УДК 004.42; 004.67 DOI: 10.31673/2786-8362.2023.010404

Гончаренко О.І., Ільїн О.О., д.т.н.,
Кравчук П.О., PhD, **Фесенко М.А.,** к.т.н.

РОЗРОБКА РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ДЛЯ ПІДБОРУ МЕРЕЖЕВОГО ОБЛАДНАННЯ НА БАЗІ СТЕКУ JAVA ТЕХНОЛОГІЙ.

Honcharenko O.I., Ilin O.O., Kravchuk P.O., Fesenko M.A. Development of a recommendation system for the selection of network equipment based on the Java technology stack. The emergence of e-commerce and the growth of the amount of available content creates an overload of large volumes of information. At the same time, recommender systems, being powerful data filtering tools and using a variety of algorithms and analysis methods, reduce this overload by generating the most relevant elements for a particular user, which contributes to more effective and efficient selection decisions.

The article discusses the main types and methods of recommender systems, methods for calculating the similarity coefficient of users and elements, metrics for evaluating the quality of work of recommender systems. The main problems of recommender systems are highlighted. The means of development of the client and server part of the system of recommendations for the web application of finding the optimal configuration of network equipment are considered, such as: HTML, CSS, JavaScript, Spring, Spring Boot, Spring Data JPA, Spring Security, Thymeleaf, MySQL. The architecture of a recommender system web application built using these tools is described.

Key words: recommendation system, Java, Spring, MySql, client-server architecture

Гончаренко О.І., Ільїн О.О., Кравчук П.О., Фесенко М.А. Розробка рекомендаційної системи для підбору мережевого обладнання на базі стеку Java технологій. У статті розглянуто основні типи та методи рекомендаційних систем, методи розрахунку коефіцієнта подібності користувачів і елементів, метрики для оцінки якості роботи рекомендаційних систем. Висвітлено основні проблеми рекомендаційних систем. Розглянуто засоби розробки клієнтської та серверної частини системи рекомендацій для веб-додатку пошуку оптимальної конфігурації мережного обладнання, такі як: HTML, CSS, JavaScript, Spring, Spring Boot, Spring Data JPA, Spring Security, Thymeleaf, MySQL. Описано архітектуру веб-додатку системи рекомендацій, побудовану з використанням цих засобів.

Ключові слова: рекомендаційна система, Java, Spring, MySql, клієнт-серверна архітектура

Вступ

Постійне збільшення обсягу інформації в мережі Інтернет, поява електронної комерції та зростання кількості доступного контенту (кінофільми, фільми, книги і т.п.) і сервісів доступу до нього, призвели до досить активного розвитку систем фільтрації інформації та появи такої їх підгрупи, як рекомендаційні системи. Будучі потужними інструментами фільтрації даних, системи рекомендацій використовують різні алгоритми та методи аналізу даних, щоб рекомендувати найбільш релевантні продукти конкретному користувачеві. На сьогодні вони перетворилися на фундаментальний інструмент для прийняття обґрунтованих, інформативних, ефективних і дієвих рішень не тільки для користувачів мережі Інтернет, але і практично в усіх сферах сучасного життя, включно з бізнесом, фінансами, освітою, розвагами, спілкуванням тощо. Вирішуючи проблему потреби фільтрування великих обсягів інформації, здатність пріоритезування та ефективно надавати відповідну інформацію, зменшуючи відповідно, час пошуку та вдосконалюючи процес прийняття рішень щодо якості вибору, розробка рекомендаційної системи є досить актуальною задачею для додатків, які несуть в собі функціонал пошуку та вибору складових елементів більш великих систем. Актуальним є розробка таких систем для допомоги у підборі окремих складових різних комп'ютерних систем та мереж, коли майже аналогічні та сумісні складові виробляють різні вендори. Розробка таких систем, які використовуються в мережі Інтернет (веб-додатків), має грунтуватись на технологічних рішеннях, притаманних цій мережі. А саме – архітектура клієнт-сервер, можливість перегляду контенту ресурсу з різних засобів доступу -комп'ютери, смартфони, реалізація програмних частин клієнта та сервера на основі відповідних технологій. І якщо підбір технологій клієнтської частини не викликають питань (HTML, CSS) то питання розробки серверної частини залишає багатий простір для вибору.

Аналіз літературних джерел та постановка задачі. В процесі дослідження теми рекомендаційних систем були проаналізовані матеріали сучасних праць таких авторів як Charu C. [1], Francesco Ricci та Lior Rokach [2], Kim Falk [3], D. Jannach, M. Zanker [4]. В роботах приділено більшу увагу теоретичним підходам дослідження рекомендаційних систем та алгоритмів, та недостатньо освітлені питання пов'язані із розробкою таких систем на базі різних цілісних стеках технологій, а саме стеку технологій Java. Також практично не висвітлена питання, пов'язані із створенням рекомендаційних систем саме для рекомендацій у підборі мережевого обладнання для створення комп'ютерних мереж з сумісних складових, але які виробляють різні вендори.

Мета дослідження. Зважаючи на результати аналізу та наявні практичні рішення в сфері розробки та застосування експертних систем для веб-середовища, визначимо, що метою даної публікації є дослідження розробки рекомендаційної системи веб-додатку для пошуку оптимальної конфігурації мережного обладнання саме на базі технологій Java.

Виклад основного матеріалу досліджень

Системи рекомендацій — це системи фільтрації інформації, які надають персоналізовані рекомендації щодо елементів користувачеві в сервісному середовищі, яке може зберігати або збирати різні дані. Система фільтрації інформації, яка в основному використовується в системах рекомендацій, пристосовується до уподобань користувача або пропонує лише елементи, які вважаються корисними для них. Надалі ми будемо розглядати такі рекомендаційні системи, які працюють в мережі Інтрнет, під виглядом звичайних веб-сайтів.

Рекомендаційні системи використовують різнобічні техніки для отримання корисної інформації шляхом виявлення кореляцій і закономірностей між даними на основі аналізу даних у великих наборах даних. Вони аналізують інформацію про певний елемент (книгу, фільм і т.п.), що дає змогу рекомендувати користувачеві елементи цього ж класу, схожий на елемент який його цікавив. Також система створює подібну групу користувачів серед користувачів, щоб ідентифікувати дані потоку кліків відвідувача на веб-сторінці, які відповідають групі користувачів. Системи також можуть рекомендувати індивідуальні параметри перегляду для задоволення потреб конкретних користувачів. Для розробки рекомендацій використовуються різні методи аналізу даних.

Тип рекомендаційних систем напряму залежить від контексту та підходу, використаного під час створення конкретної системи. Із різних типів виділяють базові, які лежать в основі всіх інших (Рис. 1). Базові моделі рекомендаційних систем працюють з двома типами даних: взаємодія користувача з елементом (рейтинги або поведінка під час вибору потрібного елементу) та атрибутивна інформація про користувачів та елементи (дані профілів або релевантні ключові слова). До базових можна віднести рекомендаційні системи в основі яких лежать колаборативна або спільна, на основі знань, на основі контенту та гібридна фільтрації.

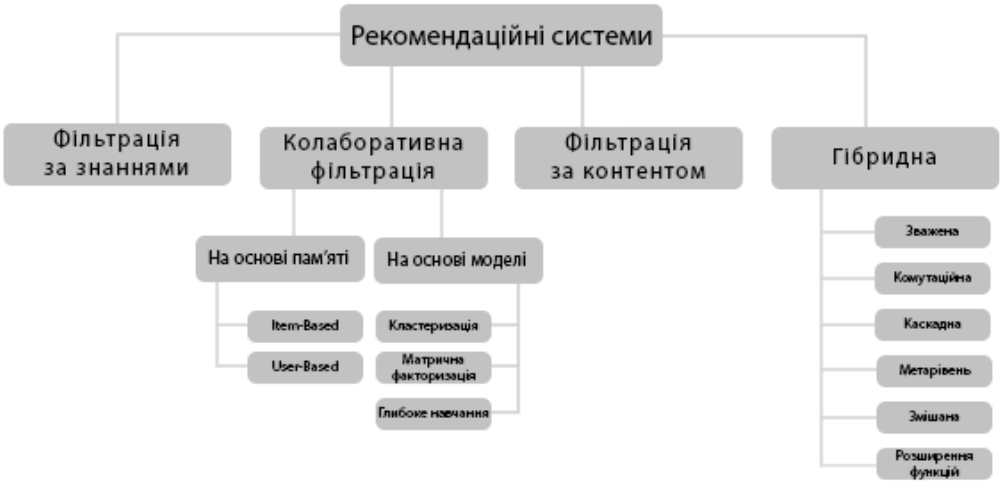

Рис. 1 Основні типи рекомендаційних систем

Колаборативна фільтрація є одним із найбільш розвинутих методів прогнозування, який працює на основі збираємих даних від багатьох користувачів та намагається, використовуючи інформацію про інших, спрогнозувати вподобання поточного користувача. Так, якщо двом користувачам сподобались однакові елементи, то є велика ймовірність того, що одному з них можуть сподобатись інші елементи другого, якими він до цього навіть не цікавився. Це найбільш затребуваний та найбільш широко впроваджуваний підхід, що використовується в розробці рекомендаційних систем. Найсильніша сторона такої фільтрації у тому, що для неї не потрібно переводити дані рекомендованих об'єктів в формат, який зрозуміє комп'ютер і вона добре працює зі складними об'єктами, де відмінності у смаку відповідальні за більшу частину відмінностей у перевагах. Спільна фільтрація ґрунтується на припущенні, що люди, які домовилися в минулому, погодяться і в майбутньому, і що їм подобаються подібні об'єкти, які їм подобалися у минулому. Існує безліч типів методів колаборативної фільтрації, але основними з них виділяють методи на основі моделі та на основі пам'яті.

Методи на основі пам'яті базуються на даних рейтингу для оцінки подібності користувачів або елементів. Рейтинг прогнозуються на основі сусідства.

Всього виділяють два типи методів на основі пам'яті: основані на користувачах та на елементах.

Методи на основі моделі призвані знаходити закономірності на основі даних. Для цього вони розробляються з використанням різного роду алгоритмів машинного навчання та засобів інтелектуального аналізу даних. Фільтрація на основі моделі дозволяє вирішити проблеми методів на основі пам'яті, які потребують великого об'єму вбудованої пам'яті, коли матриця рейтингів становиться досить великою в ситуації активного використання системи великою кількістю користувачів, що призводить до зростання споживання ресурсів та падіння загальної продуктивності системи.

В фільтрації на основі моделі рекомендації формуються шляхом вивчення параметрів статистичних моделей для оцінок користувачів, щоб якнайкраще прогнозувати рейтинги. Іншими словами, інформація витягується з набору даних в рейтингах і на її основі формується модель рекомендацій без необхідності повного аналізу даних кожного разу.

Загалом використовують декілька підходів до створення рекомендацій на основі моделей такі як: матрична факторизація, кластеризація, класифікація.

Фільтрація на основі контенту має більш тісну взаємодію з функціями або атрибутами елементів, а не з даними користувачів. Системи на основі контенту прогнозують поведінку користувача використовуючи елементи на які він реагує. Вони намагаються спрогнозувати, що сподобається користувачу беручи за основу дані про його активність.

Фільтрація на основі контенту надає рекомендації, використовуючи ключові слова та атрибути назначені елементам зіставляючи їх з профілем користувача.

Атрибути залежать від продуктів, послуг чи контенту, які рекомендуються. На основі зважування атрибутів та історії рекомендаційна система створює унікальну модель переваг кожного користувача. Модель складається з атрибутів, які можуть сподобатися користувачеві на основі минулих дій, зважених за важливістю. Моделі користувачів порівнюються з усіма об'єктами бази даних, яким потім надаються оцінки на основі їх схожості з профілем користувача. Атрибути, що з'являються в декількох об'єктах, мають більшу вагу, ніж атрибути, які з'являються рідше. Це допомагає встановити ступінь важливості, оскільки всі атрибути об'єкта рівні користувачеві. Відгуки користувачів також мають важливе значення при зважуванні елементів, тому веб-сайти, що надають рекомендації, постійно просять вас оцінити продукти, послуги або контент.

Фільтрація за контентом потребує аналізу цього контенту. Зазвичай це текстова частина контенту для якої використовують алгоритми обробки природної мови (NLP). В фільтрації на

основі контенту використовують алгоритм пошуку підходящих найбільш важливих слів TF-IDF та алгоритм LDA для виявлення схожих тенденцій в описах.

Фільтрації за знаннями дає рекомендації не на основі історії оцінок користувача, а на конкретних запитах, зроблених користувачем. Ця модель може запропонувати користувачеві надати низку правил чи вказівок щодо того, як мають виглядати результати, або приклад елемента. Ці типи системи рекомендацій використовуються в певних доменах, де історія покупок користувачів менша. У таких системах алгоритм бере до уваги знання про елементи, такі як функції, уподобання користувача, які явно запитуються, і критерії рекомендацій уточнюються, перш ніж надавати рекомендації.

Системи рекомендацій на основі знань можуть бути досить корисними, особливо в поєднанні з іншими формами систем рекомендацій. Вони можуть бути задіяні короткостроково, як рішення проблеми холодного запуску та переходити на спільну фільтрацію або системи на основі вмісту, якщо вже отримано достатню кількість оцінок. Рекомендації на основі знань також можуть бути цінними інструментами, коли простір елементів складний або використовується рідко.

Гібридні системи розробляються для усунення обмежень попередніх моделей, комбінування їх переваг та підвищення ефективності рекомендацій. Основною метою більшості моделей гібридних рекомендацій, є компенсація відсутності рейтингових даних шляхом інтеграції інформації моделей фільтрації на основі вмісту та спільної фільтрації. Модель гібридних рекомендацій ділиться на сім типів які описують методи що лежать в основі гібридизації: зважена гібридизація, комутаційна гібридизація, каскадна гібридизація, метарівень, змішана гібридизація, комбінація функцій, розширення функцій.

Оцінка схожості користувачів та елементів. Різні типи рекомендаційних систем для формування списку рекомендацій у більшості випадків, особливо для персоналізованих рекомендацій, потребують визначення схожості елементів чи користувачів між собою. Схожість користувачів може розраховуватися, як на основі бінарних даних, так і на основі рейтингів. Для розрахунку схожості в рекомендаційних системах використовують такі функції як: кореляція Пірсона, косинус подібності та коефіцієнт Жаккара.

Кореляція Пірсона вимірює ступінь, в якій дві змінні лінійно пов'язані одна з одною. Розрахунки проводяться використовуючи рейтинги користувачів для елементів. В результаті отримуємо значення того наскільки користувачі корелюють між собою.

Косинус подібності вимірює коефіцієнт подібності двох векторів. Для спільної фільтрації векторами є строки матриці рейтингів. Також косинус подібності можна використовувати в рекомендаційних системах на основі контенту де подібність між двома документами можна виміряти, розглядаючи кожен документ як вектор частот слів та обчислюючи косинус кута, утвореного векторами частот.

Відстань Жаккара надає міру подібності між двома наборами шляхом підрахунку кількості загальних елементів, які вони мають, та поділу на загальну кількість унікальних елементів між ними. Це відношення кількості елементів, якими вони користуються, до кількості елементів, якими вони потенційно можуть поділитися.

Оцінка якості роботи рекомендаційних систем. Якість роботи рекомендаційної системи вказує на те наскільки надані рекомендації відповідають вподобанням користувача. Крім того, не рекомендувати елементи, які не відповідають вподобанням користувачів, також є частиною якісної моделі системи рекомендацій. Для оцінки якості моделі рекомендацій загалом використовують такий індикатор як точність.

Найчастіше оцінку точності рекомендацій розробники проводять за допомогою метрик, які вимірюють здатність рекомендаційної системи достовірно представляти набір відомих

вподобань. Здебільшого використовують метрики для оцінки точності реального та очікуваного значення, такі як MAE та RMSE.

MAE (Mean Absolute Error) – середня абсолютна помилка. Це показник, який є результатом обчислення середнього значення всіх різниць між прогнозованою та реальною оцінкою за модулем. Чим нижче MAE, тим вища точність. MAE не дуже чутлива до великих помилок, тому використовується коли пред'явлення користувачеві рекомендацій, які можуть не найкраще відповідати його вподобанням, не критичне.

RMSE (Root Mean Squared Error) – середньоквадратична помилка. Ця метрика повідомляє квадратний корінь із середньої квадратичної різниці між прогнозованими та фактичними значеннями у наборі даних. Чим нижче RMSE, тим краще модель відповідає набору даних. RMSE більш чутливий до великих помилок, бо вони можуть значно вплинути на оцінку RMSE, роблячи цей показник найбільш цінним тоді, коли необхідно щоб жоден з користувачів не отримував не точні рекомендації.

Окрім MAE та RMSE є також ще такі загальні метрики якості рекомендаційної системи такі як: точність, повнота, акуратність, F-міра, ROC крива та AUC. Вони розраховуються використовуючи матрицю помилок яка будується на основі таких даних як: справжні позитивні результати які вказують на кількість елементів, які відповідають уподобанням користувача; справжні негативні результати, що представляють кількість елементів, перевагу яким надає користувач, але які не рекомендовані системою рекомендацій; помилкові спрацьовування, які представляють кількість випадків, коли система рекомендує елементи, яким користувач не віддає перевагу; помилково негативні результати, що представляють кількість, коли система не надає рекомендацій щодо елементів, яким користувачі не віддають перевагу. Кожен рядок в цій матриці означає елемент, який відображає переваги користувача, а кожен стовпець вказує, чи модель рекомендації рекомендувала відповідний елемент.

На основі цих даних точність виступає як співвідношення елементів які відповідають вподобанням користувача серед усіх рекомендованих користувачеві товарів. Повнота як частка релевантних елементів, знайдених у перших k рекомендаціях. Акуратність – співвідношення вдалих рекомендацій до загальної кількості рекомендацій. F-міра - середнє гармонійне значення точності та повноти. ROC крива - графік, що показує співвідношення результатів продуктивності точності та повноти. AUC – область під кривою ROC яка вимірює ймовірність того, що випадковий релевантний елемент має вищий рейтинг, ніж випадковий нерелевантний елемент.

Особливості рекомендаційних систем. Проблема «холодного старту». Коли новий користувач знайомиться з системою рекомендацій, практично неможливо надати йому індивідуальні рекомендації. Це явище відоме як проблема холодного запуску . Один із варіантів виправити це — прямо запитати користувача, які продукти його цікавлять, наприклад, показавши йому кілька елементів і запитавши, чи вони актуальні чи ні. Інший, можливо, більш прагматичний підхід полягає в тому, щоб рекомендувати лише найпопулярніші товари.

Довгий хвіст. Ще одна поширена проблема з системами рекомендацій полягає в тому, що зазвичай дуже мало елементів мають розумну кількість оцінок-взаємодій. Ці так звані популярні предмети можуть зрештою переважати в рекомендаціях, тоді як менш популярні продукти рекомендуватися рідко. Це відоме як проблема довгого хвоста і від неї особливо страждають системи спільних фільтрів.

Маштабованість. Масштабування систем рекомендацій для мільйонів користувачів і елементів може швидко стати неможливим, оскільки це потребує все більшої кількості обчислювальної потужності. Деяким технологічним гігантам пощастило мати цю потужність, але навіть у цьому випадку може бути важко надати рекомендації в реальному часі. У цій екстремальній ситуації розумною стратегією є штучне зменшення кількості елементів,

наприклад, розглядаючи елементи лише з певної категорії, або навіть використання випадкового вибору на першому етапі генерації кандидатів. Крім того, рекомендації можна попередньо розрахувати та зберегти, щоб їх можна було представити користувачеві під час наступного входу.

Оцінка систем рекомендацій. Іншою проблемою є оцінка якості систем рекомендацій. Офлайн-оцінка часто лише частково відображає реальну ситуацію - досить важко, якщо не неможливо, передбачити, як би поводився користувач, якби він побачив інші рекомендації. Насправді якісні рекомендації повинні бути не тільки актуальними , але водночас новими та різноманітними.

Проблема фільтруючих бульбашок. Якщо користувач споживає здебільшого вміст, який йому рекомендували, не тому, що рекомендації були правильними, а просто тому, що легше клацати вміст, який уже є, ніж активно переглядати новий, тоді це провокує появу так званих фільтруючих бульбашок, найбільш відомих у зв'язку зі стрічками новин із платформ соціальних мереж, таких як Twitter або Facebook. Тому особливо в цьому середовищі важливо зосередитися на новизні та різноманітності рекомендацій.

Обрання засобів розробки рекомендаційної системи. Розробка рекомендаційної системи в залежності від вимог та базової системи для якої вона розробляється потребує використання відповідних технологій. Зазвичай рекомендаційні системи розробляються для веб-додатків в основі яких лежить клієнт-серверна архітектура. Це передбачає неодмінне використання таких веб-технологій як HTML, CSS, JavaScript [5,6,7].

Також клієнт-серверна архітектура передбачає використання серверних технологій. Серверна частина може розроблятись з використанням різних мов, наприклад таких як PHP, Python, C# або Java. В рамках поточної розробки використовувався стек технологій Java, який включає в себе фреймворк Spring, Spring Boot, Spring Data JPA, Spring Security, Thymeleaf, а також система керування базами даних MySQL. Обрання саме такого стеку технологій обумовлений, по перше, технічними вимогами до проекту зі сторони замовника. Це безпосередньо мова Java. Спираючись на це, випливає застосування сумісних з мовою фреймворків та бібліотек, які дозволяють писати програмне забезпечення на стороні сервера. Тому обраний саме фреймворк Spring та Spring Boot.

Spring – це один з найпопулярніших універсальних фреймворків технології Java. Він призваний вирішити проблеми зі складністю розуміння специфікацій Jakarta Enterprise Edition (Java EE). Spring являє собою комплексне рішення в розробці та конфігурації сучасних додатків корпоративного рівня на платформі Java.

Spring Boot являє собою доповнення до фреймворку Spring. Це інструмент використовуючи який розробники можуть прискорити процес розробки веб-додатків та мікросервісів.

Для збереження великих обсягів даних, які генеруються та обробляються під час роботи веб-додатку рекомендаційної системи, застосовується база даних. Виходячи з потреби бути сумісною із стеком Java, обраний фреймворк Spring Data JPA [8]. Він відповідає за роботу додатків з базами даних. Основним його завданням є спрощення роботи з базою шляхом зменшення шаблонного коду притаманного рівню DAO (Data Access Object) та надання знайомої та узгодженої моделі доступу до даних, зберігаючи при цьому можливості та особливі базового сховища даних. В роботі застосована СУБД MySQL, реляційна система управління базами даних. Вона характеризується високим ступенем незалежності від платформи та надає можливості обробляти великі обсяги даних за короткий час і вважається системою баз даних з високим рівнем безпеки та хорошою стабільністю. Також ця система характеризується високою гнучкістю та широкою сумісністю з інтерфейсами до безлічі різних програм.

Для забезпечення елементів безпеки на веб-ресурсі, застосований фреймворк Spring Security, який надає потужні функції захисту додатків від несанкційованого доступу. Spring Security надає широкий спектр функцій безпеки, включаючи підтримку механізмів аутентифікації та авторизації, шифрування та керування сесіями.

Для генерування контенту для перегляду у веб-браузерах користувачів, це сучасний Java-шаблонізатор серверного типу Thymeleaf [9,10], розроблений з урахуванням стандартів Web. Thymeleaf використовується для створення шаблонів, які можна повторно використовувати в проекті в залежності від потреб відображення, і пропонує великий набір функцій, які роблять його потужним інструментом для розробників веб-додатків.

Архітектура веб-додатку. Розробка рекомендаційної системи відбувалась з урахуванням архітектури веб-додатка для якого додається функціонал рекомендацій. Він побудований використовуючи фреймворк Spring, доповнення до нього у вигляді Spring Boot та фреймворки Spring Security і Spring Data JPA.

В основі архітектури веб-додатку лежить модель MVC, яка пропонує спосіб розділення коду на окремі частини кожна з яких відповідає за свої окремі завдання. Розділення відбувається на такі частини як: модель, представлення та контролер.

Код додатку розділений між окремими пакетами, назва яких говорить про зберігання в них відповідних частин згідно моделі MVC, але також має додаткове розділення на код який описує бізнес-логіку.

Пакет «controller» містить в собі контролери – класи, що відповідають за обробку запитів GET або POST та описують дії сервера на відповідні запити.

Пакет «entity» містить моделі елементів об’єкти даних яких зокрема зберігаються в базі даних.

В пакеті «repository» знаходяться інтерфейси, які пов’язані з entity та імплементують CrudRepository

Пакет «service» містить класи в яких реалізована певна бізнес логіка така як розрахунки, взаємодія з різними об’єктами та репозиторіями.

Каталоги які знаходяться в «src/main/resources» відповідають за зберігання клієнтської частини. В каталозі «static» знаходяться файли каскадних таблиць стилів та файли зі скриптами JavaScript. Каталог «templates» зберігає в собі шаблони та окремі частини написані використовуючи HTML та Thymeleaf.

Організація взаємодії із базою даних. За допомогою Spring Data JPA структура бази даних формується автоматично на основі entity вказаних в інтерфейсах, що імплементують CrudRepository.

Для роботи рекомендаційної системи задля зберігання даних взаємодій користувача з елементами в пакетах entity та repository були створені entity клас UserClickLog та інтерфейс UserClickLogRepo відповідно. Також в тих самих пакетах для зберігання даних про розраховані рейтинги користувача для елемента створені клас UserEquipmentRate та інтерфейс UserEquipmentRateRepo.

Spring Data JPA на основі об’єктів цих класів створив в базі даних таблиці з колонками, які відповідають полям цих класів.

Аналіз роботи розробленої рекомендаційної системи. Для відображення рекомендацій на головній сторінці в області знайденого обладнання додано блок «Рекомендації» (Рис. 2). Рекомендації в цьому блоці помічені іншим кольором для більш явного розділення. Сортування рекомендацій відбувається за релевантністю. У разі холодного старту у списку рекомендацій відображаються найновіші елементи.

Список рекомендацій що відображається не залежить від пошукового запиту. Він формується на основі неявних рейтингів користувача для елементів мережевого обладнання.

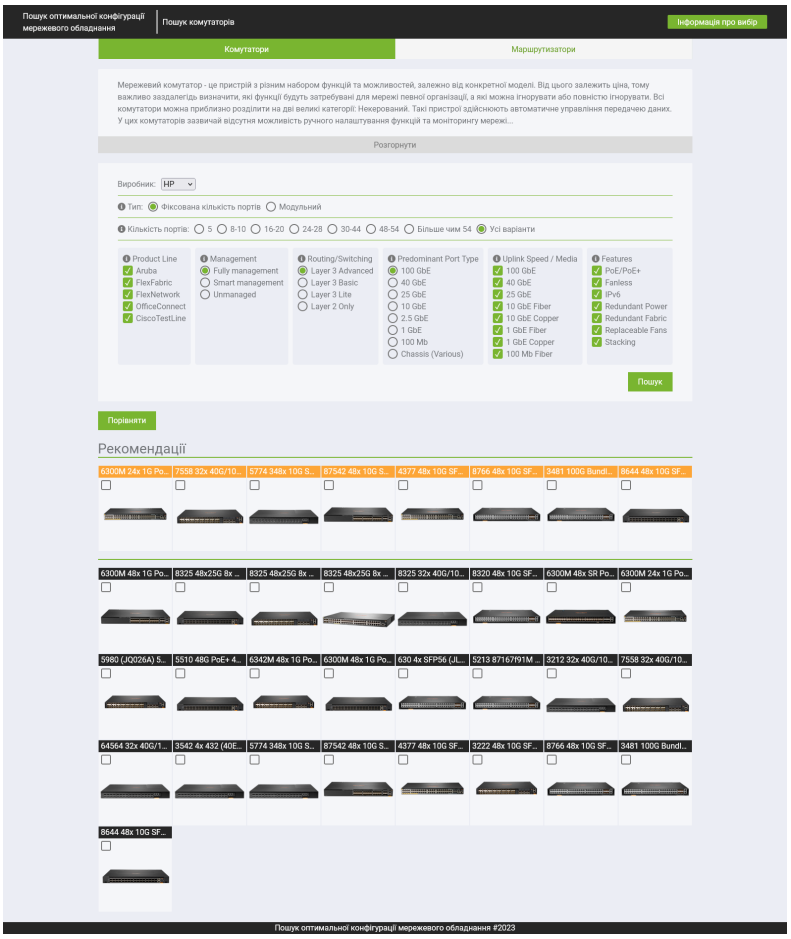

Рис. 2 Вигляд інтерфейсу рекомендаційної системи у браузері

Неявний рейтинг користувача для елементу розраховується на основі взаємодії цього користувача з елементами списку отриманого в результаті пошуку. Кожна взаємодія користувача це подія, яка реєструється сервером та записується в базу даних. Для кожного типу події на сервері записані вагові коефіцієнти які визначають її важливість. Ці коефіцієнти задаються вручну та за потреби корегуються.

Після збору даних щоб отримати неявний рейтинг користувача для елемента підраховується кількість однакових подій та проводяться розрахунки за наступною формулою:

$$IR_{i,u} = (w_1 * \text{подія}_1) + (w_2 * \text{подія}_2) + (w_3 * \text{подія}_3), \quad (1)$$

де w_k - це ваговий коефіцієнт відповідної події;
подія_k – кількість того скільки разів відбулася дана подія.

Робота підсистеми «Рекомендатор». Отримавши неявні рейтинги користувача для елементів в дію вступає рекомендатор який працює використовуючи спільну фільтрацію на основі k-найближчих сусідів з кореляцією Пірсона.

Створенню рекомендацій передує декілька етапів. В першому етапі виконується пошук сусідів: перебираючи усі рейтинги поточного користувача алгоритм знаходить інших користувачів, які мають оцінки для тих самих елементів, що й поточний користувач і записує їх в список співпадінь. Використовуючи цей список проводяться розрахунки коефіцієнта подібності інших користувачів до поточного за допомогою формули кореляції Пірсона:

$$sim(u,v) = \frac{\sum_{i\in I}(r_{u,i}-\bar{r}_u)(r_{v,i}-\bar{r}_v)}{\sqrt{\sum_{i\in I}(r_{u,i}-\bar{r}_u)^2}\sqrt{\sum_{i\in I}(r_{v,i}-\bar{r}_v)^2}}, \quad (2)$$

де i ∈ I підсумовується за елементами які були оцінені користувачами;

$r_{u,i}$ – оцінка користувача;

$\bar{r}$ – середня оцінка з усіх.

На другому етапі відбувається прогнозування рейтингів для поточного користувача: перебираючи елементи для яких у поточного користувача немає рейтингів прогнозується можливий рейтинг за формулою:

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v\in P_u(j)} \mathrm{Sim}\,(u,v)\cdot(r_{vj}-\mu_v)}{\sum_{v\in P_u(j)} |\mathrm{Sim}\,(u,v)|}, \quad (3)$$

де μ_u – середній рейтинг поточного користувача;

$P_u(j)$ – це список найближчих сусідів до поточного користувача;

$\mathrm{Sim}(u,v)$ - коефіцієнт подібності користувачів;

r_{vj} - рейтинг користувача-сусіда;

μ_v – середній рейтинг користувача-сусіда.

В результаті отримуємо список елементів з прогнозованим рейтингом, який сортирується та відбираються елементи, рейтинг яких не менший за заздалегідь заданий.

Фінальним етапом стає відображення отриманих елементів користувачу через клієнтську частину.

Висновки.

Таким чином, проведений аналіз основних типів та підходів до роботи рекомендаційних систем дозволив спроектувати архітектуру веб-орієнтованої рекомендаційної системи, яка працює з двома типами даних: взаємодія користувача з елементом (рейтинги або поведінка під час вибору потрібного елемента) та атрибутивна інформація про користувача та елементи. Для технічної реалізації проекту здійснено підбір сумісних технологій, які утворили потрібний стек Java. Розроблено рекомендаційну система для підбору ефективних конфігурацій мережевого обладнання, де застосована колаборативна фільтрація на основі сусідства з використанням кореляції Пірсона. Система надає релевантні рекомендації користувачу, зменшуючи час на пошук необхідних елементів комп'ютерної мережі та вдосконалює процес прийняття рішень, щодо якості вибору мережного обладнання. Це дозволяє підвищити ефективність роботи із подібним класом веб-додатків під час вирішення задач пошуку оптимальної конфігурації мережного обладнання.

Список використаної літератури:

1. Charu C. Aggarwal Recommender Systems: The Textbook 3st ed.– Springer, 2022. – 518 c.

2. Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B. Kantor Recommender Systems Handbook – Springer, 2011. – 845 c.

3. Kim Falk - Practical Recommender Systems – Manning, 2019. – 448 c.

4. D. Jannach, M. Zanker, A. Felfernig, G. Friedrich Recommender Systems: An Introduction – Cambridge University Press, 2011 – 352c.

5. W3schools CSS [Електронний ресурс] – Режим доступу: https://www.w3schools.com/css/default.asp (дата звернення 11.09.2023)

6. W3schools HTML [Електронний ресурс] – Режим доступу: https://www.w3schools.com/html/default.asp (дата звернення 11.09.2023)

7. MDN WEB DOCS JavaScript [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата звернення 11.09.2023)

8. Spring Data JPA - Reference Documentation [Електронний ресурс] – Режим доступу: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/ (дата звернення 11.09.2023)

9. Офіційний сайт ThymeLeaf [Електронний ресурс] – Режим доступу: https://www.thymeleaf.org/ (дата звернення 11.09.2023)

10. Документація ThymeLeaf [Електронний ресурс] – Режим доступу: https://www.thymeleaf.org/documentation.html (дата звернення 11.09.2023)

11. Офіційний сайт Spring Framework [Електронний ресурс] – Режим доступу: https://spring.io/ (дата звернення 11.09.2023)

12. Офіційний сайт MySQL [Електронний ресурс] – Режим доступу: https://www.mysql.com/ (дата звернення 11.09.2023)

Автори статті

Гончаренко Олександр - магістр, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

Ільїн Олег – доктор технічних наук, професор, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

Кравчук Петро – PhD, доцент, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

Фесенко Максим – кандидат технічних наук, доцент, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

Authors of the article

Honcharenko Oleksandr – master of science, State University of Information and Communication Technologies, Kyiv, Ukraine.

Ilin Oleh – Sciences of Doctor (technic), professor, State University of Information and Communication Technologies, Kyiv, Ukraine.

Kravchuk Petro – PhD, associate professor, State University of Information and Communication Technologies, Kyiv, Ukraine.

Fesenko Maksim - candidate of technical sciences, associate professor, State University of Information and Communication Technologies, Kyiv, Ukraine.