

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
РЕКОМЕНДАЦІЙНОЇ СИСТЕМ ПІДБОРУ ТОВАРУ НА ОСНОВІ
ВПОДОБАНЬ КОРИСТУВАЧА
RESEARCH AND SOFTWARE IMPLEMENTATION OF A SYSTEM FOR
PRODUCT SELECTION RECOMMENDATION BASED ON USER
PREFERENCES

Виконав(ла): студент(ка) 2 курсу, групи КНТ-142м
Спеціальності 121 Інженерія програмного
(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

ПЕТЕЛІН Д.Д.

(ПРИЗВИЩЕ та ініціали)

Керівник ГЛАКОВА О.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент ЗЕЛЕНЬОВА І.Я.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПЕТЕЛІНА Данила Дмитровича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація рекомендаційної систем підбору товару на основі вподобань користувача Research and Software Implementation of a System for Product Selection Recommendation Based on User Preferences

керівник проєкту (роботи) к.т.н., ГЛАДКОВА Ольга Миколаївна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 14 ” листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір інструментарію розробки. 3. Дослідження основних рішень. 4. Розроблення архітектури та структури системи.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГЛАДКОВА О.М., доцент		
Нормоконтролер	БЄЛОВА А.В., асистент		

7. Дата видачі завдання « 08 » вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2-3 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	4-5 тиждень	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7-8 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	10-11 тиждень	Додатки
8	Нормоконтроль та рецензування.	12 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

_____ Данило ПЕТЕЛІН
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Ольга ГЛАКОВА
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
70 с., 4 табл., 43 рис., 2 дод., 19 джерел.

РЕКОМЕНДАЦІЙНА СИСТЕМА, ПІДБОР ТОВАРІВ НА ОСНОВІ
ВПОДОБАНЬ КОРИСТУВАЧА, JAVA, SPRING, SPRING-BOOT, ANGULAR,
TS, INTELLIJ IDEA, POSTGRESQL.

Об'єкт дослідження – рекомендаційні алгоритми.

Предмет дослідження – застосування рекомендаційних алгоритмів для
пошуку оптимальних значень кількісних оцінок вподобань користувачами
об'єктів у рамках інтернет-ресурсів.

Мета роботи – проаналізувати предмет дослідження та реалізувати
алгоритм підбору рекомендацій на основі вподобань користувача.

Матеріали, методи та технічні засоби: функціональне програмування,
мови програмування Java, фреймворк SPRING-BOOT, фреймворк для розробки
вебзастосунків ANGULAR, база даних POSTGRESQL, персональний комп'ютер з
процесором Intel Core i5 під управлінням операційної системи Windows.

Результати. Створено максимально оптимізований програмний
вебзастосунок на основі бібліотеки Spring-boot та серверного REST API, за
допомогою якого користувач може шукати потрібний йому товар, та на основі
його вподобань йому будуть відображатись рекомендовані товари.

Висновки. Розроблено вебзастосунок інтернет магазину з
рекомендаційною системою підбору товару.

Галузь використання – програмний продукт для інтернет комерції.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 70 p., 4 tables, 43 fig., 2 appendix, 19 sources.

RECOMMENDATION SYSTEM, PRODUCT SELECTION BASED ON USER LIKES, JAVA, SPRING, SPRING-BOOT, ANGULAR, TS, JS, INTELLIJ IDEA, POSTGRESQL.

The object of the study is the recommendation algorithms.

The subject of the study is a recommendation system for product selection based on user preferences.

The purpose of the work is to research the methods of selecting recommendations for the user and the software implementation of such methods.

Materials, methods and technical means: functional programming, Java programming languages, the SPRING-BOOT framework, the ANGULAR web application development framework, the POSTGRESQL database, a personal computer with an Intel Core i5 processor running the Windows operating system.

Results. A maximally optimized software web application has been created based on the Java EE8 library and the server REST API, with the help of which the user can search for the product he needs, and based on his preferences, recommended products will be displayed to him.

Conclusions. A web application of an online store with a product selection recommendation system has been developed.

The field of use is a software product for online stores.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Аналіз предметної області	11
1.2 Дані та джерела інформації для рекомендаційних систем	13
1.2.1 Типи даних.....	13
1.3 Методи підбору рекомендацій.....	15
1.3.1 Колаборативна фільтрація.....	15
1.3.2 Технологія knowledge-based.....	16
1.3.3 Системи content-based.....	17
1.3.4 Гібридні технології	18
1.4 Огляд існуючих рішень	22
2 Вибір інструментарію розробки програмного забезпечення.....	26
2.1 Вибір засобів для розробки.....	26
2.2 Вибір мови програмування для серверної частини.....	26
2.3 Вибір фреймворку для серверної частини.....	27
2.3.1 Особливості Spring Boot.....	28
2.3.2 Простота управління залежностями	29
2.3.3 Автоматична конфігурація.....	30
2.3.4 Вбудована підтримка сервера додатків - контейнера сервлетів	30
2.4 Вибір фреймворку для розробки інтернет магазину	31
2.5 Вибір середовища розробки.....	32
3 Дослідження основних рішень щодо реалізації компонентів системи	35
3.1 Опис існуючих технологій.....	35
3.2 MIND	37

	7
3.3 PaddleRec.....	38
3.4 Milvus	39
3.5 Впровадження системи.....	42
4 Розроблення архітектури та структури системи	47
4.1 Алгоритм роботи.....	47
4.2 Архітектура вебзастосунку та серверної частини	48
4.3 Структура вебзастосунку та серверної частини на основі використання обраних фреймворків	49
4.4 Функціональні частини	51
4.5 Інтерфейси вебзастосунку	51
4.6 Вікна розробленого застосунку	57
Висновки	60
Перелік джерел посилань	61
Додаток А Код програми.....	63
Додаток Б Слайди презентації	65

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- CBF – фільтрування на основі контенту (Content Based Filtering);
- ССО – колаборативне фільтрування, яке враховує кореляцію схожості входження (Correlated Cross-Occurrence);
- CF – колаборативне фільтрування (Collaborative Filtering);
- IDE – інтегроване середовище розробки (Integrated Development Environment);
- LLR – алгоритм статичної перевірки відношенням правдоподібностей (Log Likelihood Ratio);
- REST – передача репрезентативного стану (Representational State Transfer);
- БД – база даних;
- ОС – операційна система;
- РС – рекомендаційна система.

ВСТУП

За останні десятиліття кількість інформації у інформаційному просторі значно зростає [1]. Завдяки широкому впровадженню Інтернет-технологій у всі сфери життя суспільства, доступність інформації, потребує розробки нових методів пошуку інформації. Найпопулярніші Пошукові системи Google, Yahoo, Altavista не враховують персоналізацію інформації. Системи рекомендацій – це системи, які оброблюють певний тип інформації, системою фільтрів, які рекомендують інформаційні елементи, які можуть зацікавити користувача.

Вхідним даними для рекомендаційної системи є рекомендації користувачів, вона їх агрегує та відправляє до конкретних одержувачів у вигляді рекомендацій. Завдяки цій технології користувачі для знаходження потрібної інформації в Інтернеті витрачають значно менше часу. Системи рекомендацій порівнюють зібрані від користувачів дані та генерують список елементів, які рекомендовані конкретному користувачеві. Такі системи є альтернативою алгоритму пошуку, адже вони надають змогу користувачам швидко знаходити інформацію та предмети, які б ті не змогли знайти самостійно. В електронній комерції рекомендаційні системи використовуються особливо широко[2], [3].

Використання РС все більш розширюється останнім часом на стаціонарну роздрібну торгівлю, центри довідок, пошук програмного забезпечення, наукових статтях і т. п. Це застосування являє собою характерне наданням автоматичних рекомендацій користувачам, на основі вчинених дій (відвідувань, виставлених рейтингів, покупок тощо) та отриманням від них зворотного зв'язку (перехід за посиланнями, замовлення в магазинах).

Найвідомішими інтернет-порталами, які використовують системи рекомендацій належать: Amazon [4] – найбільша в світі за оборотом серед інтернет компаній американська компанія, яка продає послуги та товари, і один з перших, орієнтованих на продаж реальних товарів масового попиту інтернет-

сервіс; eBay – американська компанія, яка надає послуги у сфері інтернет-аукціонів (основне поле діяльності), інтернет-магазинів, миттєвих платежів, управляє веб-сайтом eBay.com та його місцевими версіями в декількох країнах, володіє компанією PayPal і Ebay Enterprise; MovieLens – РС та віртуальний веб-сайт спільноти, що радить фільми для користувачів, рекомендації надаються враховуючи профілі (рейтингів) користувачів та використовують алгоритм спільної фільтрації (collaborative filtering); Rozetka.ua™ – найпопулярніший інтернет-магазин електроніки та побутової техніки в Україні, представництва компанії мають всі області України. Системи рекомендацій є одним із важливих розділів інтелектуальних систем підтримки прийняття рішення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Рекомендаційні системи використовуються у багатьох нині існуючих веб-сайтах. Для своїх користувачів такі системи створюють різні пропозиції, наприклад, товари, відео та аудіо матеріали, інших людей та ін. Такі рекомендаційні системи займаються обробкою великої кількості інформації для підбору контенту на основі вподобань користувача.

Що ж собою представляють рекомендаційні системи? Системи рекомендацій – це програми, котрі визначають що потрібно користувачу, що може його зацікавити та рекомендують цей товар або послугу. Завдяки цим технологіям було вдосконалено зв'язки юзера та вебзастосунку. Інформація надається динамічно та змінюється: для кожного користувача рекомендації створюються окремо, на основі його попередньої активності на даному ресурсі.

У багатьох випадках розвиток цих систем полягає в поліпшенні рекомендаційних алгоритмів. Метою такого прогресу виступає ідея надання найбільш точні рекомендації відвідувачам конкретних сайтів, які задовільняють їх запити. Математичні алгоритми, що лежать в основі рекомендаційних сервісів повинні постійно вдосконалюватись для досягнення цієї мети. Для цього в роботу вступає Data mining (інтелектуальний аналіз даних). Це можна описати схематично так: користувачу сервіс веб-сайту надає набір певних рекомендацій, після чого розробник отримує зворотний зв'язок від користувача, аналізує його на узгодженість даних, рекомендацій, що були до цього інтересам користувачів, математична модель перенавчається, і знову видає набір рекомендації і далі знов по колу. Для систем, які надають рекомендації ці математичні алгоритми розвиваються за наступними критеріями:

– кількість контенту, який вже рекомендовано для користувача - його повинно бути вдосталь, щоб потім було що та з чого рекомендувати;

- кількість інформації про рекомендаційний контент (наприклад, до якої категорії відноситься товар, його назва, діапазон ціни) - чим більше інформації, тим точніше можна обрати потрібний контент;
- кількість інформації про користувача (ім'я, вік, стать, країна проживання) - знов ж таки, чим більше інформації, тим точніше і конкретніше буде підібрана рекомендація;
- зручність інтерфейсу для користувача - чим простіше та комфортніше для відвідувача, тим більше інформації ми отримаємо про його реакції на наші поради.

Всесвітня мережа представляє методи збору інформації, тим самим значно спрощує використання думок громадськості за допомогою CF. Однак чим більший об'єм інформації тим складніше реалізувати ці можливості. Адже, поведінку одних досить чітко можна підставити під визначену модель поведінки, в той час як поведінка інших абсолютно хаотична та непередбачувана. Саме вони мають вплив на зміщення результатів РС, що в свою чергу знижує її ефективність. Наступний приклад: користувачі використовують систему рекомендацій для власної наживи, наприклад, для навмисного підвищення рейтингу одного з їх продукту по відношенню до іншого. Тобто вони мають змогу лишати не тільки позитивні відгуки про певний товар, але також можуть написати негативні відгуки про товари або контент своїх конкурентів. Також великим рекомендаційним системам притаманна така проблема як масштаб.

З невеликою кількістю даних добре справляються традиційні алгоритми. Та чим більше обробляється інформація, тим складніше отримати необхідні результати. Системи рекомендацій, що працюють з неявно визначеними для певного користувача оцінок, можуть провокувати різні проблеми, як захист персональних даних та конфіденційність користувачів. Ще присутня одна проблема - це проблема холодного старту, що ділиться на два види: холодний старт по відношенню до користувача (що показувати людині, яка щойно

zareestruvalasja?) або по відношенню до сайтів (для кого можна рекомендувати щойно створені вебзастосунки?). Область застосування рекомендаційних систем не дуже велика, проте важлива для сучасної мережі:

- рекомендація товару в інтернет-магазинах – напевне, це найбільш актуальний варіант на сьогоднішній день; РС пропонують в інтернет магазинах тільки ті товари користувачу, які на думку алгоритму мають бути найбільш цікавими йому, опираючись на різні фактори;

- рекомендація медіа-контенту – сервіси по типу Netflix, YouTube рекомендують авторизованим користувачам контент опираючись на їх попередню історію переглядів або на рейтинг того чи іншого медіа-контенту;

- рекомендація новин – система радить «подібні» матеріали на ті, що були прочитані раніше, опираючись на подібності ключових слів або, знову ж таки, на виборі користувачів у яких схожі інтереси.

1.2 Джерела інформації та дані для систем рекомендацій

1.2.1 Типи даних

Рекомендаційні системи використовують різних типів дані, проте їх можна звести до наступного вигляду: користувачі та продукти. Продукти є предметами, що рекомендуються.

Продукти можна характеризувати використовуючи їх властивостями. Наприклад, одяг ми можемо розділити на жіночий та чоловічий, він включає в себе певні характеристики(розмір, колір). З іншого боку клієнт може бути як задоволений так і розчарований придбаним одягом. Це також одна з характеристик продукту. Системи рекомендацій використовують усі ці характеристики по різному, що залежить від алгоритму. Часто, дані про характеристики товару використовуються алгоритмами кластеризації та

алгоритмами фільтрування за контентом (CBF). Властивості продукту є його ознаками, або опціями. У продукта обов'язково повинен бути ідентифікатор.

Користувач виступає об'єктом рекомендацій. Користувач має певний профіль для персоналізованих рекомендацій. У профілі користувача може бути різна інформація (вік, стать). Профіль користувача також розглядається як список оцінок, за допомогою яких він оцінив кілька товарів. Алгоритм кооперативної фільтрації (CF) використовує останню характеристику. Для користувача є обов'язковим його ідентифікатор. На рисунку 1.1 показано відношення між користувачами та продуктами.

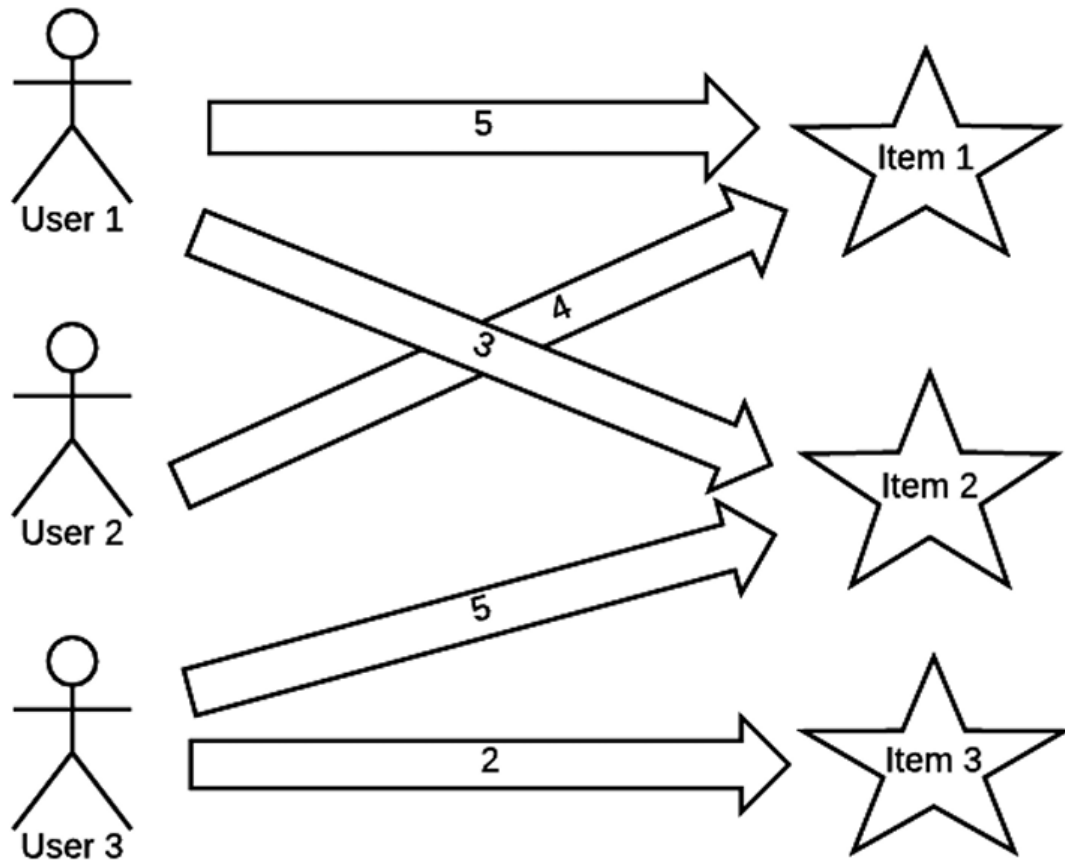


Рисунок 1.1 – Відношення між користувачами та продуктами

Користувачі придбають продукти та лишають чіткий відгук про покупки за шкалою від одного до п'яти. Перший користувач оцінив перші два продукти рейтингом 5 і 3 відповідно. Перший користувач оцінив продукт 1 на 4. Третій оцінив продукт 2 та 3 на 5 та 2 відповідно. Рекомендаційна система використовує ці дані для розрахунку рекомендацій продуктів, котрі користувач ще не переглядав та не купляв.

1.3 Методи підбору рекомендацій

Умовно, сьогодні існуючі системи рекомендацій контенту на сайті можна поділити на чотири типи.

1.3.1 Колаборативна фільтрація

Колаборативна фільтрація (collaborative filtering) [5], [6] є найпопулярнішою технологією. Вона забезпечує рекомендації контенту відштовхуючись від оцінок інших користувачів. Алгоритми котрі підбирають контент в такій схемі засновані на машинному навчанні [7]. Спочатку система дізнається, що вона може дізнатися про відвідувача. Потім шукає користувачів з подібними інтересами, для обчислення прогнозу. Після цього, для конкретного користувача вона підбирає набір рекомендацій та ранжує їх на підставі даних про CTR сторінок або розділів платформи. Завдяки кластерам та закономірностям користувачів можна збільшити залученість користувача до контенту. Недоліком такого підходу є те, що потрібно постійно отримувати інформацію про те, як саме взаємодіє кожен користувач із сайтом користуючись фільтрами. Також важливо мати велике за кількістю ядро користувачів та постійно проводити аналіз маси показників: - кількість людей, які перейшли за посиланням; - відсоток доскроллу; - час перегляду кожної одиниці контенту; - та інші фактори.

1.3.2 Knowledge-based технологія

На рисунку 1.2 зображено схему технології knowledge-based.

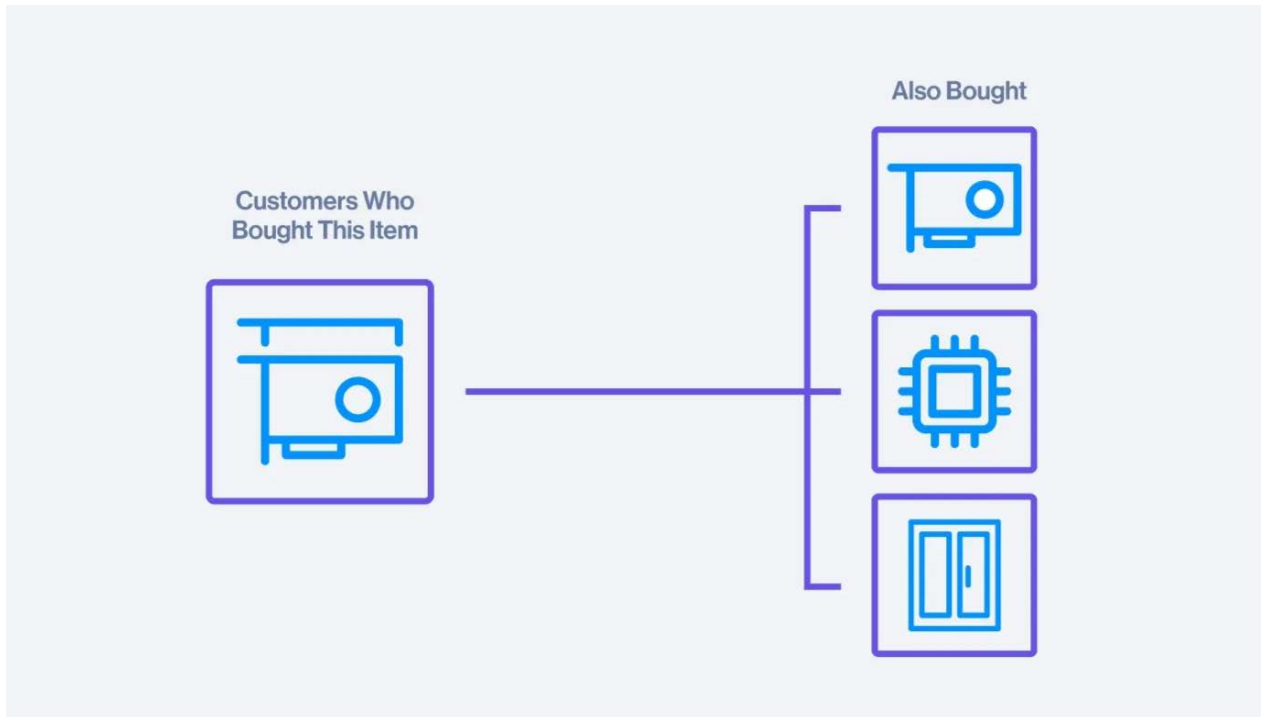


Рисунок 1.2 – Опис технології knowledge-based [5]

Такі рекомендаційні системи засновуються на знаннях про ту чи іншу предметну область (knowledge-based). Об'єднання об'єктів відбувається не за принципом подібності, а на підставі спільності категорії. Сюди ж відноситься товарний асортимент магазину. Підхід knowledgebased не випадково частіше використовується на сайтах еcommerce [8], а не на тематичних проєктах. Простий приклад: якщо ви щойно переглядали iPhone 13 на сайті популярного інтернет-магазину, то вам негайно буде запропоновано плівку на екран, захисний чохол, портативний зарядний пристрій та інші підходящі до нього аксесуари. Тобто товари які належать до інших категорій, проте підходящі до конкретною марки та навіть моделі.

1.3.3 Системи content-based

Засновані на контенті системи (content-based) – є більш узагальненою реалізацією минулої технології. Вона здійснює підбір подібних об'єктів. Точність таких рекомендацій часто прагне до нуля. Намагаючись залучити нового користувача, сайт ніби намагається тикнути палець у небо, «підкидаючи» контент випадковим чином. Програмне забезпечення завдяки більш складній реалізації та більш ретельній обробці алгоритмів тематична фільтрація здатна давати рекомендації на основі поведінки користувача на інших сайтах. В ідеалі технології, що базуються на контенті, здатні проаналізувати інформацію про сайти та блоги, які читач відвідав перед переходом на певний сайт.

Однак в житті все реалізовано більш примітивно. Успіхами роботи з content-based також не можуть похвалитися великі торгові майданчики.

Якщо ви попросите систему доставити вам термокружку, вона слухняно видасть очікуваний результат. Нижче з'явиться блок рекомендацій, покликаний показати, що ще люди котрі, які, як і ви, шукали або купляли собі термокружку. Судячи з інформації в блоці, покупцеві не цікаві ні рюкзаки, ні наметами, ні складні ножі, ні іншими товари для активного відпочинку та туризму. виправити цю ситуацію можна завдяки застосуванню сучасних рекомендаційних сервісів. Вони збирають геоположення читачів, реферер, час читання, глибину скролла, кількість переглянутих сторінок, віральність, прибутковість і кількість переходів. Та навіть дозволяють відстежити, наскільки популярні серед користувачів матеріали тих чи інших авторів.

На рисунку 1.3 зображена схема систем content-based.

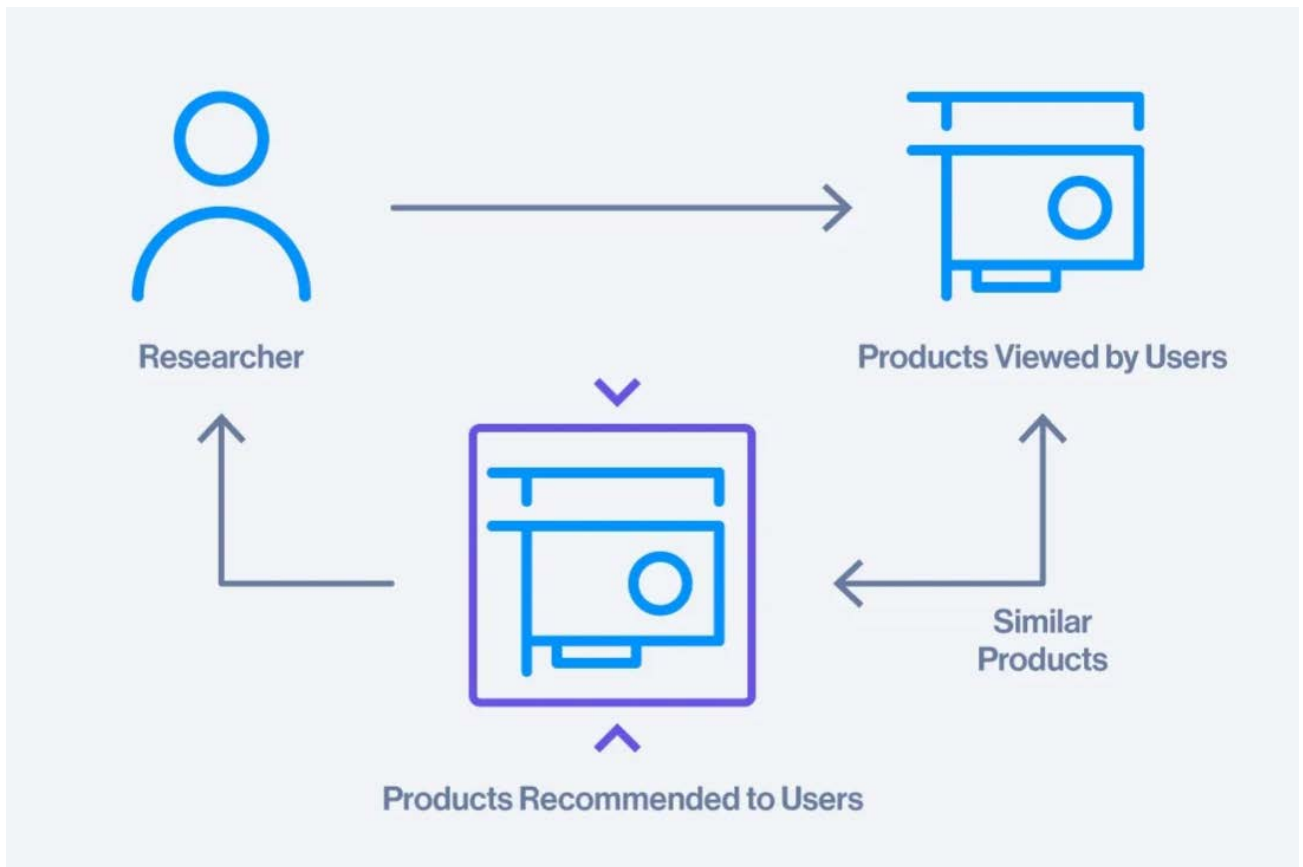


Рисунок 1.3 – Системи content-based [6]

1.3.4 Гібридні технології

Системи рекомендацій що засновані на гібридних технологій [9] дуже важкі в плані розробки. Проте є одними з найрозумніших та точних. Тут використовуються найскладніші алгоритми, котрі постійно удосконалюються. В основі такого підходу лежить ідея «взяти найліпше від кожного». Очевидно, що коштувати таке рішення буде дорожче, а дозволити використовувати його можуть лише великі корпорації типу Netflix. Нетфлікс здійснює підбір рекомендацій включаючи усі ознаки - популярні на даний момент часу, подібні переглянутим, схожі за жанрами. На рисунку 1.4 зображена схема роботи гібридної технології РС.

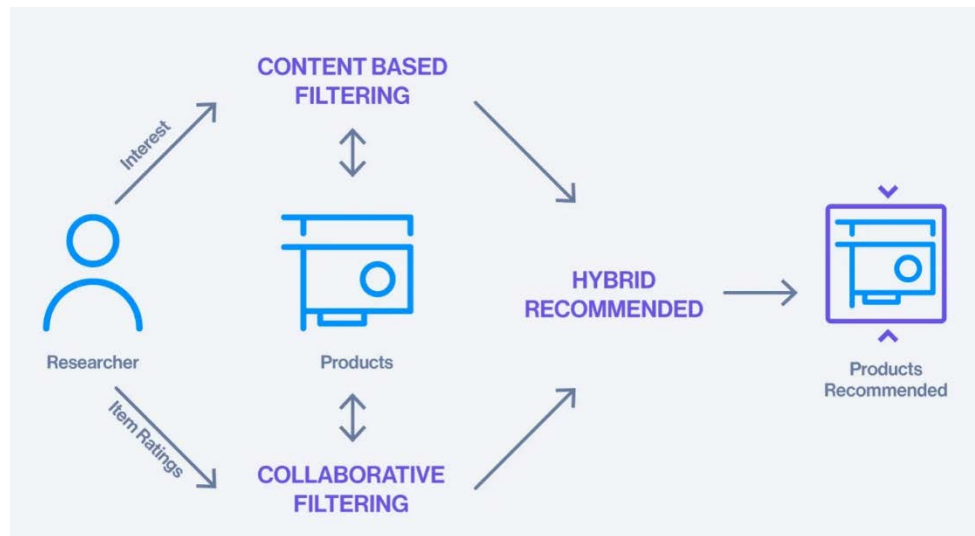


Рисунок 1.4 – Опис гібридної технології РС [9]

Гібридна РС класифікується на 7 підходів для побудови гібридної рекомендаційної системи. Зважена РС зображена на рисунку 1.5.

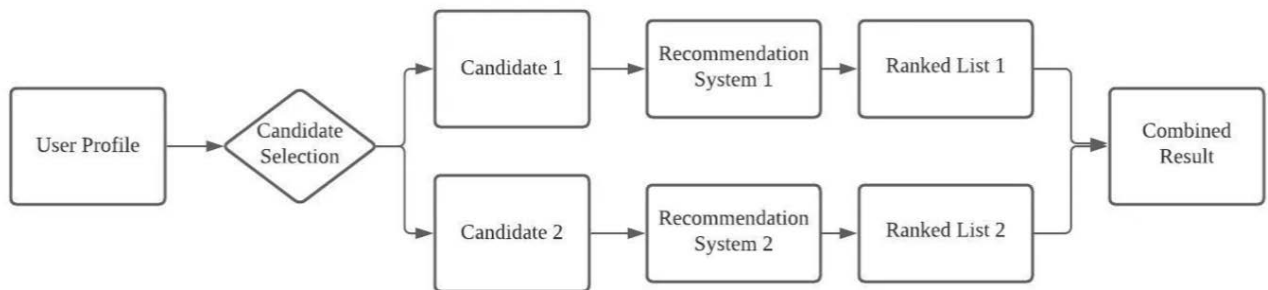


Рисунок 1.5 – Зображення зваженої гібридної системи рекомендацій автора

У системі зважених рекомендацій ми можемо визначити кілька моделей, які можуть добре інтерпретувати набір даних. Система зважених рекомендацій візьме вихідні дані з кожної моделі та об'єднає результат у статичні зважування, вага яких не змінюватиметься в поєднанні та тестовому наборі.

Наприклад, ми можемо об'єднати модель на основі вмісту та модель спільного фільтрування предметів, і кожна з них матиме вагу 50% для остаточного прогнозу.

Перевага використання зваженого гібриду полягає в тому, що ми інтегруємо кілька моделей для підтримки набору даних у процесі рекомендацій лінійним способом.

РС перемикавання вибирає єдину систему рекомендацій на основі ситуації (рис. 1.6). Модель використовується для створення конфіденційного набору даних на рівні елемента, ми повинні встановити критерії селектора рекомендацій на основі профілю користувача або інших функцій.

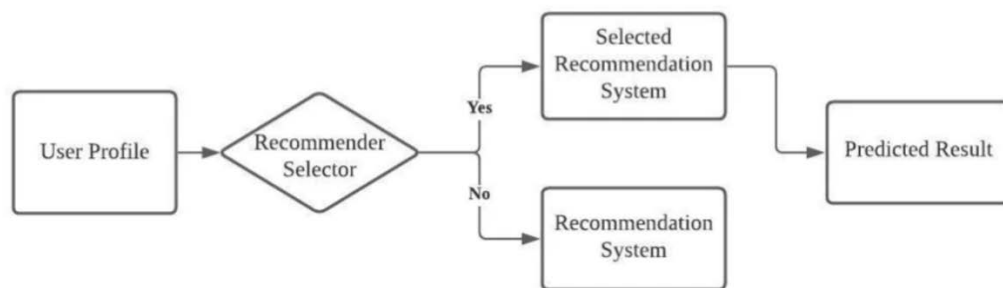


Рисунок 1.6 – Зображення системи рекомендацій перемикавання за автором

Гібридний підхід перемикавання вводить додатковий рівень у рекомендаційну модель, яка вибирає відповідну модель для використання. Система рекомендацій чутлива до сильних і слабких сторін складової моделі рекомендацій.

Змішаний гібридний підхід спочатку використовує профіль користувача та функції для створення різних наборів даних-кандидатів (рис. 1.7).

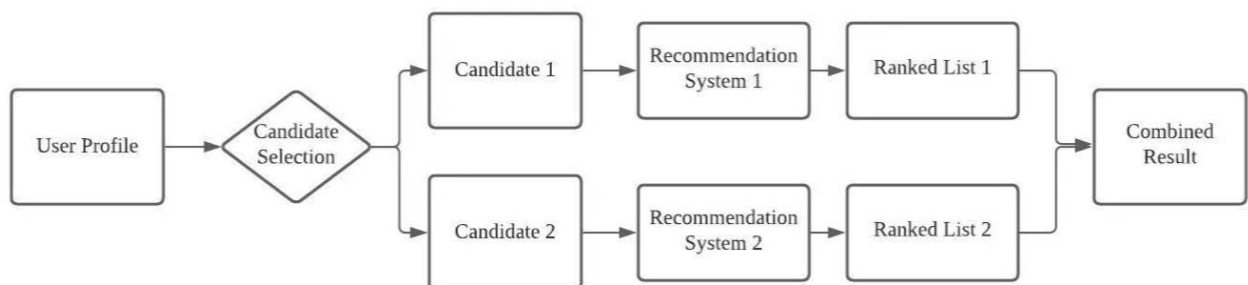


Рисунок 1.7 – Зображення системи змішаних рекомендацій

Система рекомендацій відповідно вводить інший набір кандидатів у модель рекомендації та комбінує прогноз для отримання рекомендації результату.

Змішана гібридна система рекомендацій здатна надавати велику кількість рекомендацій одночасно та адаптувати частковий набір даних до відповідної моделі, щоб мати кращу продуктивність.

У гібридній комбінації функцій ми додаємо до системи віртуальну модель рекомендацій, яка працює як розробка функцій щодо вихідного набору даних профілю користувача (рис. 1.8).

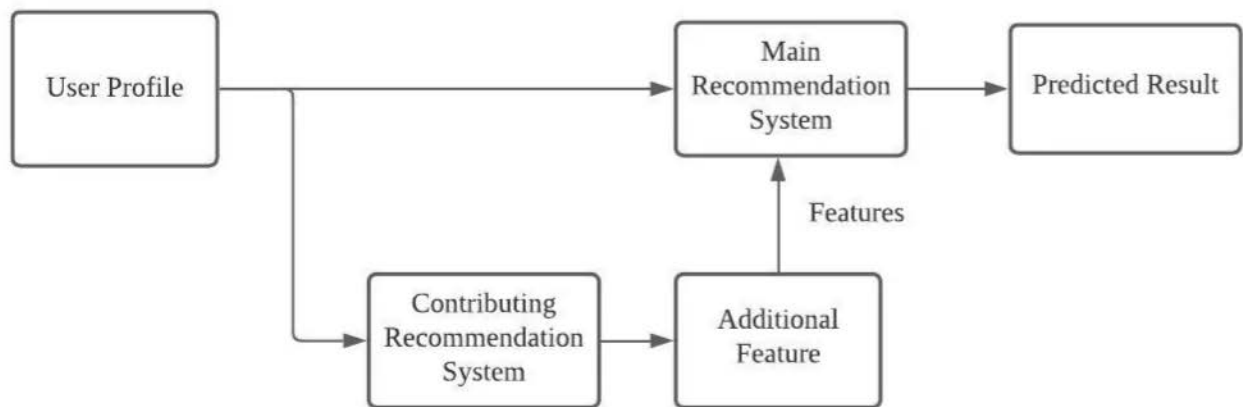


Рисунок 1.8 – Система рекомендацій комбінацій

Наприклад, ми можемо додати функції спільної моделі рекомендацій до моделі рекомендацій на основі вмісту. Гібридна модель здатна розглядати спільні дані з підсистеми, спираючись виключно на одну модель.

Каскадний гібрид (рис. 1.9) визначає сувору ієрархічну структуру системи рекомендацій, так що основна система рекомендацій дає первинний результат, а ми використовуємо вторинну модель для вирішення деяких незначних проблем первинного результату, як-от розірвання нічієї в оцінці.

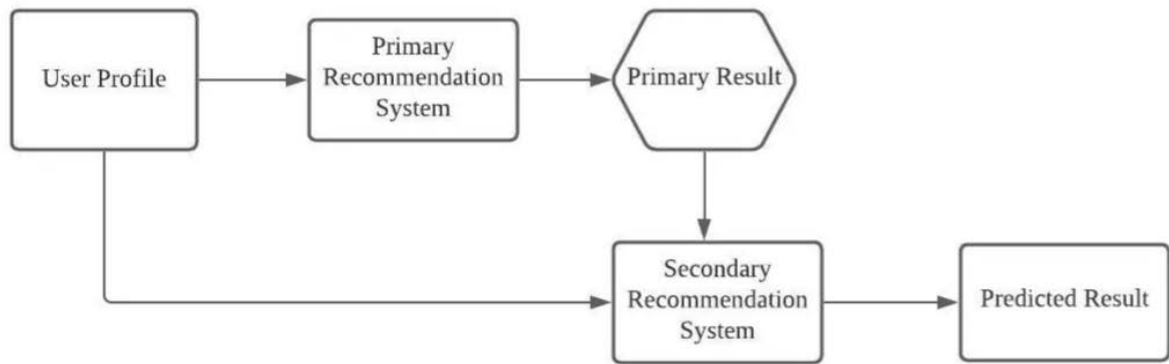


Рисунок 1.9 – Зображення каскадної системи рекомендацій

На практиці більшість набору даних є розрідженими, вторинна модель рекомендацій може бути ефективною проти проблеми рівних балів або проблеми з відсутністю даних.

Гібрид мета-рівня подібний до гібриду розширення функцій, так що допоміжна модель надає розширений набір даних для основної моделі рекомендацій. На відміну від гібриду розширення функцій, мета-рівень замінює вихідний набір даних вивченою моделлю з допоміжної моделі як вхідні дані для основної моделі рекомендацій.

1.4 Огляд існуючих рішень

Домашня сторінка – це перше, що бачить користувач коли відвідує веб-сайт використовуючи прямий трафік. Оскільки ці відвідувачі не обов'язково шукають щось конкретне, рекомендації на головній сторінці мають проінформувати клієнтів про останні пропозиції, знижки та демонстрацію популярних продуктів. Популярні продукти - це базова, але досить потужна логіка рекомендацій, яка майже у всіх магазинах електронної комерції чудово працює. Популярність продукту найлегше визначити за кількістю придбаних користувачем товарів. Проте, більш вдосконалені системи включають також інші дані, наприклад: кліки, перегляди, тощо в свою логіку, для подання ще більш

точних рекомендацій. Для змістованих сайтів (сайти новин, відео порталами) інші чинники, такі як час, витрачений на сторінку, прокручуваний відсоток, або час перегляду відео, також можуть бути важливими факторами популярності. На веб сайтах, котрі пропонують певну продукцію ця логіка більш спрощена. На (рис 1.10) приведено приклад рекомендації бестселлерів.

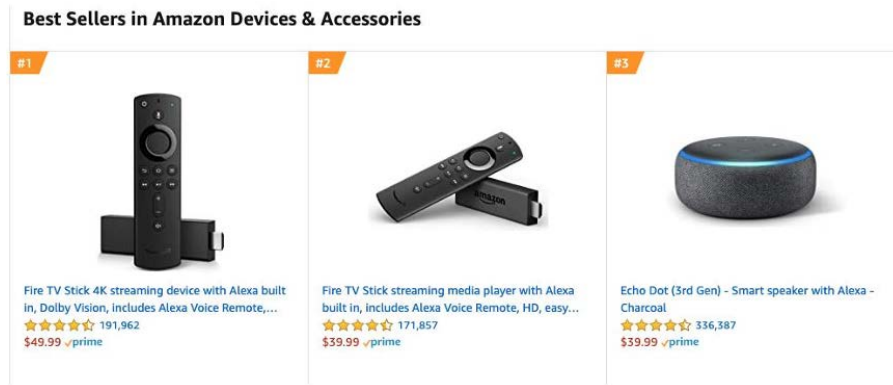


Рисунок 1.10 – Приклад рекомендації бестселерів

Для того, щоб побачити товари які інтересують користувача йому не обов'язково заходити на сайт інтернет магазину. Багато магазинів розміщують свої рекомендаційні системи на інших сайтах. Наприклад: підбір рекомендацій годинників на сайті duplichecker.com (рис. 1.11), або рекомендація від інтернет магазину Comfy[10] на сайті Baeldung (рис. 1.12).

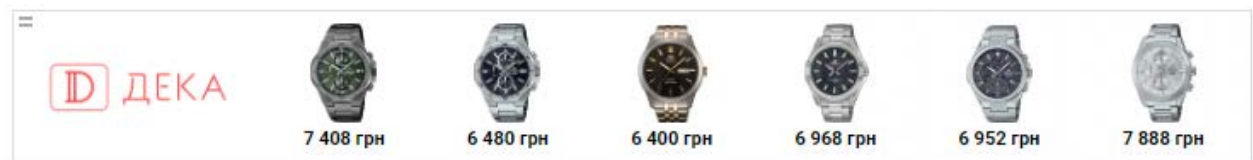


Рисунок 1.11 – Приклад рекомендації інтернет магазину Дека на сторонньому сайті [11]



Рисунок 1.12 – Приклад рекомендації інтернет магазину Comfy на сторонньому сайті

Рекомендації такого плану є рекламними рекомендаціями.

Персоналізовані рекомендації. Списки показують персоналізовані рекомендації різних продуктів для кожного користувача на основі його попередніх покупок та історії веб-перегляду. Однак персоналізовані рекомендації потребують значної кількості даних про поведінку користувачів, яких система не має для нових відвідувачів (це відоме як проблема холодного запуску RS). Щоб забезпечити персоналізований досвід користувача для нових відвідувачів веб-сайту, а також задовільнити нових відвідувачів, найкращим рішенням у галузі є визначення «іншого випадку», коли про користувача мало чи нічого не відомо. На практиці це означає, що система визначає, чи достатньо даних про конкретного користувача, щоб надати йому персоналізовані рекомендації щодо товарів. Якщо даних немає, система рекомендацій "повертається" до більш загальних рекомендацій (наприклад, популярні продукти) або логіки, що фільтрує продукти за категорією, відображаючи в рекомендаціях тільки продукти з категорії, в якій були переглянуті товари користувачем в даній або минулій сесії. Слід з великою обережністю планувати сценарій альтернативних рекомендацій та використовувати максимально ефективні дані для забезпечення точних рекомендацій для нових користувачів.

На рис. 1.13 наведено приклад персоніфікованих рекомендацій.

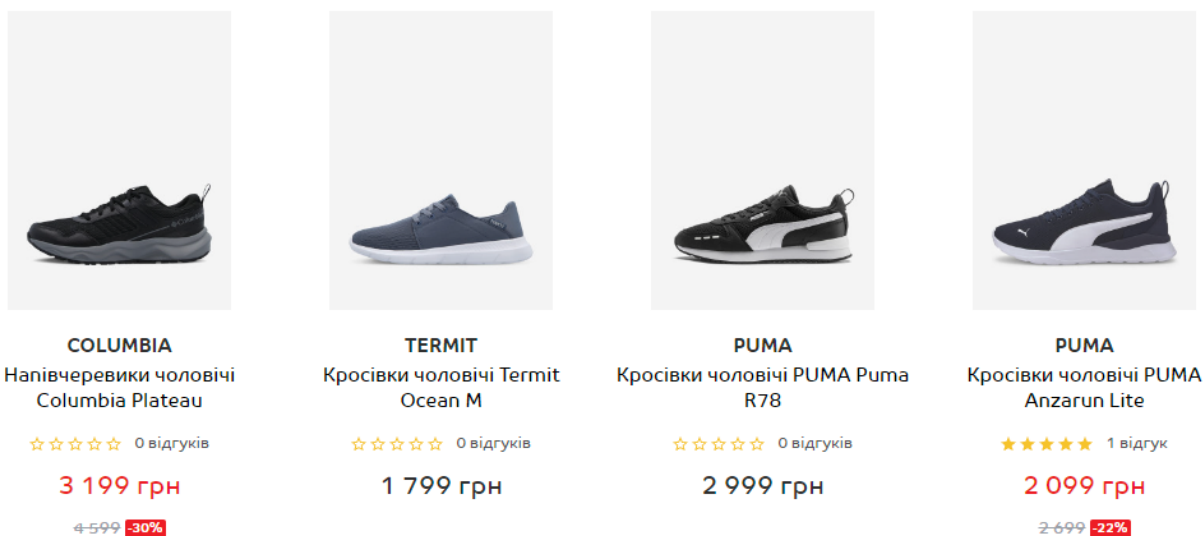


Рисунок 1.13 – Приклад персоніфікованих рекомендацій

2 ВИБІР ІНСТРУМЕНТАРІЮ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір засобів для розробки

Обираючи інструменти для розробки будь-якого застосунку для початку потрібно зрозуміти складність майбутнього проєкту, адже від цього залежить який інструмент краще обрати (рис. 2.1).

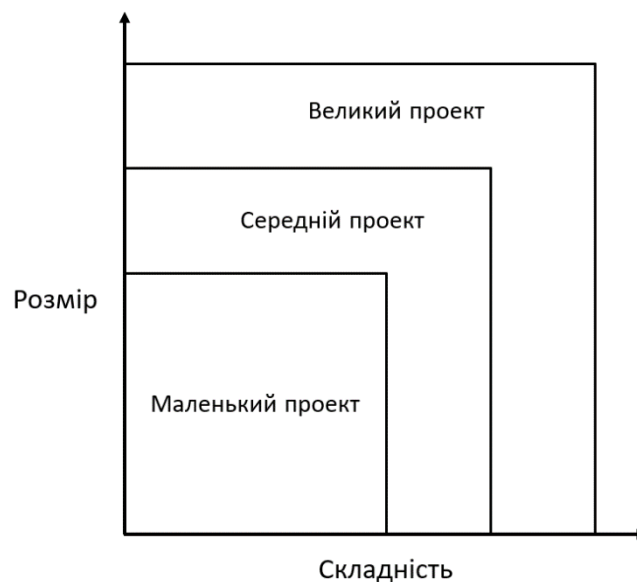


Рисунок 2.1 – Класифікація проєктів за розміром та складністю

2.2 Вибір мови програмування для серверної частини

Для створення серверного застосунку обрано мову програмування JAVA версії 1.8.

Java є суворо типізованою об'єктно-орієнтованою мовою програмування для загального призначення, розробкою якої займається компанія Sun Microsystems (надалі придбаною компанією Oracle).

Програми на Java транслюються в байт-код Java, котрий виконується в Java Virtual Machine - це програма, котра займається обробкою байтового коду та передає інструкції обладнання як інтерпретатор.

Перевагою цього способу виконання програм є повна незалежність байт-коду від операційної системи та апаратного забезпечення, що дає можливість виконувати Java-програми на будь-якому пристрої, на якому встановлена відповідна віртуальна машина.

У Java є ще один важливий аспект технології, це гнучка система безпеки, де виконання програми повністю контролюється віртуальною машиною.

Всі операції, котрі перевищують встановлені повноваження програми (наприклад, підключення з іншим пристроєм або спроба несанкціонованого доступу до даних), будуть негайно зупинені.

Часто до недоліків концепції VM відносять зниження продуктивності. Деякі удосконалення трохи підвищили швидкість виконання програм на Java:

- використання технології транслювання байт-коду в машинний код безпосередньо в момент роботи програми (JIT-технологія) з можливістю зберігання версій класу у машинному коді;
- широке використання платформно-орієнтованого коду (native-код) у бібліотеках за замовчуванням;
- апаратні засоби, що забезпечують швидку обробку байт-коду (технологія Jazelle, що підтримується деякими процесорами архітектури ARM).

2.3 Вибір фреймворку для серверної частини

Фреймворк для розробки серверного застосунку було обрано Spring-boot[12].

Розробники Spring вирішили надати їх комюніті деякі утиліти, котрі допомагають автоматизувати процедуру налаштування та прискорюють процес створення та розгортання Spring-додатків та має загальну назву Spring-boot.

Spring Boot - це корисний проект, на меті якого є спрощення написання програм на основі Spring. Він дає змогу найпростішим способом написати web-додаток, вимагаючи мінімум зусиль від розробників щодо його налаштування та написання коду. На рис. 2.2 представлено архітектуру фреймворка Spring.

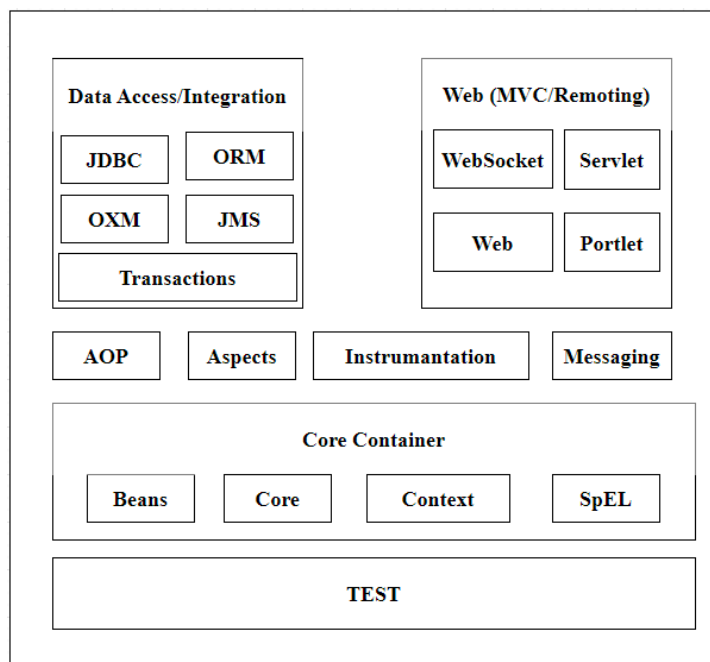


Рисунок 2.2 – Базова архітектура Spring Framework [12]

2.3.1 Особливості Spring Boot

Spring Boot має великий функціонал, але найбільш значущими особливостями є: керування залежностями, вбудовані контейнери сервлетів та автоматична конфігурація.

2.3.2 Простота керування залежностями

Для прискорення процесу управління залежностями, Spring Boot неявно упаковує усі потрібні сторонні залежності для програм кожного типу на основі Spring та надає їх розробнику завдяки так званим starter-пакетів (spring-boot-starter-web, spring-boot-starter-data-jpa і т.д.).

Starter пакети є набором зручних та корисних дескрипторів залежностей, котрі можна інтегрувати в свою програму. Це дасть змогу отримати універсальне рішення для усіх, пов'язаних зі Spring технологій, позбавляючи програміста від потреби зайвого пошуку прикладів коду та завантаження з них необхідних дескрипторів залежностей [12].

Наприклад, якщо вам потрібно почати використовувати Spring Data JPA для доступу до БД, просто додайте у свій проєкт залежність spring-boot-starter-data-jpa і все буде працювати (вам не потрібно буде шукати сумісні драйвери для баз даних та бібліотеки Hibernate).

Якщо вам потрібно створити web-додаток, просто включіть в свій проєкт залежність spring-boot-starter-web, яка підтягне в проєкт усі бібліотеки, необхідні для розробки MVC-додатків, таких як jackson-json, spring-webmvc, Tomcat та validation-api.

Іншими словами, Spring Boot збирає всі загальні залежності і визначає їх в одному місці. Це дозволяє розробникам просто використовувати їх замість того, щоб винаходити велосипед кожного разу, коли вони створюють новий додаток.

Як результат, кількість рядків у файлі pom.xml при використанні Spring Boot набагато менша, ніж для Spring-додатків [12].

2.3.3 Автоматична конфігурація

Ще однією ключовою особливістю Spring Boot є автоматична конфігурація додатків.

Коли ви виберете відповідний стартовий пакет, Spring Boot спробує автоматично налаштувати ваш Spring-додаток на основі доданих вами залежностей від контейнерів.

Наприклад, якщо ви додасте spring-boot-starter-web, Spring Boot автоматично налаштує зареєстровані контейнери, такі як DispatcherServlet, ResourceHandlers і MessagingSource.

Якщо ви використовуєте spring-boot-starter-jdbc, Spring Boot автоматично реєструє контейнери DataSource, EntityManagerFactory і TransactionManager, а також інформацію для підключення до бази даних. Він зчитує інформацію для підключення до бази даних з файлу application.properties.

Якщо ви не маєте наміру використовувати базу даних і не надасте детальну інформацію про підключення в ручному режимі, Spring Boot автоматично налаштує базу даних в пам'яті без будь-якої додаткової конфігурації з боку користувача(якщоприсутнібібліотекиH2 або HSQL) [12].

Автоматична конфігурація може бути повністю замінена налаштуваннями користувача в будь-який час.

2.3.4 Вбудована підтримка сервера додатків - контейнера сервлетів

Усі веб-програми Spring Boot мають вбудований веб-сервер. Ознайомтеся зі списком сервлет-контейнерів, які незабаром будуть підтримуватися.

Розробникам більше не потрібно турбуватися про налаштування сервлет-контейнера та розгортання свого додатку. Додатки можна запускати як виконуваний jar-файл за допомогою вбудованого сервера [12].

Якщо вам потрібно використовувати інший HTTP-сервер, просто виключіть залежність за замовчуванням; Spring Boot надає окремі стартові пакети для різних HTTP-серверів.

Створювати автономні веб-додатки з вбудованими серверами не тільки зручно для розробки, але також є прийнятним рішенням для додатків корпоративного рівня і стає все більш корисним у світі мікросервісів. Можливість швидко упакувати цілий сервіс (наприклад, автентифікацію користувачів) в окремий, повністю розгорнутий, автономний артефакт, який також надає API, значно спрощує встановлення та розгортання додатків[12].

Для демонстрації функціональності РС було створено простий веб-додаток, інтернет-магазин.

2.4 Вибір фреймворку для розробки інтернет магазину

Angular [13] – платформа для розробки, написана на TypeScript. Як платформа, Angular включає:

- заснований на компонентах фреймворк для створення вебдодатків, що масштабуються;
- набір добре інтегрованих бібліотек, що охоплюють широкий спектр функцій: маршрутизація, керування формами, клієнт-серверна взаємодія тощо;
- набір інструментів розробника, які допоможуть вам розробляти, збирати, тестувати та оновлювати ваш код.

Коли ви створюєте програму за допомогою Angular, ви користуєтеся перевагами платформи, яка може масштабуватися від проєкту, який розробляє одна людина, до програм корпоративного рівня. Angular розроблено, щоб

максимально спростити оновлення, тому ви можете використовувати останні розробки з мінімумом зусиль. А чудове те, що екосистема Angular складається з величезної спільноти, що включає більш ніж 1.7 мільйона розробників, авторів бібліотек і творців контенту [13].

2.5 Вибір середовища розробки

В якості середовища розробки було обрано IntelliJ IDEA.

IntelliJ IDEA - це інтегроване середовище розробки (IDE) для мови JVM, призначене для максимізації продуктивності розробників. Воно забезпечує інтелектуальне завершення коду, статичний аналіз коду та рефакторинг, щоб взяти на себе рутинні та повторювані завдання і дозволити вам зосередитися на задоволенні від розробки програмного забезпечення.

Перша версія була випущена в січні 2001 року і швидко завоювала популярність як перше середовище Java, що інтегрує широкий спектр інструментів рефакторингу, які дозволяють програмістам швидко реорганізувати вихідний код своїх програм. Середовище було розроблене з акцентом на продуктивність програмістів, дозволяючи їм зосередитися на функціональних завданнях, в той час як IntelliJIDEA піклується про рутинні операції.

Починаючи з шостої версії, IntelliJ IDEA пропонує інтегрований інструментарій для розробки графічних інтерфейсів користувача. Серед іншого, середовище сумісне з багатьма популярними вільними інструментами розробки, такими як CVS, Subversion, ApacheAnt, Maven, JUnit та ін. У лютому 2007 року розробники IntelliJ було анонсовано та випущено ранню версію плагіна для програмування мовою RUBY.

Community Edition, доступний за ліцензією Apache 2.0, та є повністю безкоштовною версією з повною підтримкою Java SE, Kotlin, Groovy і Scala та інтеграцією з найпопулярнішими системами контролю версій. Ultimate Edition,

доступна за комерційною ліцензією, включає підтримку Java EE, діаграм UML, розрахунки покриття коду та інших систем контролю версій, мов і фреймворків.

Конкурентом IntelliJ IDEA є середовище Eclipse. Порівняння цих двох середовищ наводиться в таблиці 2.1.

Eclipse - це інтегроване середовище розробки (IDE), що використовується для комп'ютерного програмування. Воно включає базовий робочий простір та розширювану систему плагінів для налаштування середовища. Це друге за популярністю IDE для розробки Java і було найпопулярнішим до 2016 року . Eclipse написане переважно на Java і використовується переважно для розробки Java-додатків. Використовується також для розробки додатків на інших мовах, але за допомогою плагінів для Ada, ABAP, C, C++, C#, Clojure, COBOL, D, Erlang, Fortran, Groovy, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby, Rust, Scala та Scheme. Він також може бути використаний для розробки документації з використанням пакетів LaTeX (за допомогою плагіна TeXlipse) і Mathematica. Середовища розробки включають засоби розробки Eclipse Java для Java та Scala, Eclipse CDT для C/C++ та Eclipse PDT для PHP.

Таблиця 2.1 – Порівняльна характеристика середовищ для розробки Eclipse та IntelliJ IDEA

Критерії порівняння	Eclipse	IntelliJ IDEA
Налагодження	Більше часу витрачається для налагодження коду	Більш зручне та швидке налагодження коду
Рефакторинг	Інтелектуальний рефакторинг відсутній	Включає в себе інтелектуальний рефакторинг

Продовження таблиці 2.1 – Порівняльна характеристика середовищ для розробки Eclipse та IntelliJ IDEA

Критерії порівняння	Eclipse	IntelliJ IDEA
Доступність	Повністю безкоштовне	Середовище для веб- та ентєрпрайз розробки потребує придбання Ultimate edition
Можливість працювати з БД	Не підтримує можливість взаємодії з БД	Підтримує SQL інструменти для роботи з БД

Найвідомішими програмами на основі Eclipse є різні "Eclipse IDE" для розробки програмного забезпечення на різних мовах (наприклад, найпопулярніша IDE для Java, яка спочатку підтримувалася, не покладається на закриті розширення, а використовує стандартні відкриті API для доступу до платформи Eclipse). Проаналізувавши основні переваги та недоліки цих IDE (табл. 2.1) вирішено, що IntelliJ IDEA більш зручна для розробки застосунків мовою Java.

Для розробки цього застосунку було проаналізовано існуючі та обрано найбільш підходящі фреймворки для розробки як серверної так і клієнтської частини Spring-boot та Angular відповідно, та зручне середовище для розробки IntelliJ IDEA. Мови програмування Java, TypeScript.

3 ДОСЛІДЖЕННЯ ОСНОВНИХ РІШЕНЬ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Опис існуючих технологій

Через неймовірно високий об'єм даних, котрі повинні оброблюватись вимоги до сучасних РС досить високі, і, разом з тим, користувач буде не задоволений, якщо йому треба буде очікувати кілька годин, для того, щоб отримати рекомендації.

Система повинна бути масштабованою, тобто працездатною для використання з великими обсягами даних. Якість рекомендацій, що надаються системою, можна значно покращити, використовуючи декілька показників. Іншим критерієм є швидкість отримання рекомендацій. РС отримує велику кількість запитів на отримання рекомендацій за одиницю часу, і на всі запити необхідно реагувати в режимі реального часу, щоб забезпечити позитивний досвід користувачів при використанні сервісу електронної комерції.

В даному розділі описано програмний продукт, що являє собою РС яка задовольняє вищевказані потреби, він включає в себе кілька компонентів: MIND, PaddleRec та Milvus.

Системи рекомендацій мають вибрати найкращий набір для користувача з набору мільйонів елементів. Однак це має бути зроблено в межах суворих вимог до затримки. В результаті чим складнішу модель ми навчаємо, тим більше часу йде на обробку одного запиту. Отже, РС має багатоступеневу архітектуру. Подумайте про це як про послідовність, яка починається з мільйона елементів і закінчується кількома рекомендаціями.

Ідея полягає в тому, щоб використовувати просту, легку модель у верхній частині цієї воронки, як просту модель спільного фільтрування. Ця модель повинна мати можливість вибирати кілька тисяч найрелевантніших елементів, можливо, не з найкращим рейтингом, тобто відповідні елементи мають бути

присутніми в цьому наборі з тисяч елементів, і це нормально, якщо вони не вгорі. Отже, ця модель оптимізує запам'ятовування та швидкість. Цю модель також називають генератором-кандидатом. Навіть у простій моделі спільної фільтрації переконайтеся, що розміри вбудовування не надто великі. Використання 100-х розмірів може дати вам невелике збільшення пригадування, але вплине на затримки.

Потім ці тисячі предметів надсилаються до іншої моделі під назвою Light Ranker. Як випливає з назви, завдання цієї моделі полягає в пошуку найкращого рейтингу. Модель навчена на високу точність і є більш складною, ніж потенційний генератор (наприклад, моделі з двома баштами). Він також використовує більше функцій на основі активності користувача, метаданих елементів тощо. Результатом цієї моделі є ранжирований список із сотень найпопулярніших елементів.

Нарешті ці сотні предметів надсилаються до важкого рейтингу. Цей ранкер має схожу мету з легким ранкером, за винятком того, що він важчий за легкий ранкер і використовує ще більше функцій. Оскільки він працює лише з сотнями елементів, затримками, пов'язаними з такою складною архітектурою, можн керувати. На рис. 3.1 показані усі етапи РС.

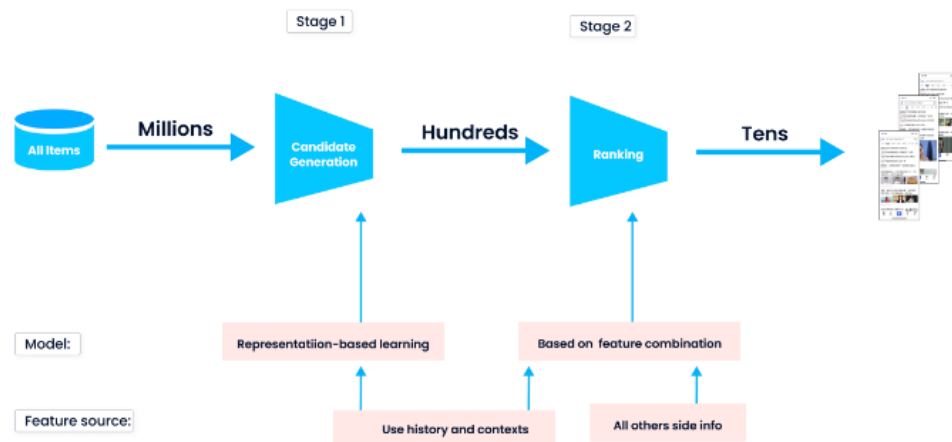


Рисунок 3.1 – Етапи рекомендаційної системи [13]

Система рекомендацій може повертати персоналізовані результати відповідно до унікальних інтересів і потреб користувачів. Щоб надати такі персоналізовані рекомендації, система проходить два етапи: формування кандидатів і рейтинг.

Перший етап — це генерація кандидатів, яка повертає найбільш релевантні або подібні дані, наприклад продукт або відео, що відповідає профілю користувача. Під час генерації кандидатів система порівнює ознаки користувача з даними, що зберігаються в її базі даних, і вилучає подібні. Потім під час ранжування система оцінює та змінює порядок отриманих даних. Нарешті, ті результати вгорі списку показуються користувачам.

У нашому випадку системи рекомендацій щодо продуктів спочатку порівнює профіль користувача з характеристиками продуктів в асортименті, щоб відфільтрувати список продуктів, які задовольняють потреби користувача. Потім система оцінює продукти на основі їх схожості з профілем користувача, ранжує їх і, нарешті, повертає користувачеві 10 найкращих продуктів.

3.2 MIND

MIND, скорочення від «Multi-Interest Network with Dynamic Routing for Recommendation at Tmall», – це алгоритм, розроблений Alibaba Group. До того, як було запропоновано MIND, більшість поширених моделей штучного інтелекту для рекомендацій використовували один вектор для представлення різноманітних інтересів користувача. Однак одного вектора далеко недостатньо для точного представлення інтересів користувача. Тому був запропонований алгоритм MIND, який перетворює численні інтереси користувача в кілька векторів [14].

Зокрема, MIND використовує мережу з кількома інтересами з динамічною маршрутизацією для обробки кількох інтересів одного користувача на етапі

створення кандидатів. Мережа з кількома інтересами — це рівень екстрактора з кількома інтересами, побудований на механізмі капсульної маршрутизації. Його можна використовувати, щоб поєднати минулу поведінку користувача з його чи її інтересами, щоб створити точний профіль користувача.

Щоб представити особливості користувачів, MIND приймає поведінку та інтереси користувачів як вхідні дані, а потім передає їх на рівень вбудовування для створення векторів користувачів, включаючи вектори інтересів користувачів і вектори поведінки користувачів. Потім вектори поведінки користувачів подаються на рівень екстрактора багатьох інтересів для створення капсул інтересів користувачів. Після об'єднання капсул інтересів користувачів із вбудовуваннями поведінки користувача та використання кількох шарів ReLU, для їх трансформації MIND виводить кілька векторів представлення користувача. Цей проєкт визначив, що MIND остаточно виведе чотири вектори представлення користувача [14].

З іншого боку, характеристики продукту проходять через рівень вбудовування та перетворюються на розріджені вектори елементів. Потім кожен вектор елемента проходить через шар об'єднання, щоб стати щільним вектором.

Коли всі дані перетворюються на вектори, для керування процесом навчання вводиться додатковий рівень уваги з урахуванням міток.

3.3 PaddleRec

PaddleRec — це широкомасштабна бібліотека моделей пошуку для рекомендацій. Це частина екосистеми Baidu PaddlePaddle. PaddleRec прагне надати розробникам інтегроване рішення для створення системи рекомендацій простим і швидким способом.

Інженерам, які розробляють системи рекомендацій, часто доводиться стикатися з проблемами поганої зручності використання та складного

розгортання системи. Однак PaddleRec може допомогти розробникам у таких аспектах:

Простота використання: PaddleRec – це бібліотека з відкритим вихідним кодом, яка інкапсулює різні популярні моделі в галузі, включаючи моделі для створення кандидатів, рейтингу, зміни рейтингу, багатозадачності тощо. За допомогою PaddleRec ви можете миттєво перевірити ефективність моделі та підвищити її ефективність за допомогою ітерації. PaddleRec пропонує вам простий спосіб навчання моделей для розподілених систем із відмінною продуктивністю. Він оптимізований для великомасштабної обробки даних розріджених векторів. Ви можете легко масштабувати PaddleRec по горизонталі та прискорити його швидкість обчислень. Тому ви можете швидко створювати навчальні середовища на Kubernetes за допомогою PaddleRec.

Підтримка розгортання: PaddleRec надає онлайн-рішення для розгортання своїх моделей. Моделі відразу готові до використання після навчання, відрізняються гнучкістю та високою доступністю.

3.4 Milvus

Milvus – це векторна база даних із хмарною архітектурою. Має відкритий вихідний код на GitHub і може використовуватися для зберігання, індексування та керування масивними векторами вбудовування, згенерованими глибокими нейронними мережами та іншими моделями машинного навчання [15].

Milvus інкапсулює кілька першокласних бібліотек пошуку приблизного найближчого сусіда (ANN), включаючи Faiss, NMSLIB і Annoy. Ви також можете масштабувати Milvus відповідно до ваших потреб. Служба Milvus є високодоступною та підтримує уніфіковану пакетну та потокову обробку.

Milvus прагне спростити процес керування неструктурованими даними та забезпечити узгоджену роботу користувача в різних середовищах розгортання. Він має такі особливості:

- висока продуктивність при проведенні векторного пошуку на масивних наборах даних;
- спільнота перших розробників, яка пропонує багатомовну підтримку та набори інструментів;
- хмарна масштабованість і висока надійність навіть у разі збою;
- гібридний пошук досягається шляхом поєднання скалярної фільтрації з пошуком векторної подібності.

Milvus використовується для пошуку схожості векторів і керування векторами, оскільки він може вирішити проблему частого оновлення даних, зберігаючи стабільність системи.

Він має 4 рівня:

- рівень доступу – складається з групи проксі-серверів без збереження стану і є переднім рівнем системи та кінцевою точкою для користувачів;
- служба координатора – призначає завдання робочим вузлам та діє як мозок системи;
- робочі вузли – діють як руки та ноги і є простими виконавцями, які дотримуються інструкцій служби координатора та виконують ініційовані користувачем команди DML/DDL;
- сховище – є основою системи та відповідає за сталість даних. Він включає метасховище, брокер журналів і сховище об'єктів.

Архітектуру Milvus представлено на рисунку 3.2.

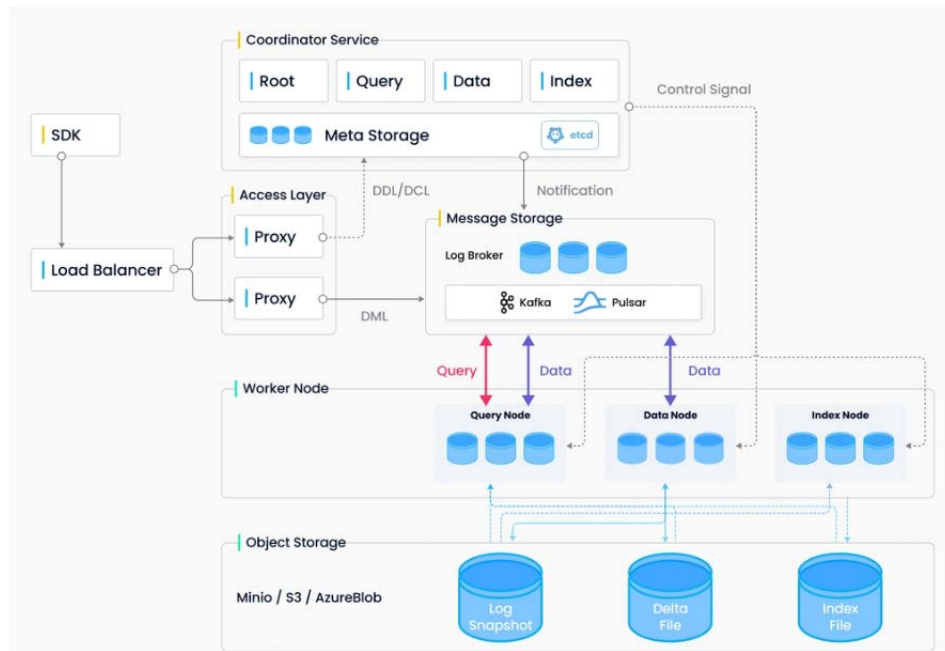


Рисунок 3.2 – Архітектура Milvus [15]

Milvus підтримує два порти, порт 19530 і порт 9091:

Порт 19530 призначений для gRPC та RESTful API. Це порт за замовчуванням при підключенні до сервера Milvus із різними SDK Milvus або HTTP-клієнтами.

Порт 9091 призначений для збору метрик, профілювання `pprof` та перевірки працездатності в Kubernetes. Він є портом управління.

У прикладі нижче показано, як встановити з'єднання з сервером Milvus, використовуючи його `localhost` як хост і 19530 порт, а потім як правильно відключитися від нього. Якщо з'єднання відмовлено, спробуйте розблокувати відповідний порт.

Підключення на мові програмування Java:

```
final MilvusServiceClient milvusClient = new MilvusServiceClient(
    ConnectParam.newBuilder()
        .withHost("localhost")
        .withPort(19530)
        .build()
);
```

3.5 Впровадження системи

Щоб побудувати систему рекомендацій продуктів необхідно пройти такі кроки:

- обробка даних;
- модельне навчання;
- тестування моделі;
- створення кандидатів на позицію продукту;
- зберігання даних: вектори елементів виходять за допомогою навченої моделі та зберігаються у Milvus;
- пошук даних: чотири вектора користувача, згенеровані MIND, передаються в Milvus для пошуку за схожістю векторів;
- ранжування даних: кожен із чотирьох векторів має свої власні top_k подібні вектори елементів, а чотири набори top_k векторів ранжуються для отримання остаточного списку top_k найбільш схожих векторів.

Крок 1. Обробка даних. Вихідний набір даних взятий із набору даних книг Amazon, наданого ComiRes. Однак у цьому проєкті використовуються дані, що завантажуються та обробляються PaddleRes.

Очікується, що набір даних для навчання матиме наступний формат, де кожен стовпець представлятиме:

- Uid: ID користувача;
- item_id: ідентифікатор продукту, на який користувач натиснув;
- Time: тимчасова мітка або порядок кліків.

Очікується, що набір даних для тестування матиме наступний формат, де кожен стовпець представлятиме:

- Uid: ID користувача;

- `hist_item`: ідентифікатор продукту в історії кліків користувача. Якщо їх дещо `hist_item`, вони сортуються за тимчасовою міткою;
- `eval_item`: фактична послідовність, у якій користувач натискає продукти.

Крок 2. Навчання моделі. При навчанні моделі використовуються оброблені дані на попередньому етапі та застосовується модель генерації кандидатів MIND, побудована на PaddleRec.

Введення моделі.

У `dygraph_model.py` запустіть наступний код, щоб обробити дані та перетворити їх на вхідні дані моделі. Цей процес сортує елементи, на які натискав той самий користувач у вихідних даних, відповідно до мітки часу і об'єднує їх у послідовність. Потім випадковим чином виберіть `item``_`id` з послідовності як `target_item` і вийміть 10 елементів до `target_item` цього в `hist_item` якості вхідних даних моделі. Якщо послідовність недостатньо довга, її можна встановити як 0. `seq_len` Це має бути фактична довжина послідовності `hist_item`. Зверніться до сценарію `/home/aistudio/recommend/model/mind/mind_reader.py` для отримання коду для читання вихідного набору даних.

Модель мережі.

Наступний код є фрагментом `net.py`. `Mind_Capsual_Layer` визначає рівень екстрактора декількох інтересів, побудований на механізмі маршрутизації капсули інтересів. Функція `label_aware_attention()` реалізує техніку уваги з урахуванням міток в алгоритмі MIND. Функція `forward()` моделює `class MindLayer` характеристики користувача та генерує відповідні вагові вектори.

Зверніться до сценарію `/home/aistudio/recommend/model/mind/net.py` для конкретної мережної структури РОЗУМУ.

Оптимізація моделі.

У цьому проєкті як оптимізатор моделі використовується алгоритм Адама.

Крім того, PaddleRec записує гіперпараметри у форматі `config.yaml`. Отже, вам просто потрібно змінити цей файл, щоб побачити чітке порівняння ефективності двох моделей для підвищення ефективності моделі. Під час навчання моделі поганий ефект моделі може бути результатом недостатнього або переоснащення моделі. Тому ви можете покращити його, змінивши кількість раундів навчання. У цьому проєкті вам потрібно лише змінити параметри `epoch`, `config.yaml` щоб знайти ідеальну кількість раундів навчання. Крім того, ви також можете змінити оптимізатор моделі `optimizer.class` або `learning_rate`, виконати налагодження. Нижче показано частину параметрів у `config.yaml`. Для отримання детальної реалізації зверніться до сценарію `/home/aistudio/recommend/model/mind/dygraph_model.py`.

Модельне навчання.

Виконайте наступну команду, щоб розпочати навчання моделі. Дивитися `/home/aistudio/recommend/model/trainer.py` проєкт навчання моделі.

Крок 3. Тестування моделі. На цьому етапі використовується набір тестових даних для перевірки продуктивності, наприклад, швидкості відкликання навченої моделі.

Під час тестування моделі всі вектори елементів завантажуються з моделі, а потім імпортуються до Milvus, бази даних векторів з відкритим вихідним кодом. Вважайте тестовий набір даних через скрипт `/home/aistudio/recommend/model/mind/mind_infer_reader.py`. Завантажте модель на попередньому кроці та введіть тестовий набір даних у модель, щоб отримати чотири вектори інтересів користувача. Знайдіть 50 векторів елементів, найбільш схожих на чотири вектори інтересів у Milvus. Ви можете рекомендувати повернені результати користувачам.

Запустіть наступну команду, щоб протестувати модель.

Під час тестування моделі система надає кілька показників для оцінки ефективності моделі, наприклад, `Recall@50`, `NDCG@50` та `HitRate@50`. У цій

статті описано зміну лише одного параметра. Однак у вашому власному сценарії програми вам необхідно навчити більше епох для кращого ефекту моделі. Ви також можете підвищити ефективність моделі, використовуючи різні оптимізатори, встановлюючи різні швидкості навчання та збільшуючи кількість тестових раундів. Рекомендується зберегти кілька моделей з різними ефектами, а потім вибрати ту, яка має найкращі характеристики та найкраще підходить для вашої програми.

Під час тестування моделі система надає кілька показників з метою оцінки ефективності моделі, наприклад `Recall@50, N`.

Крок 4. Генерація кандидатів на позицію продукту. Для створення служби створення кандидатів на продукт використовується навчена модель, отримана на попередніх етапах у поєднанні з `Milvus`. Під час створення кандидатів `FASTAPI` використовується для забезпечення інтерфейсу. Коли служба запускається, ви можете безпосередньо запускати команди у терміналі.

Сервіс надає чотири типи інтерфейсів (рис 3.4).

Вставити: виконайте наступну команду, щоб прочитати вектори елементів з вашої моделі та вставити їх у колекцію `Milvus`.

Створіть попередні кандидати. Введіть послідовність, у якій користувач натискає на продукти, та визначте наступний продукт, на який користувач може натиснути. Ви також можете генерувати кандидатів на позиції продукту в пакетному режимі для декількох користувачів за один раз. `hist_item` в наступній команді - це двовимірний вектор, і кожен рядок представляє послідовність продуктів, на які користувач натискав у минулому. Ви можете визначити довжину послідовності. Результати, що повертаються, також являють собою набори двовимірних векторів, кожен рядок представляє повертаються `item_id` значення для користувачів.

Запитайте загальну кількість товарів: виконайте наступну команду, щоб отримати загальну кількість векторів товарів, що зберігаються у базі даних Milvus.

Видалити: Запустіть наступну команду, щоб видалити всі дані, що зберігаються у базі даних Milvus.

Якщо ви запустите службу генерації кандидатів на своєму локальному сервері, ви можете отримати доступ до вищевказаних інтерфейсів за адресою 127.0.0.1:8000/docs. Ви можете пограти, клацнувши чотири інтерфейси та ввівши значення параметрів. Потім натисніть "Спробувати" (рис. 3.5), щоб отримати результат рекомендації.

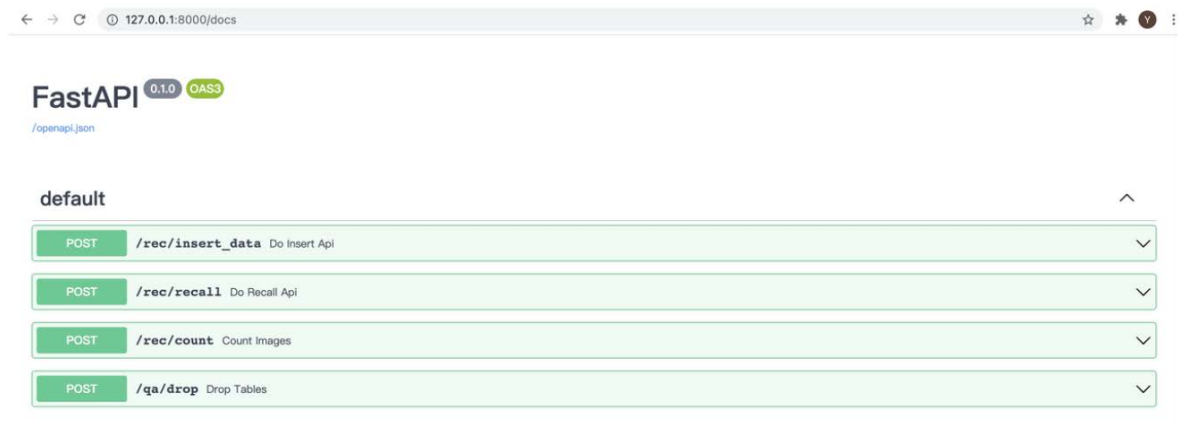


Рисунок 3.4 – Типи інтерфейсів



Рисунок 3.5 – Результат виконання запиту

4 РОЗРОБЛЕННЯ АРХІТЕКТУРИ ТА СТРУКТУРИ СИСТЕМИ

4.1 Алгоритм роботи

Для початку написання коду програми необхідно скласти алгоритм, за яким буде працювати ця програма. Для цього в цьому підрозділі буде створено схему роботи алгоритму. Тому на рисунку 4.1 зобразимо блок-схему алгоритму взаємодії всередині програми та зв'язок з БД.

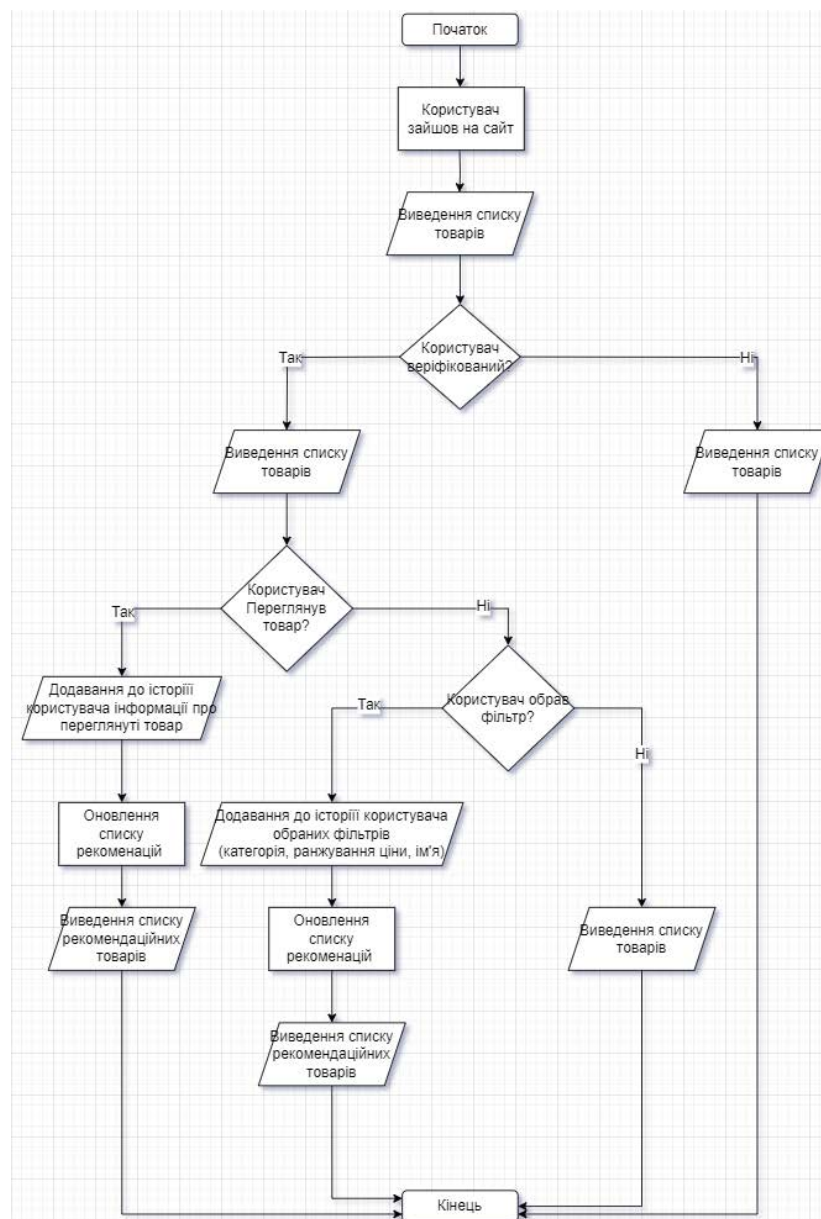


Рисунок 4.1 – Алгоритм роботи застосунку

Коли користувач заходить на сайт, йому одразу буде відображено список усіх товарів. Якщо користувач авторизований, йому додатково буде відображено список рекомендаційних товарів.

Список рекомендацій наповнюється коли користувач починає шукати якийсь конкретний товар, наприклад: користувачу цікава блешня EOS P0001A, для її пошуку він стає на строку пошуку та вводить її назву, якщо такий товар є в наявності користувач може його переглянути тим часом до історії пошуку по цьому користувачу було створено запис, який містить в собі строку пошуку, яку ввів користувач, та айді товару, який він потім переглянув. На основі таких записів і буде формуватися майбутній список рекомендацій.

4.2 Архітектура вебзастосунку та серверної частини

Основна мета фреймворків це полегшити процес розробки системи за рахунок використання вже створених інструментів. Це не лише зручно для програміста, що розробляє, а і для того, хто буде перероблювати чи працювати з таким кодом завдяки тому, що проєкт буде слідувати заданій фреймворком структурі, будуть використовуватись стандартні бібліотеки.

Для реалізації вебзастосунку та серверної частини для нього було обрано фреймворки Angular, Spring-boot та архітектуру MVC через широку застосовність саме для вебзастосунків, вбудованість у фреймворк. Разом з ним надає можливість для реалізації застосунків, що використовуючи найостанніші стандарти програмування, а також застосовувати модульне тестування. Архітектура розроблюваної системи зображена на рисунку 4.2.

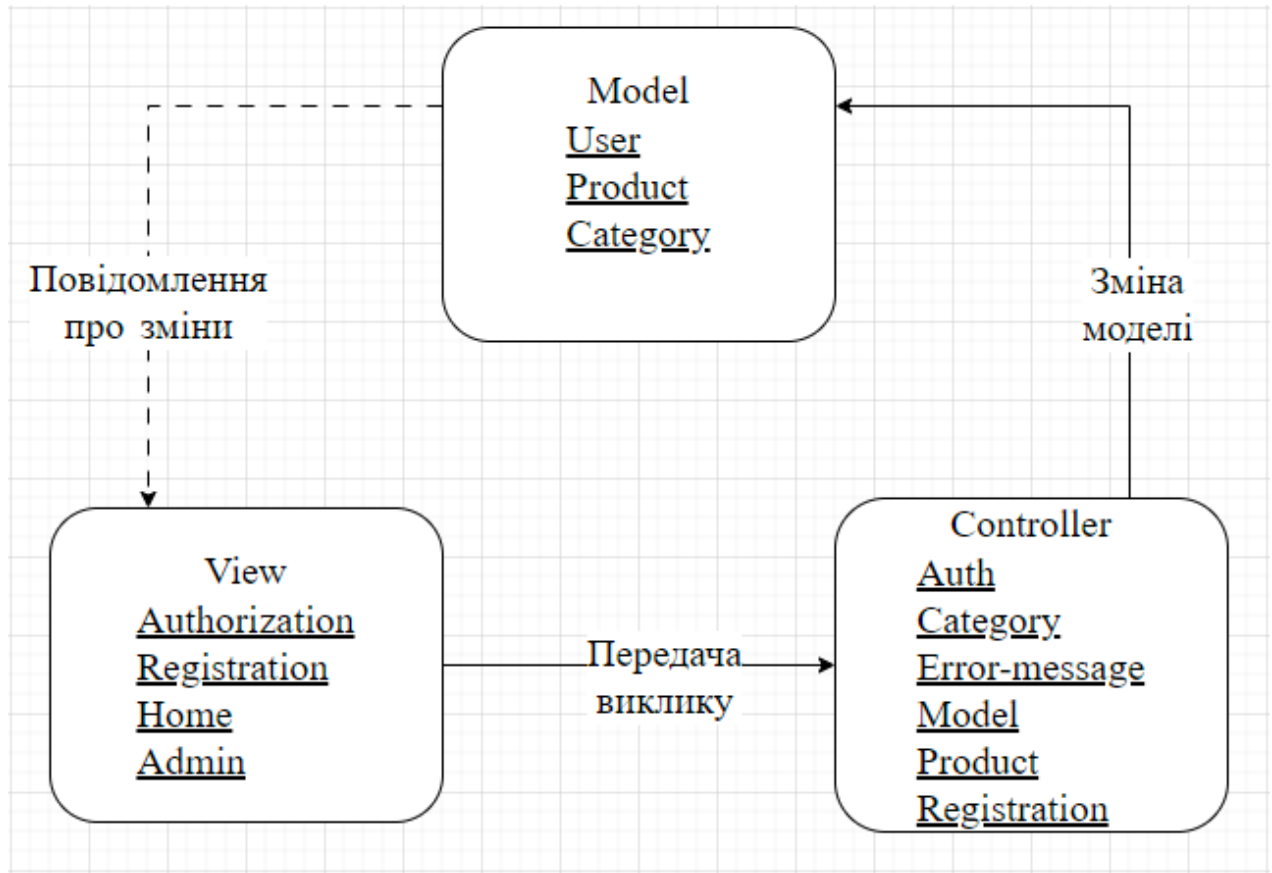


Рисунок 4.2 – Зображення архітектури на основі шаблону MVC

4.3 Структура вебзастосунку та серверної частини на основі використання обраних фреймворків

Відправною точкою для будь-якої програми Angular стає створення структури папок. Команда `ng new` створює структуру, в якій всі файли програми знаходяться в папці `src`, а файли Angular - в папці `src / app`.

Щоб визначити структуру проєкту, було створено папки, перераховані в таблиці 4.1.

Також було створено структуру папок для серверної частини застосунку (табл. 4.2).

Таблиця 4.1 – Папки необхідні для проєкту Store

Папка	Опис
client/src/app/model	Папка для коду моделі даних.
client/src/app/components	Папка яка містить основні компоненти.
client/src/app/admin	Папка для функціонування адміністрування.
client/src/app/guards	Папка для функціонування захисників.
client/src/app/interceptors	Папка для функціонування перехоплювачів.
client/src/app/pages	Папка, що містить додаткові компоненти.
client/src/app/pipes	Папка для каналів.
client/src/app/services	Папка, що містить сервіси, шлють запити до сервера.

Таблиця 4.2 – Структура папок для серверної частини

Папка	Опис
client/src/main/java/cf.zp.danil.store/config	Папка для додаткового налаштування конфігурації
client/src/main/java/cf.zp.danil.store/controller	Папка яка містить контролери для обробки запитів
client/src/main/java/cf.zp.danil.store/converters	Папка для функціонування маршалінгу/емаршалінгу
client/src/main/java/cf.zp.danil.store/filter	Папка для функціонування логера
client/src/main/java/cf.zp.danil.store/model	Папка для коду моделі даних
client/src/main/java/cf.zp.danil.store/repository	Папка для коду репозиторія
client/src/main/java/cf.zp.danil.store/service	Папка що містить сервіси та їх реалізації
client/src/main/java/cf.zp.danil.store/storage	Папка що містить сервіси та їх реалізації

4.4 Функціональні частини

В залежності від рівня доступу користувачам доступні певні функції (рис. 4.3).



Рисунок 4.3 – Перелік функцій за типом користувача

4.5 Інтерфейси вебзастосунку

У таблиці 4.3 представлено інтерфейси, які реалізовано.

Таблиця 4.3 – Інтерфейси вебзастосунку

Інтерфейс	Тип інтерфейсу	Опис
http://127.0.0.1:8080/api/v1/basicauth	GET	Інтерфейс для авторизації. Потребує логін та пароль користувача (рис. 4.4).
http://127.0.0.1:8080/api/v1/info	GET	Відображає інформацію по авторизованому користувачу (рис. 4.5).
http://127.0.0.1:8080/api/v1/logout	GET	Вихід з аккаунту.
http://127.0.0.1:8080/api/product-recommendation?i=1	GET	Повернення рекомендацій (рис. 4.6).
http://127.0.0.1:8080/api/product/{id}	GET	Відображення продукту за його айді (рис. 4.7).
http://127.0.0.1:8080/api/product?category_id={category_id}&page=0&size=3&search=воблер&min=400&max=500	GET	Відображення продуктів за заданими фільтрами (рис. 4.8).
http://127.0.0.1:8080/api/product-count-category	GET	Повернення кількості товарів за певної категорії.
http://127.0.0.1:8080/api/adm/product	POST	Створення нового товару (може виконувати тільки адміністратор).
http://127.0.0.1:8080/api/adm/product	PUT	Редагування товару (може виконувати тільки адміністратор).
http://127.0.0.1:8080/api/adm/product/{id}	DELETE	Видалення товару (може виконувати тільки адміністратор).

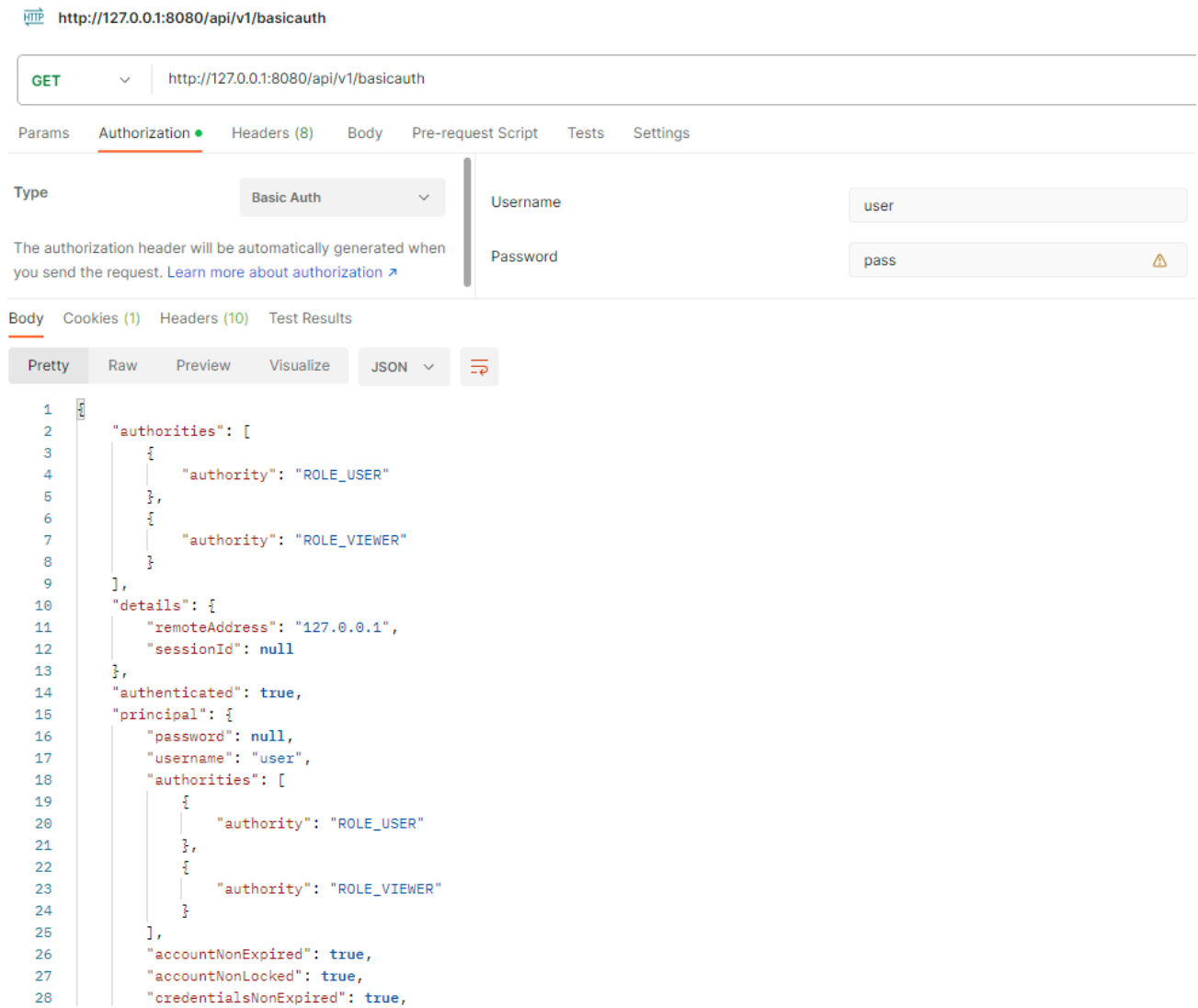


Рисунок 4.4 – Інтерфейс авторизації

HTTP <http://127.0.0.1:8080/api/v1/info>

GET ▼ <http://127.0.0.1:8080/api/v1/info>

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

Query Params

Key	Value
Key	Value

body Cookies (1) Headers (9) Test Results

Pretty Raw Preview Visualize JSON ▼ 

```

1  {
2    "authorities": [
3      {
4        "authority": "ROLE_USER"
5      },
6      {
7        "authority": "ROLE_VIEWER"
8      }
9    ],
10   "details": {
11     "remoteAddress": "127.0.0.1",
12     "sessionId": null
13   },
14   "authenticated": true,
15   "principal": {
16     "password": null,
17     "username": "user",
18     "authorities": [
19       {
20         "authority": "ROLE_USER"
21       },
22       {
23         "authority": "ROLE_VIEWER"
24       }
25     ],
26     "accountNonExpired": true,
27     "accountNonLocked": true,
28     "credentialsNonExpired": true,
29     "enabled": true

```

Рисунок 4.5 – Інтерфейс виводу інформації про користувача

HTTP <http://127.0.0.1:8080/api/product-recommendation?i=1>

GET <http://127.0.0.1:8080/api/product-recommendation?i=1>

Params • Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	Key	Value
☒	i	1
	Key	Value

Body Cookies (1) Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```

1  {
2    {
3      "id": 87,
4      "name": "Сироп риболовний Robin CSL, об'єм 1,0л, смак Ананас",
5      "description": "Сироп риболовний Robin CSL, об'єм 1,0л, смак Ананас",
6      "remain": 1,
7      "price": 120.97,
8      "categoryId": 1,
9      "categoryName": "Ароматизатори, аттрактанти, клейковина"
10   },
11   {
12     "id": 70,
13     "name": "Меляса Art Fishing, об'єм 500мл, смак Малина",
14     "description": "Меляса Art Fishing, об'єм 500мл, смак Малина",
15     "remain": 20,
16     "price": 29.6,
17     "categoryId": 1,
18     "categoryName": "Ароматизатори, аттрактанти, клейковина"
19   },
20   {
21     "id": 71,
22     "name": "Меляса Art Fishing, об'єм 500мл, смак Мед",
23     "description": "Меляса Art Fishing, об'єм 500мл, смак Мед",
24     "remain": 20,
25     "price": 29.6,
26     "categoryId": 1,
27     "categoryName": "Ароматизатори, аттрактанти, клейковина"

```

Рисунок 4.6 – Інтерфейс виводу рекомендаційних товарів

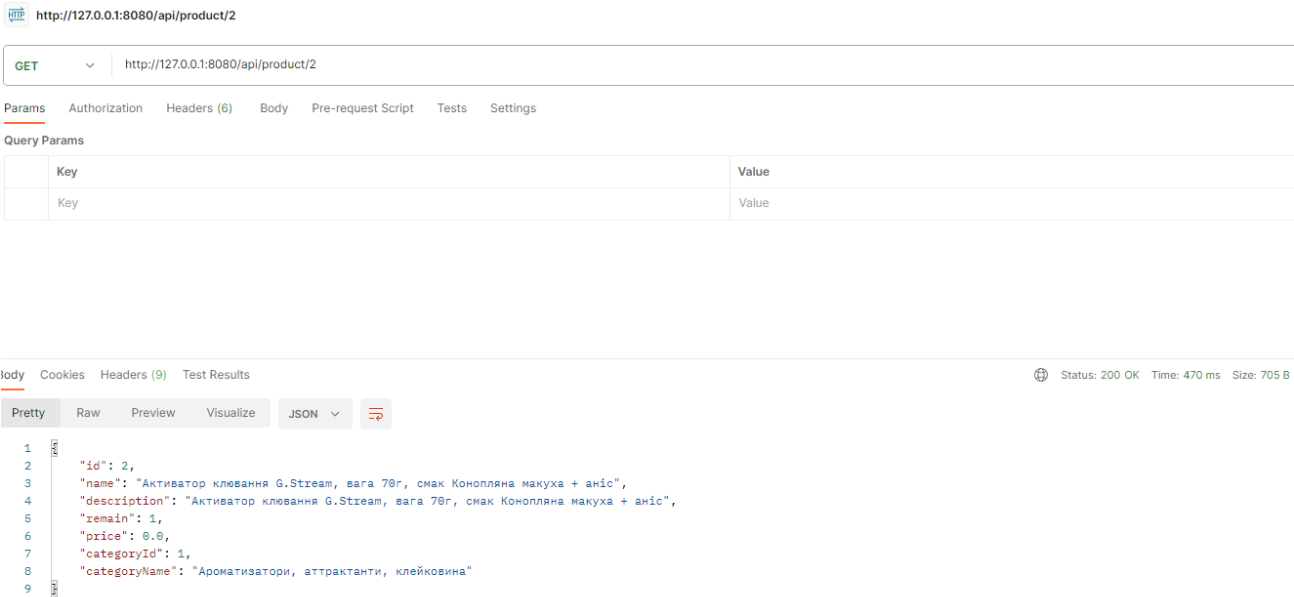


Рисунок 4.7 – Інтерфейс виводу інформації про товар за його айді

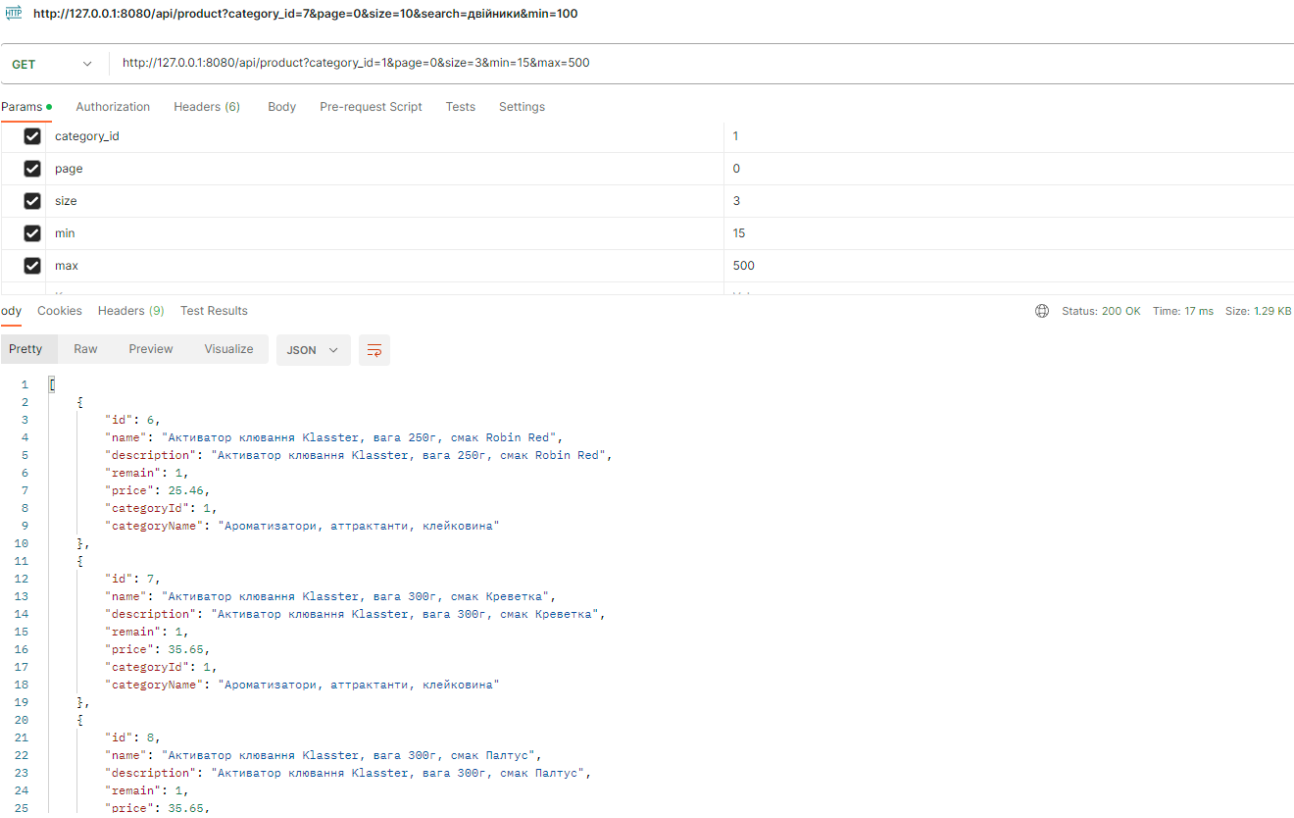


Рисунок 4.8 – Інтерфейс виводу товарів за критеріями пошуку(категорія, ранжування ціни)

4.6 Вікна розробленого застосунку

Інтерфейс користувача (UI)-це набір інструментів, що використовуються користувачами для взаємодії з різними пристроями (комп'ютерами і технікою) та іншими складними інструментами (системами).

На рис. 4.9 – 4.12 зображено вікна веб-застосунка.

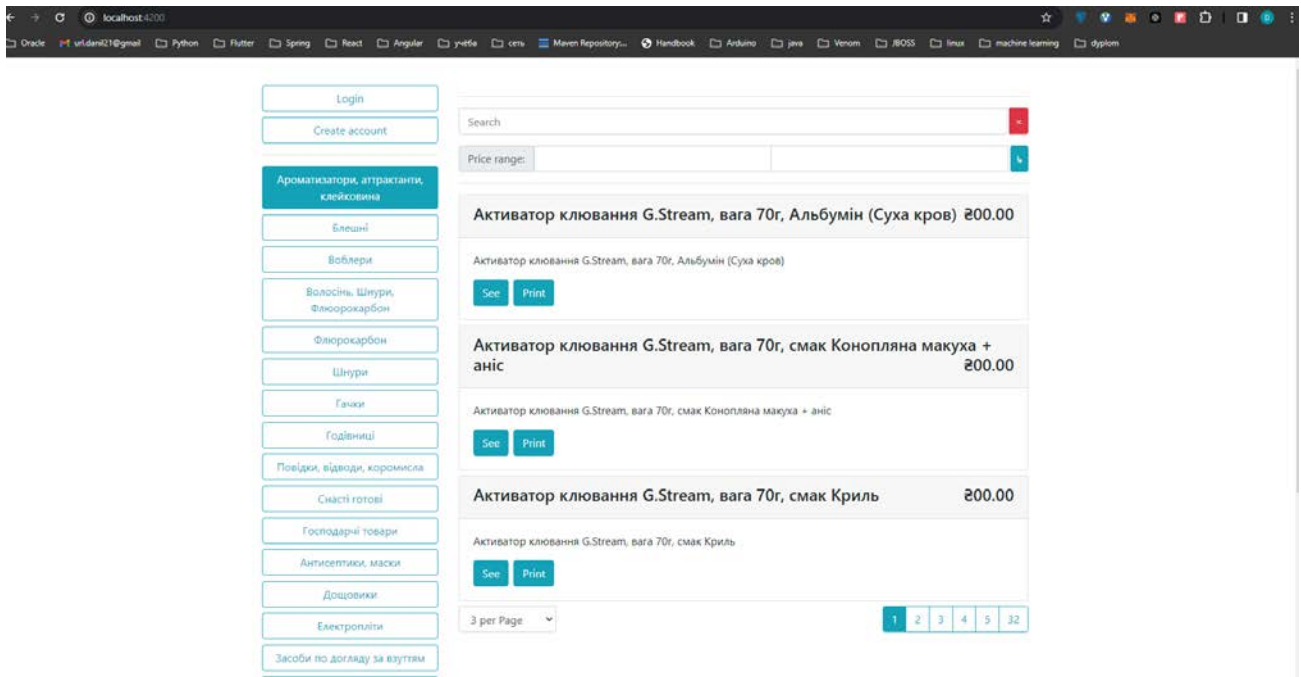


Рисунок 4.9 – Головна сторінка магазину для не авторизованого користувача

Для не авторизованого користувача буде виведено список усіх товарів. Він може їх переглядати та шукати за допомогою фільтрів, але історія пошуку для такого користувача зберігатися не буде.

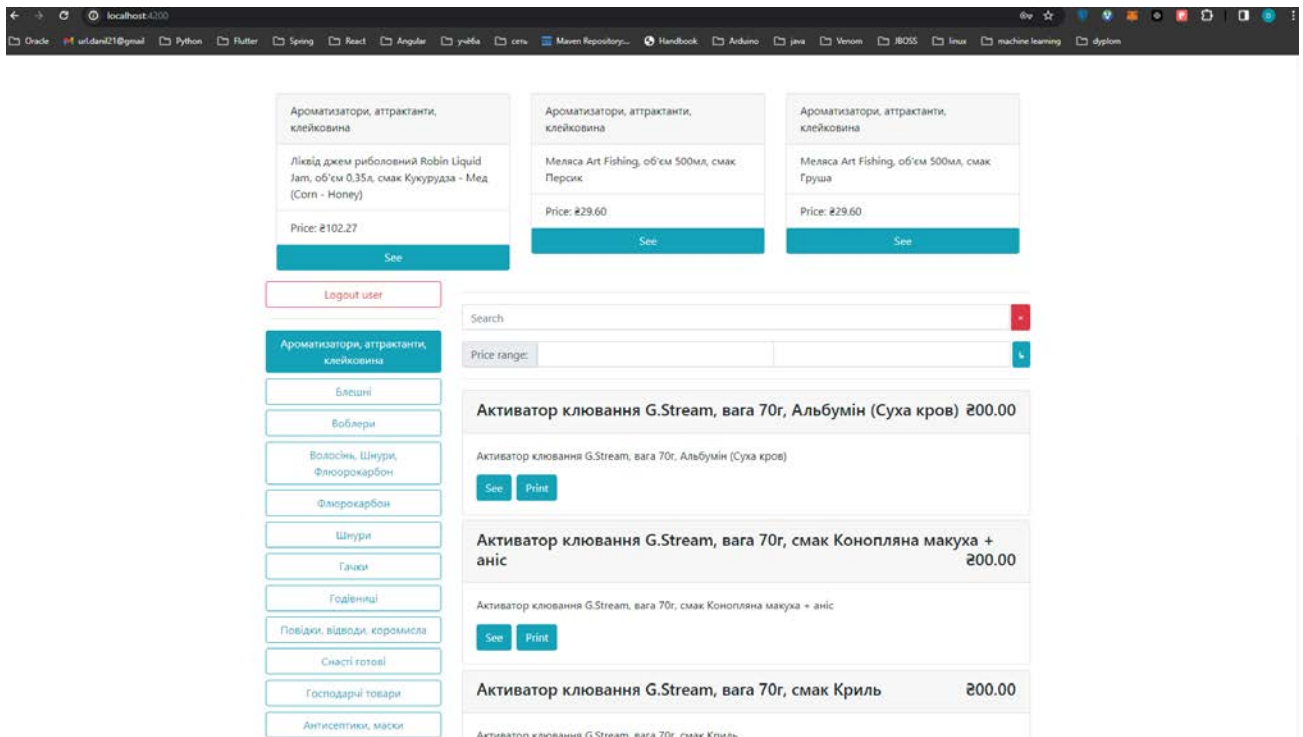


Рисунок 4.10 – Головна сторінка з рекомендаціями для авторизованого користувача

Зайшовши на свій аккаунт користувачу, у якого вже є історія пошуку або переглядів додатково буде виведено список рекомендацій на основі історії.

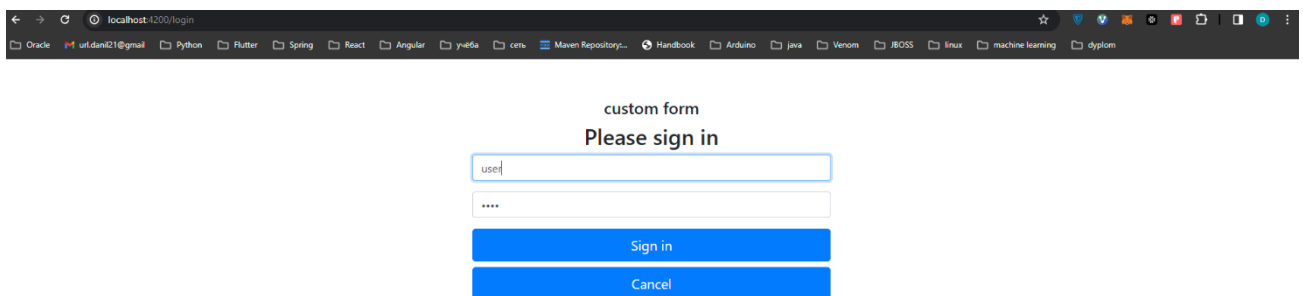


Рисунок 4.11 – Сторінка авторизації

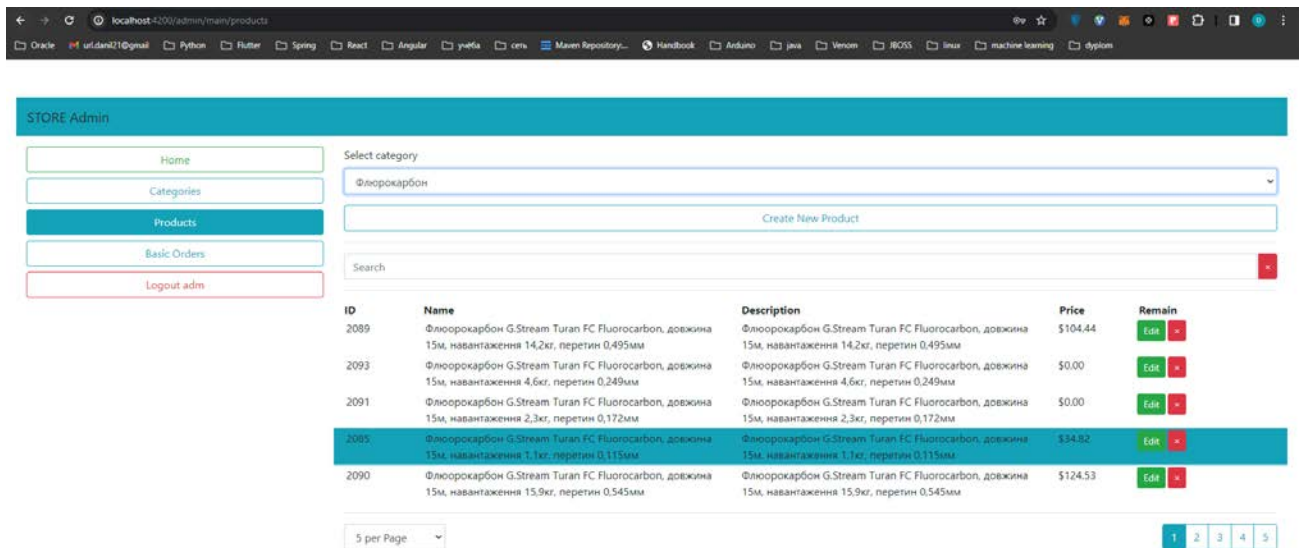


Рисунок 4.12 – Сторінка адміністратора

Адміністратор має право на редагування, створення та видалення товарів та категорій. Для адміністратора також зберігається історія пошуку та переглядів.

ВИСНОВКИ

Рекомендаційні системи – це системи фільтрації інформації. Сьогодні важко уявити життя без таких помічників. У розділі 1 було описано базові поняття теорії рекомендаційних систем і те, як вони працюють. Типи даних, з якими мають справу РС - користувачі, товари та рейтинги.

Описано основні алгоритми та підходи до розрахунку рекомендацій. Проведено порівняння сучасних методів розрахунку рекомендацій та визначено їх переваги та недоліки; розглянуто підходи до побудови РС, обговорено основні алгоритми та підходи до розрахунку рекомендацій, проведено порівняння основних алгоритмів та підходів до розрахунку рекомендацій та визначено їх переваги та недоліки, обговорено основні алгоритми та підходи до розрахунку рекомендацій. Використано гібридний підхід до розрахунку рекомендацій, який дозволяє покрити недоліки одного алгоритму та переваги іншого. Визначено точність РС та її основні показники-точність, відкликання та AUC, для оцінки точності РС необхідний механізм оцінки похибки результатів, запропоновано підхід до побудови рекомендаційної системи в секторі електронної комерції з використанням алгоритму ССО. Практична значущість роботи полягає у тому, що був розроблений вебзастосунок інтернет магазину, який рекомендує найбільш підходящі для певного користувача товари завдяки інтегрованій рекомендаційній системі, яка реалізована на основі методу content-based.

Наукова новизна роботи полягає у тому, що отримав подальший розвиток метод content-based рекомендаційних систем в розробці вебзастосунку для рекомендації товарів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Обробка даних реальних користувачів для створення рекомендаційної системи [Електронний ресурс] - Режим доступу: <https://dou.ua/forums/topic/39070/>.
2. Introduction to recommender systems [Electronic resource]. – Access mode: <https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada>
3. 10 Product Recommendation Techniques to Improve UX and Conversions [Electronic resource]. – Access mode: <https://conversionxl.com/blog/productrecommendations/>.
4. Amazon [Electronic resource]. – Access mode: <https://www.amazon.com/>.
5. Collaborative Filtering for Information Recommendation Systems [Electronic resource]. – Access mode: http://research.create.usc.edu/cgi/viewcontent.cgi?article=1101&context=nonpublished_reports.
6. Building a Correlated Cross-Occurrence (CCO) Recommenders with the Mahout CLI [Electronic resource]. – Access mode: <http://mahout.apache.org/users/recommender/intro-cooccurrence-spark.html>.
7. Dunning, T. Practical Machine Learning: Innovations in Recommendation / T. Dunning, E. Friedman, A. Felfernig // O'Reilly Media, Inc., 2014. – 56 с.
8. Approaches, Issues and Challenges in Recommender Systems: A Systematic Review [Electronic resource]. – Access mode: https://www.researchgate.net/publication/312507062_Approaches_Issues_and_Challenges_in_Recommender_Systems_A_Systematic_Review.
9. Hybrid Recommender Systems: Survey and Experiments. [Electronic resource]. – Access

mode: <https://pdfs.semanticscholar.org/5880/b9bc3f75f4649b8ec819c3f983a14fca9927.pdf>.

10. Comfy.ua [Електронний ресурс] – Режим доступу: <https://comfy.ua/>.
11. DeKa.ua [Електронний ресурс] – Режим доступу: <https://deka.ua/>.
12. Козміна, Ю. Spring 5 для професіоналів / Ю.Козміна, Р. Харроп, К. Шефер, К. Хо. – Видавництво Вільямс, 2019. – 1120 с.
13. Angular для професіоналів. [Електронний ресурс] – Режим доступу: https://balka-book.com/ua/javascript_jquery_dojo-313/angular_dlya_professionalov-58998.
14. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall [Electronic resource]. – Access mode: <https://dl.acm.org/doi/10.1145/3357384.3357814>.
15. Milvus [Electronic resource]. – Access mode: <https://milvus.io/>.
16. Recommender Systems: Metrics and Business Metrics [Electronic resource]. – Access mode: <https://neptune.ai/blog/recommender-systems-metrics>
17. Rozetka.ua [Електронний ресурс] – Режим доступу: <https://rozetka.com.ua/>.
18. Retailrocket recommender system dataset [Electronic resource]. – Access mode: <https://www.kaggle.com/retailrocket/ecommercedataset#events.csv>.
19. Koren, Y. Matrix Factorization Techniques for Recommender Systems / Y. Koren, B. Robert, V. Chris. // Journal Computer. – 2009. – №42 (8). – P. 30-37.

ДОДАТОК А**Код програми**

```

@PreAuthorize("hasAnyRole('ADMIN', 'USER', 'ANONYMOUS')")
@PostAuthorize("hasAnyRole('ADMIN', 'USER', 'ANONYMOUS')")
@RequestMapping(value = "/product-recommendation", method = RequestMethod.GET, produces = {MediaType.APPLICATION_JSON_VALUE})
public List<Product> listRecommendation(@RequestParam(value = "i", required = true, defaultValue = "1") Integer i) {

    String userName = SecurityContextHolder.getContext().getAuthentication().getName();
    logger.info("--- userName: {} ", userName);

    List<Product> list = productService.listRecommendation(userName, i);
    return list;
}

```

Рисунок А.1 – Контролер, що оброблює запит рекомендаційних товарів

```

@Override
public List<Product> listRecommendation(String userName, Integer i) {

    List<Product> recommendationProducts;

    if (!userName.equals("anonymousUser")) {
        User user = userRepository.findByUserName(userName);
        Long userId = user.getId();
        logger.info(" --- userId: {}", userId);

        List<UserHistory> userHistories;
        userHistories = historyRepository.findAllByUserIdOrderByViewDateDesc(userId);
        // userHistories = historyRepository.findAll();
        userHistories.stream().forEach(h -> logger.info(" --- {} {}", h.getUser().getUserName(), h.getPriceMin()));

        recommendationProducts = productStorage.listRecommendation(userHistories, i);

        return recommendationProducts;
    }

    return new ArrayList<>();
}

```

Рисунок А.2 – Сервіс, що формує список рекомендацій

ДОДАТОК Б
Слайди презентації



Рисунок Б.1 – Слайд 1

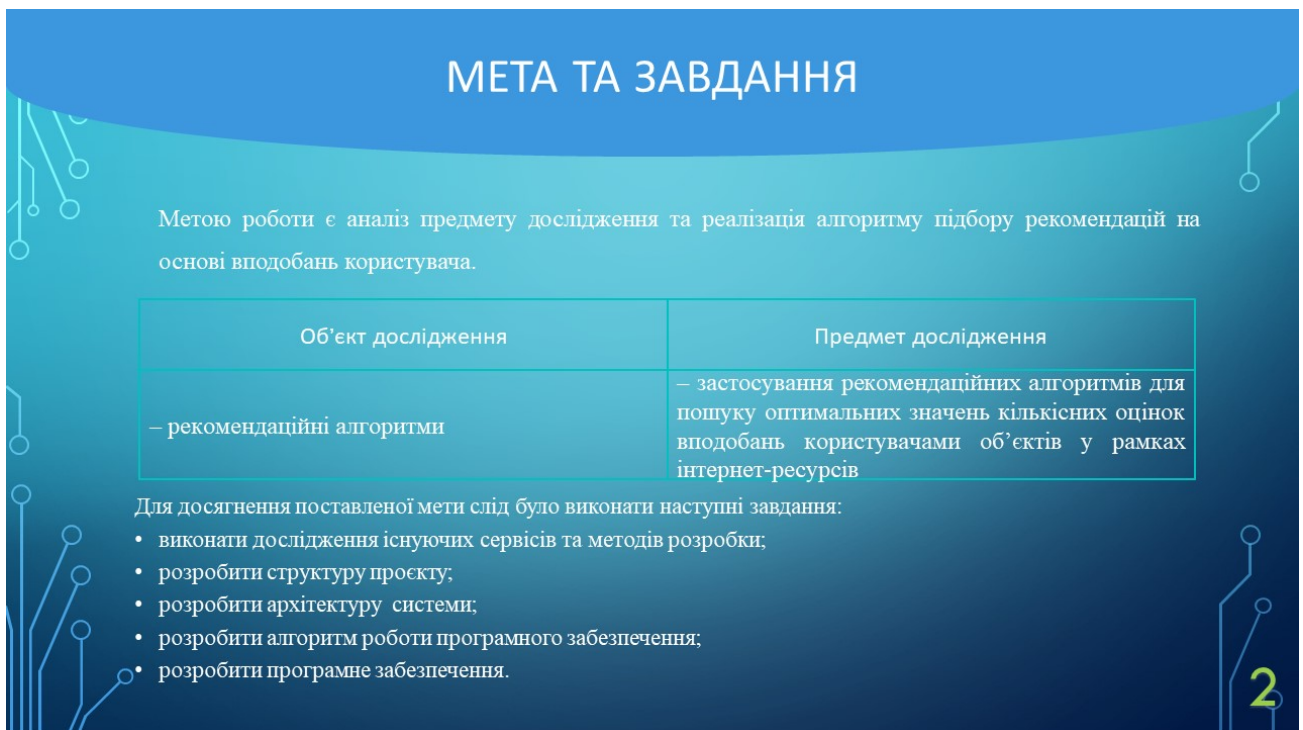


Рисунок Б.2 – Слайд 2



Рисунок Б.3 – Слайд 3

ПОРІВНЯННЯ СЕРЕДОВИЩ ДЛЯ РОЗРОБКИ

Критерій порівняння	Eclipse	IntelliJ IDEA
Налагодження	Налагодження виконується довше	Більш зручне налагодження коду
Рефакторинг	Інтелектуальні рефакторинги відсутні	Має інтелектуальні рефакторинги
Доступність	Повністю безкоштовне	Якщо вам потрібне середовище для веб- або enterprise-розробки, то вам потрібно придбати Ultimate edition
Можливість працювати з базами даних	Відсутня	Присутні SQL інструменти для роботи з базами даних

4

Рисунок Б.4 – Слайд 4

ОСНОВНІ ФУНКЦІОНАЛЬНІ ЧАСТИНИ ПРОГРАМИ

В залежності від рівня доступу користувачам надаються певні можливості.



Рисунок Б.5 – Слайд 5

АЛГОРИТМ РОБОТИ СИСТЕМИ

Алгоритм роботи застосунку полягає в тому, щоб зібрати достатньо інформації про користувача і його вподобання та на основі цих даних запропонувати йому найбільш підходящий продукт.

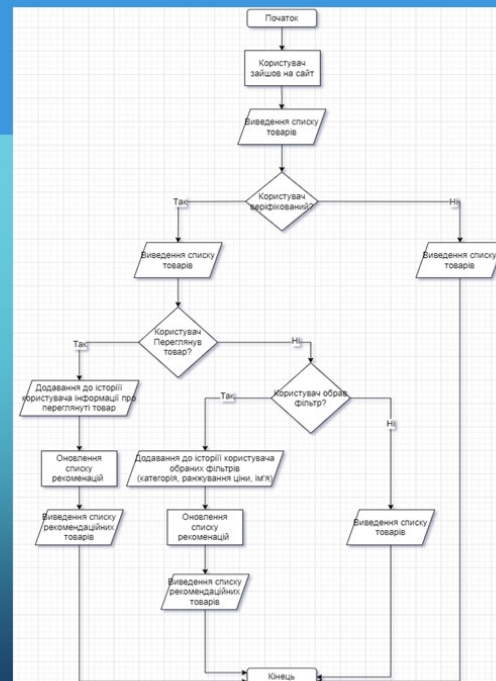


Рисунок Б.6 – Слайд 6



Рисунок Б.7 – Слайд 7



Рисунок Б.8 – Слайд 8

РЕЗУЛЬТАТИ ЗАПИТІВ ДО СЕРВЕРА

The image displays three screenshots of a REST client interface, likely Swagger or Postman, showing the results of API requests to a server.

- Left Screenshot:** Shows a GET request to `http://127.0.0.1:8080/api/v1/auth/auth`. The response is a JSON object containing authentication details, including a token and user information.
- Middle Screenshot:** Shows a GET request to `http://127.0.0.1:8080/api/v1/product/recommendation/1`. The response is a list of recommended products, each with details like name, price, and description.
- Right Screenshot:** Shows a GET request to `http://127.0.0.1:8080/api/v1/product/filtering_by/Name=50000-100000&min=50000`. The response is a list of products filtered by name and price range.

Below the screenshots, there are labels in Ukrainian:

- Запит на авторизацію (Request for authentication)
- Запит рекомендацій (Request for recommendations)
- Запит товарів за заданими фільтрами (Request for products by given filters)

Рисунок Б.9 – Слайд 9

ВИСНОВКИ

Створено вебзастосунок на основі бібліотеки Spring-boot та серверного REST API, за допомогою якого користувач може шукати потрібний йому товар, та на основі його вподобань йому будуть відображатись рекомендовані товари.

Практична значущість роботи полягає у тому, що був розроблений вебзастосунок інтернет магазину, який рекомендує найбільш підходящі для певного користувача товари завдяки інтегрованій рекомендаційній системі, яка реалізована на основі методу content-based.

Наукова новизна роботи полягає у тому, що отримав подальший розвиток метод content-based рекомендаційних систем в розробці вебзастосунку для рекомендації товарів.

Рисунок Б.10 – Слайд 10