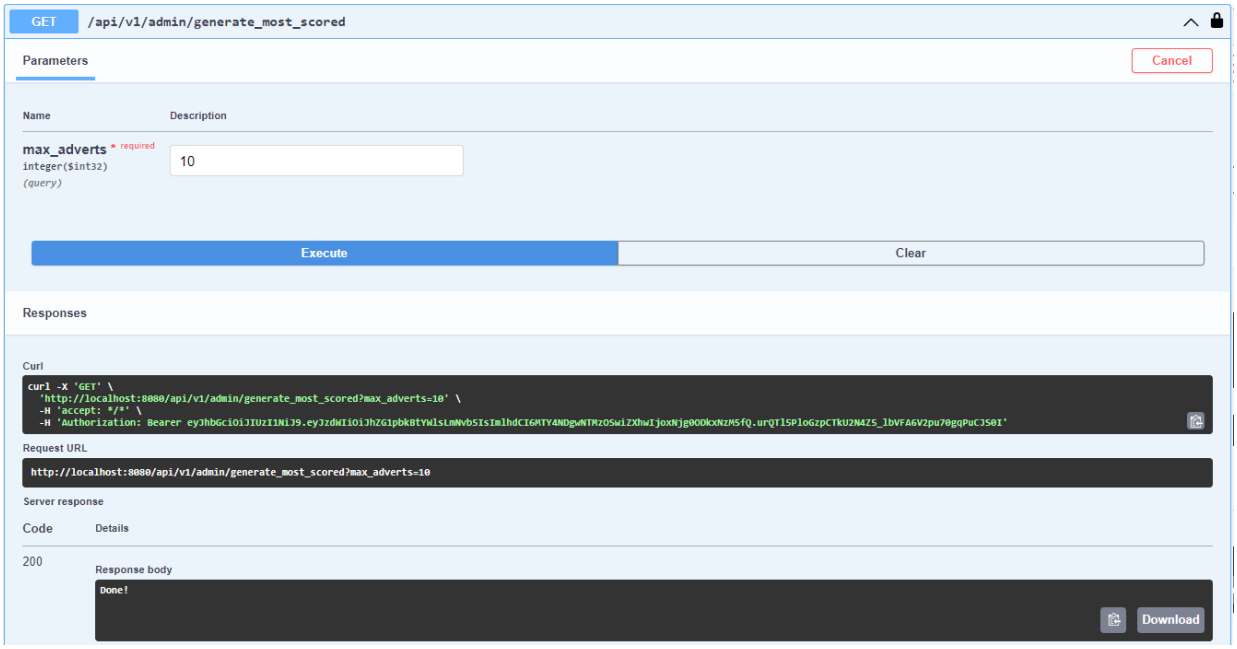

Рис. 6.3 інтерфейс open API з викликом методу генерації рекомендацій

4) Зареєструємо нового користувача та використаємо згенерований токен при реєстрації. (на рис. 6.4)

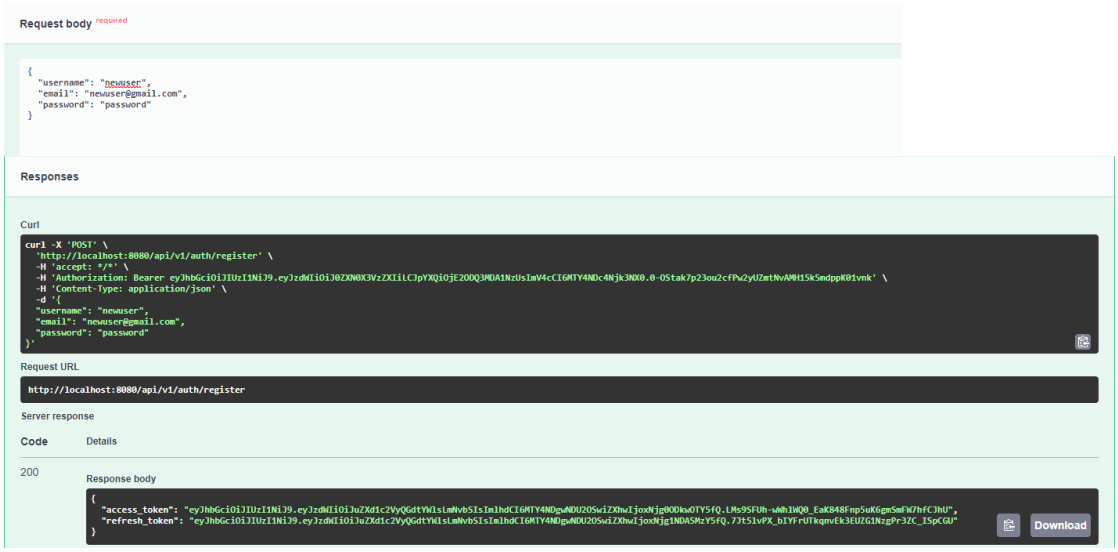

Рис. 6.4 інтерфейс open API з викликом методу реєстрації нового користувача

5) Додамо нове оголошення. (на рис. 6.5)

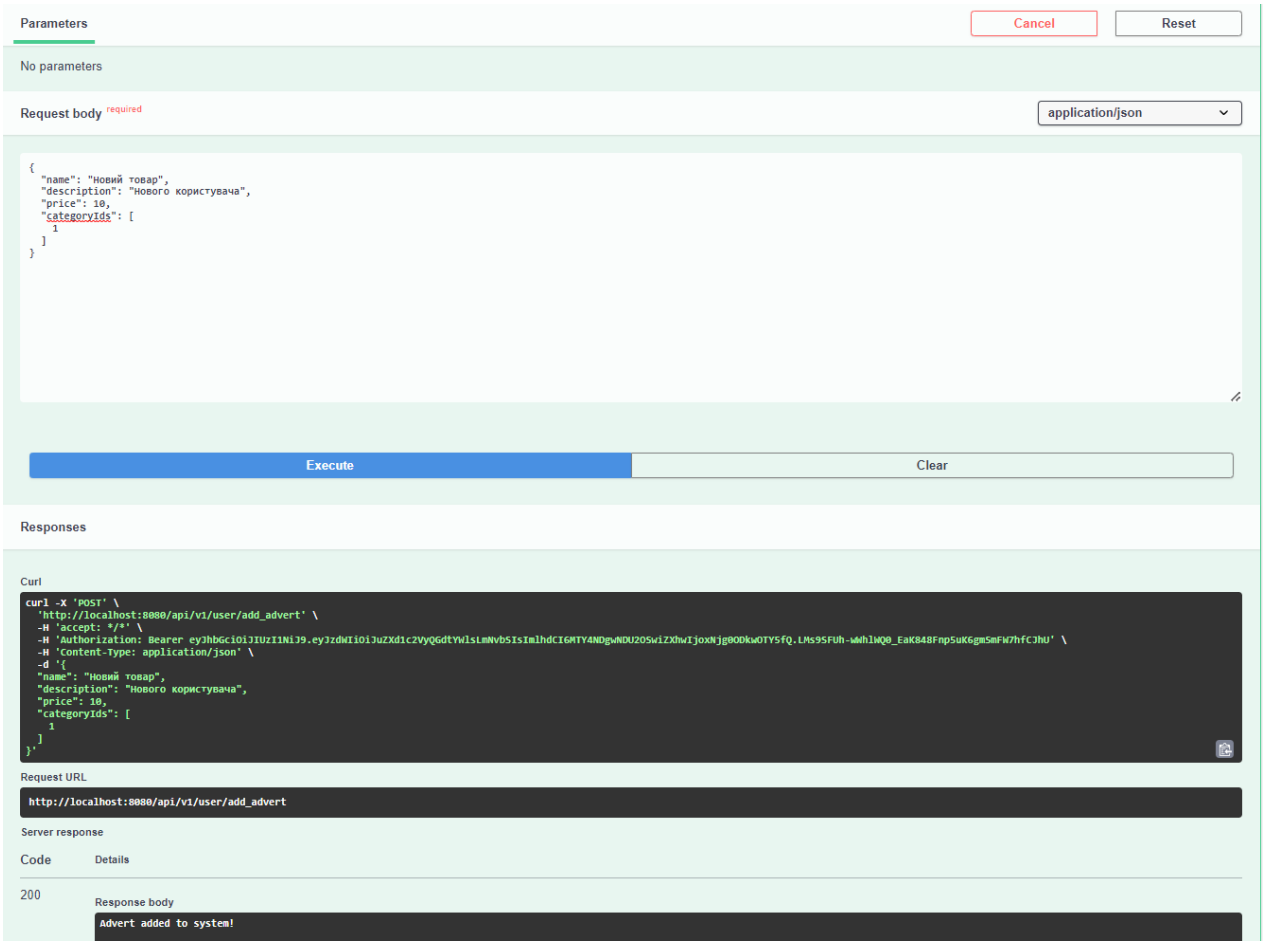

Рис. 6.5 інтерфейс open API з викликом методу створення оголошення

6) Переглянемо оголошення які належать користувачу. (на рис. 6.6)

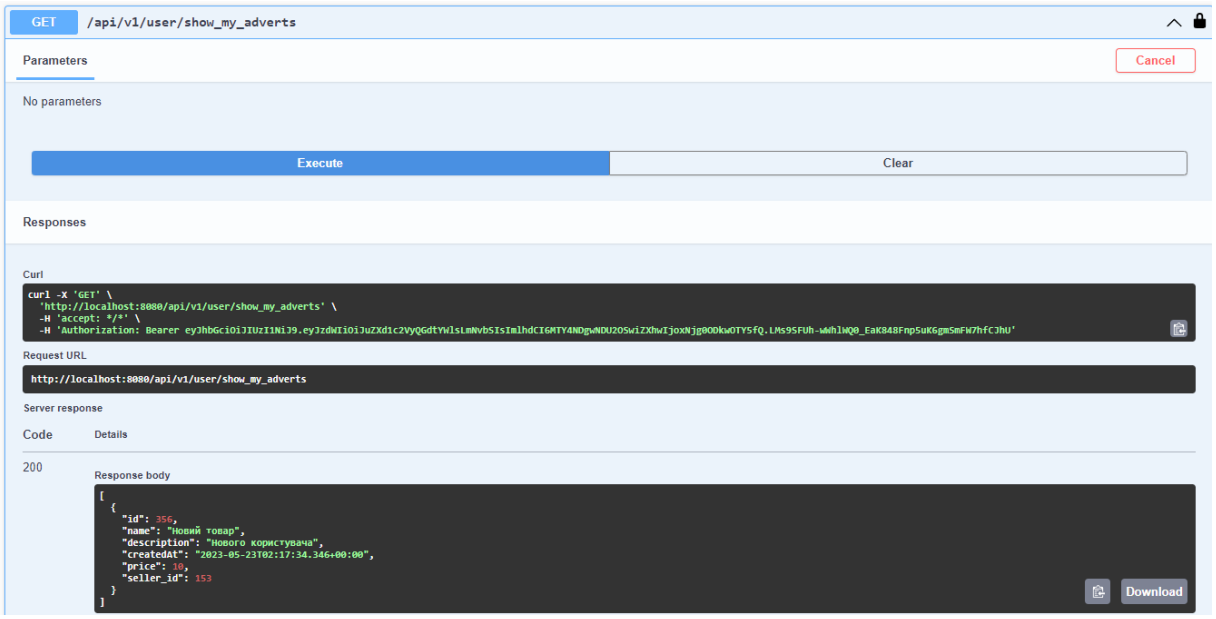

Рис. 6.6 інтерфейс open API з викликом методу перегляду оголошень користувача

7) Спробуємо отримати персональні рекомендації для нового користувача, отримали пустий список, адже користувач ще не взаємодіяв з оголошеннями. (на рис. 6.7)

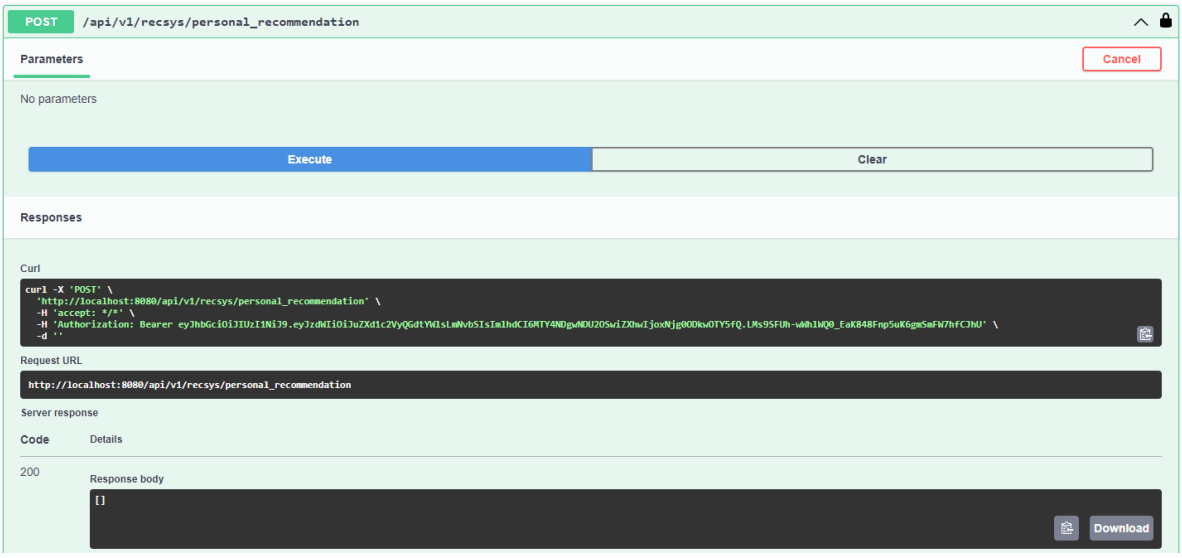

Рис. 6.7 інтерфейс open API з викликом методу генерації персоналізованих рекомендацій (для користувача без взаємодії з оголошеннями)

8) Отримаємо неперсоналізовані рекомендації які створив адміністратор в 3тьому кроці. (на рис. 6.8)

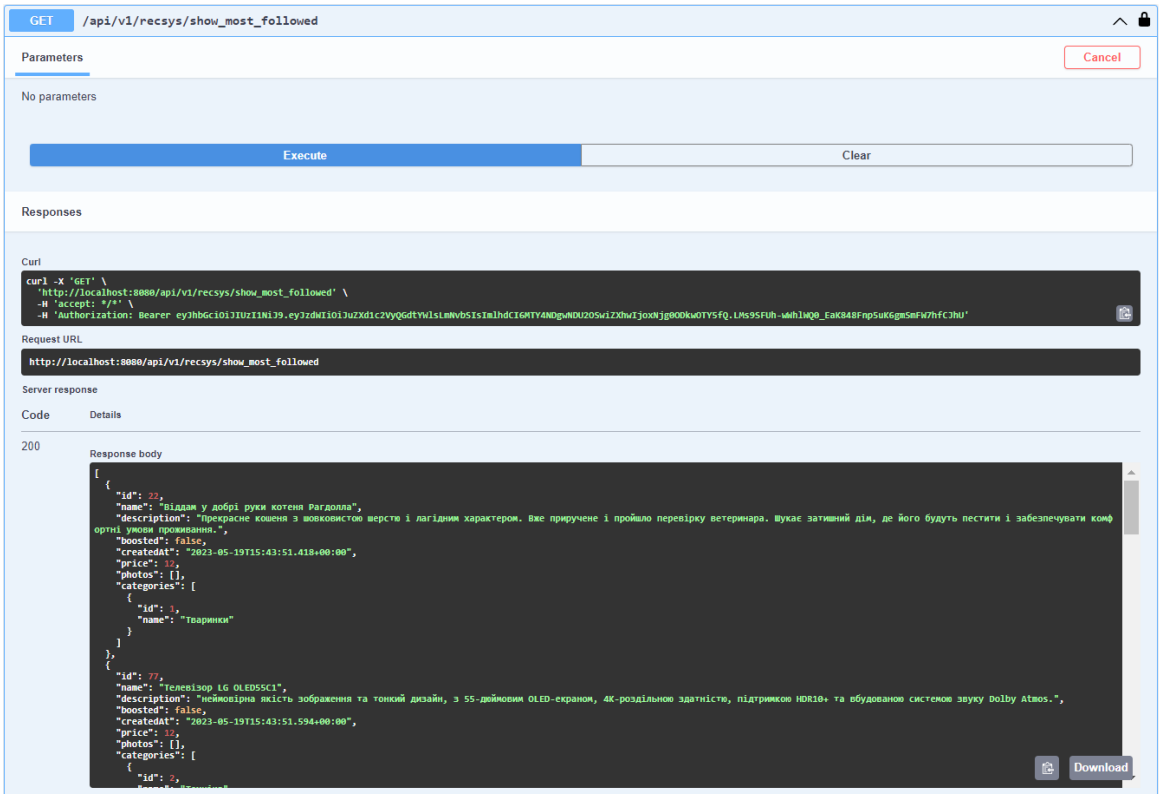

Рис. 6.7 інтерфейс open API з викликом методу перегляду неперсоналізованих рекомендацій

9) Взаємодіємо з оголошеннями за допомогою методів контроллера збирача даних. Вказавши що взаємодіємо з елементом з id 22 та з id 77. (на рис. 6.9)

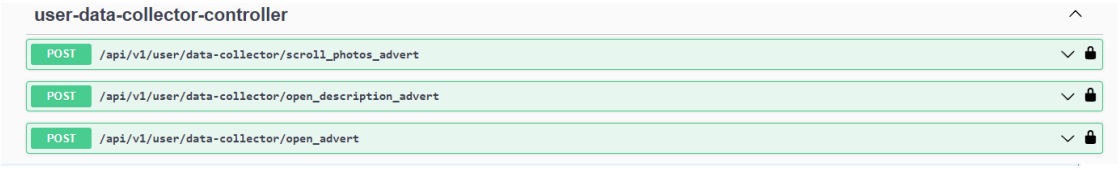

Рис. 6.10 інтерфейс open API з методами створення активностей користувача з оголошенням

10) Спробуємо знову отримати персоналізовані рекомендації, отримуємо в результаті оголошення на основі яких створені рекомендації та 10 подібних оголошень до оголошення з id 22(на рис. 6.10) та id 77(на рис. 6.11).

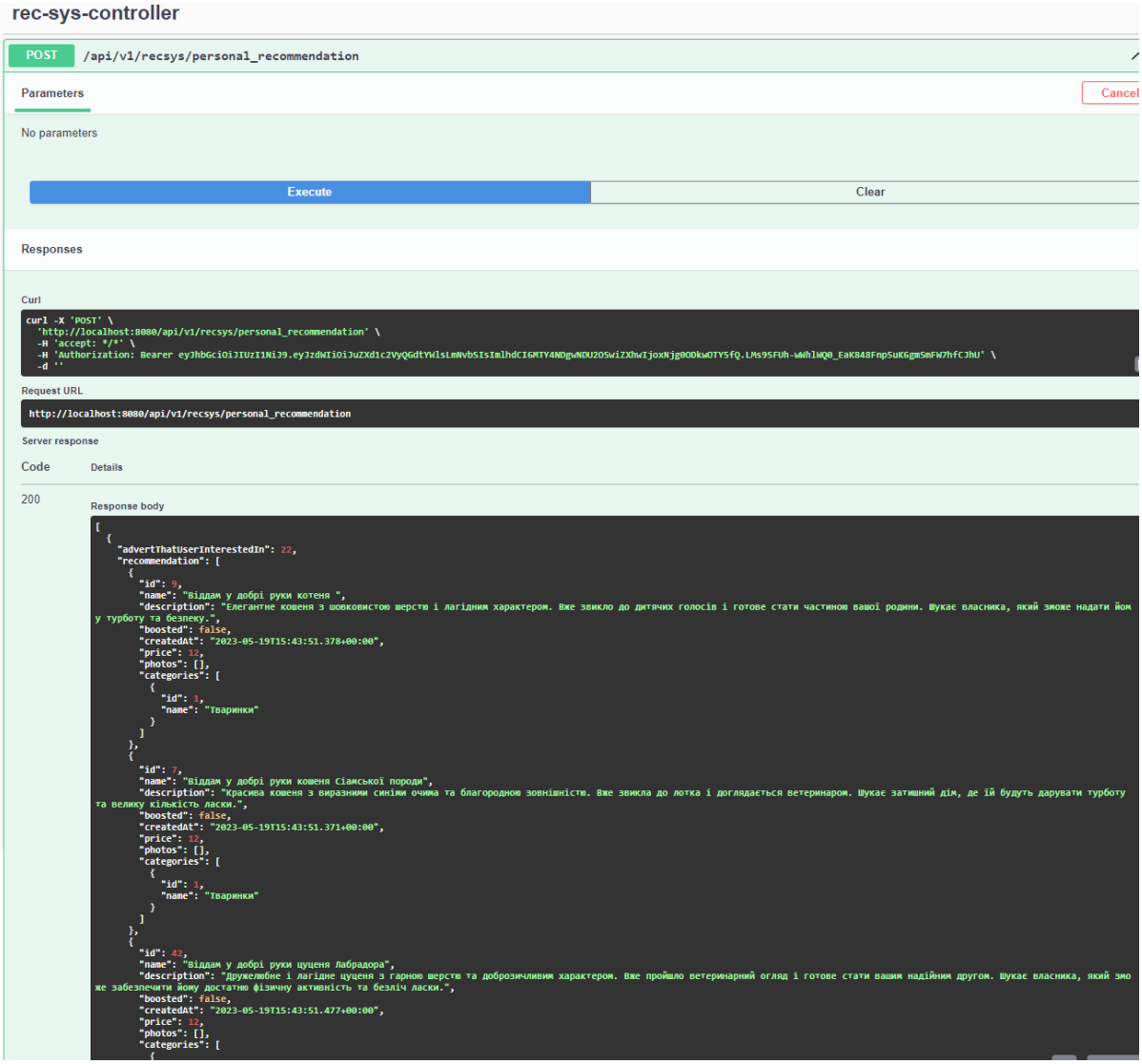

Рис. 6.10 інтерфейс open API з викликом методу генерації персоналізованих рекомендацій (частина рекомендацій на основі зацікавленості користувача в оголошенні з id 22)

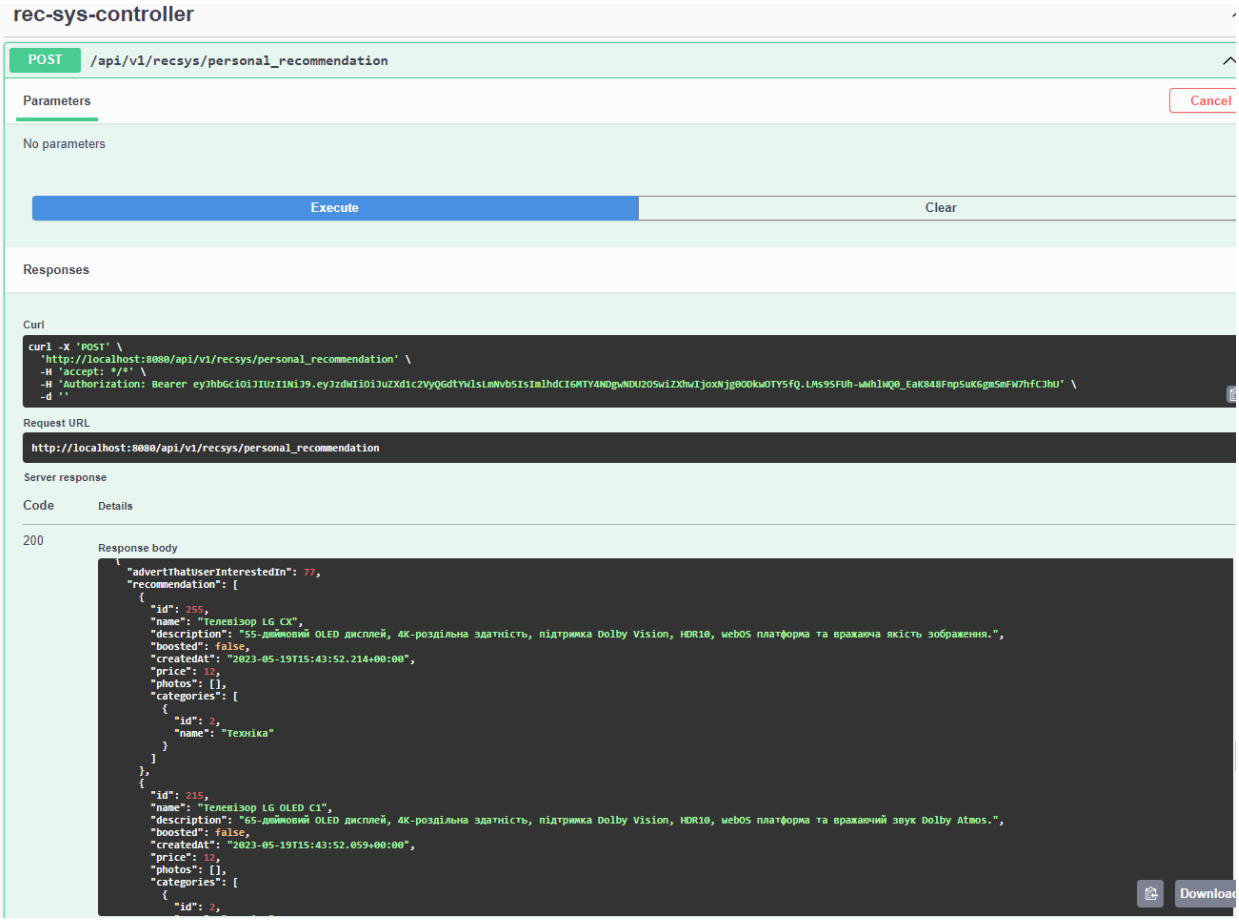

Рис. 6.11 інтерфейс open API з викликом методу генерації персоналізованих рекомендацій (частина рекомендацій на основі зацікавленості користувача в оголошенні з id 77)

7. Висновок

Ця робота присвячена розробці рекомендаційної системи для рекламного маркетплейсу з використанням фреймворку Spring Boot та методів фільтрації контенту. Метою дослідження є створення ефективного та персоналізованого та неперсоналізованого механізму рекомендацій, який покращує користувацький досвід та збільшує взаємодію користувачів з платформою. У процесі розробки було використано фреймворк Spring Boot, що дозволило швидко та ефективно розробити маркетплейс та інтегрувати систему рекомендацій. Крім того, підхід до фільтрації контенту дозволив системі аналізувати характеристики та особливості оголошень і генерувати рекомендації на основі схожості контенту. Розроблена система рекомендацій дозволяє користувачам взаємодіяти з платформою, надаючи релевантні та цікаві оголошення. Персоналізований підхід до рекомендацій сприяє підвищенню задоволеності користувачів та збільшенню кількості успішних транзакцій. Отримані результати підтверджують ефективність використання фільтрації контенту в контексті рекомендаційної системи маркетплейсу оголошень.

8. Джерела

1. Маркетплейс простими словами: що таке, як це працює і яка від нього вигоду. URL: https://xn--90aamhd6acpq0s.xn--j1amh/teoriya/marketpleys-prostymy-slovamy-shcho-take-iak-tse-pratsiuie-i-iaka-vid-noho-vyhoda.
2. Як працюють і на чому заробляють маркетплейси. *Асоціація ритейлерів України – The profile association of retail market players*. URL: https://rau.ua/novyni/chto-takoe-marketplejsy/.
3. Український ринок e-commerce у першому півріччі 2021 року: аналітика hubber. *Впливове медіа про ритейл – RetailersUA*. URL: https://retailers.ua/news/menedjment/12129-ryinok-e-commerce-v-pervom-polugodii-2021-analitika-hubber.
4. «Щодня блокуємо 3 тисячі оголошень після ручної перевірки»: OLX розповів, як створюється безпека в онлайні – офіційний блог olx.ua. *Офіційний блог OLX.ua*. URL: https://blog.olx.ua/25411/shhodnya-blokuyemo-3-tisyachi-ogoloshen-pislya-ruchnoi-perevirki-olx-rozpoviv-yak-stvoryuyetsya-bezpeka-v-onlajni/.
5. Положення про обробку і захист персональних даних – ROZETKA. *Довідковий центр – ROZETKA*. URL: https://help.rozetka.com.ua/p/69-polozhennya-pro-obrobku-i-zahyst-personalnyh-danyh.
6. Kim Falk, Practical Recommender Systems, 2019, 1.2 Taxonomy of recommender systems, ст.15
7. Учасники проектів Вікімедіа. Бульбашка фільтрів – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Бульбашка_фільтрів.
8. Учасники проектів Вікімедіа. TF-IDF – вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/TF-IDF.

9. Working with Relationships in Spring Data REST | Baeldung. *Baeldung*. URL: https://www.baeldung.com/spring-data-rest-relationships.

10. A Guide to JPA with Spring | Baeldung. *Baeldung*. URL: https://www.baeldung.com/the-persistence-layer-with-spring-and-jpa.