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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

Система підтримки обміркованого вибору товару на основі виявлених переваг покупця

Галузь знань **12 «Інформаційні технології»**

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

Освітня програма **«Аналітика даних»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи АнД-41

_____Мінін І.Б._____
(прізвище та ініціали)

Керівник _____Циганок В.В._____
(прізвище та ініціали)
____Доктор технічних наук, с.н.с.____
(науковий ступінь, звання)

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

Київ – 2023

Зміст

Анотація

Мінін Ігор Борисович виконав випускну кваліфікаційну роботу на тему «Система підтримки обміркованого вибору товару на основі виявлених переваг покупця» за спеціальністю 122 – «Комп'ютерні науки». У випускній кваліфікаційній роботі проведено аналіз існуючих методів багатокритеріального аналізу, розроблено програмне забезпечення виду системи підтримки прийняття рішень, що реалізує один із методів критеріального аналізу та надає користувачу персоналізовані рекомендації.

Ключові слова: система підтримки прийняття рішень, критерій обміркованого вибору, реклама товару, рекомендація.

Summary

The degree project: "A system of support for a deliberate choice of goods based on the identified preferences of the buyer" in the specialty 122 - "Computer Science" has completed by Minin Ihor specialty 122 – «Computer Sciences»

In the final qualification work, an analysis of the existing methods of multi-criteria analysis was carried out, software of a kind of decision support system was developed, which implements one of the methods of criterion analysis and provides the user with personalized recommendations.

Keywords: criterion, decision support system, recommendation.

ВСТУП

У сучасному світі, де доступ до різноманітних товарів стає все більшим, процес вибору оптимального товару може стати викликом для багатьох покупців. Завантажений ринок, розмаїття варіантів та інформаційний шум ускладнюють прийняття розумних рішень. В таких умовах, системи підтримки обміркованого вибору товару стають надзвичайно корисними інструментами, що допомагають покупцям зорієнтуватися та зробити оптимальний вибір.

Об'єктом дослідження є процес індивідуального вибору товару покупцем за власними перевагами під впливом реклами.

У цьому випадку, процес обміркованого вибору товару це коли покупець розглядає різні варіанти, аналізує їх характеристики та розглядає свої власні потреби та вимоги. Об'єкт дослідження передбачає дослідження психологічних та економічних аспектів, що впливають на прийняття рішення покупцем, такі як особисті уподобання, бюджет, якість товару, його вартість та інші фактори, що враховуються під час вибору.

Предметом дослідження є розробка програмної системи підтримки обміркованого вибору товару на основі виявлених переваг покупця.

Предмет дослідження включає розробку програмного забезпечення, яке забезпечує збір та аналіз даних про вимоги та переваги покупця, автоматичне порівняння різних товарів, враховуючи їх характеристики та параметри, та надання рекомендацій покупцю щодо найкращого варіанту вибору товару, відповідно до його потреб і вимог. Така система має допомагати покупцям зробити обґрунтовані рішення під час покупки товару та підвищити задоволення від покупки.

Основною метою розробки системи підтримки обміркованого вибору товару на основі виявлених переваг покупця є надання можливості вибору обгрунтованого рішення, що сприяє ефективному та обгрунтованому прийняттю рішеня покупцем під час покупки товару.

Одним із ключових аспектів системи є виявлення переваг покупця. Кожен покупець має свої унікальні вимоги, вподобання та критерії, які відіграють важливу роль у процесі вибору. Система буде заснована на використанні аналітичних методів, алгоритмів для виявлення найкращих альтернатив товару для покупця. Це дозволить системі ефективно аналізувати та ранжувати товари відповідно до виявлених переваг, забезпечуючи зручну та персоналізовану взаємодію з покупцем.

Очікується, що результати цієї дипломної роботи не тільки допоможуть покупцям у виборі товару, але й принесуть користь бізнесу, сприяючи покращенню взаємодії між покупцями та постачальниками, а також забезпечуючи підвищення конкурентоспроможності та задоволення потреб ринкових учасників.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ІСНУЮЧИХ СИСТЕМ ІНТЕЛЕКТУАЛЬНОЇ РЕКЛАМИ ТА СИСТЕМ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

1.1 Аналіз предметної області

У фізичних магазинів є обмеження - розмір. Як би ми не хотіли, але дві тисячі телевізорів у торгову залу виставити неможливо. Для вирішення цієї задачі довелося би купити значні площі території та розширити магазин, але по-перше люди фізично не зможуть подивитися всі запропоновані телевізори, а по-друге купити території, побудувати торгові площі та найняти велику кількість персоналу буде коштувати дуже і дуже дорого. Що потрібно щоб розмістити дві тисячі телевізорів в інтернет-магазині? 10 мБ пам’яті, яка коштує копійки, та веб-сторінку з пагінацією. Компанії, які створюють певні товари, почали продавати більшу різноманітність товарів, адже тепер покупці можуть швидше передивитися доступні варіанти та обрати той, який їм більше до вподоби.

Як приклад товару надалі будуть використані смартфони, оскільки вони є одними з найпопулярніших торгових одиниць на сьогодні. Всі тези стосовно смартфонів стосуються й більшості інших можливих товарів.

Стрімке збільшення населення землі, особливо у таких промислово розвинених країнах як Китай та Індія, а також суцільна цифровізація збільшує і кількість користувачів смартфонів. Наглядно це зростання можна побачити на рисунку 1.1.

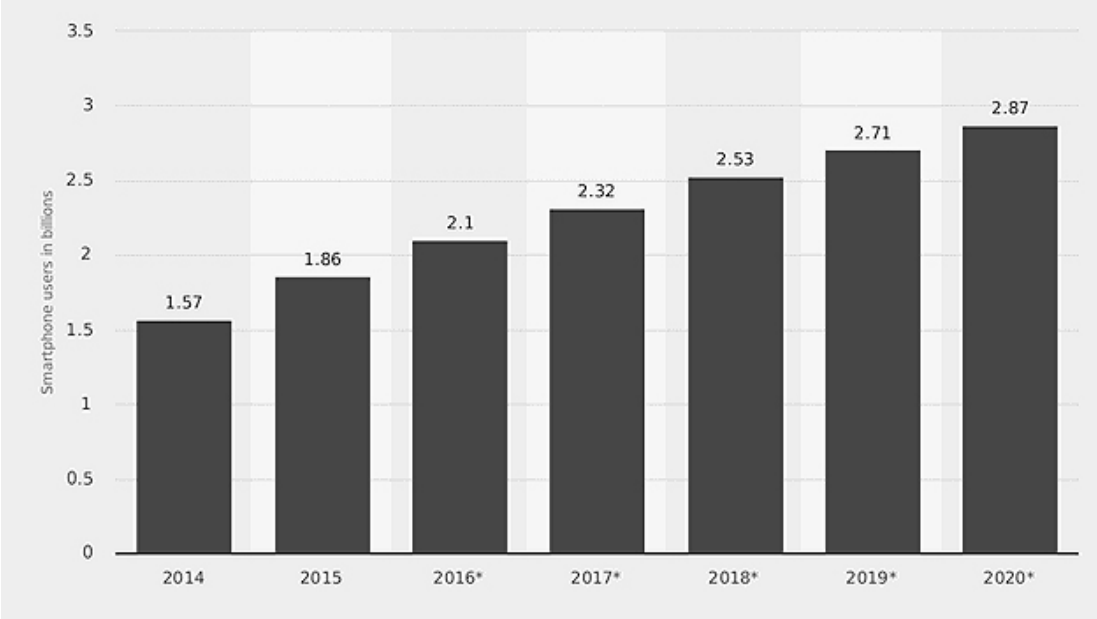


Рисунок 1.1 - кількість користувачів смартфонів у період 2014-2020 років[9]

Є попит - є пропозиція. Як можна побачити на рисунку 1.2 за 13 років кількість проданих смартфонів за рік зросла майже в 7 разів. Дана тенденція є сприятливою для створення великої кількості нових компаній, які розширюють і без того широкий асортимент товарів.

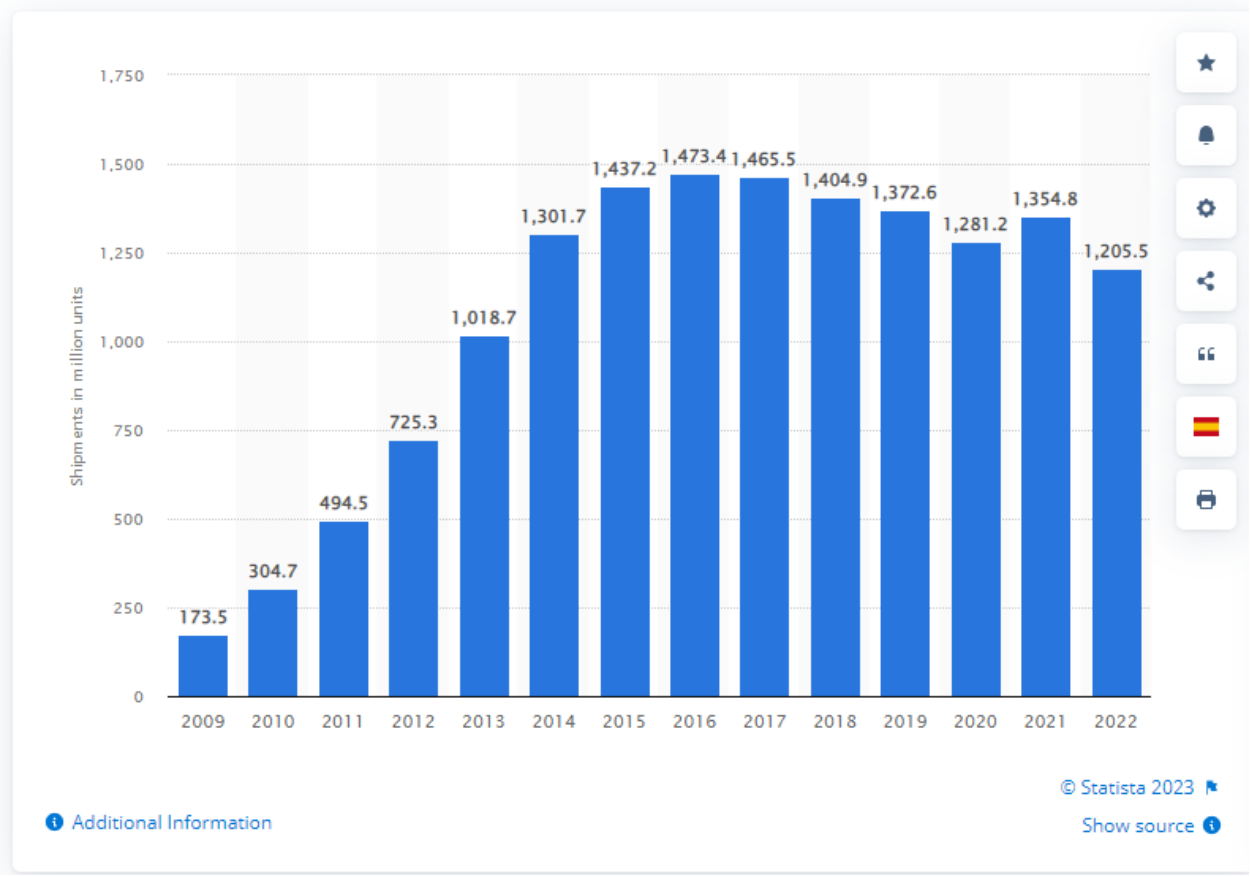


Рисунок 1.2 - кількість проданих смартфонів у мільйонах штук у період 2009-2022 років[10]

Приблизно десять років назад було декілька відомих виробників телефонів. Це такі компанії як Apple, Samsung, Nokia, Research in Motion (більш відомий як BlackBerry), HTC. Вони домінували і займали майже три чверті ринку смартфонів. За даними [12] за 2011 рік топ компаній за продажами смартфонів представлено в табл. 1.1.

Таблиця 1.1 - Світові продажі мобільних телефонів кінцевим споживачам у 2011 р.

Компанія	Кількість проданих смартфонів, млн. шт.	Доля ринку в 2011, %	Країна
Samsung	97.4	20.1	Південна Корея
Apple	93	19.2	США
Nokia	77.3	15.6	Фінляндія
Research in Motion	51.5	10.6	Канада
HTC	44.4	9.1	Тайвань
Sony Ericsson	21.1	4.4	Японія
LG	20.2	4.2	Південна Корея
Motorola	18.6	3.8	США
Huawei	18.4	3.8	Китай
ZTE	11.5	2.3	Китай
Інші	30.5	6.3	
Загалом	483.8	100	

На сьогодні, до перших двух доєдналася велика кількість інших компаній, в основному китайських: Xiaomi(який в свою чергу ділиться ще й на Redmi та Poco), Realme, Vivo, Oppo, інші. За даними[13] за 2022 рік топ компаній за продажами вже інший (див. Таблиця 1.2).

Таблиця 1.2 - Світові продажі мобільних телефонів кінцевим споживачам у 2022 р.

Компанія	Кількість проданих смартфонів, м	Доля ринку в 2022, %	Країна
Samsung	259	21	Південна Корея
Apple	231	19	США
Xiaomi	152	13	Китай
Oppo Group	107	9	Китай
vivo	98	8	Китай
Transsion	68	6	Китай
Honor	59	5	Китай
Realme	53	4	Китай
Motorola	47	4	США
Huawei	28	2	Китай
Інші	104	9	
Загалом	1207	100	

Як можна побачити з статистики, китайські виробники смартфонів зайняли майже половину ринку. Вони виробляють велику кількість смартфонів, та, через їх дешевизну, продаються у величезних обсягах. За 11 років, з 2011 по 2022, кількість проданих смартфонів зросли майже в 2.5 рази, з 491 млн одиниць до 1.2 млрд.

В кожного з виробників є декілька модельних рядів, а в кожної моделі є різні підмоделі, кількість яких за десять років також значно збільшилася.

Наприклад, в 2011 продукт компанії Apple мав варіанти iPhone 4 та 4s, сучасна модель має вже 4 варіації 14, 14 plus, 14 pro, 14 pro max, при чому в продажі ще залишаються моделі попередніх років.

Десять років назад покупець, зайшовши на сайт, міг досить легко визначити серед невеликої кількості фірм та моделей найкращий для себе варіант за якісними та кількісними характеристиками. Сьогодні можна витратити години, або навіть дні, аналізуючи сотні доступних варіантів, порівнюючи їх характеристики та передивляючись огляди чи коментарі інших покупців.

Більшість сайтів мають рекомендаційну систему на основі найпопулярніших товарів серед покупців та товарів, переглянутих користувачем. Перший вид рекомендацій може зовсім не відповідати потребам користувача, адже не покладається на попередній досвід конкретного користувача, а другий може робити випадкові рекомендації, які не допоможуть клієнту в пошуці, адже все рівно доведеться переходити на сторінку із списком товарів і робити часозатратний аналіз найбільш підходящих варіантів . Тому написання програмного модулю для пришвидшення прийняття рішення покупцем є надзвичайно важливою та перспективною задачею, адже, як відомо, час - це гроші.

1.2 Теоретична основа існуючих систем інтелектуальної реклами

У цьому розділі розглянуто існуючі системи інтелектуальної реклами, що використовують адаптацію до переваг покупця. Проведений аналіз допоможе зрозуміти, які принципи та методи використовуються в цих системах, як вони працюють та які переваги вони можуть мати.

1.2.1 Системи рекомендацій

Системи рекомендацій є одними з найбільш поширених і відомих систем інтелектуальної реклами. Вони використовуються в онлайн-магазинах, сервісах потокової передачі музики та відео, соціальних мережах та інших платформах. Системи рекомендацій надають користувачам персоналізовані рекомендації на основі їхньої історії переглядів, вподобань та інших даних.

Системи рекомендацій зазвичай використовують алгоритми машинного навчання, крім того, можуть бути використані і інші методи, такі як аналіз тексту та зображень.

1.2.2 Системи персоналізованої реклами

Системи персоналізованої реклами також використовують адаптацію до переваг покупця. Вони надають персоналізовані рекламні пропозиції на основі даних про інтереси та поведінку користувача. Системи персоналізованої реклами можуть використовувати аналіз даних, алгоритми машинного навчання та інші методи для збору та обробки даних.

Перевагою систем персоналізованої реклами є те, що вони надають рекламу, яка краще відповідає інтересам та потребам користувача, що підвищує ефективність рекламної кампанії та знижує витрати на рекламу.

1.2.3 Системи динамічної реклами

Системи динамічної реклами використовують адаптацію до переваг покупця, щоб надати рекламу, яка змінюється в залежності від поведінки користувача. Наприклад, якщо користувач переглядає товар на сайті, система

динамічної реклами може відобразити рекламу зі схожим товаром. Це дозволяє залучити більше уваги користувача та підвищити ймовірність покупки.

Системи динамічної реклами можуть використовувати різні методи адаптації, включаючи географічне місцезнаходження користувача, погодні умови та інші параметри. Це дозволяє надавати рекламу, яка відповідає конкретним умовам, що підвищує її ефективність та знижує витрати на рекламу.

1.2.4 Системи реклами з використанням штучного інтелекту

Останнім часом все більшу популярність набувають системи реклами з використанням штучного інтелекту. Ці системи використовують алгоритми машинного навчання, щоб надавати рекламу, яка адаптується до переваг покупця.

Наприклад, системи реклами з використанням штучного інтелекту можуть аналізувати дані про користувачів, включаючи їхню історію покупок, поведінку в Інтернеті та інші параметри, щоб надавати персоналізовані рекламні пропозиції.

Перевагою систем реклами з використанням штучного інтелекту є те, що вони можуть адаптуватися до переваг покупця набагато ефективніше, ніж традиційні системи реклами. Вони також можуть надавати рекламу з високою точністю та швидкістю, що знижує витрати на рекламу та підвищує її ефективність.

Проаналізувавши різні види інтелектуальної реклами, було обрано до реалізації систему рекомендацій. Оскільки, найбільш достовірні вподобання покупця можна отримати лише від нього самого, наприклад, шляхом опитування, то варіанти персоналізованої та динамічної реклами розглядатись не будуть, оскільки вони працюють на основі автоматичного збору інформації

про клієнта. Системи з використанням штучного є складними в реалізації та не є оптимальними для вирішення поставленої задачі.

Як перспектива для покращення рекомендаційного модулю можна поєднати результати рейтингу альтернатив при опитуванні користувача та при автоматичному зборі даних про дії користувача в інтернет-магазині, порівнявши результати явних та неявних уподобань користувача можна найбільш точно виявити переваги покупця і запропонувати бажаний товар.

1.3 Системи підтримки прийняття рішень

СППР – це інтерактивна прикладна система, яка забезпечує кінцевим користувачам, котрі приймають рішення, легкий і зручний доступ до даних і моделей з метою прийняття рішень у ситуаціях напівструктурованих і неструктурованих даних з різних галузей людської діяльності.

За типами СППР поділяються на:

1) Комунікативні

   Призначені для одночасної роботи декількох експертів, які працюють над одним спільним завданням.

2) Інформаційні

   Орієнтовані на збір і обробку корисних даних, насамперед на аналіз часових рядів

3) Документальні

   Призначені для обробки та аналізу документів у різних форматах із структурованими та неструктурованим даними.

4) Інтелектуальні

Містять дані про рішення однотипних завдань та правила, за якими вони прийняті, пропонують готові алгоритми, засновані на накопиченому досвіді.

5) Моделювальні

Підбирають моделі бізнес-процесів за заданими критеріями (статистичні, фінансові, аналітичні).

Дуже спрощено загальна схема більшості СППР виглядає як на рисунку 1.3.

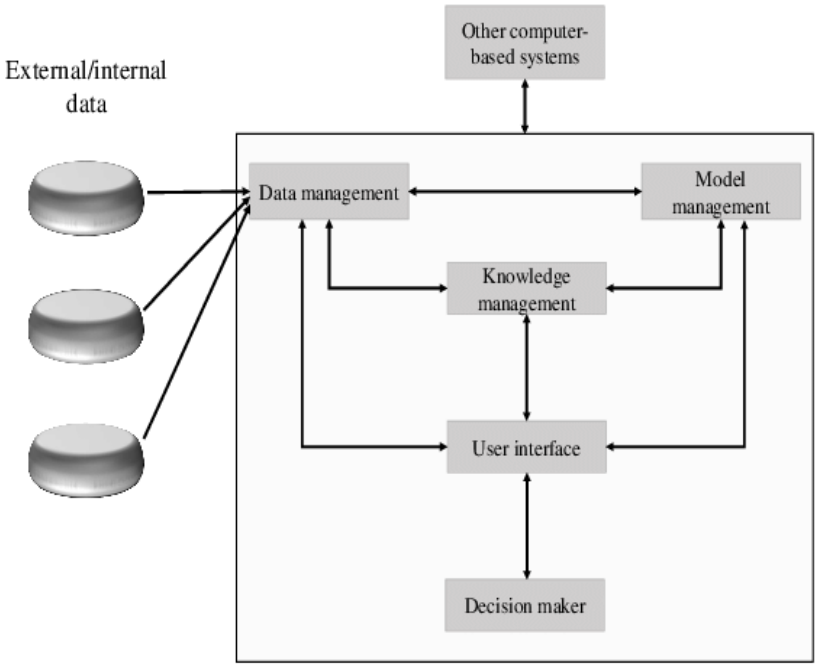


Рисунок 1.3 - загальна схема СППР[11]

Головна задача СППР допомогти прийняти рішення. Прийняття рішення – це процес вибору найбільш преференційного рішення з множини припустимих рішень або упорядкування множини рішень. Прийняття рішень можливо на підставі знань про об’єкт управління, процеси, що в ньому відбуваються і

можуть відбутися з перебігом часу, а також за наявності множини показників, що характеризують ефективність та якість прийнятого рішення

Характерними ознаками рішення є:

- можливість вибору з набору альтернативних варіантів, оскільки за відсутності альтернатив, відсутній і вибір, а тому відсутнє й рішення;
- наявність мети, оскільки вибір без цілі не розглядають як рішення;
- необхідність вольового акту особи, яка приймає рішення, при виборі найкращих альтернатив, оскільки дана людина формує рішення в умовах боротьбі мотивів і думок.

СППР можна також класифікувати по об’єкту їх керування. Вперше таку селекцію ввів Стевен Альтер. Він розрізняв два види СППР:

- Система керована даними (Data-Driven DSS, Data-oriented DSS) – в основному орієнтується на доступ і маніпуляції з даними;
- Система керована моделями (Model-Driven DSS) – забезпечує доступ і маніпуляції з математичними моделями (статистичними, фінансовими, оптимізаційними, імітаційними).

Ця класифікація на сьогодні є застарілою. Вона була розширена Деном Павером і, крім двох “драйверів” вище, розрізняють також наступні:

- Система керована документами (Document-Driven DSS) – здійснює пошук і маніпулювання неструктурованою інформацією, заданої в різних форматах;
- Система керована знаннями (Knowledge-Driven DSS) – забезпечує рішення задач у виді фактів, правил, процедур;

- Система керована повідомленнями (Communication-Driven DSS) – підтримує групу користувачів, що працюють над виконанням загальної задачі;

Однозначно зробити класифікацію СППР майже неможливо через велику кількість зовсім різних реалізацій рекомендаційних систем.

1.4 Постановка задачі

Отже, з усього вищенаписаного можна чітко окреслити проблему і мету цієї роботи.

Проблема - перенасиченість ринку великою кількістю схожих товарів, що створює складності при їх виборі для покупки.

Мета - розробити систему підтримки прийняття рішень для допомоги покупцю із вибором товару.

Рекомендаційна система для даної задачі має бути побудована у вигляді системи підтримки прийняття рішень, яка буде використовувати критеріальні методи вибору альтернатив. Це є найпростішим для користувача способом напряму заявити про свої бажання та отримати відповідний результат. Суто машинні методи збору інформації про вподобання користувача орієнтуються лише на попередній досвід покупця, який в свою чергу міг передивлятися зовсім не ті товари, які би він хотів купити, проте машинний алгоритм вже засмічений даними.

Функціональні вимоги:

1) Система повинна використовувати алгоритми критеріального аналізу для виявлення переваг покупця на основі зібраних даних.
2) Система повинна надавати персоналізовані рекомендації покупцям, враховуючи їхні індивідуальні переваги. Рекомендації повинні бути точними, зрозумілими та корисними для покупців.
3) Система має забезпечувати можливість оновлення та покращення алгоритмів виявлення переваг та рекомендацій з метою підтримки актуальності та ефективності системи.
4) Система повинна мати інтерфейс

Нефункціональні вимоги:

Надійність: Система повинна бути стійкою до помилок та збоїв, забезпечуючи безперебійну роботу і надійність результатів рекомендацій.

Швидкість: Система повинна працювати ефективно та швидко, забезпечуючи мінімальні часові затримки при обробці даних та генерації рекомендацій.

Точність: Система повинна мати високу точність та достовірність рекомендацій, забезпечуючи релевантні та коректні висновки на основі зібраних даних та аналізу.

Зрозумілість: Інтерфейс має бути простим у використанні та надавати зрозумілу інформацію щодо рекомендацій та можливостей налаштування вподобань.

Процес створення рекомендаційної системи буде складатися з наступних етапів:

1) Пошук даних

   Рекомендаційна система повинна вміти збирати, обробляти та аналізувати дані. Вони повинні збиратися автоматично. Найкращим чином це можна зробити парсингом реальних сторінок деякого інтернет-магазину.

2) Очищення та зберігання даних

   Перед додаванням даних в базу потрібно зробити стандартну процедуру аналітики даних, а саме очистку. Отримані дані повинні відповідати типам даних з програми, дані не повинні містити дублікати.

3) Створення бази даних

   Перш за все потрібно розробити архітектуру бази даних, продумати зв’язки між таблицями та між полями цих таблиць. Першим етапом буде створення концептуальної моделі бази даних, яка примітивно описує наявні сутності та зв’язки між ними. Зробити аналіз нормальності цієї концепції, за потреби привести її до нормальної форми. На основі цієї моделі і буде розроблена реляційна база даних.

4) Створення графічного інтерфейсу для користувачів, який буде складатися з
   a) Головної сторінки
   b) Сторінки з рекомендаційним модулем
   c) Сторінки товарів

5) Написання рекомендаційного модулю, його сутностей та методів.

   Безпосередньо рекомендаційний модуль буде працювати у вигляді опитувань. Користувач буде повинен на першому етапі обрати за якими характеристиками буде виконуватися фільтрація товарів, потім поставити фільтри, щоб зменшити кількість можливих варіантів товару, відкинути більшість з існуючих на сайті. Наприклад, прибрати всі товарі, що коштують більше ніж певну суму та мають менше певної кількості оперативної пам'яті. Для зручності користувача бажано залишити 4-6 альтернатив, серед яких і будуть застосовані рекомендаційні методи. На другому - вибрати критерії для порівняння та пройти опитування на вибір: повне чи адаптивне. Під адаптивним мається на увазі часткове опитування, коли користувач в будь-який час може закінчити його проходження.

РОЗДІЛ 2. ВИБІР МЕТОДУ БАГАТОКРИТЕРІАЛЬНОГО ВИБОРУ АЛЬТЕРНАТИВ ТА ПОБУДОВА АРХІТЕКТУРИ ЗАСТОСУНКУ

2.1 Аналіз методів багатокритеріального вибору альтернатив

Для вирішення задачі багатокритеріального вибору альтернатив існують десятки різних методів, розглянемо лише декілька з них.

2.1.1 Метод аналітичних мереж

При структуруванні однієї й тієї ж мети різні люди можуть збудувати різні ієрархії. При цьому в багатьох проблемах ухвалення рішень існують залежності між елементами різних рівнів ієрархії, що призводить до неможливості їх уявлення як ієрархічних структур. В ієрархічних структурах важливість критеріїв впливає пріоритети альтернатив. У структурах, відмінних від ієрархічних, важливість альтернатив впливає пріоритети критеріїв. МАС вирішує завдання прийняття рішень шляхом виконання наступних етапів:

- Етап 1: структурування проблеми прийняття рішень у вигляді аналітичної мережі;
- Етап 2: використання системи парних порівнянь для вимірювання ваги компонентів структури;
- Етап 3: ранжування альтернатив у вирішенні.

Цей метод не підходить для вирішення поставленої задачі, оскільки між елементами ієрархії нашої задачі не має залежностей, поставлену мету можна представити у вигляді ієрархічної структури.

2.1.2 Best Worst Method

BWM — це метод на основі парного порівняння, який пропонує структурований спосіб проведення порівнянь.

Алгоритм методу наступний:

1) Визначити критерії
2) Визначити найкращий(найважливіший для експерта) критерій
3) Визначити найгірший критерій
4) Створити вектори оцінок переваг найкращого та найгіршого критерію над іншими.
5) Знайти вектор оптимальних ваг. Для виконання цієї дії потрібно провести minmax оптимізацію.
6) Порахувати оцінку відповідності
7) Створити матрицю оцінок критеріїв для альтернатив
8) Помножити матрицю оцінок на вектор ваг критеріїв.

Отримані значення є рейтингом альтернатив.

Перевагами метода є його простота для користувача, адже йому не потрібно заповнювати багато матриць попарних порівнянь, а лише два вектори для найкращої та найгіршої альтернативи.

На жаль, даний метод не підходить для вирішення поставленої задачі, оскільки minmax операція є ресурсозатратною, що може вплинути на продуктивність застосунку.

2.1.3 Метод аналізу ієрархій

Метод аналізу ієрархій — це структурований метод організації та аналізу складних рішень, заснований на математиці та психології.

Метод полягає у декомпозиції проблеми на дедалі прості складові частини та подальшій обробці послідовності суджень особи, яка приймає рішення, за парними порівняннями. В результаті може бути виражений відносний ступінь взаємодії елементів у ієрархії. Ці міркування потім виражаються чисельно. MAI включає процедури синтезу множинних суджень, отримання пріоритетності критеріїв і знаходження альтернативних рішень. Цей метод найкраще підходить для вирішення поставленої задачі, оскільки дозволяє зібрати найбільшу кількість інформації з користувача і точно оцінити найкращу альтернативу. Недоліком методу є велика кількість порівнянь, проте метод можна модифікувати метод для роботи з неповними матрицями попарних порівнянь. Користувач буде попереджений про обов’язкове заповнення принаймні частини порівнянь в матриці попарних порівнянь, це є необхідною умовою для зв’язності графу відповідного матриці попарних порівнянь, всі інші порівняння є опціональними. Це, звичайно, вплине на точність результату, проте дозволить користувачу не витрачати час на повне заповнення матриці, якщо він цього не бажає. Пропущені порівняння будуть прийняті за одиницю, що дозволить просто знехтувати цими характеристиками при підрахунку ваг рядків матриць попарних порівнянь. Також вирішено не перевіряти узгодженість порівнянь, адже вибір смартфону не

потребує дуже велику точність розрахунків, проте значно вплине на кількість часу взаємодії користувача із сервісом та розробки.

2.2 Алгоритм методу аналізу ієрархій

2.2.1. Загальна модель MAI

Побудова загальної моделі MAI зазначена на рисунку 2.1

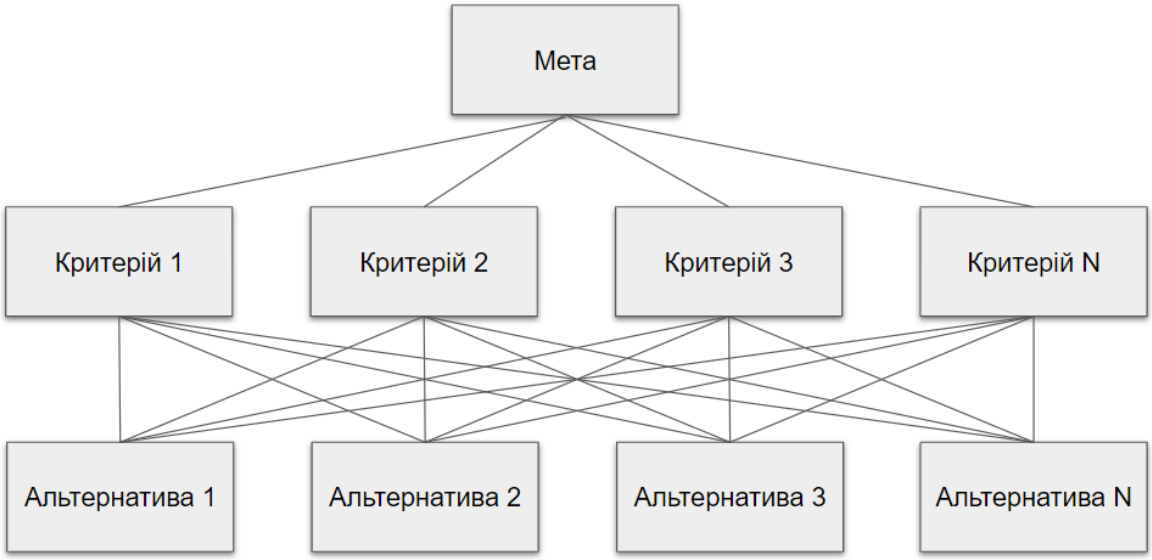


Рисунок 2.1 - модель MAI

2.2.2. Визначення пріоритетів елементів ієрархічної структури

Пріоритети визначаються на основі експертних оцінок в діапазоні [1, 9]

Таблиця 2.1 - ступені важливостей

Ступінь важливості	Визначення
1	Однакова значимість
3	Певна перевага значимості однієї дії перед іншою(слабка значимість)
5	Істотна чи сильна значимість
7	Дуже сильна чи очевидна значимість
9	Абсолютна значимість
2, 4, 6, 8	Проміжні значення між сусідніми значеннями шкали

Чому саме [1,9]? Автор методу (Томас Сааті) в своїй книзі “Багатокритеріальне прийняття рішень”[1], де він власне і описав MAI, пояснює це так: “Використання шкали парних порівнянь у діапазоні від 0 до ∞ може виявитися марним, оскільки передбачає, що людське судження якимось чином здатне оцінити відносну перевагу будь-яких двох об'єктів, що зовсім не так. Як добре відомо з досвіду, наша здатність розрізняти знаходиться в дуже обмеженому діапазоні, і коли є значна непомірність між порівнюваними об'єктами або діями, наші припущення тяжіють до того, щоб бути довільними, і,

зазвичай, виявляються далекими від дійсності.”

Після визначення ступенів важливості потрібно побудувати матрицю попарних порівнянь

	E_1	E_2	…	E_n
E_1	1	$1/a_{21}$	…	a_{1n}
E_2	a_{21}	1	…	$1/a_{n2}$
…	…	…	…	…
E_n	$1/a_{1n}$	a_{n2}	…	1

Таблиця 2.2 - матриця попарних порівнянь

На головній діагоналі розміщуються одиниці, оскільки кожен об’єкти/альтернатива не порівнюється сама з собою. Дана матриця оберненосиметрична: кожному значенню a є відповідне йому значення $1/a$, що розміщене симетрично відносно головної діагоналі.

2.2.3. Знаходження вектору пріоритетів

$W = [w_1, w_2, ..., w_n]$ - вектор пріоритетів.

Для кожного рядку матриці попарних порівнянь знаходимо середнє геометричне:

$$\square_{\square} = \sqrt[\square]{\prod_{\square=1}^{\square} \square_{\square\square}}, \quad (2.1)$$

де *n* - кількість рядків матриці попарних порівнянь

Отримані значення нормуємо: кожен елемент вектору *W* ділимо на суму елементів даного вектору.

$$\square_{\square} = \square_{\square} / \sum_{\square=1}^{\square} \square_{\square}, \quad (2.2)$$

де *n* - кількість елементів вектору *W*

Знаходимо вектори пріоритетів для всіх матриць попарних порівнянь альтернатив та критеріїв

2.2.4. Оцінювання узгодженості суджень експертів

В кожного вимірювання чи оцінювання є похибка. В випадку MAI це неузгодженість експертів. Наприклад експерт визначає, що критерій товару А кращий за критерій товару В, В краще за С, але С кращий за А. Це неможливо, експерт зробив помилку при визначенні оцінок. Оцінкою узгодженності матриць попарних порівнянь є відносна узгодженість α.

$$OY = (\square_{\square\square\square} - \square)(\square - 1), \quad (2.3)$$

де $\lambda_{\square\square\square}$ - найбільше власне значення матриці

$$\lambda_{\square\square\square} = \quad \square^{\square}\square\square, \quad (2.4)$$

де e^T - одиничний транспонований вектор, *A* - матриця оцінок експерта, *W* - нормалізоване значення сум рядків матриці А.

Визначимо випадкову узгодженість за таблицею[1]

Порядок матриці	1	2	3	4	5	6	7	8	9	10
ВУ	0	0	0.382	0.946	1.22	1	1.468	1.4	1.35	1.46

Таблиця 2.3 - випадкові узгодженості

Розрахуємо відносну узгодженість:

$$\alpha = ОУ/ВУ \quad (2.5)$$

Якщо $\alpha > 0.1$, то оцінки нашого експерту надто неузгоджені і йому варто зробити переоцінку.

2.2.5. Розрахунок глобальних пріоритетів

$$W = W_a * W_k, \quad (2.6)$$

де W_k - вектор локальних пріоритетів матриці критеріїв, W_a - матриця векторів локальних пріоритетів матриць альтернатив.

Найкращою альтернативою буде альтернатива з найбільшим значенням W.

2.3 Архітектура програмного застосунку

Застосунок написаний на мові Java, для backend частини, з використанням популярного фреймворку Spring (а саме Spring Boot, MVC та Data) та JavaScript, для frontend.

При розробці застосунку використовувався паттерн MVC для зв’язку між різними частинами додатку.

Для отримання тестових даних було вирішено створити парсер, який збирає відкриту інформацію про характеристики смартфонів з відомого інтернет магазину. Реалізацію цього парсера було вирішено реалізувати на мові Python, через наявність зручних бібліотек для парсингу. Він не є частиною основного Java-застосунку, при імплементації на реальний інтернет-магазин даний парсер може бути видалений, адже дані для відображення будуть братися безпосередньо із раніше сформованої бази даних. Була розроблена UML-діаграма класів для візуалізації архітектури застосунку. Діаграма розміщена на додатку А.

Візуально архітектуру застосунку можна зобразити як на рисунку 2.2

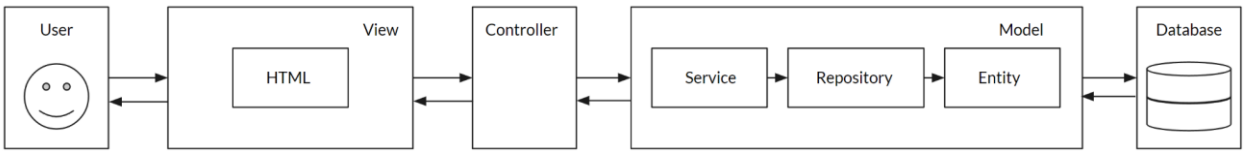


Рисунок 2.2 - загальна схема архітектури застосунку

2.3.1 Використання патерну проектування MVC

Застосунок побудований із використанням патерну MVC.

MVC[8] (Model-View-Controller) — це шаблон у розробці програмного забезпечення, який зазвичай використовується для реалізації інтерфейсів користувача, даних і логіки керування. Він підкреслює розмежування між бізнес-логікою програмного забезпечення та відображенням. Такий «розподіл завдань» забезпечує кращий розподіл праці та покращення технічного обслуговування.

Три частини шаблону проектування програмного забезпечення MVC можна описати наступним чином:

Модель: керує даними та бізнес-логікою.

Перегляд: керує макетом і відображенням.

Контролер: направляє команди до частин моделі та виду.

2.3.2 Модель, що реалізована в застосунку

Модель містить 3 класи, що описують бізнес-об’єкти застосунку: Characteristic, PMCharacteristics, PairwiseMatrix та Phone, які є класами-сутностями(entity) ORM-фреймворку Hibernate, який створює базу даних на основі атрибутів цих класів.

Розглянемо детальніше ці класи:

Characteristics містить дані про характеристики смартфонів доступні для фільтрації. Містить 2 атрибути: name та checked. name - назва характеристики(наприклад розмір екрану чи процесор). checked - булева змінна, яка містить інформацію про те чи обрав користувач дану характеристику до порівняння(true) чи ні(false).

PMCharacteristics має всі ті самі атрибути, що й клас Characteristics, містить характеристики смартфонів доступні для порівняння.

PairwiseMatrix зберігає матриці попарних порівнянь. Містить два атрибути: name та matrixAsString. name - ім’я матриці попарних порівнянь, за яким її можна ідентифікувати. matrixAsString - рядок, що зберігає матрицю.

Було створено методи перетворювачі матриці в рядок і навпаки. Як це працює:

Нехай є матриця

1 2 3

4 5 6

7 8 9

Між елементами кожного рядку є пробіл, який позначається метасимволом \s. В кінці кожного рядку є перехід на новий рядок, цей перехід позначається \n. Мій алгоритм перетворює матрицю вище в рядок 1s2s3n4s5s6n7s8s9, цей уніфікований запис матриць можна конвертувати назад до матриці.

Phone зберігає інформацію про товари, містить такі поля як name, capacity, ram, cpu, price, camera, company, display_size, selected, mergedName. Серед особливостей цих атрибутів, mergedName не є частиною entity і не зберігається в базі даних. він за запитом створюється для кожного телефону окремо і використовується для відображення тексту в api при відкритті сторінки телефону. selected позначає телефони, які відповідають критеріям користувача.

На початковому етапі розробки проекту Phone зберігає невелику кількість характеристик. При подальшій імплементації додатку до інтернет-магазину з’явиться можливість брати дані для характеристик безпосередньо з бази даних, що дозволить порівнювати більшу їх кількість.

Будь-які дані від бази даних до веб-сторінки проходять наступний шлях:

База даних - репозиторій - сервіс - контролер. Репозиторій та сервіс також є частинами моделі.

Додаток містить 4 репозиторії для кожної із сутностей. Вони є інтерфейсами і наслідуються від інтерфейсу CrudRepository, який є частиною Spring Data. Ці інтерфейси дозволяють виконувати базові операції роботи з базою даних без написання sql запитів.

Класи, в яких відбувається вся бізнес-логіка нашого застосунку - сервіси. В нас їх 6:

CharacteristicService, PMCharacteristicsService та PhoneService виконують базові операції(отримати, оновити) над відповідними їм сутностям.

PairwiseMatrixService містить методи для обробки матриць попарних порівнянь: конвертації з матриці в рядок і навпаки, конвертацію з рядкової матриці до цифрової(для виконання розрахунків), а також запити на отримання матриць попарних порівнянь.

SearchFilterService виконує фільтрування смартфонів за заданими фільтрами користувача.

RecommendationService реалізує метод аналізу ієрархій.

Також в моделі є два допоміжних класи: MathUtil - містить операції множення матриць та векторів. RecommendedPhone - dto(data transfer object), задача якого передати рекомендовані користувачу смартфони та їх ваги на відображення.

2.3.3 Клас Controller

Зв’язок між моделлю та користувацьким інтерфейсом забезпечує клас Controller. Endpoints цього класу:

GET-запити:

1) ‘/’ - відкриває головну сторінку;

2) ‘/phones’ - отримує через PhoneService список всіх смартфонів, та передає на веб-сторінку та відкриває сторінку зі списком смартфонів;

3) ‘/suggestme’ - відкриває сторінку рекомендаційного модулю;

4) ‘/error’ - сторінка, на яку йде перенаправлення за будь-якої помилки застосунку.

5) ‘/phone/{phone name}’ - відкриває сторінку обраного смартфону

POST-запити:

1) ‘/suggestme/select-characteristics’ - зберігає вибір характеристик смартфонів від користувача

2) ‘/suggestme/filter’ - застосовує фільтрування смартфонів за вибраними параметрами користувачем

3) ‘suggestme/make-recommendation’ - на основі заповнених користувачем матриць попарних порівнянь виконує розрахунки методом аналізу ієрархій та надає користувачу рейтинг смартфонів серед обраних при фільтруванні.

4) ‘/suggestme/add-pm-characteristic’ - додає власну користувацьку характеристику для порівняння смартфонів.
5) ‘/suggestme/select-characteristics-for-pm’ - створює матриці попарних порівнянь на основі обраних характеристик.

2.3.4 Користувацькій інтерфейс

Інтерфейс буде побудований на Java шаблонізаторі Thymeleaf(який використовує HTML), стилі будут написані з використанням CSS, анімації та AJAX запити будуть реалізовані з використанням мови JavaScript.

Наявні HTML-сторінки:

main - головна сторінка з навігацією.

phones - сторінка, на якій відображається список смартфонів.

phone - сторінка із інформацією про обраний смартфон.

suggestme - сторінка рекомендаційного модулю.

error - на цю сторінку йде переадресація за будь-яких помилок.

2.3.5 Валідація даних

В застосунку є 2 типи валідації:

1) Frontend валідація.

JavaScript перевіряє кожну зміну в матриці попарних порівнянь, якщо введене значення оцінки вище за 9 або менше за 0 - виводиться повідомлення про помилку, поле введення підфарбовується червоним, а кнопка рекомендації зникає до тих пір, доки значення не буде змінено на валідне. Також використовуючи JS були заблоковані для зміни діагональні елементи матриць попарних порівнянь, адже вони є константними і дорівнюють одиницям.

2) Entity валідація

Якщо атрибут класу-сутності не може приймати null значення, назначається аннотація @NotNull, яка створює обмеження на поле в базі даних і не дозволяє зберігання запису із не валідним(null) значенням.

2.3.6 Аутентифікація та авторизація користувачів у системі

В даному застосунку не передбачено авторизації та аутентифікації на даному етапі розвитку проекту. В нашому тестовому варіанті є лише один користувач, тому на даний момент створювати розділення ролей та акаунтів не є необхідним, проте є в перспективі на майбутнє.

2.4 База даних

Як система управління базами даних використаний PostgreSQL, зв’язок між базою даних та застосунком відбуватися з використанням Spring Data. Оскільки використовується ORM-фреймворк Hibernate, створення різних типів моделей бази даних(концептуальна, фізична, тощо) немає сенсу, адже всі таблиці будуть створюватися автоматично при запуску застосунку за правилами Hibernate. Зображення згенерованої бази даних на основі entity класів та полів всередині них показано на рисунку 2.3

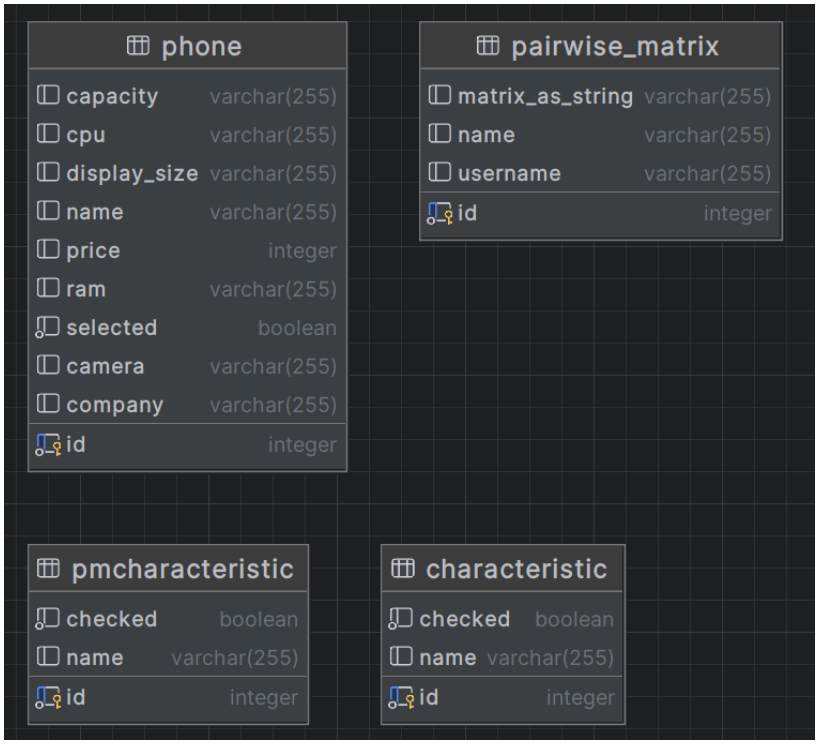


Рисунок 2.3 - діаграма сутностей бази даних

Як можна побачити з діаграми, всі наші класи є незалежними, не з’єднані один з одним.

2.5 Висновки до розділу

Отже, мною був проведений аналіз методів критеріального аналізу. Були досліджені велика кількість методів, декілька з яких описані у пункті 2.1. Нагадаємо, це метод аналітичних мереж, який нам не підійшов через те, що між елементами ієрархії нашої задачі не має залежностей, кожна характеристика смартфона незалежна від інших, кожен смартфон не залежить від інших. Це метод Best-Worst method, який ми відкинули через затратність ресурсів. І метод аналізу ієрархій, який є найоптимальнішим для нашої задачі.

Він не потребує великих ресурсів для розрахунків та дає точний результат навіть за невеликої кількості даних від користувача. Єдиним недоліком даного методу є велика кількість порівнянь, які повинен зробити користувач, хоча й це вирішено в нашій модифікації метода. Наш метод є адаптивним, тому користувач не повинен, якщо на бажає, заповнювати всі порівняння. всі незаповнені значення по-замовчуванню мають вагу 1 і просто не враховуються в розрахунках. Також було вирішено не робити перевірку узгодженості, адже це значно збільшує час на заповнення полів і час на розробку, адже потрібно було б створити додатковий алгоритм, який шукав би місце неузгодженості та підсвічував би його. Даний алгоритм є перспективною задачею для покращення нашего рекомендаційного сервісу.

Був проведений аналіз популярних технологій для створення веб додатків, було вирішено писати програму на мові Java із використанням фреймворку Spring, а також JavaScript для роботи із браузерними подіями та надсилання AJAX запитів. Відображення відбувається із використанням шаблонізатора Thymeleaf, стилі - CSS. Як система управління базами даних використовується PostgreSQL, а безпосередньо робота із таблицями виконується із використанням ORM фреймворку Hibernate. Власне через це було вирішено відмовитися від планування архітектури баз даних у вигляді моделей: концептуальна, фізична, тощо, адже Hibernate сам створює таблиці на основі класів-сутностей позначених

анотацією @Entity. Дані для тестування використовуються реальні. На мові Python написаний парсер, який збирає інформацію із відомого інтернет магазину та записує у csv файл, звідки дані вручну копіюються в базу даних. Даний спосіб є лише тимчасовим, адже при імплементації сервісу в реальний інтернет магазин буде отриманий доступ до бази даних із товарами, звідки можна буде безпосередньо брати інформацію.

В розділі була описана архітектура застосунку, а саме MVC патерн, на якому все й побудовано. Є опис що таке модель, які класи до неї входять і яка в них роль. Нагадаємо, до моделі відносяться всі класи, в яких(чи які) реалізується бізнес-логіка. В нашому проекті це класи-сутності, які описують бізнес об’єкти, і з яких будується база даних. Це інтерфейси-репозиторії, які відповідають певному бізнес-об’єкту і реалізують методи, які дозволяють виконувати запити до бази даних. І також це сервіси, в яких відбувається вся бізнес-логіка застосунку. Після моделі в нас йде відображення, в пункті 2.4.4 є список сторінок, які є в нашому застосунку. Більше інформації про їх наповнення буде в третьому розділі. І останнє це контролер, він отримує запити від відображення та делегує отримані дані до сервісів. І навпаки отримує дані із сервісів та відправляє їх на UI. Вище в пункті про контролер описані всі кінцеві точки нашого застосунку.

РОЗДІЛ 3. ОПИС ІНТЕРФЕЙСУ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Опис інтерфейсу користувача

В застосунку є 4 основних сторінки, три з яких інформативні і одна функціональна, на якій власне і міститься основна логіка, що реалізує вимоги викладені в першому розділі цієї роботи.

Сторінка main:

Головна сторінка, що розміщена на URI “/”. Ця сторінка містить навігацію до інших сторінок. В перспективі її можна зробити більш інформаційно насиченою, та навіть продублювати деяку логіку з інших сторінок. На сторінці знаходяться 3 посилання:

1) Лого з картинкою та назвою магазину в хедері. Посилання веде до головної сторінки і є зручним інструментом навігації з інших сторінок на головну. Це посилання розміщено на всіх сторінках в системі.
2) Блок Phones. Містить посилання на сторінку phones, на якій можна передивитися всі смартфони з бази даних.
3) Блок suggestme. Містить посилання на сторінку suggestme, яка є головним функціональним ядром нашого застосунку.

Зовнішній вигляд можна побачити на рисунку 3.1

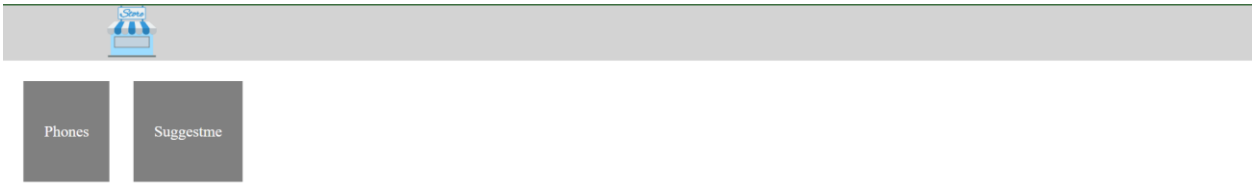


Рисунок 3.1 - сторінка main

Сторінка phones:

Сторінка, яка є сховищем інформації про всі доступні смартфони в системі. Розміщена на URI “/phones” В хедері міститься посилання на головну сторінку. Перед собою можна побачити список із усіх смартфонів. На кожному інформативному блоці є дані про продукт, а саме його назва, фото і найпопулярніші, на думку розробника застосунку, характеристики смартфону із відповідними їм значеннями. Фото смартфону є посиланням на сторінку даного товару із більш докладною інформацією про продукт.

Дана сторінка зображена на рисунку 3.2

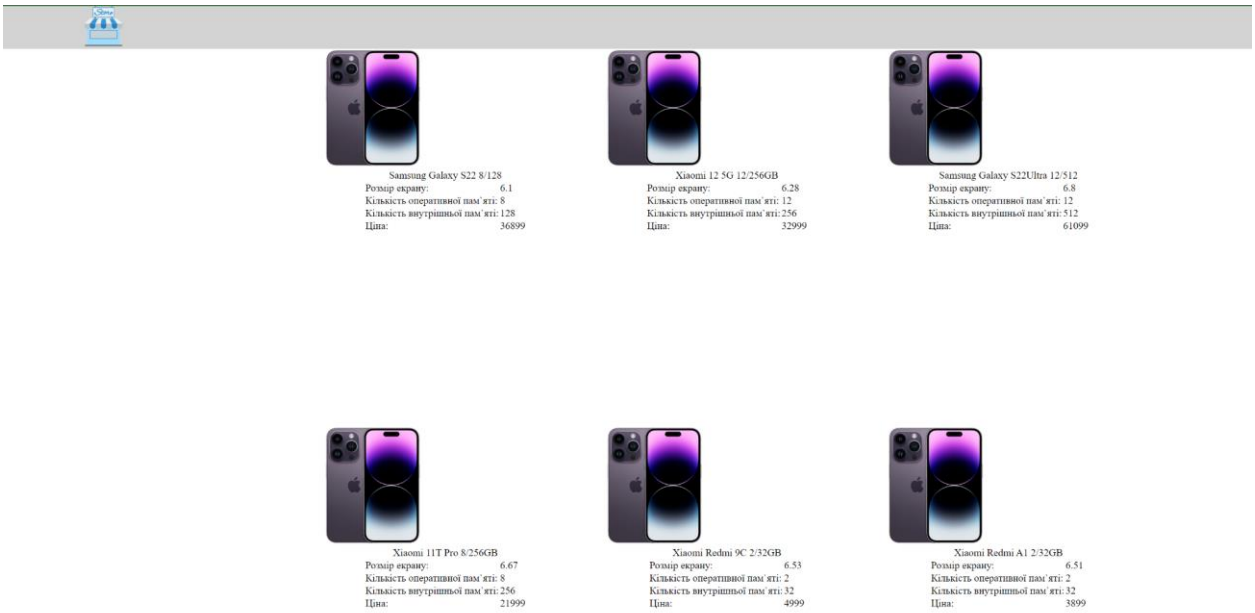


Рисунок 3.2 - сторінка phones

Сторінка phone:

Сторінка, що містить в собі докладну інформацію про вибраний смартфон. Як і всі інші містить посилання на головну сторінку. З особливостей цієї сторінки є архітектурне рішення при побудові URI. Розміщена на URI “/phones/{phone.mergedName}”, де {phone.mergedName} є власним значенням поля відповідного об’єкту смартфона. Використовувати поле id класу Phone, чи створювати ще один унікальний ідентифікатор смартфону не є інформативним. Тому було вирішено ініціалізувати змінну mergedName в класі Phone, анотувати її @Transient, що означає, що ця змінна є частиною класу, проте не є частиною сутності, а тому не потрапляє в базу даних. Ця змінна реалізує паттерн Singleton, і містить в собі назву смартфону, де всі слова розділені дефісами. Саме це значення можна побачити в URI.

Інтерфейс сторінки зображений на рисунку 3.3

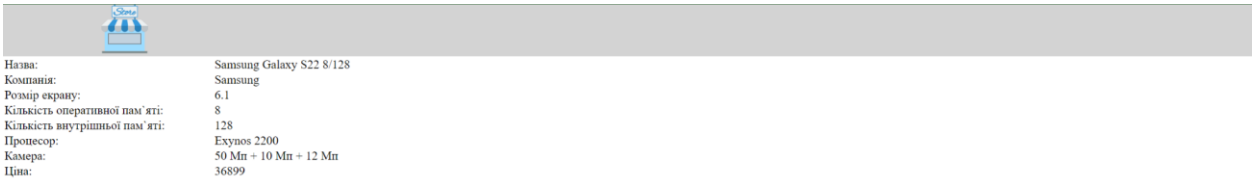


Рисунок 3.3 - сторінка phone

Сторінка suggestme:

Ця сторінка є ядром реалізації нашого рекомендаційного модулю, саме тут і відбувається більшість задач бізнес-вимог. Розміщений на URI “/suggestme”.

Можна виділити наступні блоки цієї сторінки:

1) Знак питання. Це кнопка, яка відкриває інструкцію користуванням рекомендаційним модулем suggestme. Після прочитання, для закриття, потрібно натиснути кнопку “Ознайомився”.

2) Вибір характеристик. Да даному блоці є список доступних для фільтрації. Користувач повинен обрати ті, в яких він розбирається та натиснути “Select”

3) Фільтрація. До кожної з обраних характеристик створені блоки багатоваріантного вибору. Користувач повинен обрати характеристики, обрати діапазон ціни та натиснути “Search”.

4) Вибрані смартфони. Справа від блоку фільтрації повинні з’явитися декілька смартфонів, які є відповідними по характеристикам до вибраних користувачем фільтрів.

В даному блоці є декілька ситуацій, на які варто звернути увагу.

Оскільки чим більше вибраних смартфонів, тим більш масивні таблиці матриць попарних порівнянь будуть створенім, тим більше порівнянь користувачу потрібно буде зробити. Тому було введено обмеження на не більше семи смартфонів, серед яких можна порівняння. Звідси в нас є наступні варіанти подій:

a) Кількість смартфонів від 2 до 7 включно. Цей варіант можна назвати happy path, оскільки він є успішним результатом фільтрування.

b) Кількість смартфонів більше 7. Смартфони не будуть відображені у блоці справа від блоку фільтрації, а нижче з’явиться напис “Знайдено забагато смартфонів, оберіть інші фільтри.”

c) Кількість смартфонів дорівнює 1. В цьому випадку застосовані користувачем фільтри однозначно визначили найбільш відповідний його потребам і запитам товар, посередині екрану буде виведений цей смартфон.

d) Кількість смартфонів дорівнює 0. За заданими фільтрами не знайдено жодного товару. Буде виведений текст “Не знайдено, застосуйте фільтри”. Також цей напис буде відображений при першому візиту сторінки, до застосування фільтрів.

5) Вибір характеристик для порівняння. Користувач повинен вибрати характеристики між якими буде порівнювати смартфони

6) Матриці попарних порівнянь. В цьому блоці створюються два типи матриць попарних порівнянь на основі отриманих раніше даних

a) Матриця критеріїв. Ця матриця попарних порівнянь розміру n, де n - кількість обраних характеристик в другому блоці “Вибір характеристик”.

b) Матриці альтернатив. Для кожного з критеріїв створюється матриця попарних порівнянь розміру m, де m - кількість смартфонів, які ми отримали після фільтрації.

Обмеження:

Матриці попарних порівнянь мають складатися лише з чисел 1-9. На сайті реалізована валідація, користувач не зможе розпочати розрахунки кращих альтернатив доки не виправить введене число.

Діагональні елементи завжди дорівнюють одиниці, тому вони були заблоковані на зміну, користувач не зможе виправити.

Має бути хоча б одне порівняння кожного елемента з іншим. Для точності розрахунків всі альтернативи мають бути зв’язані. Це можна наглядно показати на графі побудованому з матриці. Наприклад маємо наступну матрицю попарних порівнянь:

Критерій 1	Альтернатива 1	Альтернатива 2	Альтернатива 3
Альтернатива 1	1	1	3
Альтернатива 2	1	1	1
Альтернатива 3	1/3	1	1

Таблиця 3.1 - приклад матриці попарних порівнянь

У графовому вигляді отримаємо наступне:

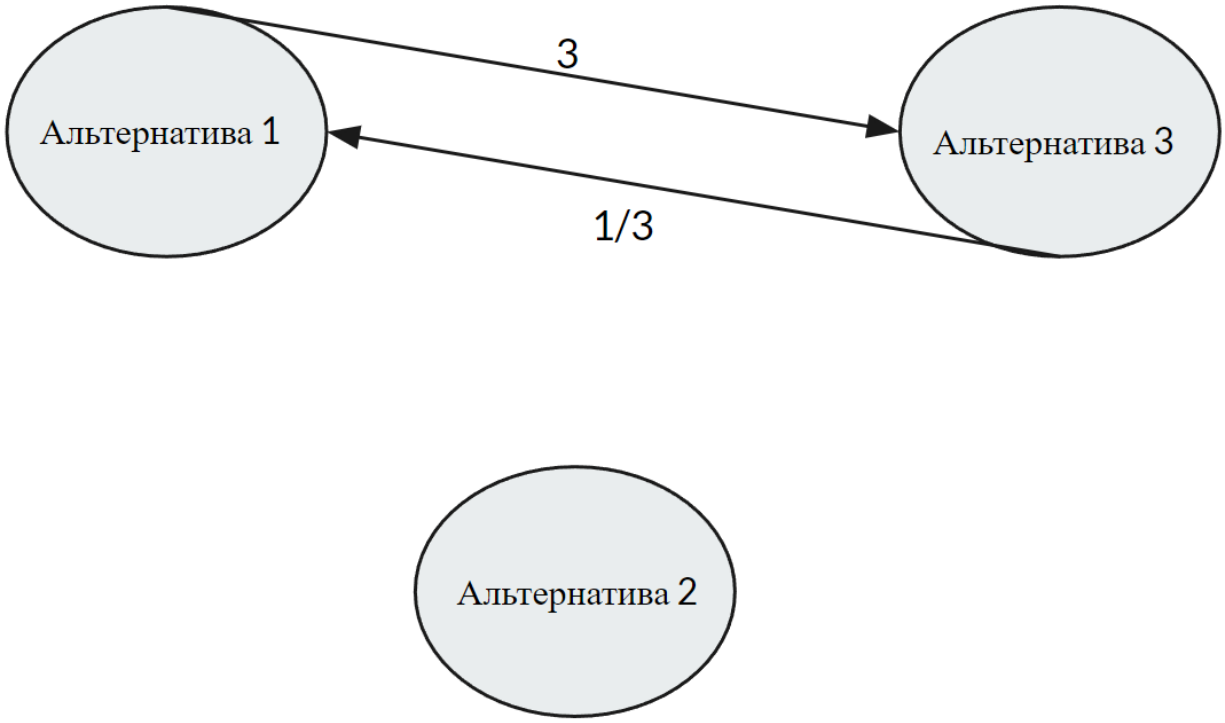


Рисунок 3.4 - графове представлення матриці попарних порівнянь

Видно, що через відсутність порівнянь будь-якого елементу із альтернативою 2 не має зв’язку і на графі. Через що будь-які твердження щодо альтернативи 2 не мають ваги.

Після заповнення матриць попарних порівнянь потрібно натиснути кнопку “Рекомендувати”.

7) Рекомендація. Даний блок складається з двох деталей: блоки з смартфонам та цифри над ними. Ці цифри - ваги рекомендації. Чим вище - тим більш відповідним до ваших запитів є смартфон. Ці значення відсортовані за спаданням, тому першим продуктом є найбажаніший, останній навпаки найменш. Оскільки ми пропонуємо користувачу саме рейтинг товарів, а не певний товар, то наша програма може називатися

системою підтримки прийняття рішення, і користувач може обрати іншу альтернативу з високим значенням ваг, не тільки найкращий.

Сторінка suggestme зображена на рисунках 3.5 та 3.6

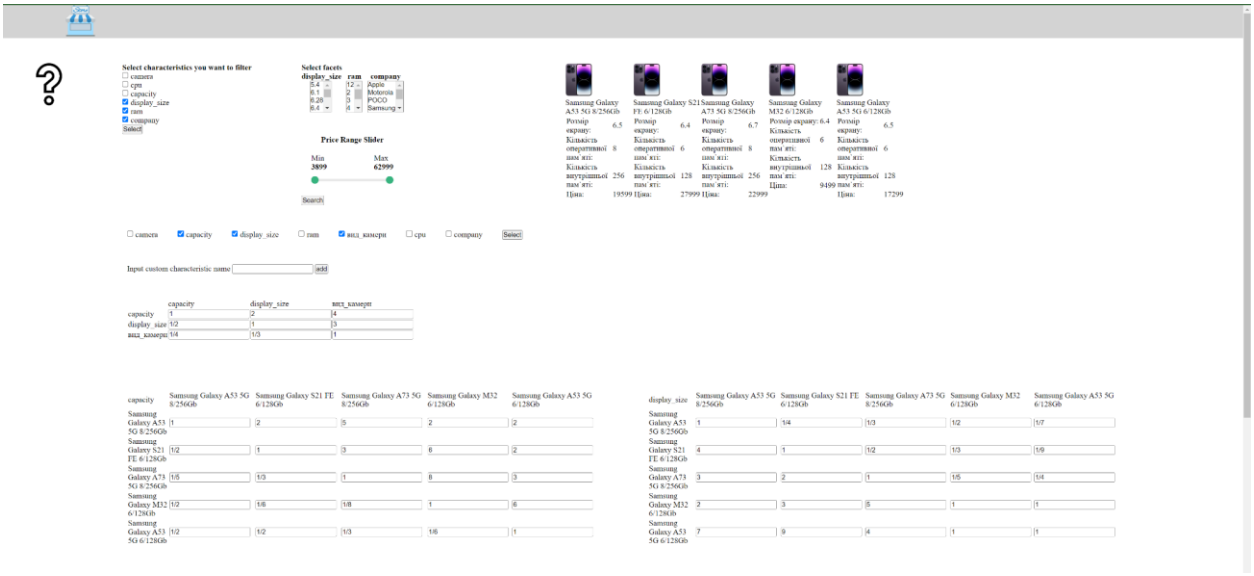


Рисунок 3.5 - сторінка suggestme

Рисунок 3.6 - продовження сторінки suggestme

Наглядно сторінки і переходи між ними можна зобразити на діаграмі(Рисунок 3.7). Як вище було написано, зі всіх сторінок можна прийти до main, на сторінці main два посилання на phones та suggestme, і з цих сторінок можно перейти на сторінку phone натиснувши на посилання у вигляді фото смартфону. На діаграми переходів зазвичай ще зазначаються події на стрілках, за яких відбувається переход, проте єдина подія у всіх варіантах, що змушує сторінку змінюватися - це безпосереднє натискання на посилання.

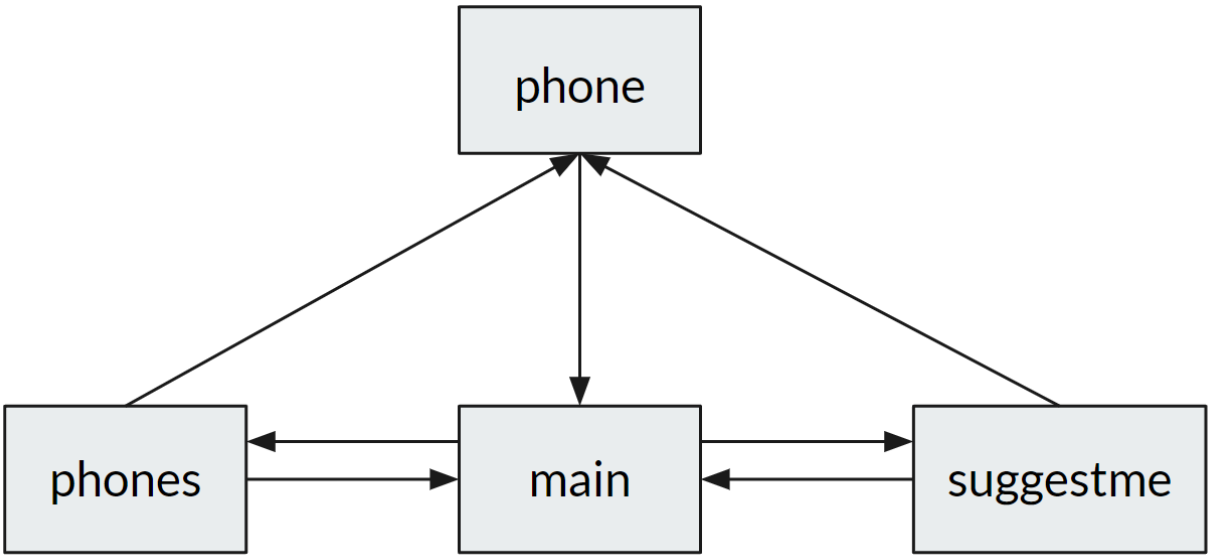


Рисунок 3.7 - діаграма переходів між сторінками веб-застосунку

3.2 Мануальне тестування

Як відомо, в кожної програми є два результати виконання: happy path та alternative path. Перший - те, як система повинна працювати в ідеалі. Користувач ввів всі дані правильно, система вірно відпрацювала та надала правильний результат. Другий - всі інші варіанти: користувач ввів неправильні дані, система стикнулася з проблемою та повідомила про помилку, інше.

3.2.1 Тестування доступності сторінок

1) Перейти на сторінку main

Варіанти подій:

1. Сторінка main відкрилася, на ній є хедер та два посилання - успіх
2. Сторінка error відкрилася, в вікні “network” інструментів користувача браузера можна побачити 400-405 http response - провал

2) Перейти на сторінку phones

Варіанти подій:

1. Сторінка phones відкрилася, на ній є список із смартфонами - успіх
2. Сторінка phones відкрилася, списку смартфонів, за наявності їх в базі даних, немає - провал
3. Сторінка error відкрилася, в вікні “network” інструментів користувача браузера можна побачити 400-405 http response - провал

3) Перейти на сторінку phone

Варіанти подій:

1. Сторінка phone відкрилася, на ній є інформація про вибраний смартфон - успіх
2. Сторінка phone відкрилася, на ній є інформація не про вибраний смартфон - провал
3. Сторінка error відкрилася, в вікні “network” інструментів користувача браузера можна побачити 400-405 http response - провал

4) Перейти на сторінку suggestme

Варіанти подій:

1. Сторінка suggestme відкрилася, на ній є знак питання та чекбокси для вибору характеристик смартфону - успіх
2. Сторінка suggestme відкрилася, на ній не має знаку питанні та\або чекбоксів з характеристиками - провал
3. Сторінка error відкрилася, в вікні “network” інструментів користувача браузера можна побачити 400-405 http response - провал

3.2.2 Тестування сторінки suggestme

1) Тестування інструкції

Шаги:

a) Натиснути знак питання
b) Впевнитися, що вікно із інструкцією з’явилося на екрані
c) Натиснути “Ознайомився”

Варіанти подій:

1. При натисканні знаку питання інструкція з’явилася, при натисканні “Ознайомився” зникла - успіх
2. При натисканні на знак питання інструкція не з’явилася - провал
3. При натисканні знаку питання інструкція з’явилася, при натисканні “Ознайомився” не зникла - провал

2) Тестування вибору характеристик

Шаги:

a) Впевнитися, що характеристики існують
b) Вибрати характеристики
c) Натиснути “Select”

Варіанти подій:

1. Характеристики вибрані, після натискання кнопки “Select” з’являється форма з полями багатоваріантного вибору значень для кожної з характеристик - успіх
2. Характеристики вибрані, після натискання кнопки “Select” відкривається сторінка error, в вікні “network” інструментів користувача браузера можна побачити 400-405 http response - провал
3. Характеристики вибрані, після натискання кнопки “Select” вибрані чекбокси характеристик скидаються, форма із полями багатоваріантного вибору значень для кожної з характеристик не з’являється - провал

3) Тестування вибору значень характеристики

Шаги:

a) Впевнитися, що форма з полями багатоваріантного вибору існує, а заголовки полів відповідають їх наповненню
b) Вибрати значення характеристик
c) Натиснути “Search”

Варіанти подій:

1. Значення характеристик вибрані, заголовки полів відповідають їх наповненню, після натискання кнопки “Search” з’являється список смартфонів та таблиці для матриць попарних порівнянь, ибо інформативні написи - успіх
2. Значення характеристик вибрані, заголовки полів не відповідають їх наповненню - провал
3. Значення характеристик вибрані, після натискання кнопки “Search” відкривається сторінка error, в вікні “network” інструментів користувача браузера можна побачити 400-405 http response - провал

4) Тестування вибору характеристик для попарних порівнянь

Шаги:

a) Впевнитися, що характеристики існують
b) Вибрати значення характеристик
c) Натиснути “Select”

Варіанти подій:

1. Характеристики вибрані, після натискання кнопки “Select” з’являються матриці попарних порівнянь - успіх
2. Характеристики не вибрані, після натискання кнопки “Select” не з’являються матриці попарних порівнянь і не з’являється помилка - успіх

3. Характеристик вибрані, після натискання кнопки “Select” відкривається сторінка error, в вікні “network” інструментів користувача браузера можна побачити 400-405 http response - провал

5) Тестування додавання власних характеристик користувача

Шаги:

a) Впевнитися, що поле для введення характеристик існує
b) Ввести значення
c) Натиснути “add”

Варіанти подій:

1. Назва характеристики введена, після натискання кнопки “add” введена назва з’являється у списку характеристик для порівняння - успіх
2. Назва характеристики введена, після натискання кнопки “add” відкривається сторінка error, в вікні “network” інструментів користувача браузера можна побачити 400-405 http response - провал

6) Тестування матриць попарних порівнянь

Шаги:

a) Впевнитися, що таблиці із матрицями попарних порівнянь критеріїв та альтернатив існують
b) Заповнити матриці попарних порівнянь
c) Натиснути “Recommend”

Варіанти подій:

1. Матриця попарних порівнянь критеріїв існує, її порядок дорівнює кількості вибраних характеристик. Матриці попарних порівнянь

альтернатив існують, їх кількість дорівнює кількості вибраних характеристик, а порядок кожної дорівнює кількість знайдених смартфонів, при натисканні “Recommend” з’являється список рекомендованих смартфонів - успіх.

2. Порядок матриці попарних порівнянь критеріїв не співпадає з кількістю вибраних характеристик та\або порядок будь-якої з матриць попарних порівнянь альтернатив не співпадає з кількістю вибраних смартфонів - провал
3. Кількість матриць попарних порівнянь альтернатив не співпадає із кількістю вибраних характеристик - провал
4. Матриці заповнені, при натисканні “Recommend” не з’являється список рекомендованих смартфонів - провал
5. Матриці заповнені, при натисканні “Recommend” відкривається сторінка error, в вікні “network” інструментів користувача браузера можна побачити 400-405 http response - провал

7) Тестування рекомендованих смартфонів

Шаги:

а) Впевнитися, що рекомендовані смартфоні існують на сторінці

Варіанти подій:

1. Рекомендовані смартфони існують на сторінці, значеннях їх ваг відсортовані від більшого до меншого - успіх
2. Рекомендовані смартфони не існують на сторінці - провал
3. Рекомендовані смартфони не відсортовані за вагами - провал

3.2.3 Тестування рекомендації

Алгоритм побудований таким чином, що непередбачуваних ситуацій виникати не може. Всі винятки оброблені. Єдине, що може вплинути на можливість виконання алгоритму - фільтрація. Вище були описані варіанти подій, що відбувається при різних кількостях знайдених смартфонів. Розглянемо їх на прикладах більш детально.

1) Смартфонів 2-7
    a) Вибрані характеристики
    b) В фільтрах обрано параметри фільтрації
    c) Після фільтрації з’являються 2-7 варіантів смартфонів та матриці попарних порівнянь
    d) Заповнюємо матриці попарних порівнянь
    e) Натискаємо “Recommend”
    f) Впенюємося, що смартфоном з найбільшою вагою є той смартфон, в матрицях попарних порівнянь якого ми виставили найбільші значення, відносно інших.
2) Смартфонів 0
    a) Вибрані характеристики
    b) В фільтрах обрано параметри фільтрації
    c) За заданими параметрами не знайдено смартфонів
    d) З’являється напис “Не знайдено, застосуйте фільтри”, матриць попарних порівнянь на сторінці немає
3) Смартфон 1
    a) Вибрані характеристики
    b) В фільтрах обрано параметри фільтрації

c) За заданими параметрами фільтрації знайдений один смартфон, він представление посередині екрану, матриць попарних порівнянь немає

4) Смартфонів більше семи

a) Вибрані характеристики
b) В фільтрах обрано параметри фільтрації
c) За заданими параметрами фільтрації знайдено велика кількість смартфонів, на екрані з’являється напис “Знайдено забагато смартфонів, оберіть інші фільтри.”

5) Валідація даних в матриці попарних порівнянь

a) Вибрані характеристики
b) В фільтрах обрано параметри фільтрації
c) Після фільтрації з’являються 2-7 варіантів смартфонів та матриці попарних порівнянь
d) Вводимо в поле випадкової матриці попарних порівнянь число 10 або число 0
e) З’являється push-повідомлення із попередженням про помилку, комірка в яку введено невалідне значення підкрашується червоним, а кнопка “Recommend” зникає
f) Виправляємо значення цього поля на валідне 1-9 значення
g) Комірка окрашується в білий, кнопка “Recommend” з’являється

3.3 ВИСНОВКИ

Отже, в третьому розділі була задокументована інформація про застосунок, який ми розробляли для вирішення задачі поставленої у темі цієї роботи. Були описані інтерфейс користувача, а також тести за якими перевіряється працездатність системи. В першому пункті докладно розкладені сторінки даного застосунку, описані елементи цих сторінок, переходи між ними. Для останньої задачі була розроблена діаграма переходів між сторінками, яка дозволяє швидко оцінити ієрархію сторінок в системі.

В пункті про тестування до найдрібніших деталей розписано види тестування, а саме: тестування доступності сторінок, тестування до відповідності вимогам найбільш важливої та складної сторінки в системі suggestme, а також працездатність рекомендаційного модулю, головною ціллю написання якого власне і є дана робота.

Під час написання атестаційної роботи були досліджені численні джерела інформації такі як: література наближена до предметної області теми роботи, статті на веб-сайтах та блоги, документації технологій для розробки, інше.

Були досліджені алгоритми критеріального вибору і обраний найкращий - метод аналізу ієрархій. Він був модифікований відповідно до обмежень нашої системи, адже оригінальний алгоритм є часозатратним, що не є прийнятним для користувача системи.

Наступною задачею було визначитися з технологічним стеком. Пріоритет був наданий мові Java з використанням фреймворку Spring, тому що він є популярним і надійним інструментом розробки веб-застосунків. Проте не все зручно робити саме на Java, тестові дані для нашої системи зібрані парсингом з відомого інтернет-магазину, найзручніше створювати парсер на мові Python. Дана програма не є частиною нашого веб-застосунку, і використовується

незалежно. Отримані дані копіюються в базу даних вручну. При імплементації на реальний інтернет-магазин з доступом в базу даних, потреби в парсері та ручному копіюванню даних не буде. Також для наших вимог були написани деяка кількість скриптів на JavaScript. Цю мову ми використовуємо для валідації даних безпосередньо на веб-сторінці, а також для застосування технології AJAX, що дозволило нам збирати заповнені дані зі сторінки у файл формату JSON та надсилати до серверу.

Щодо допоміжних технологій, окрім мов програмування, були використані Java-шаблонізатор Thymeleaf, який побудований на HTML із можливістю отримувати дані з серверу та динамічно виводити їх, для відображення даних і CSS для накладання стилів на нашу розмітку.

Важливою частиною роботи є створення бази даних. Оскільки для вирішення поставлених задач велика та складна база даних не потрібно, було вирішено будувати її автоматично, використовуючи ORM фреймворк Hibernate. Весь цей комплекс роботи: дослідження, планування та реалізація, дозволив створити застосунок, який відповідає всім вимогам та може практично застосовуватися для полегшення вибору товару користувачем.

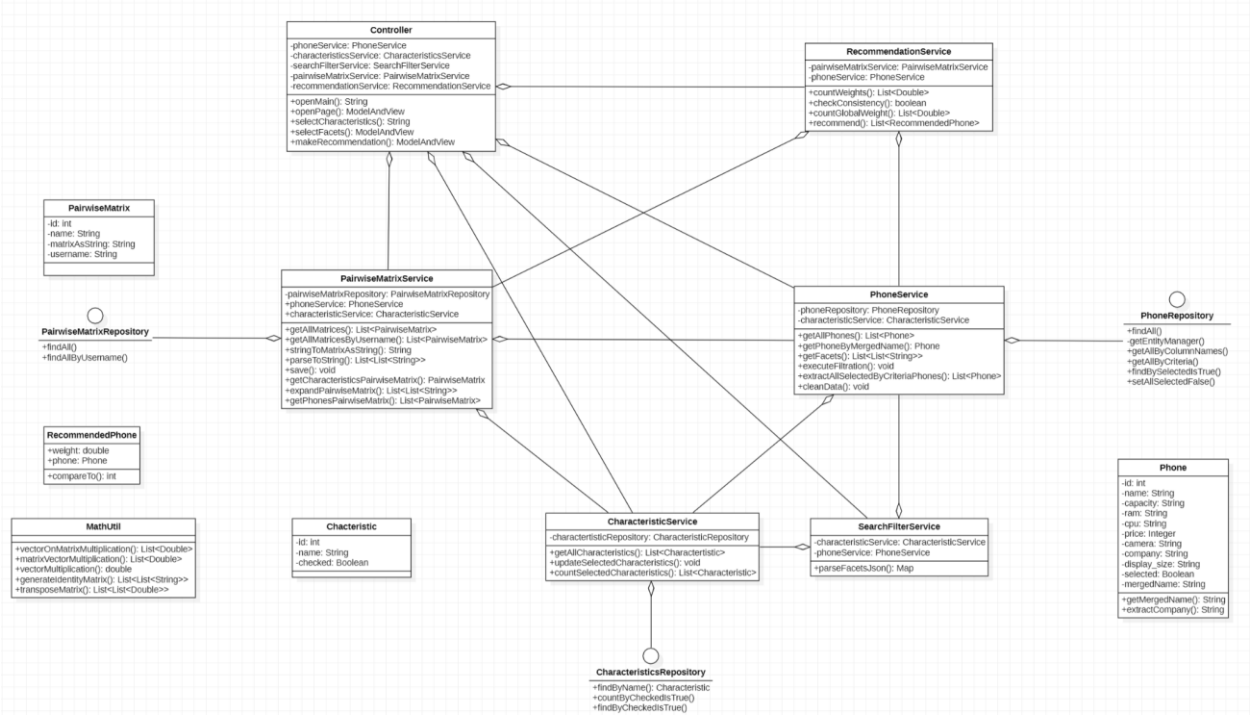
Список використаних джерел

1. ‘Multicriteria Decision Making’ Thomas L. Saaty
2. “Застосування методу аналізу ієрархій для оцінки маркетингової активності торговельних підприємств” - Євстрат Д.І., Кушнерук Ю.І.
3. ‘The Analytic Hierarchy Process – What It Is and How It Is Used’ Rozann Whitaker
4. ‘ANALYTIC HIERARCHY PROCESS (AHP) TUTORIAL’ Kardi Teknomo
5. Дослідження методу аналізу ієрархій для задач з великою кількістю альтернатив - Якимчук С.О., Франчук О.В.
6. Jafar Rezaei ‘BWM: Best Worst Method’ // Delft University of Technology
7. https://developer.mozilla.org/en-US/docs/Glossary/MVC
8. https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%A1%D0%BC%D0%B0%D1%80%D1%82%D1%84%D0%BE%D0%BD%D1%8B_(%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D1%80%D1%8B%D0%BD%D0%BE%D0%BA)
9. https://www.statista.com/statistics/271491/worldwide-shipments-of-smartphones-since-2009/
10. https://www.researchgate.net/figure/Schematic-view-of-DSS-components-2_fig2_313772963
11. https://mobiforge.com/research-analysis/2011-handset-and-smartphone-sales-statistics-worldwide-big-picture
12. https://gagadget.com/en/226668-cool-things-on-amazon/

13.https://docs.spring.io/spring-framework/reference/

14.https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html

Додаток А


Controller
-phoneService: PhoneService
-characteristicsService: CharacteristicsService
-searchFilterService: SearchFilterService
-pairwiseMatrixService: PairwiseMatrixService
-recommendationService: RecommendationService
+openMain(): String
+openPage(): ModelAndView
+selectCharacteristics(): String
+selectFacets(): ModelAndView
+makeRecommendation(): ModelAndView
RecommendationService
-pairwiseMatrixService: PairwiseMatrixService
-phoneService: PhoneService
+countWeights(): List<Double>
+checkConsistency(): boolean
+countGlobalWeight(): List<Double>
+recommend(): List<RecommendedPhone>
PairwiseMatrix
-id: int
-name: String
-matrixAsString: String
-username: String
PairwiseMatrixService
-pairwiseMatrixRepository: PairwiseMatrixRepository
+phoneService: PhoneService
+characteristicService: CharacteristicService
+getAllMatrices(): List<PairwiseMatrix>
+getAllMatricesByUsername(): List<PairwiseMatrix>
+stringToMatrixAsString(): String
+parseToString(): List<List<String>>
+save(): void
+getCharacteristicsPairwiseMatrix(): PairwiseMatrix
+expandPairwiseMatrix(): List<List<String>>
+getPhonesPairwiseMatrix(): List<PairwiseMatrix>
PhoneService
-phoneRepository: PhoneRepository
-characteristicService: CharacteristicService
+getAllPhones(): List<Phone>
+getPhoneByMergedName(): Phone
+getFacets(): List<List<String>>
+executeFiltration(): void
+extractAllSelectedByCriteriaPhones(): List<Phone>
+cleanData(): void
PhoneRepository
+findAll()
-getEntityManager()
+getAllByColumnNames()
+getAllByCriteria()
+findBySelectedIsTrue()
+setAllSelectedFalse()
PairwiseMatrixRepository
+findAll()
+findAllByUsername()
RecommendedPhone
+weight: double
+phone: Phone
+compareTo(): int
Phone
-id: int
-name: String
-capacity: String
-ram: String
-cpu: String
-price: Integer
-camera: String
-company: String
-display_size: String
-selected: Boolean
-mergedName: String
+getMergedName(): String
+extractCompany(): String
MathUtil
+vectorOnMatrixMultiplication(): List<Double>
+matrixVectorMultiplication(): List<Double>
+vectorMultiplication(): double
+generateIdentityMatrix(): List<List<String>>
+transposeMatrix(): List<List<Double>>
Chacteristic
-id: int
-name: String
-checked: Boolean
CharacteristicService
-charactertisticRepository: CharacteristicRepository
+getAllCharacteristics(): List<Charactertistic>
+updateSelectedCharacteristics(): void
+countSelectedCharacteristics(): List<Characteristic>
SearchFilterService
-characteristicService: CharacteristicService
-phoneService: PhoneService
+parseFacetsJson(): Map
CharacteristicsRepository
+findByName(): Characteristic
+countByCheckedIsTrue()
+findByCheckedIsTrue()