КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття освітнього рівня магістра
за спеціальністю 121 Інженерія програмного забезпечення
на тему:

Розробка пошукового та рекомендаційного веб-додатку для приготування кави

Виконала студентка 2-го курсу
Таїсія ВЕЛИЧКО

(підпис)

Науковий керівник:
доцент, кандидат фізико-математичних наук
Оксана ШКІЛЬНЯК

(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студентка

(підпис)

Роботу розглянуто й допущено до захисту
На засіданні кафедри інтелектуальних програмних систем
«___» _____ 2023р.,
протокол №
Завідувач кафедри
Олександр ПРОВОТАР

(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 50 сторінок, 4 рисунки, 17 використаних джерел.

Ключові слова: ВЕБ-ДОДАТОК, МІКРОСЕРВІСИ, КАВА, ПОШУК, РЕКОМЕНДАЦІЇ, JAVA, PYTHON, SPRING BOOT, ANGULAR, POSTRESQL, ELASTICSEARCH.

Об’єктом роботи є система для пошуку та отримання рекомендацій кави від українських виробників на основі мікросервісної архітектури з використанням баз даних. Для розробки використовувався рушій для повнотекстового пошуку ElasticSearch, СУБД PostgreSQL для збереження даних користувачів, бібліотека sklearn для формування рекомендацій, фреймворки Spring Boot та Django для побудови мікросервісів для бекенду та Angular для розробки фронтенду.

Метою кваліфікаційної роботи є створення веб-додатку для пошуку та отримання рекомендацій кави за попередніми вподобаннями. Користувачам буде надаватися можливість шукати, обирати улюблену каву та створювати власні рецепти.

Результат кваліфікаційної роботи: розглянуто та проаналізовано подібні веб-додатки; проаналізовано алгоритми та методи для формування рекомендацій та обрано affinity propagation для застосування; побудовано моделі баз даних, щоб зберігати інформацію про каву, рецепти та інформацію про користувачів; розроблено мікросервісну архітектуру з спроєктованим API, з реалізованим мікросервісом пошуку та рекомендацій, мікросервісом користувацького контенту, мікросервісом періодичного завдання для оновлення інформації про каву; розроблено фронтенд для взаємодії користувачем з системою.

Сфера застосування: кава є одним з товарів, що тільки починає вивчатися та надходити до України в більших масштабах. Цей веб-застосунок призначений для того, щоб люди могли легко шукати потрібну каву,

знаходити рецепти приготування та отримувати рекомендації за власними вподобаннями. Також це може бути корисним для кавової індустрії. Виробники кави будуть отримувати впізнаваність серед населення і зможуть охоплювати ширшу аудиторію.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

СУБД	– система управління базами даних
API	– Application Programming Interface
JSON	– JavaScript Object Notation
REST	– Representational State Transfer

ВСТУП

Пандемії COVID-19 та початок повномасштабного вторгнення мали серйозний вплив на бізнес у всьому світі, зокрема і в Україні. Багато малих бізнесів зазнали труднощів, тому їх розвиток і підтримка є важливими. Кавова індустрія у світі та, зокрема, і в Україні відіграє важливу роль, оскільки кава є другим найбільш споживаним напоєм у світі. З розвитком технологій люди частіше замовляють товари в інтернеті з доставкою до найближчої до них точки. Це є також актуальним після пандемії, оскільки багато людей перейшли на дистанційний режим роботи та звикли замовляти усі необхідні товари.

Актуальність роботи та її підстави для виконання. Обсмажчики та платформи з продажу кави в Україні зазвичай продають лише свій продукт та наразі не існує додатку, який би агрегував каву від різних користувачів та надавав йому можливість знайти каву схожу до його попередніх вподобань. Це знижує впізнаваність окремих брендів у випадку, якщо людина до цього не здійснювала у них замовлень. Схожі застосунки, що агрегують каву від різних виробників у межах однієї країни існують у деяких країнах Європи та є успішними.

Мета й завдання роботи. Метою роботи є створення веб-додатку для пошуку та отримання рекомендацій кави за попередніми вподобаннями. Користувачам буде надаватися можливість шукати, обирати улюблену каву. Для побудови веб-додатку було розв’язано такі завдання:

- огляд схожих веб-додатків для пошуку та отримання рекомендацій для кави;
- аналіз методів та алгоритмів для формування рекомендацій;
- побудова моделі бази даних;
- проєктування API;
- аналіз та розробка мікросервісної архітектури;
- розробка мікросервісу для взаємодії з користувачем;

– тестування додатку.

Об’єкт, методи й засоби дослідження або розроблення. Об’єктом роботи є засоби розробки рекомендаційної системи на основі мікросервісної архітектури з використанням баз даних. Для розробки використовувався рушій для повнотекстового пошуку ElasticSearch, СУБД PostgreSQL для збереження даних користувачів, бібліотека sklearn для формування рекомендацій, фреймворки Spring Boot та Django для побудови бекенд мікросервісів та Angular для розробки фронтенду.

Сфера застосування. Кава є другим найбільш вживаним напоєм у світі після води. Цей продукт ще тільки починає вивчатися та надходити до України в більших масштабах. За допомогою створеного веб-застосунку, люди можуть легко шукати потрібну каву, знаходити рецепти приготування та отримувати рекомендації за власними вподобаннями. Також це може бути корисним для розвитку кавової індустрії. Виробники кави будуть отримувати впізнаваність серед населення України і зможуть охоплювати ширшу аудиторію. Таким чином це може мати позитивний вплив на український ринок кави та економіку України.

РОЗДІЛ 1. ТЕОРИТИЧНІ ВІДОМОСТІ

1.1 Споживання кави в світі та Україні

Споживання кави є дуже поширеним у світі і має велике значення як соціальний, культурний і економічний феномен (рис 1.1). Є найбільш споживаним напоєм у світі після води, щороку люди по всьому світу споживають близько 400 мільярдів чашок кави або близько 80 мільйонів 60-кг мішків [1] (рис. 1.2). Для Україні релевантні дані відсутні, проте до початку повномасштабного вторгнення, тобто станом на 2021 рік споживання кави становило 3,97% від світового, тобто 6,57 мільйонів 60-кг мішків.

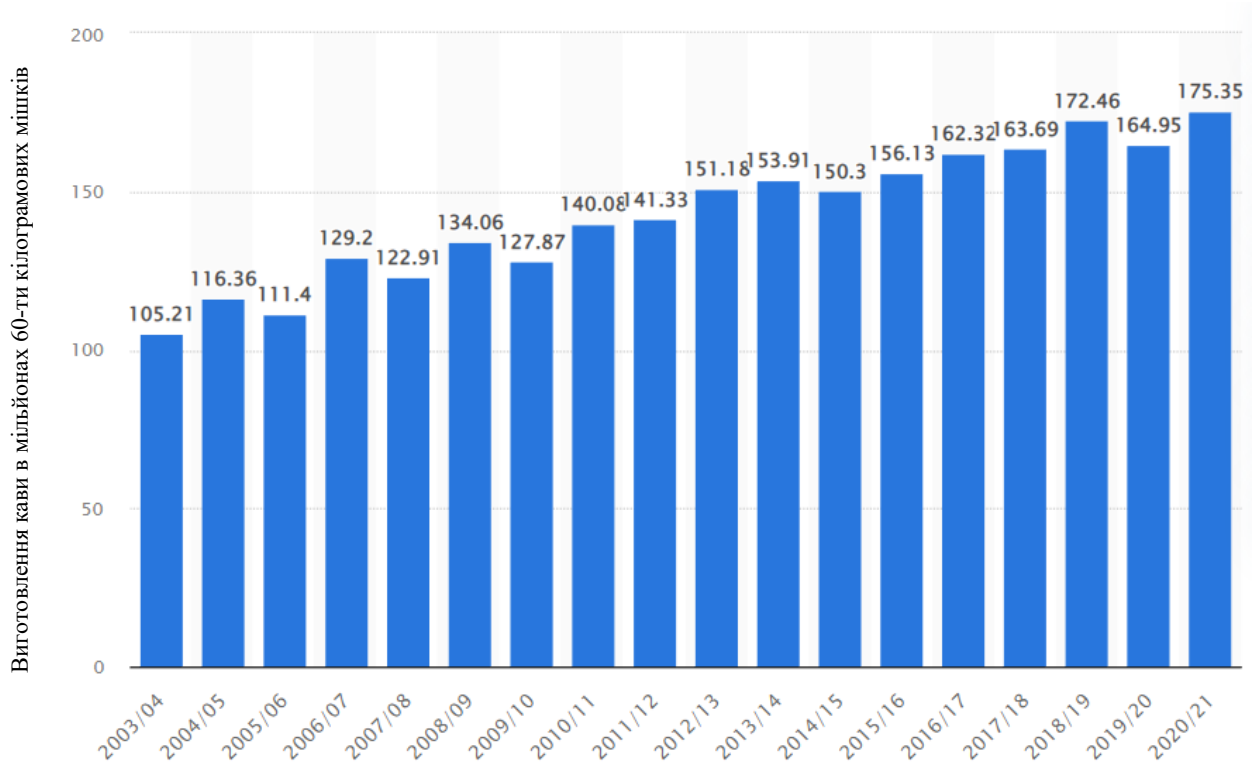

Рисунок 1.1 – Зріст світового об’єму виготовлення кави. [2]

Отже кавова індустрія відіграє значну роль у світовій економіці та економіці України. Частина обсмажчиків кави в Україні експортує продукцію до Європи, що гарантує впізнаваність українських виробників на ринку. Ця індустрія також має позитивний вплив на розвиток туристичної галузі. Хоча

наразі через війну це не є релевантним, проте після її закінчення це може бути важливим.

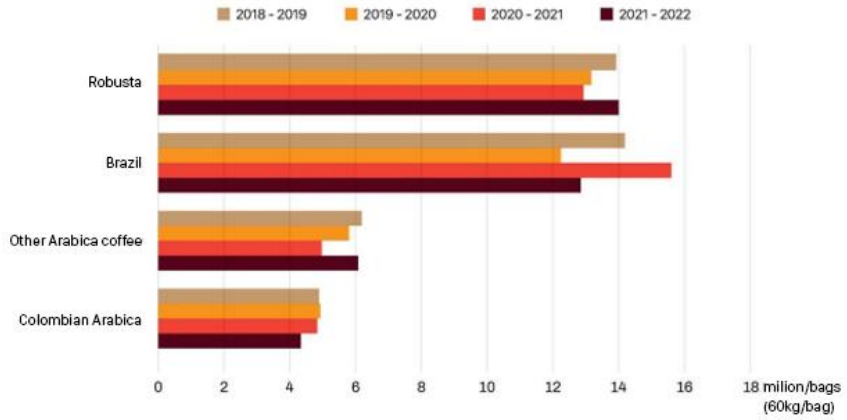

Рисунок 1.2 – Зріст світового об’єму споживання кави. [3]

1.2 Розвиток кавового бізнесу в Україні

Кавовий бізнес в Україні переживає значний розвиток та зростання в останні роки. Україна також відома своєю кавовою культурою та зростанням споживчої активності. Таким чином цей набір факторів створює сприятливі умови для розвитку кавової індустрії. Тенденції світового ринку кави також мають вплив на індустрію кави в Україні, відбувається стабільний зріст попиту. [4]

Не зважаючи на кризу через несприятливі погодні умови для врожаю та зростання ціни на зелене зерно під час пандемії, кавова культура в Україні доволі висока та випереджає чимало країн Європи. В останні роки в Україні споживання кави класів преміум та спешелті стрімко зростає. Оскільки Україна розміщена поза кавовим поясом, то це імпортований продукт. Проте в Україні каву завозять, обсмажують та деяка частина також іде на експорт.

Пандемія COVID-19 та початок повномасштабного російського вторгнення негативно вплинули на середній та малий бізнес в Україні. [5] Проте за стабільних умов у великих містах люди намагаються відновити нормальний перебіг життя, до якого часто входить відвідування закладів

громадського харчування, зокрема і кав’ярень. Крім того, під час пандемії багато працівників перейшли на дистанційний режим роботи та, за неможливості відвідувати заклади, почали більше готувати каву вдома.

Ринок кавової індустрії зростатиме та насичуватиметься після припинення повномасштабної війни.

1.3 Мови програмування для імплементації логіки веб-додатку

Для розробки різних мікросервісів було обрано різні мови виходячи з потреб та наявних бібліотек. Основною мовою для розробки бекенд мікросервісів було обрано Java.

Для імплементації рекомендаційного мікросервісу було обрано мову розробки Python, оскільки для формування рекомендацій найбільш зручним було використовувати бібліотеку, яка наявна для цієї мови.

1.4 Система управління базами даних

Для збереження різних типів даних для роботи застосунку було використано два інструменти. ElasticSearch було використано для збереження даних про каву, а всі інші дані, які стосуються користувачів зберігаються у PostgreSQL.

Оскільки для збережених сутностей кави було необхідно забезпечити пошук по визначеним полям та врахувати можливість помилок у написаних користувачами запитах, то було обрано ElasticSearch. Це програмне забезпечення є наразі найбільш популярним рушієм для повнотекстового пошуку та розподіленим сховищем документів. ElasticSearch використовує потужну індексацію та алгоритми ранжування для швидкого пошуку та отримання найбільш релевантних результатів. Для багатьох мов програмування існують бібліотеки, що працюють з ElasticSearch API, що полегшує його інтеграцію в проєкті. Усі дані за їх типом зберігаються до

індексів, що є аналогами таблиць да баз даних для реляційних СУБД. Дані зберігаються у JSON представленні.

Для швидкого пошуку використовується структура даних, що називається інвертованим індексом. Перед індексуванням усі текстові поля аналізуються, розбиваються на токени, виконуються такі операції для покращення пошуку, як нормалізація, стемінг, видалення стоп-слів та інші. [6] Серед інструментів також є і фільтрація чи агрегація даних, отже міститься усе необхідне для створення веб-додатку.

Оскільки для даних, що пов’язані з користувачем не потрібні інструменти для швидкого повнотекстового пошуку, то в якості СУБД було обрано PostgreSQL. Це об’єктно-реляційна система управління базами даних, що заснована на POSTGRES та підтримує стандарт SQL. [7] Обрана СУБД має наступні переваги:

- має власну безкоштовну ліцензію, оскільки Postgres було створено як проєкт з відкритим кодом, тому додаткове ліцензіювання не потрібне;
- дотримується стандарту SQL та використовує стандартні запити та команди, що дозволяє легко переносити застосунки між різними СУБД та спрощує міграцію даних;
- PostgreSQL має набір корисних інструментів, дозволяє використання наслідування таблиць, підтримку таких напівструктурованих типів даних як JSON та XML, географічних даних;
- є надійною та стабільною СУБД, має механізми відновлення після відмови, які відновлюють дані після непередбачуваних ситуацій та збоїв системи;
- PostgreSQL має розширення для специфічних потреб, які можна при розширенні функціоналу застосунку за необхідності додати для полегшення роботи з даними;
- сильною стороною є ефективна обробка великих обсягів даних та високе навантаження, підтримує паралельну обробку запитів, реплікацію та горизонтальне масштабування.

1.5 Spring Boot

Більшість бекенд мікросервісів написані мовою Java. Таким чином в якості веб-фреймворка для розробки відповідних мікросервісів було обрано Spring Boot. Цей інструмент надає можливість пришвидшити розробку, не витрачаючи зайвий час на конфігурацію. Фреймворк включає вбудовані компоненти для керування залежностями. Є можливість під'єднувати необхідні пакети, для використання потрібних технологій. Фреймворк спрощує інтеграцію із Spring та надає доступ до його інструментів. Завдяки вбудованому контейнеру сервлетів не потрібно витрачати зусилля на налаштування та розгортання застосунку.

1.6 Django

Оскільки для розробки рекомендаційного мікросервісу було обрано мову Python, то постала необхідність обрати веб-фреймворк для нього. Найбільш розповсюдженими інструментами для створення веб-застосунків мовою Python є Flask та Django. Обидва фреймворки є потужними інструментами для розробки веб-застосунків, обидва розвиваються та мають достатньо велику спільноту прихильників, тому вибір було здійснено на основі потреб проєкту.

Таким чином було обрано Django, оскільки Flask є легковаговим фреймворком та не має такої кількості вбудованих інструментів. Наприклад, Django має вбудовані компоненти для підтримки кешування, налаштування для роботи з базами даних та масштабування серверів. Однією з переваг є те, що готові компоненти дозволяють швидше створити функціональний веб-застосунок з меншими витратами зусиль та часу. Усі компоненти працюють разом та підтримують стандартизовану структуру проєкту. Крім того, передбачено, що проєкт міститиме складну логіку та його функціонал надалі може розширюватись.

1.7 Angular

Для розробки фронтенду було обрано фреймворк Angular, оскільки це потужний фреймворк для розробки веб-додатків різного рівня складності. Обраний фреймворк має наступні переваги для використання:

– підтримка модульної компонентної архітектури, що дозволяє легко розширювати фронтенд-мікросервіс за умови появи нових функцій, додаючи невеликі самодостатні компоненти;
– підтримка використання сторонніх бібліотек та модулів, що дозволяє розширити його можливості потрібними інструментами;
– має вбудовано систему маршрутизації, валідації даних та керування станом додатку тощо.

1.8 Affinity propagation

Для формування рекомендацій було використано частину пакету scikit-learn, а саме клас AffinityPropagation. [8] Цей пакет є потужним інструментом машинного навчання. Відповідний клас, що реалізує алгоритм кластеризації з назвою "Affinity Propagation" [9] призначений для знаходження "епіцентрів" або "представників" в масиві даних та призначення кожному елементу свого "екземпляра". Таким чином дані про каву розбиваються на кластери та можна швидко дістати подібні кави до наданої кави або до списку таких за їх належністю до певних кластерів кави. Обраний інструмент надає реалізацію алгоритму та додаткові функції:

– підтримка різних метрик для вимірювання схожості між елементами, що дало змогу імплементувати найбільш підходящу метрику для існуючого набору даних кави;
– налаштування параметрів алгоритму, таких як максимальна кількість ітерацій, демпфування та афінність;
– можливість працювати з різними даними;

– не потребує визначення кількості кластерів наперед, алгоритм автоматично визначає кількість кластерів на основі вхідних даних;
– алгоритм не залежить від вибору початкового центроїда кластера.

Для вибору вказаного вище алгоритму було проаналізовано інші алгоритми для формування рекомендацій, враховуючи недостатню кількість даних користувачів на початку роботи додатка. Також було проаналізовано вимоги для формування рекомендацій та обрано алгоритм кластеризації на основі спорідненості об’єктів, affinity propagation, виходячи з них.

Кожен елемент даних про каву містить достатню кількість характеристик для того, щоб порівнювати їх смак і таким чином можна визначити схожість між елементами та рекомендувати користувачу елементи, що найбільш схожі за смаком на ті, що він вподобав. Також було враховано можливості подальшого розвитку та покращення алгоритму рекомендацій. Таким чином на основі проведеного аналізу було отримано наступні порівняння з іншими відомими підходами для формування рекомендацій:

– колаборативний фільтр та affinity propagation мають дещо різний підхід. Колаборативний фільтр використовує історію взаємодії між користувачами та об’єктами, щоб знайти подібність між ними, проте історії взаємодії на початку роботи додатка немає і такий підхід для рекомендацій може не працювати вдало. Необхідно мати дані про достатню кількість взаємодій. У affinity propagation немає прямих зв’язків із історією взаємодій користувачів і він використовує тільки схожість між об’єктами, що може бути обчислені на основі їх характеристик [10, с. 11] [11, с. 8-9];
– алгоритми рекомендацій на основі контенту часто можуть стикатись з проблемою рекомендацій нових об’єктів, оскільки для них немає достатньої історії або кількості взаємодій. В таких випадках системи рекомендацій на основі контенту можуть мати обмежену здатність для рекомендацій. З іншого боку affinity propagation може бути менш чутливим до цієї проблеми, оскільки він враховує схожість

характеристик елементів, незалежно від історії чи взаємодій. [10, с. 11-12] [11, с. 7-8]

Отримане рішення є достатньо гнучким, оскільки його можна надалі поєднати з одним або обома із описаних, оскільки на основі кластеризації можна оцінити характеристики об’єктів та їх схожість. [10, с. 13-14] Це дозволяє уникнути проблем, які можливі для інших методів рекомендацій. Крім того affinity propagation було використано для того, щоб одразу отримувати схожі елементи на каву, що переглядає користувач. Тобто, якщо користувач переглядає якусь каву, то він одразу може побачити список кави із того ж кластера та мають схожі характеристики, що й наведена.

1.9 Аналіз подібних застосунків

Було проаналізовано додатки, що частково мають схожі функції та оцінено їх можливості, переваги та недоліки для подальшого розвитку створеного застосунку. На українському ринку наразі таких застосунків небагато і більшість є веб-сайтами з продажу кави власного обсмаження та не агрегують кави від різних брендів.

1.9.1 Аналіз Sipsome.coffee

Sipsome.coffee – це українська веб-платформа з продажу кави, що об’єднувала певну кількість обсмажчиків кави. [12] На ній була доступна підписка для отримання кави з обраною періодичністю та в обраній кількості. Підписка базувалась на певних вподобаннях користувачів для кави. Для визначення своїх вподобань користувач мав пройти короткий тест, який визначав, до якої групи належить користувач. Для кожної групи користувачів була визначена підмножина кави, яка є рекомендованою для неї. Користувач не міг оцінювати чи вподобати каву на платформі. Тобто таким чином можна визначити, що платформа не мала окремого рушія для отримання

рекомендацій, попередні вподобання користувача не враховувались. Для пошуку кави були реалізовані фільтрація по виробниках та певних характеристиках кави.

Платформа надавала можливість замовляти каву або оформлювати підписку. Обравши підписку користувач мав можливість отримувати каву з обраною періодичністю. Кава, яка йому була надіслана обиралась на основі пройденого тесту з врахуванням його вподобань. За оформлення замовлень та підписки відповіли Sipsome.coffee, а доставка здійснювалась окремими обсмажчиками кави. Наразі платформа припинила свою роботу через повномасштабне вторгнення.

1.9.2 Аналіз Fresh Black

Одна із платформ з продажу кави, Fresh Black, що також була обрана для отримання даних, також має певні налаштування для отримання рекомендацій для кави. Проте оскільки це плафторма безпосередньо обсмажчика кави, то вона містить лише каву обсмажену компанією. [13] Платформа також не має можливостей для оцінки кави та отримання рекомендацій на їх основі. Так само як і для Sipsome.Coffee для вибору плану підписки використовується невеликий тест для визначення групи кави, яка буде рекомендована користувачу.

1.9.3 Аналіз Trade Coffee

Одна із платформ з продажу кави, яка базується у Нью-Йорку. [14] Вона має схожі функції до Sipsome.coffee, проте має більш гнучкі налаштування для отримання кави за підпискою, хоча також не враховує попередні вподобання користувача. Після проходження невеликого тесту для реєстрації, користувач має налаштування групи та підгруп для кави, які він також може змінити у разі наявності неточностей. Користувач може оцінювати та коментувати каву,

проте це не впливає на формування рекомендацій. Пошук на платформі здійснюється лише по назві, проте може враховувати помилки у вводі. Наявна також додаткова фільтрація по різним характеристикам кави. Крім того платформа містить блог, в якому також є збережені наперед рецепти, а також розділ для продажу обладнання для приготування кави.

Таким чином основною перевагами застосунку, як і для попередньо оглянутого Sipsome.coffee, є велика кількість даних про каву, які отримані напряму від обсмажчиків, можливість коментувати каву та блог, який наповнюється адміністраторами.

1.9.4 Аналіз Acaia Coffee

Мобільний додаток Acaia Coffee не має засобів для рекомендації кави, це платформа, що дозволяє створювати та зберігати рецепти для себе та готувати за ними з допомогою вагів Acaia. [15] Ваги є необхідними для повноцінної взаємодії з користувачем, проте доволі дорогі. Інтеграція з вагами є одночасно перевагою та недоліком додатку. Тобто здебільшого застосунок лише для розширення функцій “розумних” ваг. За допомогою додатку не можливо переглядати рецепти запропоновані іншими користувачами або зберігати каву, для якої вони були створені, лише її назву чи текстовий опис, нотатки. Також можна поширювати лише графік за допомогою інших соціальних мереж, який був побудований за допомогою вагів під час приготування кави. Оцінка, яку можна поставити лише власному рецепту не доступна для перегляду іншими користувачами і не впливає на взаємодію з ними лише як маркування.

1.9.5 Аналіз Coffeely

Coffeely також є додатком розробленим для мобільних платформ. [16] Основним недоліком взаємодії із застосунком є великі затримки під час його

використання та помилки під час виконання. Ця платформа не містить рекомендацій для кави.

Усі дані про каву надаються користувачами із описом та фотографією. Таким чином дані про каву також є недостовірними, оскільки не отримані напряму від обсмажчиків та дистриб'юторів кави. На жаль, застосунок майже не містить кави від українських виробників, отже користувачі з України мало зацікавлені в його використанні. Крім того усі функції для взаємодії з кавою є платними. Пошук кави можливий за назвою та характеристиками кави, він враховує помилки при вводі. Можливостей для додаткової фільтрації за обраними характеристиками кави при пошуку не надано. Також є функція для пошуку кави за фотографією пакунка, що є основною перевагою додатка.

Користувач має можливість створювати рецепти із гнучкими налаштуваннями для ваги кави, температури води, методу приготування та часу для виконання кроку. Проте створені рецепти не є доступними для перегляду іншими користувачами.

Також додаток містить стрічку для перегляду дописів інших користувачів. Дописи можуть містити нотатки про каву з фотографіями, користувачі мають можливість вподобати допис та прокоментувати його. Є можливість підписатися на обраного користувача для отримання оновлень від нього.

РОЗДІЛ 2. РОЗРОБКА БЕКЕНДУ

2.1 Модель бази даних

Для збереження даних було спроєктовано модель бази даних таким чином, щоб кава була індексована до ElasticSeach та можна було здійснювати повнотекстовий пошук за її характеристиками. Оскільки для даних користувачів це не потрібно, то окремо було спроєктовано базу даних для них. Дані про взаємодію користувачів із кавою було інтегровано до користувацьких даних за допомогою ідентифікаторів кави, індексованої у ElasticSearch.

2.1.1 ElasticSearch та дані про каву

Для подальшого пошуку кави та функціонування рекомендаційного мікросервісу кава зберігається до індексу “coffee” в ElasticSearch. В ньому міститься усі дані про каву отримані з відповідних платформ з її продажу. Дані складаються з такого переліку полів:

- id – унікальний ідентифікатор кави, формуються за допомогою конкатенації назви мікросервісу, звідки було отримано дані кави та назви кави(title) з нижнім підкресленням в якості роздільника;
- url – посилання на сторінку оригінальної платформи, звідки було отримано дані про каву;
- imageUrl – посилання на зображення кави;
- source – назва платформи звідки було отримано каву;
- title – назва кави;
- price – ціна кави у гривнях, без копійок;
- description – розгорнутий опис кави;
- roast – тип обсмаження кави;
- tastes – список рядків, що описують смако-ароматичні характеристики кави;

– recommendedBrewMethods – список рядків, з рекомендованими методами для приготування кави;
– country – країна походження кави;
– processingMethod – спосіб обробки;
– variety – сорт або вид кави;
– score – оцінка кави, визначена за специфікацією Specialty Coffee Association;
– enabled – булеве поле, що характеризує наявність кави на платформі, з якої її було отримано. Під час оновлення або додавання кави полю встановлюється значення true. Якщо поле має значення false, то дані про таку каву будуть з’являтись нижче у пошуку. Проте необхідно, щоб вони продовжували зберігатись для того, щоб бути використаними для формування рекомендацій для користувача;
– lastUpdated – час, коли дані про каву були оновленні останній раз. Якщо кава не була оновлена під час останнього оновлення, то дата буде меншою за поточну. Для такої кави полю enabled буде встановлено значення false.

2.1.2 PostgreSQL та користувацькі дані

Для збереження даних про користувачів, їх рецепти та вподобану каву було створено наступні таблиці:

a) users – таблиця, що містить дані про всіх користувачів, які зареєструвались у застосунку. Містить наступні поля:
    1) id – унікальний ідентифікатор користувача;
    2) email – унікальна електронна пошта користувача на випадок необхідності зворотнього зв’язку;
    3) login – унікальний нікнейм користувача, який використовується для ідентифікації користувача у додатку;
    4) password – пароль користувача у зашифрованому вигляді;

5) name – ім’я користувача на випадок необхідності зворотнього зв’язку;
6) surname – прізвище користувача на випадок необхідності зворотнього зв’язку;
7) reset_token – унікальний токен, що використовується для оновлення паролю;
8) reset_expiration – час виділений для використання токену для оновлення паролю, 5 хвилин з моменту створення токену.

b) coffee_likes – таблиця, що містить дані про вподобану користувачами каву. Містить наступні поля:
   1) id – унікальний ідентифікатор вподобання;
   2) user_id – унікальний ідентифікатор користувача. Відповідає полю id у таблиці User;
   3) coffee_id – унікальний ідентифікатор кави. Відповідає полю id кави у індексі “coffee” в ElasticSearch;
c) recipes – таблиця, що містить загальні дані про рецепти для приготування кави, створені користувачами. Містить наступні поля:
   1) id – унікальний ідентифікатор рецепта;
   2) title – назва рецепту;
   3) user_id – унікальний ідентифікатор користувача. Відповідає полю id у таблиці User;
   4) coffee_id – унікальний ідентифікатор кави. Відповідає полю id для кави, що збережена у індексі “coffee” в ElasticSearch;
   5) brew_method – метод приготування, що описаний у рецепті;
   6) description – загальний текстовий опис рецепту;
d) recipe_step – таблиця, в якій збережений текстовий опис кроків рецепту. Містить наступні поля:
   1) id – унікальний ідентифікатор кроку рецепта;
   2) order – порядок кроку, який необхідний для послідовного відображення кроків;

3) recipe_id – унікальний ідентифікатор рецепту, до якого належить крок. Відповідає полю id у таблиці Recipe;
4) text – текстовий опис дій, які необхідно виконати в межах кроку рецепту.
5) time – час, за який необхідно виконати крок.

Відповідно усі дані, з даними про каву, що індексуються окремо до ElasticSearch модель бази даних. Схематичне зображення моделей подано на рис. 2.1.

Рисунок 2.1 – Модель бази даних.

2.2 Періодичне завдання для оновлення даних

Для того, щоб отримати дані про каву з веб-сайтів з її продажу та мати змогу підтримувати їх релевантність, було створено мікросервіс, що працює як періодичне завдання(англ. cron job). Мікросервіс реалізує єдиний ендпоінт /run, який викликається з певною періодичністю. Було обрано період раз на тиждень вночі, оскільки дані про наявність кави оновлюються не часто. Часте оновлення даних заважало б користувачам взаємодіяти із застосунком, оскільки під час оновлення даних неможливо отримати до них доступ. Запит /run не вимагає жодних вхідних даних та повертає у разі успіху HTTP код 204 No Content.

Першим кроком виконання періодичного завдання є виклик відповідних веб-скраперів, що збирають дані з веб-сторінок платформ з продажу кави за допомогою бібліотеки JSoup. Було обрано кілька платформ, для яких дані містять більшість необхідних полів та представлені у вигляді, для якого ця дія є можливою. Тобто веб-сайти мають містити HTML теги, які можна розрізняти за допомогою CSS селекторів, тобто мають містити унікальні класи або ідентифікатори, щоб їх вміст можна було отримати за допомогою бібліотеки JSoup. Такими чином використовуються веб-сторінки CafeBoutique, Mad Heads та Fresh Black.

Першочергово за відповідним посиланням виконується завантаження документа з URL та отримується HTML представлення головної сторінки з переліком усієї кави. Наступним кроком є вибір елементів за допомогою CSS-селекторів або за тегами. Після вибору елементів отримують їх вміст та необхідні атрибути. У разі виникнення помилки при роботі парсера виконується перехоплення та обробка помилок. З цієї сторінки збираються необхідні HTML теги, що містять посилання на кожну окрему представлену каву. Надалі для кожного посилання виконується завантаження документа сторінки з URL та отримується HTML представлення сторінки, що містить усі необхідні характеристики відповідної кави. Так само для кожної сторінки

знаходяться необхідні HTML теги, що містять характеристики кави та здійснюється їх обробка наступним чином:

- id – формується як текстовий рядок, що складається з назви кави, та назви платформи, що зберігається у константі відповідного парсера тобто “[title]_[source]”;
- source – назва платформи;
- price – прибираються зайві нулі та текст;
- tastes та recommendedBrewMethods – формуються у вигляді списку текстових рядків, шляхом розділення рядка за комами;
- country – приводиться до необхідного вигляду, тобто рядок має містити усі літери крім першої у нижньому регістрі. Після цього усі країни перекладаються на українську мову для уникнення проблем з порівнянням їх значень на етапі формування рекомендацій. Оскільки кількість країн, що вирощують каву обмежена та бібліотеки для перекладу можуть дати не точний результат, то для наявних країн було додано хеш-мапу, що містить назви країн англійською у якості ключів та українською у якості значень.

Наступним кроком всі отримані дані індексуються до ElasticSearch невеликими групами з кожної платформи, з якого вони були отримані, з полем enabled зі значенням true та поточною датою для lastUpdated. Після цього всі дані, що не були оновлені та мають застаріле значення для поля lastUpdated оновлюється для встановлення значення false поля enabled.

Після цього формується запит POST /reset до рекомендаційного мікросервісу, для поширення сповіщення про необхідність оновлення рекомендацій для кави. Для цього створення клієнта було використано бібліотеку Feign Client та описано невеликий інтерфейс з відповідним ендпоінтом /рекомендаційного мікросервісу.

У випадку, якщо на одному з етапів виконання відбувається помилка, то причина логується та опис помилки повертається у відповіді на запит з HTTP 500 Internal Server error. Такими можуть бути помилки під час взаємодії з

зовнішніми ресурсами за допомогою бібліотеки JSoup для кожного зі скраперів та парсерів. А також помилки при індексації даних до ElasticSearch.

2.3 Мікросервіс для рекомендацій

Мікросервіс для рекомендації відповідає за формування та отримання користувачем рекомендацій для кави, а також її пошук. Він складається з рушія для формування рекомендацій кави за її схожістю, шару для доступу до сховищ даних та REST API для отримання результатів.

2.3.1 Формування рекомендацій

Після оновлення даних кави у ElasticSearch необхідне оновлення параметрів моделі рекомендацій. Щоб запустити переобчислення алгоритму було додано ендпоінт /reset. Для того, щоб під час оновлення користувач не отримав спотворені дані, на збережених результатах алгоритму застосовується блокування читання-запису. Доступ до ендпоінта налаштовується таким чином, щоб його можна було викликати з хоста, на якому працює періодичне завдання, що займається оновленням даних про каву. Після POST виклику ендпоінту /reset усі дані про каву отримаються з ElasticSearch за допомогою виклика методу search без тіла запиту.

Побудова рекомендацій відбувається за допомогою обраного алгоритму affinity propagation. Це алгоритм кластеризації для групування об’єкті на основі їх взаємної спорідненості, що не вимагає визначення наперед кількості кластерів під початком роботи алгоритму. На вхід подається набір даних кави $D = \{d_1, d_2, \dots d_n\}$. Для цих даних обчислюється матриця спорідненості s розмірності $N \times N$. У якості метрики спорідненості було обрано таку: $taste\_similarity + processing\_method\_similarity + origin\_similarity$, де вказані параметри обчислюються наступним чином:

– $taste\_similarity(coffee1, coffee2)$ – обчислюється як кількість елементів параметру tastes, які співпадають для coffee1 та coffee2;
– $processing\_method\_similarity(coffee1, coffee2)$ – має значення 1, якщо processing_method співпадають для coffee1 і coffee2 та 0, якщо ні;
– $origin\_similarity(coffee1, coffee2)$ – має значення 1, якщо country співпадають для coffee1 і coffee2 та 0, якщо ні.

Надалі алгоритм почергово виконує два кроки передачі повідомлень для оновлення двох наступних матриць:

– матриця відповідальності R зберігає значення $r(i,k)$, які визначають наскільки добре значення x_k відповідає тому, щоб служити зразком для значення x_i, порівняно з іншими зразками-кандидатами для нього;
– матриця доступності A зберігає значення $a(i,k)$, які вказують наскільки доцільно для значення x_i обрати x_k як приклад, узявши до уваги перевагу інших точок щодо x_k як зразка.

Обидві матриці ініціалізуються нулями та заповнюються ітеративно. Ітерації виконуються допоки межі кластера не залишаться незмінними, або якщо кількість ітерацій перевищує задану кількість кроків. Було обрано максимальну кількість ітерацій 200. Ітерації визначаються наступним чином:

– спочатку поширюються оновлення відповідальності:

$$r(i,k) \leftarrow s(i,k) - \max_{k' \neq k}\{a(i,k') + s(i,k')\};$$

– наступними поширюються оновлення доступності:

$$a(i,k) \leftarrow \min\left(0, r(k,k) + \sum_{i' \notin \{i,k\}} \max\big(0, r(i',k)\big)\right) for\ i \neq k$$

та

$$a(k,k) \leftarrow \textstyle\sum_{i' \neq k} \max(0, r(i',k)).$$

Далі з кінцевих отриманих матриць вилучаються центри кластерів, так звані “екземпляри”, для яких сума значень відповідальності та доступності більша за 0. [17] [18]

Оскільки матриця спорідненості не зберігає значення id для кави, то було додано хеш-мапу, що зберігає співвідношення id кави до позиції рядка в матриці. Результати роботи алгоритму було вирішено зберігати в оперативній пам’яті, оскільки її розмір за даної кількості даних не перешкоджає виконанню програми. Також таким чином запити з отримання рекомендацій для користувача не потребують додаткового читання з файлу і виконуються швидше. При збільшенні кількості даних можна розглянути збереження результатів роботи алгоритму до файлу в рекомендаційному мікросервісі.

2.3.2 Отримання рекомендацій

Для отримання рекомендацій було додано два ендпоінти:

– рекомендації за кавою – GET /recommendations/coffee/[coffee_id];
– рекомендації за наявним авторизованим користувачем – GET /recommendations/user/[user_id].

При отриманні рекомендацій за кавою спершу з хеш-мапи знаходиться ключ, що відповідає наданому coffee_id. Надалі знаходиться кластер, до якого належить дана кава та повертаються усі інші сутності в ньому, окрім наданої. Дані відсортовані таким чином, щоб першими у списку були відображені ті, що мають значення true для параметра enabled.

Отримання рекомендацій для користувача також вимагає, щоб користувач, який його викликає був авторизований. Для цього користувач має надіслати в хедері запита “Authentication” сесійний jwt токен. Рекомендаційний мікросервіс має перевірити токен для користувача. Якщо токен є помилковим або недійсним, то наступні дії виконані не будуть, у відповідь на запит буде надіслана відповідь з HTTP кодом 401 Unauthorized. Для отримання рекомендацій за користувачем операції виконуються аналогічно. Проте для кожної кави, уподобаної користувачем, знаходиться кластер, до якого вона належить. Результати відсортовані за кількістю кави, що належала кластеру та полем enabled. Тобто зверху будуть результати з

кластера, до якого належало найбільше кави уподобаної користувачем та для якої поле enabled має значення true.

В обох випадках як результат повертається список записів кави в тому ж вигляді, як вони збережені в ElasticSearch.

2.3.3 Пошук кави

Для пошуку та отримання необхідних сутностей кави імплементовано наступні ендпоінти:

– загальний ендпоінт, що повертає усю каву – GET /coffee/all/;
– повернення конкретної кави за id – GET /coffee/[coffee_id];
– пошук кави за наданим пошуковим рядком – GET /coffee/[text].

Для отримання усієї кави як і з випадком для налаштування рекомендацій здійснюється виклик search до ElasticSearch. У тілі запиту можуть бути вказані параметри from та size для пагінації, якщо вони були надіслані користувачем. Якщо вони не вказані, то буде повернуто перші 10 документів кави.

Для отримання конкретної кави здійснюється запит get до ElasticSearch з вказаним ідентифікатором документа кави. У відповідь користувачу надсилається єдиний документ потрібної кави. Якщо такого документу не було знайдено в ElasticSearch, то для користувача буде повернуто повідомлення про помилку із HTTP кодом 400 Bad Request.

Для пошуку кави за текстовим рядком виконується валідація довжини вхідної текстової стрічки, вона не має перевищувати 50 символів, інакше користувач отримає повідомлення про відповідну помилку із HTTP кодом 400 Bad Request. Для пошуку кави за пошуковим рядком використовується повнотекстовий пошук ElasticSearch за полями title, country, description та tastes. Для пошуку використовується запит search, що містить оператори match з відповідними полями, які поєднуються за допомогою операторів bool та should. Тобто хоча б одне з полів має містити співпадіння із пошуковою

стрічкою, надісланою користувачем. При розширенні кількості важливих полів, їх можна легко додати до списку необхідних для пошуку.

Алгоритми ElasticSearch оцінюють, наскільки часто терміни з запиту зустрічаються в документах та наскільки вони важливі в контексті колекції даних. Пошук може бути здійсненим не зважаючи на помилки та описки у запиті. Таким чином у відповіді будуть отримані найбільш релевантні до пошуку сутності, першими з яких будуть ті, що мають значення true для поля enabled.

2.4 Користувацький мікросервіс

Користувацький мікросервіс містить увесь функціонал для взаємодії користувача зі створеним застосунком. Були імплементовані функції реєстрації та авторизації, оцінки кави, а також створення, отримання та фільтрація рецептів.

2.4.1 Реєстрація та авторизація

Для реєстрації та автентифікації було додано наступні ендпоінти:

– створення користувача, якщо такого не існує у базі даних за полями email або login – POST /authentication/register;
– авторизація користувача за логіном там паролем, якщо такий існує в базі даних за полем email – POST /authentication /login;
– запит на зміну пароля – POST /password-reset/request;
– зміна паролю за отриманим токеном – POST /password-reset/reset.

При реєстрації користувача у тілі запиту надсилаються усі поля, які необхідні зберегти до таблиці user бази даних PostgreSQL, окрім ідентифікатора та даних для зміни паролю. Для отриманої сутності здійснюється валідація на те, що всі обов’язкові поля не нульові. Крім того поля мають відповідати наступним вимогам:

- email – обов'язкове поле, має містити до 40 символів та відповідати формату електронної пошти;
- login – обов'язкове поле, допустима довжина рядка від 3 до 30 символів;
- password – обов’язкове поле, допустима довжина рядка від 8 до 20 символів;
- name – обов’язкове поле, допустима довжина рядка до 30 символів;
- surname – обов’язкове поле, допустима довжина рядка до 30 символів.

Якщо якась із валідацій не пройшла успішна, то у відповідь повертається повідомлення про неправильні поля з HTTP кодом 400 Bad Request.

Перед обробкою та збереженням даних користувача здійснюється пошук по полю email та login в таблиці user. У випадку, якщо користувач з наданою електронною поштою або login вже зареєстрований у базі даних, то у відповідь буде надіслано повідомлення про помилку із HTTP кодом 400 Bad Request. Для збереження паролю до бази даних його значення хешується за допомогою BCryptPasswordEncoder, що використовує алгоритм bcrypt. Це рекомендований підхід для хешування паролів за допомогою Spring Security. Він автоматично використовує унікальну сіль (salt) та проводить багато ітерацій для ускладнення атак на паролі. Для хешування та перевірки співпадіння з незахешованим паролем необхідно здійснити лише по одному виклику функцій. Після хешування паролю новий користувач додається до таблиці users. Токен не генерується при реєстрації, тому користувач має викликати наступний ендпоінт /login з використаними електронною поштою та паролем для того, щоб отримати доступ до обмежених ресурсів.

Для авторизації користувач має викликати ендпоінт /login та надіслати електронну пошту(email) та пароль(password) у тілі запита. Ці поля є обов’язковими та мають такі ж обмеження як і при створенні користувача. Після валідації запиту мікросервіс має перевірити чи існує користувач з таким email у базі даних та порівняти пароль із збереженим зашифрованим у базі даних за допомогою BCryptPasswordEncoder. Якщо користувача не існує у базі

даних або пароль є неправильним, то у відповіді на запит буде отримано помилку з HTTP кодами 404 Not Found або 400 Bad Request відповідно. У випадку, якщо помилок не виникло, то мікросервіс створює новий jwt токен для користувача та повертає його(sessionToken) та нікнейм користувача(login) у тілі відповіді з HTTP кодом 200 OK.

Для зміни паролю користувач має виконати два кроки. Першим є крок з створенням запиту для оновлення паролю. Користувач має надіслати запит зі своїм email на ендпоінт /password-reset/request. Після цього мікросервіс перевіряє наявність користувача з зазначеною електронною адресою у базі даних. Якщо користувача не існує, то у відповідь повертається відповідне повідомлення з HTTP кодом 400 Bad Request. Інакше створюється токен, як UUID, його значення зберігається до поля reset_token та до поля reset_expiration встановлюється час через 5 хвилин після поточного. Після цього на пошту користувача надсилається лист, що містить посилання складене наступним чином: [frontend_url]/resetPassword/[reset_token]. При переході за посилання користувач матиме форму для введення нового паролю. У разі успішної роботи запит повертає HTTP код 202 Accepted.

Наступним кроком для зміни паролю є встановлення нового паролю з отриманим токеном для його зміни. Для цього користувач має надіслати отриманий токен та новий пароль у тілі до запиту на ендпоінт /password-reset/reset. Якщо користувач для такого токена не існує або токен не є дійсним, що перевіряється за полем reset_expiration, то у відповідь на запит він отримає повідомлення про помилку з HTTP кодом 400 Bad Request. Інакше пароль буде заново зашифрований та збережений до бази даних, та користувач отримає у пусту відповідь із HTTP кодом 202 Accepted. Для продовження роботи з додатком йому буде необхідно наново авторизуватися з новим паролем.

2.4.2 Створення та отримання рецептів

Для створення та отримання даних про рецепти було імплементовано наступні ендпоінти:

- створення рецепту – POST /recipe;
- зміна параметрів для рецепту – PATCH /recipe/[recipe_id];
- видалення рецепту та усіх існуючих для нього кроків за recipe_id – DELETE /recipe/[recipe_id];
- створення кроків рецепту – POST /recipe-steps;
- зміна параметрів кроків рецепту – PATCH /recipe-steps;
- видалення кроків, що належать до списку – DELETE /recipe-steps;
- отримання та фільтрація рецептів за методом приготування, що використовується – GET /recipes;
- отримання рецептів за користувачем – GET /recipes/[user_id];
- отримання згрупованих методів для приготування кави, що використані в наявних рецептах – GET /recipe/brewMethods;
- отримання рецепту з його кроками – GET /recipe/[recipe_id].

Для будь-якого з цих запитів, якщо користувач не авторизований, то має повернутися повідомлення з помилкою та HTTP кодом 401 Unauthorized. Якщо рецепт чи його кроки не належать йому, то повернеться помилка та HTTP кодом 400 Bad Request. Для цього здійснюється перевірка є відповідними ідентифікатори користувача, якому належить сесійний токен та той, що був надісланий у запиті.

Для створення рецепту авторизований користувач має надіслати на ендпоінт /recipe дані про опис рецепту та метод, який використовується для приготування кави. Тобто необхідно надіслати усі дані, які потрібні для збереження сутності до бази даних, окрім ідентифікатора, який генерується автоматично. Вони мають бути не нульові та не пусті. Після отримання об’єкт запиту проходить валідацію, усі поля мають бути не пусті та мати допустиму довжину:

- user_id – ідентифікатор користувача, який створив рецепт має бути присутнім та відповідати авторизованому користувачу, який надіслав рецепт. Тобто jwt токен, який надісланий у хедері запиту має належати користувачу з таким самим ідентифікатором;
- coffee_id – кава має бути наявна серед даних збережених у ElasticSearch;
- brew_method – метод приготування має бути вказаний та його довжина не має перевищувати 20 символів;
- description – опис є опціональним, текст не має перевищувати довжину у 500 символів.

У випадку, якщо якась із валідацій не пройшла успішно, то у відповідь повертається повідомлення про неправильні поля з HTTP кодом 400 Bad Request. Якщо ж усі валідації пройшли успішно то сутність для рецепту зберігається до бази даних та запит має повернути змінену сутність, якою вона збережена в базі даних з HTTP кодом 200 OK.

Для зміни рецепту має бути надісланий запит з ідентифікатором існуючого рецепту з полями, які необхідно змінити. Якщо рецепт за ідентифікатором не було знайдено, то у відповіді буде повідомлення про помилку з HTTP кодом 404 Not Found. Усі поля можуть бути відсутніми, проте мають містити допустиму кількість символів, якщо вони наявні. Якщо хоча б одна з цих валідацій не проходить, то має повернутися повідомлення про помилку із HTTP кодом 400 Bad Request. Якщо ж всі валідації були пройдені успішно та сутність було успішно збережено до бази даних, то запит має повернути змінену сутність, якою вона збережена в базі даних з HTTP кодом 200 OK.

Для видалення рецепту має бути здійснений запит до ендпоінта /recipe/[recipe_id]. Необхідно, щоб були здійснені перевірки на наявність рецепту в базі даних та належність користувачу. В разі невідповідності у відповіді міститься повідомлення про помилку з HTTP кодами 404 Not Found та 401 Unauthorized. Надалі видаляються сутність рецепту та усі кроки, які

мають recipe_id, що відповідає ідентифікатору рецепта. У разі успіху повертається пуста відповідь із HTTP кодом 202 Accepted.

Аналогічно до операцій з сутністю рецепту виконуються операції над його кроками, окрім того, що всі операції виконуються для групи сутностей, які мають бути надіслані у вигляді масиву. Для них виконуються наступні валідації для створення:

– order – порядок кроку має бути унікальним, має зберігатись послідовність чисел починаючи з 0;
– recipe_id – усі сутності в групі мають мати однаковий ідентифікатор рецепту, який має належати авторизованому користувачу, який надсилає запит;
– text – текст не має перевищувати 200 символів;
– time – час має бути вказаний у правильному форматі та бути меншим за годину.

Для оновлення сутностей кроків зберігаються ті самі валідації, проте якщо якийсь крок не був у списку змінених сутностей, то валідація для послідовності порядків усіх кроків має бути здійснена для усіх кроків рецепту лише за зміни порядку якихось із них.

При видаленні кроків рецепту необхідним є також зсунення порядку наступних кроків рецептів на порядок нижче таким чином, щоб їх послідовність зберігалась та була правильною.

При отриманні списку рецептів користувачу повертається список рецептів, який містить лише загальні відомості про рецепт без його кроків та загальний час приготування обчислений з його кроків. Для встановлення обмеження на кількість результатів користувач може вказати наступні параметри:

– limit – максимальна кількість результатів для повернення користувачу, має бути більше 0, за замовчуванням 10;
– offset – номер рядка першого елемента для повернення користувачу, має бути більше 0, за замовчуванням 0;

– brewMethod – параметр для фільтрації рецептів за методом приготування.

Якщо валідація не була виконана успішна для одного з параметрів, то для користувача буде повернуто повідомлення про помилку з HTTP кодом 400 Bad Request.

Аналогічним чином працює запит для отримання рецептів для користувача, лише фільтрація результатів для отримання їх з бази даних здійснюється за допомогою ідентифікатора користувача.

Запит для отримання методів для приготування кави повертає список значень поля brew_method для рецептів, відсортованих кількістю рецептів для них. Він є необхідним для отримання даних для фільтрації.

Запит на отримання конкретного рецепту повертає всю його структуру із кроками, якщо рецепт було знайдено за наданим ідентифікатором. Інакше користувач отримає повідомлення про помилку із HTTP кодом 404 Not Found.

2.4.3 Вподобання кави користувачем

Для реалізації функції вподобання кави користувачем було імплементовано наступні ендпоінти:

– Збереження вподобання користувача – POST /like/[coffee_id];
– Видалення вподобання користувача – DELETE /like/[coffee_id]
– Отримання вподобання користувача для конкретної кави – GET /like/[coffee_id];
– Отримання списку вподобань користувача – GET /likes.

При виклику кожного з ендпоінтів перевіряється сесійний токен користувача, за яким отримується його ідентифікатор. Якщо токен неправильний або недійсний, тоді повертається відповідне повідомлення про помилку із HTTP кодом 401 Unauthorized. Наступним кроком перевіряється наявність кави у ElasticSearch за допомогою виклику get за ідентифікатором кави та отримання конкретного документу кави. Якщо кави за наданим

ідентифікатором не існує, то користувач отримає помилку із HTTP кодом 404 Not Found. Та ж помилка отримується, якщо користувач для надісланого токена не існує в базі даних.

Для створення вподобання кави в базі даних перевіряється чи вже є запис про вподобання кави користувачем. Якщо такий вже існує, то новий запис не додається. Інакше створюється новий запис про вподобання користувача, що складається з отриманих ідентифікаторів кави та користувача. Для видалення вподобання відповідно здійснюється спроба вилучення запису, що містить надані coffee_id та user_id.

У разі успішного виконання запитів з створення чи видалення вподобання, користувач отримає пусту відповідь із HTTP кодом 202 Accepted.

Для перевірки вподобань користувача створено два ендпоінти для отримання списку або окремого вподобання користувача для наданої кави. Дані про користувача для отримання вподобань з бази даних отримуються з надісланого jwt токена. У випадку запиту для отримання усіх вподобань користувача відповідь містить список, що складається з елементів з ідентифікаторами вподобаної кави для користувача.

Для запиту з вподобання конкретної кави відповідь містить один елемент, що складається з ідентифікаторів кави та користувача та булевого поля liked, яке характеризує наявність чи відсутність вподобання. Обидві відповіді користувач отримує із HTTP кодом 200 OK у разі успіху.

РОЗДІЛ 3. ФРОНТЕНД

3.1 Взаємодія користувача з додатком

Користувач має доступ до 3 основних вкладок: головної, вкладки з рекомендаціями для нього та вкладки з рецептами. Усі шляхи, до яких авторизований користувач має доступ:

- /main – головна сторінка, що містить перелік кави;
- /coffee/[coffee_id] – сторінка вибраної кави зі списку;
- /recommendations/[user_id] – сторінка з рекомендаціями для користувача;
- /recipes – сторінка з усіма рецептами;
- /recipe/new – сторінка для створення нового рецепту;
- /recipe/[recipe_id]/edit – сторінка для заповнення та внесення змін до кроків рецепту;
- /recipe/[recipe_id] – сторінка для перегляду інформації про конкретний рецепт;
- /user/[user_id] – сторінка для перегляду інформації про конкретного користувача.

Усі запити до бекенд мікросервісів використовують HTTPS протокол для безпеки передачі чутливих даних. Для авторизованих користувачів сесійний токен, нікнейм та ідентифікатор зберігаються у локальне сховище для того, щоб користувач залишався авторизованим при випадковому закритті вкладки чи браузера. Ендпоінти за допомогою яких, фронтенд взаємодіє із бекенд мікросервісами описані для кожного з них окремо. Таким чином джерелами з яких отримуються дані для відображення та взаємодії з користувачем є мікросервіс для отримання рекомендацій та пошуку кави та користувацький мікросервіс. Мікросервіс, що виконує періодичне завдання не має ендпоінтів для взаємодії з користувачем і його виклик здійснюється лише оркестратором (рис. 3.1).

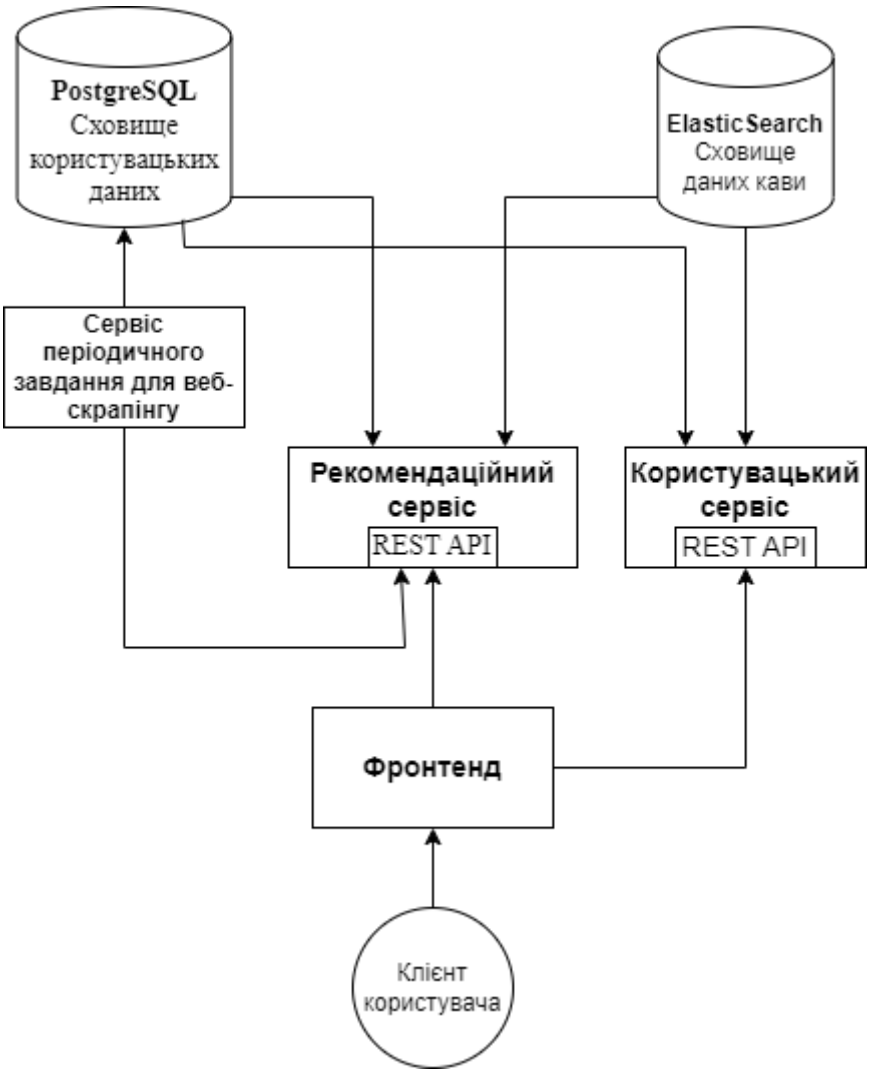

Рисунок 3.1 – Діаграма взаємодії мікросервісів.

3.2 Головна сторінка та сторінка рекомендацій

Головна сторінка містить перелік усієї кави, дані про яку є в системі, отримані з рекомендаційного мікросервісу. Дані відображені у вигляді карток, що містять базову інформацію: фото, назву та смако-ароматичні характеристики. Користувач може ввести свій запит для пошуку у пошуковій стрічці, після цього у переліку буде відображена кава, яка є релевантною до заданого пошуку. Користувач може вподобати каву натиснувши на відповідну кнопку на картці обраної кави. Кава, що вже була вподобана користувачем буде мати заповнену кольором кнопку для вподобання. Якщо він натисне кнопку знову, то кава буде прибрана з його вподобань. Також користувач може

перейти на сторінку з розгорнутою інформацією про вибрану каву натиснувши на саму її картку.

Сторінка з рекомендаціями для користувача працює аналогічно до головної, проте обмежену кількість карток з кавою, які є схожими на каву, яку він відмічав як вподобану. Якщо користувач ще не має вподобань, то сторінка буде пустою.

3.3 Сторінка кави

На сторінці конкретної кави відображені усі її розгорнуті характеристики та картки з кавою, яка є найбільшою схожою на обрану. На цій сторінці для карток з відображеними даними кави також присутня кнопка для вподобання кави. Картки для схожої кави працюють аналогічно до тих, що зображені на головній сторінці і по ним можна перейти на сторінку відповідної кави. Кава, що вже була вподобано користувачем не відображається серед списку рекомендацій.

Також сторінка містить кнопку для створення рецепту з використанням вибраної кави. При переході на сторінку зі створенням рецепту кава для нього буде встановлена автоматично та користувач може продовжити заповнення необхідних даних для створення рецепту.

3.4 Сторінки для створення рецепту

Сторінка для створення рецепту містить поля для заповнення основних характеристик рецепту та місце для додавання карток з кроками для створення рецепту. Порядок карток з кроками можна змінювати та видаляти непотрібні. Користувач може створити рецепт після завершення заповнення полів або відмінити дію.

Якщо користувач перейде у власний профіль, то може переглянути створені ним рецепти та перейти на сторінку для зміни одного з них, де також буде присутня кнопка для видалення рецепту.

Сторінка для перегляду рецептів містить список карток з рецептами розподіленими по номерам сторінок та список для вибору методів для фільтрації. При застосуванні фільтрації користувачу будуть відображені тільки рецепти, що відповідають критеріям фільтрації. При виборі картки з рецептом користувач перейде на сторінку з повним відображенням інформації про рецепт. Також користувач може перейти за посиланням на профіль користувача, що створив рецепт, та переглянути його загальні дані та інші створені ним рецепти.

3.5 Взаємодія з неавторизованим користувачем

У випадку, якщо користувач не зареєстрований або не є авторизованим у системі на момент її використання, він має доступ до обмеженої кількості функцій: перегляд кави та пошук на головній сторінці, перегляд та пошук рецептів створених іншими користувачами. Також він має доступ до форм для реєстрації та авторизації, а також відновлення паролю, у разі його втрати.

Для неавторизованого користувача відсутні функції для створення рецептів, уподобання кави та перегляд власного профілю. Вкладка для рецептів є прихованою та при спробі переходу на неї користувач переадресовується на головну сторінку. На місці кнопки для переходу на власний профіль для неавторизованого користувача відображено випадаючий список з вибором із авторизації чи реєстрації на платформі. Вибір одного з елементів у списку відкриває для користувача відповідну форму для реєстрації та заповнення особистих даних або авторизації, якщо користувач вже має акаунт. Якщо користувач з певних причин втратив пароль, то платформа надає йому можливість відновити пароль, за електронною поштою.

Якщо під час взаємодії з платформою токен перестає бути дійсним і бекенд мікросервіси повертають повідомлення помилку з HTTP кодом 401, то дані про користувача будуть стерті з локального сховища та користувача буде переадресовано на головну сторінку, де він знову може авторизуватись, отримати новий токен та продовжити роботу на платформі.

РОЗДІЛ 4. ПЛАН ПОДАЛЬШОГО РОЗВИТКУ ВЕБ-ДОДАТКУ

4.1 Отримання безпосереднього доступу до даних

Рішення з використанням веб-скраперів для створення веб-застосунку є робочим, проте розширення його для інших платформ потребує більше часу. Необхідно аналізувати веб-сторінки кожної такої платформи та створювати унікальний веб-скрапер. Крім того з часом HTML-представлення сторінок може змінюватись, отже необхідна постійна підтримка, аналіз та внесення змін. Кожен крок роботи веб-скраперів та парсерів потребує більше часу на обробку даних, аніж їх отримання безпосередньо з необхідної бази даних чи веб-апі. Для цього необхідно отримати дозвіл від власників платформ з продажу кави. Таким чином необхідно було б лише отримувати дані в тому вигляді, як їх зберігає відповідна платформа та приводити до вигляду моделі даних, що збережені у створеному веб-додатку.

У разі успіху створеного застосунку він може бути використаний в якості реклами враховуючи успіх агрегаторів для інших продуктів. Наприклад, для розміщення своїх товарів на HotLine, платформи з продажу платять кошти і таким чином їх товар більше продається. Таким чином можна забезпечити отримання коштів на подальший розвиток веб-додатку, отримання більшої кількості даних для його наповнення і полегшення його підтримки.

4.2 Замовлення кави

Серед розглянутих застосунків, що мають споріднені функції зі створеним додатком, також є ті на яких безпосередньо можна замовляти кави чи створювати підписку для отримання кави на основі вподобань з певною періодичністю. Єдина платформа, що агрегувала каву від різних українських виробників та мала подібну підписку наразі припинила роботу, відповідно тому для додатку немає конкурентів у цій сфері в Україні. Для цього

необхідним є налагодження зв’язку із платформами з продажу кави. Оскільки створений застосунок вже містить систему для формування рекомендацій, то створення підписки також є можливим. Для цього також необхідне створення систем для оплати та відслідковування замовлення.

4.3 Покращення алгоритму для отримання рекомендацій

При збільшенні кількості користувачів стане можливим покращення системи рекомендацій, оскільки стане можливим будувати рекомендації не лише на спорідненості відомих характеристик кави. Можна доповнити алгоритм таким чином, щоб на рекомендації також впливали вподобання користувачів, що оцінюють схожу каву, тобто використати підхід колаборативного фільтрування. При цьому для нових користувачів, які ще вподобали невелику кількість кави та нової кави, що ще не отримала достатньої кількості оцінок від користувачів можна буде використовувати попередній підхід, тобто це допоможе уникнути проблеми холодного старту для системи рекомендацій. Таким чином при поєднанні двох методів можна буде отримати більш якісні рекомендації.

Також для користувачів, які з якоїсь причини не ставлять оцінки для кави можливим покращенням є збереження даних про переглянуту каву, кількість разів, коли користувач відкривав сторінку певної кави, використання певної кави в рецептах, які завантажив користувач. Додатково можливе створення невеликого тесту для проходження при реєстрації. Таким чином можливо отримати мінімальну кількість даних для отримання рекомендацій.

4.4 Розвиток елементів соціальної мережі

Соціальні мережі є потужним інструментом для швидкого поширення інформації. Вони дозволяють користувачам легко ділитися новинами,

фотографіями, відео та іншими контентом зі своїми друзями та прихильниками.

Оскільки люди, що користуватимуться застосунком вже точно мають щонайменше одну спільну тему інтересів, то елементи соціальної мережі можуть заохотити їх більше користуватись застосунком та спілкуватись між собою. Крім того користувачі мають змогу розміщувати рецепти, тому було б доцільно розглянути створення фунціоналу, який би забезпечував можливість для користувачів підписатись на користувачів, які їм цікаві та отримувати сповіщення при отриманні оновлень від них. Для цього необхідний окремий мікросервіс для сповіщень, щоб розсилати їх підписаним користувачам. Також кілька з розглянутих для порівняння платформ також містили блог із статтями та рецептами, які було створено та розміщено адміністраторами, що також можна розглянути як одну із можливих додаткових функцій для взаємодії із користувачами.

4.5 Модерація контенту створеного користувачами

Оскільки користувачі можуть публікувати недоцільний або образливий контент замість передбачених рецептів, то необхідно додати можливість створення користувачів із роллю модератора, які матимуть змогу переглядати створений контент, видаляти недоцільний та надсилати сповіщення користувачам. Для цього необхідно створити нове API для функцій модератора та налаштувати контроль доступу до них. Оскільки наразі з платформою працює лише один тип користувачів, то використання такого підходу не було необхідним.

4.6 Розширення на інші платформи

Для привернення уваги більшої кількості користувачів та зручності використання можливим шляхом розвитку є створення мобільного додатку.

Хоча користуватись додатком через веб-сторінку можливо і на мобільних платформах, проте користувачам, яким про нього не відомо може бути важко про нього дізнатись. Крім того це надає можливість використати функції, які можливо додати в зручному вигляді лише на мобільній платформі, наприклад, сповіщення у зручній формі. Також можна додати якісь статистичні функції, які не зручно використовувати з веб-сторінки. Наприклад, для відслідковування кількості спожитої кави користувачем на день.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було розроблено додаток, який дає можливість шукати каву, відмічати улюблену та отримувати рекомендації за утвореними вподобаннями.

Було здійснено аналіз подібних веб-додатків та отримано їх переваги та недоліки. Також були створенні вимоги до системи, на основі яких було обрано інструментарій для виконання проєкту.

Першопочатково було проведено аналіз платформ з продажу кави та обрано ті з них, для яких дані були розміщенні у доступному та розгорнутому вигляді. Для кожної з цих платформ було розроблено веб-скрапери та парсери для отримання даних кави та приведення їх до спільного вигляду, щоб зробити можливим їх оцінку та порівняння при побудові рекомендацій.

При аналізі цих даних, було визначені ключові параметри для кожної кави і на їх основі були побудовані моделі баз даних. Також було побудовано моделі для зберігання інформації користувачів веб-додатку, їх аутентифікаційні дані, дані щодо вподобань по каві та їхні власні рецепти приготування. Було проведено аналіз можливих способів взаємодії користувача з отриманою моделлю бази даних.

На основі вимог до системи було проведено аналіз та розробку мікросервісної архітектури, що також потребувало проектування API для мікросервісів. Загалом було створено чотири мікросервіси: для періодичного оновлення даних з обраних платформ з продажу кави, для отримання рекомендацій та пошуку за визначеними параметрами, для взаємодії з користувачем та створення рецептів, а також фронтенд.

Також під час виконання кваліфікаційної роботи було проаналізовано методи та алгоритми для отримання рекомендацій для користувачів і було обрано та реалізовано алгоритм affinity propagation.

Результат роботи може бути корисним як для поціновувачів кави, так і для тих хто вперше шукає каву, оскільки тут зібрані різні сорти та смаки кави в одному місці з описаними рецептами, і не треба шукати їх по різним сайтам та порівнювати характеристики кави власноруч для вибору тої, що задовільнить смакові потреби.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ICO I. C. O. Historical data on the global coffee trade //International Coffee Organization-Historical Data on the Global Coffee Trade (ico. org). – 2022.
2. Statista – Coffee production worldwide from 2003/04 to 2020/21 (in million 60 kilogram bags) [Електронний ресурс], – Режим доступу до ресурсу: statista.com/statistics/263311/worldwide-production-of-coffee/
3. Sybil Agri – Global coffee market report In February 2022 https://sybilagri.com/market/global-coffee-market-report-in-february-2022/
4. Громик, О. (2022). Аналіз сучасного стану кавової індустрії в Україні. Ресторанний і готельний консалтинг. Інновації, 5(2), 250–266. https://doi.org/10.31866/2616-7468.5.2.2022.270105
5. Coffee-story – Вплив російської війни в Україні на кавовий ринок України та світу [Електронний ресурс], – Режим доступу до ресурсу: https://coffeestory.in.ua/uk/articles/industriya/vliyanie-rossiiskoi-voiny-v-ukraine-na-kofeinyi-rynok-ukrainy-i-mira/
6. Elastic – What is an Elasticsearch Index? [Електронний ресурс], – Режим доступу до ресурсу: https://www.elastic.co/blog/what-is-an-elasticsearch-index
7. PostgreSQL [Електронний ресурс], – Режим доступу до ресурсу: https://www.postgresql.org/docs/
8. Scikit-learn – sklearn.cluster.AffinityPropagation [Електронний ресурс], – Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AffinityPropagation.html
9. Frey B. J. Clustering by passing messages between data points /B. J. Frey, D. Dueck //science. – 2007. – Vol. 315. – №. 5814. – P. 972-976.

10. Ricci F. Introduction to recommender systems handbook /F. Ricci, L. Rokach, B. Shapira //Recommender systems handbook. – Boston, MA : springer US, 2010. – P. 1-35.
11. Terveen L. Beyond recommender systems: Helping people help each other /L. Terveen, W. Hill //HCI in the New Millennium. – 2001. – Vol. 1. – №. 2001. – P. 487-509.
12. Sipsome.coffee | Instagram [Електронний ресурс], – Режим доступу до ресурсу: https://www.instagram.com/sipsome.coffee/
13. Fresh.Black [Електронний ресурс], – Режим доступу до ресурсу: https://fresh.black/
14. Coffee Trade [Електронний ресурс], – Режим доступу до ресурсу: https://www.drinktrade.com/about-us
15. Our Apps – Acaia [Електронний ресурс], – Режим доступу до ресурсу: https://acaia.co/pages/apps
16. Coffeely - Your Coffee App [Електронний ресурс], – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=io.appery.project320490&hl=uk&gl=US
17. Brendan J. Frey; Delbert Dueck (2007). "Clustering by passing messages between data points". Science. 315 (5814): 972–976.
18. GeeksforGeeks – Affinity Propagation in ML | To find the number of clusters [Електронний ресурс], – Режим доступу до ресурсу: https://www.geeksforgeeks.org/affinity-propagation-in-ml-to-find-the-number-of-clusters/