

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

“До захисту допущено”
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА

“ ____ ” _____ 2024 р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою
“Цифрові технології в енергетиці”
зі спеціальності 122 **“Комп’ютерні науки”**

на тему: **Сервіс пошуку закладів харчування за смаковими
вподобаннями користувача**

Виконала:
студентка IV курсу, групи ТМ-02

_____ ГУРИН Ірина Андріївна
(прізвище, ім’я, по батькові)

_____ (підпис)

Керівник: доцент каф. цифрових технологій в
енергетиці, доцент, к.т.н., Володимир ТИХОХОД
(посада, вчене звання, науковий ступінь, ім’я, ПРІЗВИЩЕ)

_____ (підпис)

Рецензент: _____

_____ (посада, вчене звання, науковий ступінь, ім’я, ПРІЗВИЩЕ)

_____ (підпис)

Н.контроль: асистент Володимир РУДИК
(посада, ім’я, ПРІЗВИЩЕ)

_____ (підпис)

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

Студентка _____
(підпис)

Київ – 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)

спеціальність 122 “Комп’ютерні науки”

Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ

Завідувач кафедри ЦТЕ

Наталія АУШЕВА

«__» _____ 2024 р.

ЗАВДАННЯ
на дипломну роботу студенту

ГУРИН Ірині Андріївні

(прізвище, ім’я, по батькові)

1. Тема роботи **Сервіс пошуку закладів харчування за**
смаковими вподобаннями користувача

керівник роботи **Тихоход Володимир Олександрович, к.т.н., доцент**

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__» _____ 202__ р. № _____

2. Термін подання роботи студентом **07.06.2024 року**

3. Вихідні дані до роботи мова програмування Java, фреймворки Spring, Vaadin та
Lombok, СКБД MySQL, середовище розробки IntelliJ Idea

4. Перелік питань, які потрібно розробити _____

1) провести аналіз серед наявних аналогів

2) веб-систем для пошуку закладів харчування за смаковими вподобаннями
користувачів

3) аргументувати вибрані інструменти та технології для реалізації програмного
забезпечення

4) побудувати архітектуру програмного забезпечення, модель бази даних, сховища
файлів та використання прецедентів.

5) представити процес застосування веб-інтерфейсу

5. Орієнтований перелік ілюстративного матеріалу діаграми, що демонструють використання прецедентів, архітектуру системи, бази даних та сховища файлів, взаємодію користувачів із системою, приклади роботи з програмним продуктом.

6. Дата видачі завдання «15» вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Затвердження теми роботи	17.05.24	
2.	Вивчення та аналіз задачі	17-20.04.24	
3.	Розробка архітектури та загальної структури системи	21-25.04.24	
4.	Розробка частин окремих підсистем	25.04-02.05.24	
5.	Програмна реалізація системи	03-07.05.24	
6.	Оформлення пояснювальної записки	08-12.05.24	
7.	Захист програмного продукту	13-17.05.24	
8.	Передзахист	03-06.06.24	
9.	Подання готової роботи на кафедру	07.06.2024	
10.	Захист	17-21.06.2024	

Студент

(підпис)

Ірина ГУРИН

(ім'я, ПРІЗВИЩЕ)

Керівник роботи

(підпис)

Володимир ТИХОХОД

(ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота виконана на 67 сторінках, містить 46 ілюстрацій, 2 додатки та 13 джерел в переліку посилань.

Мета роботи - розробка сервісу для пошуку закладів харчування, який дозволить гнучко налаштовувати параметри повнотекстового пошуку та забезпечувати швидке отримання ранжованих результатів.

Методи та засоби: мова програмування Java, середовище розробки IntelliJ IDEA, фреймворки Spring, Vaadin та Lombok, система управління базами даних MySQL, прикладні програмні інтерфейси Places API, Routes API та Maps API.

Результат – веб-застосунок для пошуку закладів харчування за персональними смаковими вподобаннями користувача.

Роботу було апробовано на конференції «XXI Міжнародна науково-практична конференція молодих вчених та студентів “Сучасні проблеми наукового забезпечення енергетики”» [1].

Ключові слова: ВЕБ-ДОДАТОК, JAVA, ЗАКЛАД ХАРЧУВАННЯ, СМАКОВІ ВПОДОБАННЯ.

ABSTRACT

The thesis consists of 67 pages, 46 illustrations, 2 appendixes and 13 references.

The purpose of the work is to develop a service for searching for catering establishments that will allow you to flexibly configure full-text search parameters and provide fast, ranked results.

Methods and tools: Java programming language, IntelliJ IDEA development environment, Spring, Vaadin and Lombok frameworks, MySQL database management system, Places API, Routes API and Maps API application programming interfaces.

The result is a web application for searching for restaurants according to the user's personal taste preferences.

The work was tested at the conference «XXI Міжнародна науково-практична конференція молодих вчених та студентів “Сучасні проблеми наукового забезпечення енергетики”».

Keywords: WEB APPLICATION, JAVA, CATERING ESTABLISHMENT, TASTE PREFERENCES.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ.....	7
ВСТУП.....	8
1 ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ СИСТЕМИ ПОШУКУ ЗАКЛАДІВ ХАРЧУВАННЯ ЗА СМАКОВИМИ ВПОДОБАННЯМИ КОРИСТУВАЧА.....	10
2 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ДЛЯ ПОШУКУ ЗАКЛАДІВ ХАРЧУВАННЯ.....	12
3 ВИБІР ЗАСОБІВ ДЛЯ РОЗРОБКИ СИСТЕМИ.....	21
3.1 Вибір мови програмування.....	21
3.2 Вибір середовища розробки	23
3.3 Вибір фреймворків та бібліотек.....	24
3.4 Вибір бази даних.....	26
3.5 Вибір технології для повнотекстового пошуку.....	29
4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ.....	30
4.1 Моделювання прецедентів (Use case View).....	30
4.2 Представлення процесів	31
4.3 Архітектура програмної системи.....	34
4.4 Структура проєкту.....	35
4.3 Опис моделі даних.....	37
5 ВЗАЄМОДІЯ КОРИСТУВАЧА З СИСТЕМОЮ.....	44
5.1 Функціональні можливості для авторизованого користувача.....	44
5.1.1 Пошук закладів харчування за смаковими вподобаннями користувача.....	45
5.1.2 Перегляд та додавання відгуків	50
5.2 Функціональні можливості для адміністратора системи.....	50
5.2.1 Додавання страв та інгредієнтів в меню	51
5.2.2 Зміна інформації про страви на інгредієнти.....	55
5.2.3 Видалення інформації про страви на інгредієнти.....	56
ВИСНОВКИ.....	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	59
ДОДАТОК А.....	61
ДОДАТОК Б.....	67

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

СКБД (Система Керування Базами Даних) – це програмне забезпечення для створення та керування базами даних.

БД (База Даних) – це колекція даних, що організована для зберігання, пошуку, зміни та видалення інформації.

MySQL – реляційна система керування базами даних, що базується на мові структурованих запитів.

Java – це високорівнева об'єктно-орієнтована мова програмування, що заснована на класах. Вона розроблена таким чином, щоб містити якомога менше залежностей реалізації.

Spring – це фреймворк розробки програмного забезпечення на Java з відкритим кодом.

API (Application Programming Interface) – це набір правил та протоколів, що дозволяють програмам спілкуватися між собою для обміну даними та функціями.

JSON (JavaScript Object Notation) – це формат для зберігання та транспортування даних.

HTTP (HyperText Transfer Protocol) - це протокол для обміну інформацією через інтернет.

DTO (Data Transfer Object) – це об'єкт для передачі даних, найчастіше використовується для передачі інформації в інтерфейс користувача.

ВСТУП

Сервіси пошуку інформації відіграють ключову роль у сучасному цифровому суспільстві, забезпечуючи швидкий доступ до великого обсягу даних. Вони допомагають користувачам ефективно знаходити необхідну інформацію, покращуючи продуктивність і зручність у повсякденному житті. Завдяки розвитку технологій ці сервіси стають все більш персоналізованими, задовольняючи індивідуальні потреби кожного користувача.

Все більшу популярність серед населення на сучасному етапі набуває здорове харчування, тому заклади харчування можуть привабити таку категорію клієнтів, запропонувавши меню з корисними стравами. Але пошук клієнтом закладів, що повинні задовольнити його вподобання, викликає складнощі через недосконалість механізмів та засобів пошуку. Важливим критерієм під час пошуку закладу громадського харчування є різноманітність характеристик і особливостей, які допомагають користувачам знайти ідеальний заклад, що відповідає їхнім потребам. Ці критерії включають смакові вподобання, що дозволяють знайти заклади з бажаною кухнею, можливість пошуку в меню, наявність повнотекстового і комбінованого пошуку, пошуку за ключовими словами у відгуках та вибір закладів з бажаним рейтингом та розташуванням.

Існують різні онлайн сервіси, що пропонують вибір закладів харчування, проте більшість з них спеціалізуються на додаткових послугах, таких як бронювання столика в обраному закладі, замовлення страв та доставки. Вони зазвичай надають базові параметри пошуку, що враховують загальні категорії та відгуки користувачів. Однак жоден з цих сервісів не надає повноцінний персоналізований пошук за вподобаннями користувача, що враховує такі специфічні параметри як кулінарні вподобання клієнта, його смакові переваги, культурні традиції, дієтичні обмеження тощо.

Саме тому, метою роботи є розробка сервісу пошуку закладів харчування, що забезпечуватиме користувачів унікальними можливостями пошуку. Система повинна надавати можливість гнучкого регулювання пошукових параметрів та здійснюватиме швидке отримання ранжованих результатів.

Розроблений програмний продукт враховує персональні смакові вподобання користувачів, містить великий вибір категорій, фільтрів, а також надає можливість комбінованого та повнотекстового пошуку.

Результати роботи було обговорено на конференції «XXI Міжнародна науково-практична конференція молодих вчених та студентів “Сучасні проблеми наукового забезпечення енергетики”» [1].

Структура кваліфікаційної роботи включає перелік скорочень, умовних позначень та термінів, вступ, п'ять основних розділів, які описують постановку задачі створення програми, аналіз аналогічних систем, вибір засобів для розробки, проектування системи, приклади взаємодії користувача з програмним продуктом, висновки та список використаних джерел. Загальний обсяг даної роботи становить 67 сторінок, що містить 47 ілюстрацій, 2 додатки та 13 джерел в переліку посилань.

1 ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ СИСТЕМИ ПОШУКУ ЗАКЛАДІВ ХАРЧУВАННЯ ЗА СМАКОВИМИ ВПОДОБАННЯМИ КОРИСТУВАЧА

Метою роботи є створення сервісу пошуку закладів харчування, що надаватиме можливість гнучкого регулювання пошукових параметрів повнотекстового пошуку та здійснюватиме швидке отримання ранжованих результатів пошуку

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- провести дослідження щодо порівняння функціональності подібних систем, визначення їх переваг та недоліків;
- провести дослідження щодо найбільш популярних пошукових запитів, визначити набір критеріїв, структуру запитів та модель проблемної області;
- провести дослідження щодо організації оптимальних налаштувань індексування в системі Elasticsearch, що підтримуватиме повнотекстовий пошук, пошук за ключовими словами, підтримки української та англійських пошукових термів.

До вхідних даних системи входять дані про заклади харчування, що буде отримано з Places API.

Вихідними даними системи є отримані результати пошуку.

До основних функціональних вимог слід віднести:

- система повинна надавати гнучкі параметри повнотекстового пошуку;
- система повинна здійснювати пошук в українській та англійській морфології;
- система повинна надавати можливість зручного комбінованого пошуку за різними критеріями, основними з яких є:
 - за географічною віддаленістю;
 - за тривалістю шляху;
 - за типом кухні;
 - за стравами меню;

- за інгредієнтами;
- за ключовими словами у відгуках;
- за рейтингом.

Розроблена система буде мати назву FoodieFinds.

Потенційними користувачами цього застосунку можуть бути люди будь-якого віку, соціального статусу та рівня доходу, які люблять смачно поїсти та зацікавлені у пошуку закладу харчування відповідно до своїх індивідуальних вподобань.

2 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ДЛЯ ПОШУКУ ЗАКЛАДІВ ХАРЧУВАННЯ

Одним з важливих етапів перед розробкою продукту є вивчення наявних аналогів пошуку закладів харчування за смаковими вподобаннями користувача. Перш за все, це відбувається з метою створення конкурентоздатного та унікального застосунку, що містить необхідний, раніше не присутній функціонал та не має недоопрацьованостей.

У даному розділі буде розглянуто три аналоги, що частково схожі за функціоналом та ідеєю з розробленим веб-застосунком.

2.1 Glovo

Glovo — це платформа, яка пропонує широкий спектр послуг доставки. В основному вона зосереджена на доставці продуктів і товарів повсякденного попиту з різних магазинів, ліків і продуктів з багатьох ресторанів і кафе, а також включає варіанти доставки додому. Продукт працює в більш ніж 20 країнах світу, в тому числі і в Україні [2].

Щоб скористатися системою, необхідно відвідати офіційний сайт компанії або завантажити додаток на смартфон. Веб-додатком може користуватися як неавторизовані, так і авторизовані користувачі. Однак слід зазначити, що функціонал для першого випадку буде обмежений лише переглядом різних пропозицій, без оформлення замовлення.

Після переходу на домашню сторінку продукту користувачеві пропонується вибір доступних послуг, які надає компанія (рис. 2.1). Крім того, у цьому розділі можна вибрати бажане місце доставки та отримати доступ до пропозицій щодо потенційних угод.

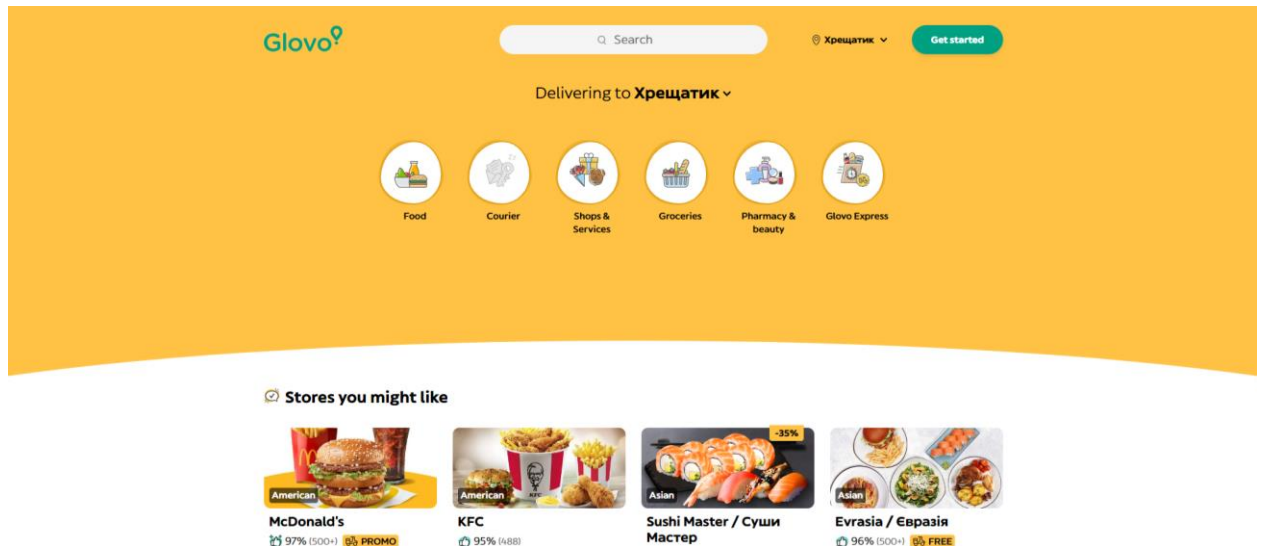


Рисунок 2.1 – Головна сторінка

Перейшовши на сторінку доставки з ресторанів (рис. 2.2), можна помітити широкий спектр фільтрів та можливостей, що надані для пошуку найкращого місця для харчування. Серед них можна виділити основні категорії для сортування:

- за рейтингом: ресторани сортуються на основі рейтингів користувачів;
- за розташуванням (біля мене): відображаються ресторани, що знаходяться найближче до місцезнаходження користувача;
- за безплатною доставкою: відображаються лише ресторани з безплатною доставкою.

Також можна здійснювати фільтрацію за категоріями:

- за популярними запитамі: відображає ресторани, які найбільше користуються популярністю в користувачів;
- за кухнями: фільтрує ресторани за приналежністю до певної кухні. Наприклад: азійська, європейська, українська, веганська тощо;
- за деякими стравами: фільтрує ресторани за наявністю деяких страв, таких як піца, морепродукти, шаурма тощо.

Окрім цього, для кожного закладу харчування, також містяться його особисті фільтри, які обробляють запити лише відносно обраного місця та його страв.

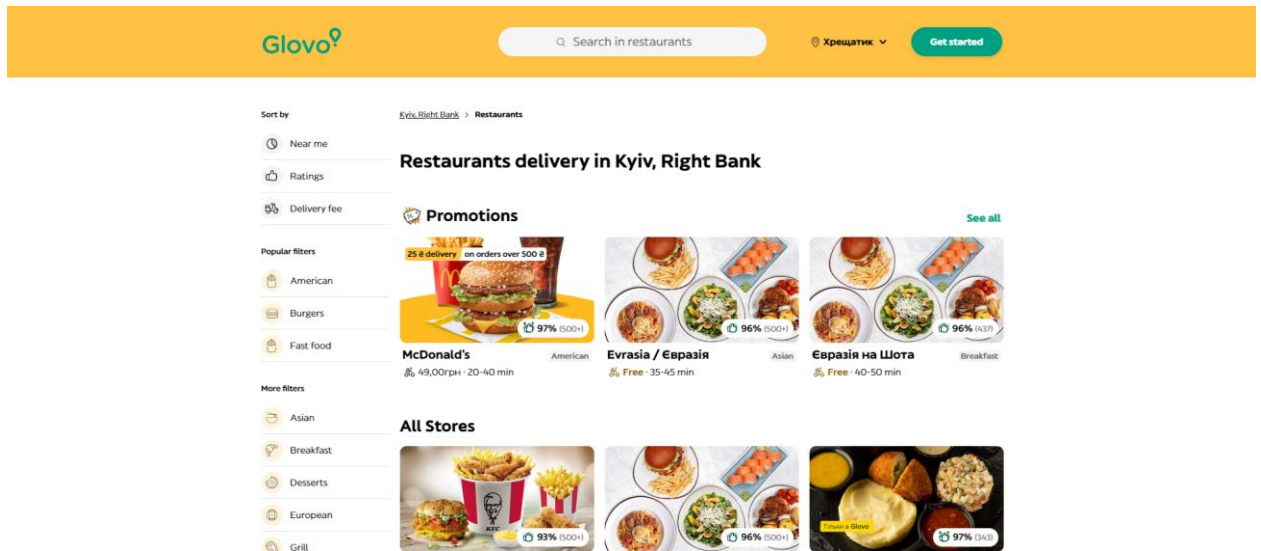


Рисунок 2.2 – Сервіс доставки із закладів харчування

Однак, даний сервіс має низку недоліків, які роблять пошук закладів харчування проблематичним та не персоналізованим. Як було зазначено вище, даний сервіс надає можливість пошуку за деякими категоріями, проте їх насправді дуже мало і вони є більш загальними. Наприклад, у випадку, якщо користувач хоче знайти заклади з найнижчим рейтингом, у нього не буде можливості цього зробити, тому що сервіс надає можливість пошуку лише закладів з найкращими результатами. Ще одним недоліком є відсутність групування фільтрів за їх категоріями. Всі кухні, страви та їх типи змішані між собою, що ускладнює вибір необхідних характеристик. З цього і випливає ще одна проблема, а саме відсутність комбінації різних опцій для отримання більш точних результатів пошуку.

Отже, головною перевагою даного додатка є його великий спектр способів доставки, що охоплюють в собі широкий діапазон закладів харчування, магазинів, аптек, тощо. Додаток містить функціональність для обох неавторизованих та авторизованих користувачів, що надає людині можливість оцінити функціонал продукту та якщо є охота (бажання) скористатися ним. Проте, сервіс не надає персоналізований пошук, адже хоч і містить різні фільтри, вони не містять широкий вибір опцій та можливість комбінації між собою для отримання закладів, що відповідатимуть індивідуальним вподобанням користувача.

2.2 Trip Advisor

Trip Advisor – веб-сервіс для планування та організації подорожей у будь-яку країну світу. Це ідеальний додаток, який дозволяє вибрати все, від готелю до місць, які варто побачити в тому чи іншому регіоні (рис. 2.3).

Як і його попередник, цей продукт, окрім веб-застосунка, має додатки для Android та IOS, які дозволяють користувачам зручніше взаємодіяти з представленим функціоналом. Така ж ситуація з розподілом ролей. Неавторизовані користувачі можуть переглядати пропозиції та фільтрувати їх на основі особистих уподобань, проте у них немає можливостей додавати бажані місця в обране або будь-яким чином взаємодіяти з окремими закладами [3].

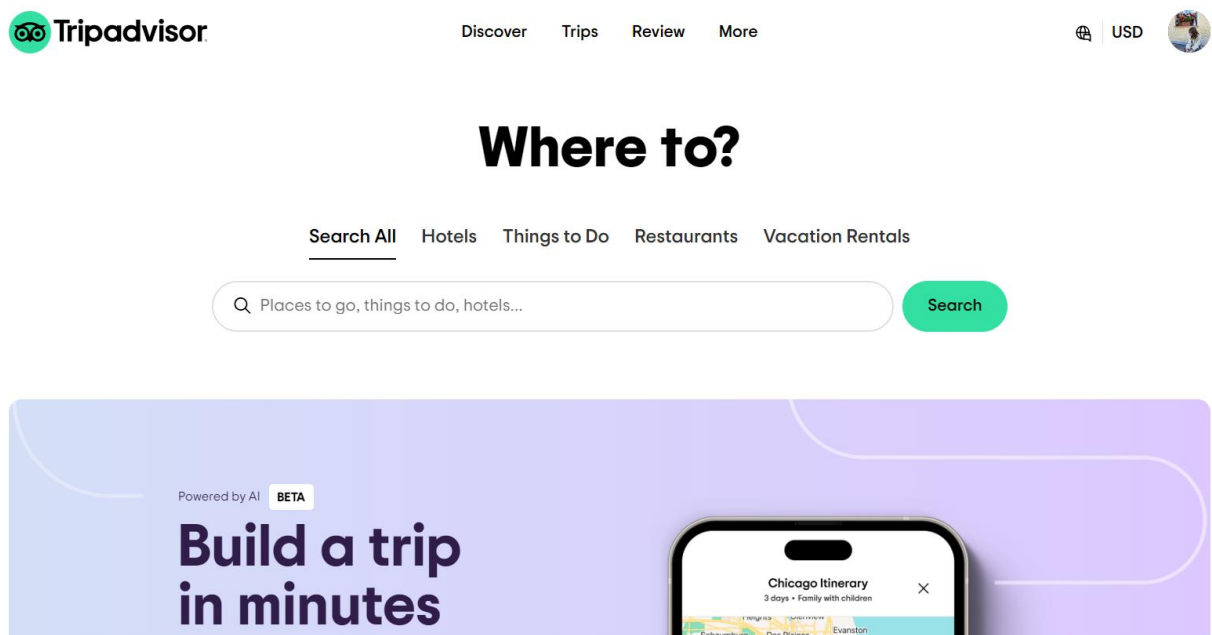


Рисунок 2.3 – Головна сторінка

Окрім цього, даний продукт також містить велику базу закладів харчування і надає користувачам можливість вибору ресторанів, які вони хочуть відвідати. Тут присутні такі категорії для вибору (рис. 2.4):

- тип закладу, наприклад: ресторан, кафе тощо;
- тип харчування, наприклад: сніданок, обід, вечеря, інші прийоми їжі;

- онлайн-опції та пропозиції, а саме можливість замовлення онлайн та спеціальні пропозиції і знижки;
- ціна (діапазон цін, що підходить бюджету користувачу);
- рейтинг, за оцінкою користувачів системи;
- кухня, наприклад: азійська, європейська, українська, веганська тощо;
- страви, наприклад: піца, морепродукти, шаурма тощо;
- дієтичні обмеження, наприклад: безглютенові або веганські страви;
- особливості закладу, мається на увазі наявність особливих сервісів;
- орієнтація на певні групи, наприклад сім'я з дітьми, великі групи, побачення тощо;
- визначення місцезнаходження закладу за деякими районами або околицями.

Важливою особливістю цього продукту є також те, що він містить вбудовану систему відгуків. Тобто, кожен авторизований в системі користувач має можливість поділитися своїми враженнями з іншими людьми та поставити власну оцінку відвіданому місцю.

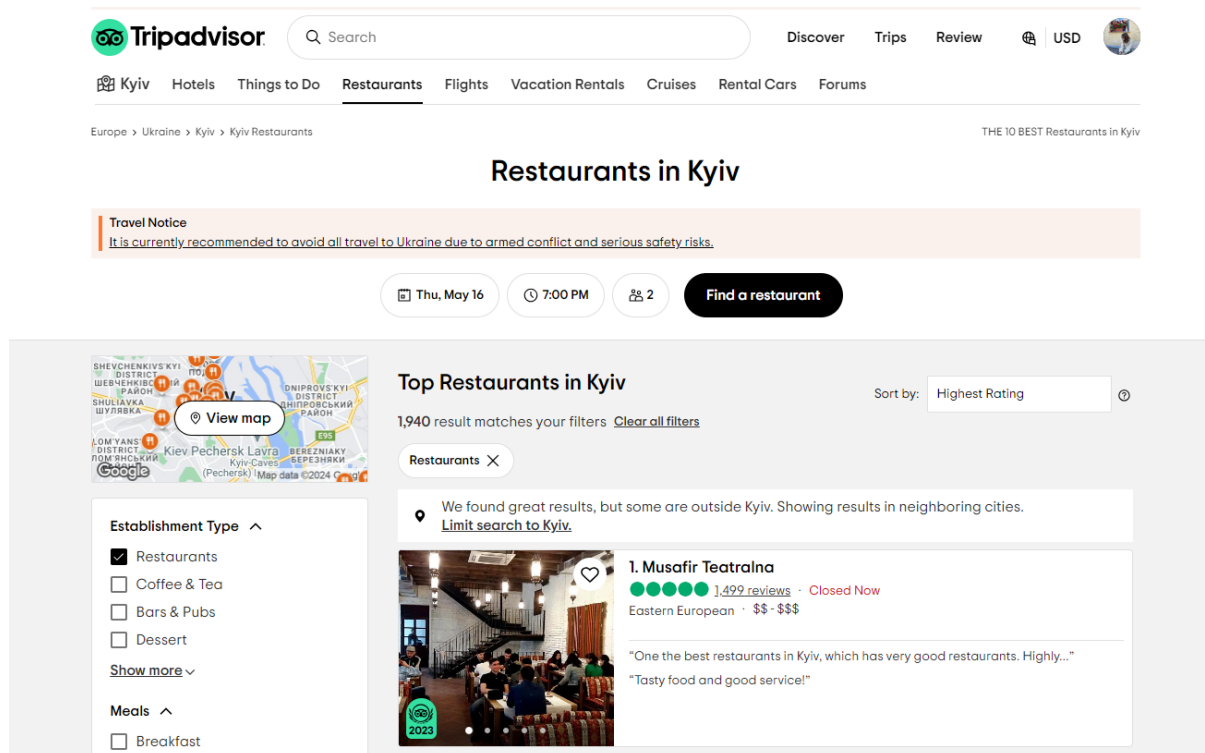


Рисунок 2.4 – Сторінка пошуку закладу харчування

Даний сервіс надає широкі можливості пошуку закладів харчування за смаковими вподобаннями користувача, проте він не враховує повною мірою пошук відносно розташування. Як було зазначено вище, з його допомогою можна переглянути заклади, що знаходяться в околицях міста, проте він не враховує на якій відстані даний ресторан знаходиться від користувача, або скільки часу потрібно для того, щоб добратися до нього.

Отже, даний додаток має великий функціонал, який містить пошук ресторанів за деякими критеріями. Тут важливо зазначити, що сервіс дозволяє комбінувати бажані фільтри, для отримання більш точних бажаних результатів. Продукт надає можливість користуватися загальними функціями вибору та пошуку неавторизованим та авторизованим користувачам. Однак, даний сервіс не надає широкий спектр пошуку закладів відносно місцезнаходження користувача, або розташування самого місця.

2.3 Open Table

Онлайн-сервіс, який спеціалізується на послугах онлайн-бронювання столиків у 80 країнах світу. Завдяки їх співпраці з великою кількістю закладів харчування, користувачі системи можуть з легкістю обрати місце та зручний для них час відвідування [4].

Застосунком можна користуватися як у веб-браузер, так і завантаживши додаток на мобільний пристрій. Система доступна усім користувачам, проте у випадку, якщо людина хоче оформити бронювання, їй необхідно авторизуватися в додатку для надання своїх даних.

На сторінці бронювання місць в закладах харчування (рис. 2.5), користувачі можуть відфільтрувати об'єкти за допомогою багатьох фільтрів та категорій. Серед них є такі:

— варіанти харчування, наприклад: в закладі та можливість виносу;

- варіанти місць для сидіння, наприклад: бар, серед відкритого неба, стандартний тощо;
- ціна (діапазон цін, що підходить бюджету користувачу);
- пошук закладу за околицею або регіоном;
- кухня, наприклад: азійська, європейська, українська, веганська тощо;
- фільтрація ресторанів за популярними словами, що присутні в відгуках користувачів;
- доступність для людей з особливими потребами;
- повнотекстовий пошук;
- пошук за ключовими словами у відгуках.

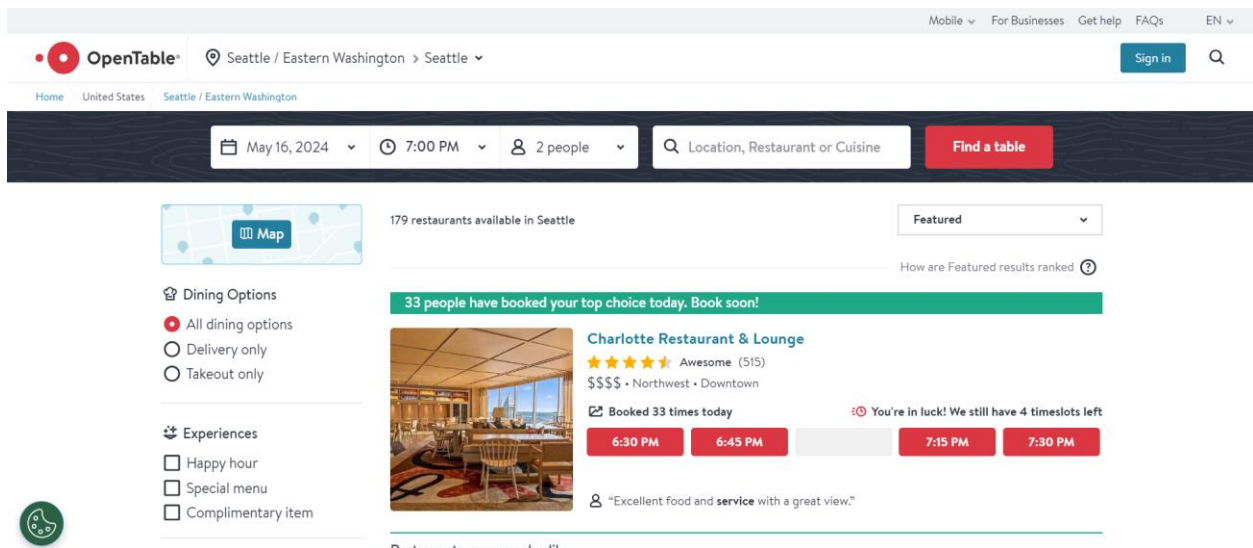


Рисунок 2.5 – Сторінка бронювання столиків в закладах харчування

Окрім вибору місця, користувачі також можуть залишити свої відгуки та поставити рейтинг відвіданим ресторанам. Завдяки цьому, користувачі пізніше зможуть шукати серед цих відгуків ті, що містять слова, які цікавлять користувачів.

Проте, так само, як його попередник, цей сервіс обмежений у виборі закладів за місцем розташування. Він не має можливості враховувати відстань та час переміщення при пошуку.

Отже, даний застосунок є прекрасним сервісом бронювання столиків, де люди можуть знайти місце для харчування і насолодитися поведеним там часом.

Система містить велику базу закладів та користувачів, які можуть ділитися своїми враженнями через відгуки та оцінки. Єдиним його недоліком є відсутність персоналізованого вибору, що враховуватиме місце розташування закладу та користувача.

2.4 Результат порівняння

Після дослідження аналогічних систем можна зробити висновок, що всі вони відрізняються один від одного додатковим функціоналом та містять базові фільтри для пошуку закладів харчування. Проте не всі з них надають можливість вибору закладу який буде точно відповідати індивідуальним смаковим вподобанням користувача. Для зручного порівняння розглянутих систем створимо таблицю, де буде відзначено наявність або відсутність певних функціональних можливостей у кожній з них.

Таблиця 2.1 – Результати порівнянь аналогів

	Glovo	Trip Advisor	Open Table	FoodieFinds (Кваліфікаційна робота)
Пошук за кухнею	+	+	+	+
Пошук в меню	+	+	-	+
Повнотекстовий пошук	+	+	+	+
Комбінований пошук	-	+	+	+
Пошук за ключовими словами у відгуках	-	+	+	+
Пошук за рейтингом	+-	+	+	+
Пошук за розташуванням	+-	+-	+-	+

Проаналізувавши порівняльну таблицю можна зробити висновок, що розроблена система, охоплюватиме весь функціонал для пошуку закладів харчування за індивідуальними вподобаннями користувача. Вона надаватиме розширені можливості для фільтрації та вибору закладів відповідно до індивідуальних потреб і смакових уподобань, що є конкурентною перевагою на ринку ресторанних послуг.

3 ВИБІР ЗАСОБІВ ДЛЯ РОЗРОБКИ СИСТЕМИ

Після визначення основних вимог до системи та аналізу наявних рішень важливим кроком є вибір інструментів для її розробки. Правильний їх вибір є ключовим фактором ефективності, надійності та масштабованості майбутніх систем

У цьому розділі йдеться про розгляд та обґрунтування вибору технологій, мови програмування, середовища розробки, фреймворків, бібліотек та БД, які найкраще відповідають встановленим цілям і завданням розроблюваної системи.

3.1 Вибір мови програмування

Існує багато мов програмування, що зосереджені на розробці веб-додатків. Серед них можна виділити такі як: JavaScript, Python, Ruby, PHP, C#, Java тощо. Для розробки даної системи необхідно обрано об'єктно-орієнтовану мову програмування, що буде виділятися своєю надійністю та простотою.

Серед усіх згаданих мов, що використовуються в розробці веб-додатків, варто виділити Java. За статистичними показниками [5] мова Java входить в п'ятірку мов, що найчастіше використовуються для веб-розробки (рис. 3.1). Це об'єктно-орієнтована мова програмування, що відома своєю простотою, надійністю та функціями безпеки [6].

Однією з переваг Java для нашої системи є наявність великої кількості бібліотек і фреймворків, які значно спрощують розробку необхідного функціоналу. Ці інструменти дозволяють не лише ефективно реалізовувати складні завдання, але й скорочуючи час і зусилля, необхідні для створення додатків. Більш детально про їх використання буде згадано в наступних підрозділах.

Web Languages-What language are most commonly used for web development

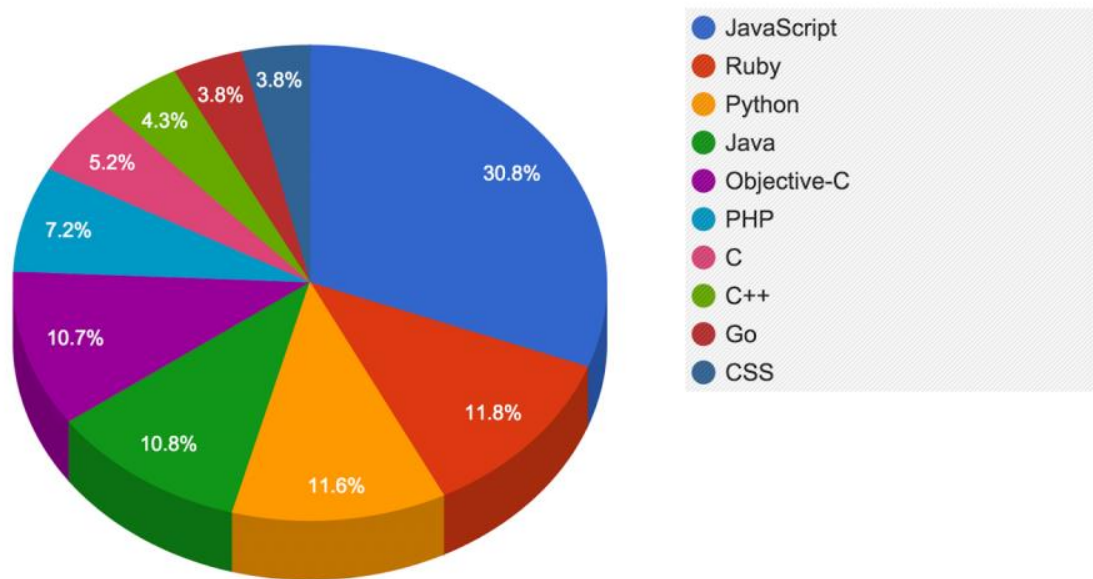


Рисунок 3.1 - Мови програмування, що найчастіше використовуються для веб-розробки

Розглянемо ключові можливості Java, що відіграють важливу роль в розробці системи пошуку закладів харчування за смаковими вподобаннями користувача.

1. Незалежність від платформи. Компілятор перетворює вихідний код у байт-код, що виконується JVM. Це дозволяє запускати програми на будь-якій платформі (Windows, Linux, macOS) з однаковими результатами.

2. Об'єктно-орієнтована мова програмування. Використовує 4 основні концепції ООП, а саме: абстракцію, інкапсуляцію, успадкування та поліморфізм.

3. Простота. Не містить складних функцій, таких як перевантаження операторів, множинне успадкування та явний розподіл пам'яті, що можуть ускладнити розробку додатку.

4. Надійність. Такі особливості як: раннє виявлення помилок, збір сміття, обробка винятків та ефективний розподіл пам'яті сприяють стабільності та надійності системи в процесі її роботи.;

5. Безпека. Відсутність вказівників у Java дозволяє уникнути вразливостей, таких як пошкодження стека або переповнення буфера. Окрім цього, Java-програми

працюють в середовищі, яке не залежить від операційної системи, що підвищує їхню безпеку.

3.2 Вибір середовища розробки

При виборі середовища розробки важливо покладатися на його зручність на його зручність та функціональні можливості, для забезпечення продуктивної роботи. Окрім цього важливо враховувати сумісність середовища з потрібними технологіями та інтеграцію з іншими інструментами розробки.

Згідно з Звітом про продуктивність розробників Java за 2024 рік [7], IntelliJ IDEA визнано найпопулярнішим Java IDE цього року, займаючи 41% ринку. Eclipse займає друге місце із 23%, тоді як Microsoft VSCode та Visual Studio Code посідають третє місце з рейтингом 19% (рис.3.2).

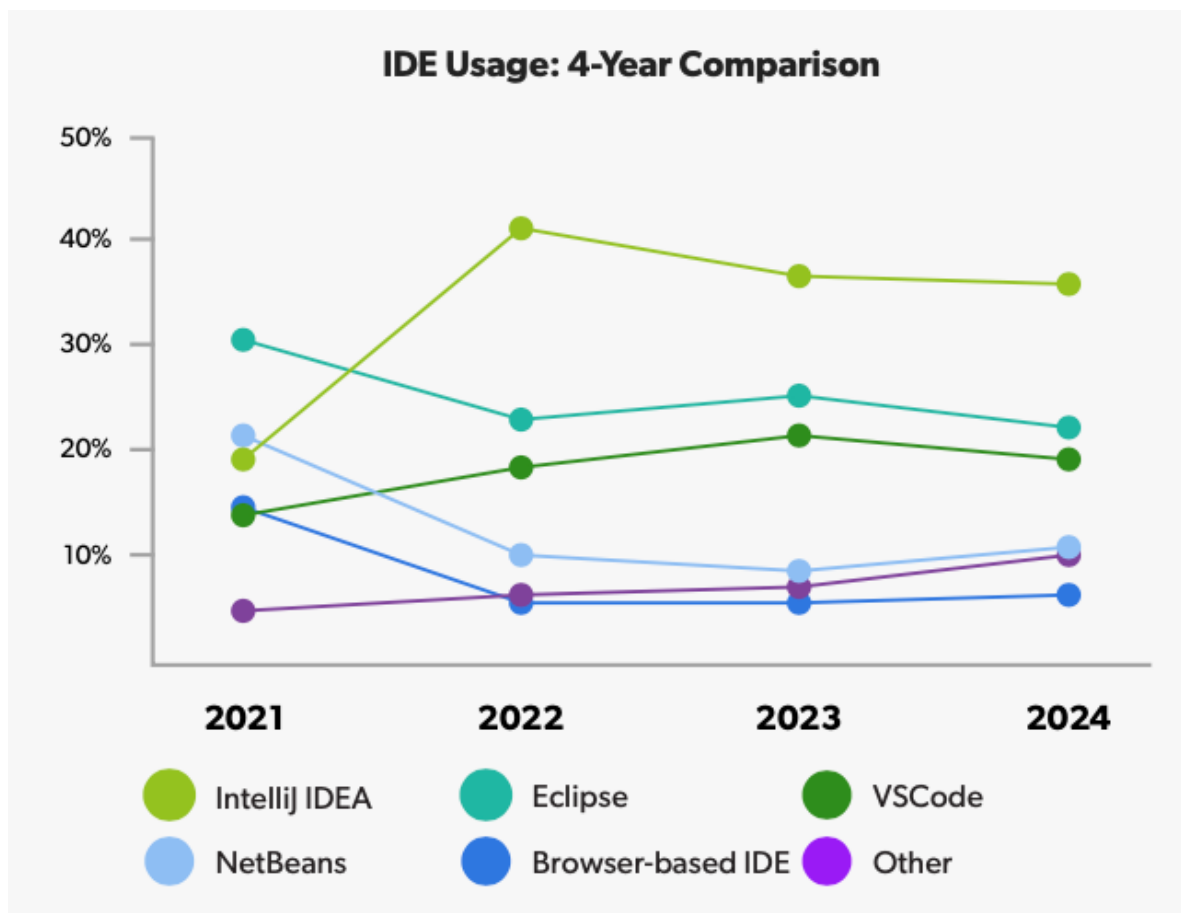


Рисунок 3.2 - Найпопулярніші Java IDE 2024 року

У виборі середовища розробки було віддано перевагу IntelliJ IDEA, оскільки воно не тільки забезпечує зручне написання коду, але й є потужним інструментом, який допомагає розробникам у реалізації їхніх програм та рішень.

Найголовнішою перевагою даної платформи над усіма конкурентами є те, що вона пропонує користувачу велике різноманіття базових та розширених функцій [8], таких як:

- виявлення помилок в реальному часі;
- інтелектуальне автозавершення;
- статичний аналіз коду;
- швидке створення коду;
- вбудовані інструменти контролю версій;
- рефакторинг;
- вбудовані інструменти тестування.

Крім того, варто відзначити, що це середовище забезпечує зручну взаємодію між усіма компонентами розробленої системи, наприклад інтеграцію з базами даних, підключення Docker та роботу з Github. Ця програма не лише спрощує взаємодію з цими компонентами, але й допомагає у їх підключенні та подальшій роботі з ними.

3.3 Вибір фреймворків та бібліотек

Під час реалізації даної системи буде використано два фреймворки, а саме Spring та Vaadin. За результатами досліджень [9] дані фреймворки входять в топ трьох найпопулярніших Java фреймворків (рис. 3.3) та забезпечують швидку і ефективну розробку, пропонуючи високий рівень функціональності та зручний інтерфейс.

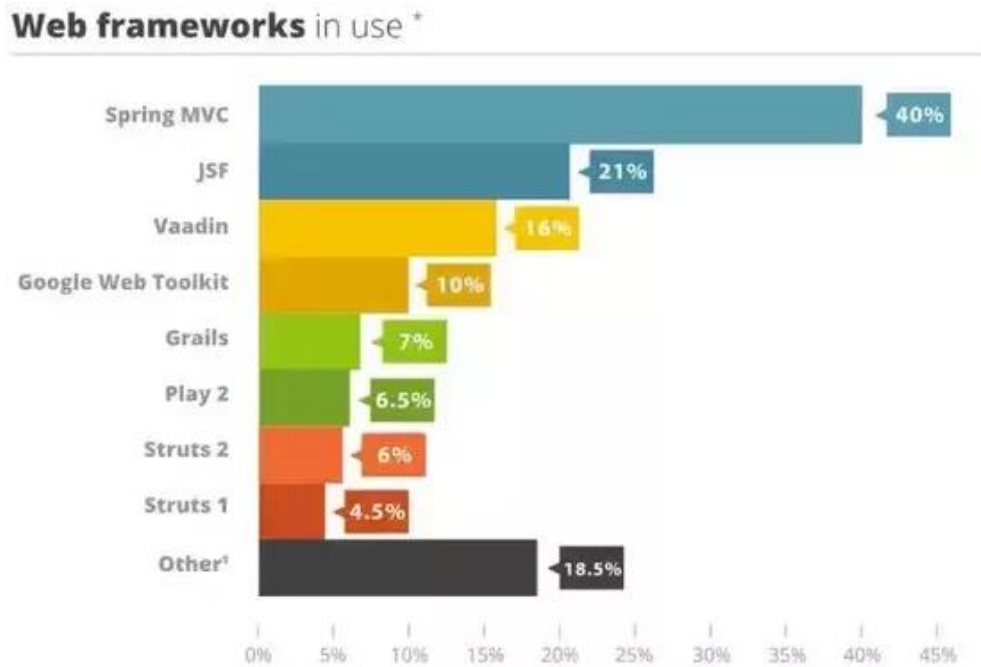


Рисунок 3.3 – Найпопулярніші Java фреймворки для розробки веб-додатків

Spring – це фреймворк з відкритим кодом для платформи Java. Він ґрунтується на принципах впровадження залежностей і аспектно-орієнтованого програмування. Об'єктам надаються ресурси та необхідні об'єкти за допомогою впровадження залежностей, що усуває необхідність в їх власному пошуку. Spring Framework пропонує ряд розширень і функцій, таких як Spring Boot, Spring MVC, Spring Security, Spring Data і Spring Cloud [10]. Даний фреймворк було обрано завдяки його можливості полегшувати розробку та написання коду, а також його вбудованих функціях і розширеннях, які було згадано раніше.

Vaadin – це потужна платформа веб-розробки для створення користувацького інтерфейсу на Java без необхідності у використанні HTML, CSS та JS. Вона надає зручний інтерфейс для роботи з веб-компонентами та різними макетами, що дозволяє розробникам швидко створювати стильні та функціональні веб-додатки. Даний фреймворк відіграє важливу роль в розроблюваному додатку, адже саме він відповідає за створення користувацьких інтерфейсів. Великою його перевагою є те, що для написання клієнтської частини програми потрібно використовувати Java, так само як і для серверної частини. Окрім цього тут існує дві вбудовані теми: Lumo і Material, які містять уже готові стилі для всіх елементів сторінки.

Також однією з важливих використаних бібліотекою під час розробки є Lombok. З її допомогою можна згенерувати сетери, гетери, методи toString(), конструктори та багато інших. Дана бібліотека є незамінною під час розробки системи, адже вона спрощує розробку програми, самостійно генеруючи методи, які вимагають написання великої частини коду. Вона використовує різні анотації, що якраз і спрямовані на заміну шаблонного Java коду.

Окрім цього під час розробки даної системи буде використано такі клієнтські бібліотеки для Google Maps Services як Places API та Routes API. Вони надають та полегшують пошук закладів харчування, а також визначають відстань та часовий проміжок для подолання маршруту між заданими місцями.

3.4 Вибір бази даних

Головною складовою розроблюваної системи є база даних, в якій будуть зберігатися всі необхідні дані для реалізації продукту.

За результатами опитування розробників StackOverflow у 2022 році [11] найбільш використовуваними системами баз даних є PostgreSQL, MySQL та SQLite (рис. 3.4).

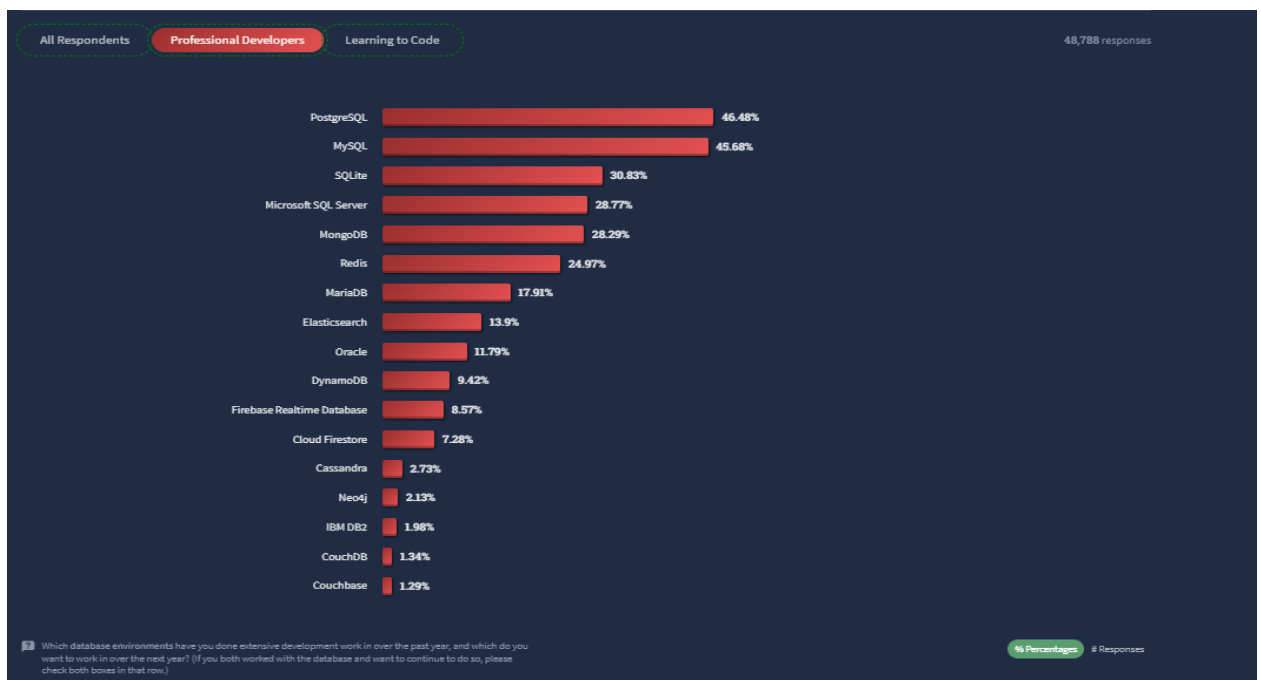


Рисунок 3.4 - Найбільш популярні системи баз даних

PostgreSQL — це потужна відкрита система керування реляційною базою даних, яка підтримує різноманітні функції, такі як складні запити SQL, транзакції, цілісність даних і реплікація. Вона часто використовується у великих проектах, де вимагається високої продуктивності та надійності.

MySQL – це друга за популярністю реляційна СУБД, яка також має відкритий код. Вона відомий своєю швидкістю та простотою використання, особливо для веб-проектів, і підтримує багато функцій, такі як транзакції та індексування [12].

SQLite — це легка вбудована реляційна СУБД, яка використовується переважно для локального зберігання даних у мобільних додатках, вбудованих системах і простих веб-сайтах. SQLite займає мало ресурсів і не потребує окремого сервера, що робить її привабливою для простих проектів і швидкого розгортання.

Для більш детального порівняння вище згаданих систем баз даних розглянемо таблицю 3.1.

Таблиця 3.1 – Порівняння систем баз даних

Назва	SQLite	MySQL	PostgreSQL
Архітектура	На основі файлів (вбудована)	Клієнт/Сервер	Клієнт/Сервер
Серверна операційна система	-	FreeBSD, Linux, OS X, Solaris, Windows	FreeBSD, HP-UX, Linux, NetBSD, OpenBSD, OS X, Solaris, Unix, Windows
Послідовність транзакцій	ACID	ACID	ACID
Мова програмування (базовий код)	C	C, C++	C

Продовження таблиці 3.1

Підтримувані мови програмування	ActionScript, Ada, Basic, C. C#, CH, D, Delphi, Forth, Fortran, Haskell, Java, JavaScript, Lisp. Lua. MATLAB, PHP, PL/SQL	C, C++, Delphi, Perl, Java, Lua, NET, Node.js	ST. C. C++, Delphi Java, Perl, PHP, Python, Tcl
Популярні випадки використання	Веб-сайти з низьким і середнім трафіком, IoT та вбудовані пристрої, тестування та розробка	Веб-сайти, веб-програми, стек LAMP, програми на основі OLTP	Аналітика, аналіз даних, сховища даних. Бізнес-аналітика, Hadoop
DBaaS	-	База даних Azure для MySQL, Amazon RDS для MySQL, Google Cloud SQL для MySQL	База даних Azure для PostgreSQL, Amazon RDS для PostgreSQL Google Cloud SQL для PostgreSQL

Проаналізувавши розглянуті дані, було обрано Mysql, як систему баз даних для системи пошуку закладів харчування за смаковими вподобаннями користувача. Дана система є ідеальним вибором для розроблюваного додатку, завдяки її зручному інтерфейсу, добре задокументованих ресурсам, а також високому рівню безпеки даних, включаючи шифрування паролів.

3.5 Вибір технології для повнотекстового пошуку

Одною з головних особливостей розроблюваної системи є можливість надання повнотекстового пошуку та пошуку за ключовими словами. Зважаючи на те, що реалізація такого функціоналу з використанням бази даних не забезпечить достатньої ефективності, планується використання спеціалізованої технології для досягнення необхідної продуктивності, швидкості та точності результатів.

Elasticsearch — це розподілена пошукова та аналітична система з відкритим вихідним кодом, створена на базі Apache Lucene. З її допомогою можна швидко зберігати, шукати та аналізувати великі обсяги даних, забезпечуючи майже миттєві відповіді. Elasticsearch використовує документно-орієнтовану структуру та надає розширені REST API для роботи з даними у форматі JSON [13].

У Elasticsearch документи є основною одиницею інформації, яка може бути індексована у форматі JSON і може представляти структуровані дані, такі як числа, рядки та дати. Кожен документ має унікальний ідентифікатор і певний тип даних, що визначає його суть.

Завдяки своїй гнучкості та потужності Elasticsearch можна застосовувати в багатьох різних сферах, адже вона має широкий спектр застосування. У випадку розроблюваної системи, було обрано дану технологію, тому що вона надає можливість здійснювати пошук у різних форматах даних. Головними можливостями пошуку, що буде використано під час розробки є:

- повнотекстовий пошук дозволяє швидко знаходити необхідну інформацію у великій кількості текстових даних: дана особливість корисна у випадках, коли необхідно знайти важливу інформацію в журналах, новинах, статтях та будь-яких інших текстових ресурсах;

- пошук за ключовими словами: можливість знаходити документи, що містять певні ключові слова або фрази.

4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

У цьому розділі буде продемонстровано програмну реалізацію розробленого застосунку. Представлені матеріали допоможуть зрозуміти загальну структуру проєкту, його функціональність та взаємодію з різними компонентами системи. Буде показано діаграми варіантів використання (Use case), процеси системи, архітектурну схему програми, схему структури проєкту та модель даних.

4.1 Моделювання прецедентів (Use case View)

Усього в даному додатку наявні 3 типи користувачів:

- гість (неавторизований користувач): для отримання доступу до основних функцій системи, необхідно зареєструватися за допомогою облікового запису Google;
- користувач (авторизований): зареєстрований гість, що може увійти в систему за допомогою попереднього доданого в систему облікового запису Googleж для нього доступний основний функціонал системи, а саме: можливість пошуку закладу харчування за допомогою різних фільтрів, використання повнотекстового пошуку, пошуку за ключовими словами, за розташуванням, перегляду інформації обраного ресторану, його відгуків та додавання нових;
- адміністратор: відповідає за додавання, редагування та видалення страв та інгредієнтів в меню ресторану.

Змоделюємо діаграму прецедентів повної інформаційної системи (рис. 4.1).

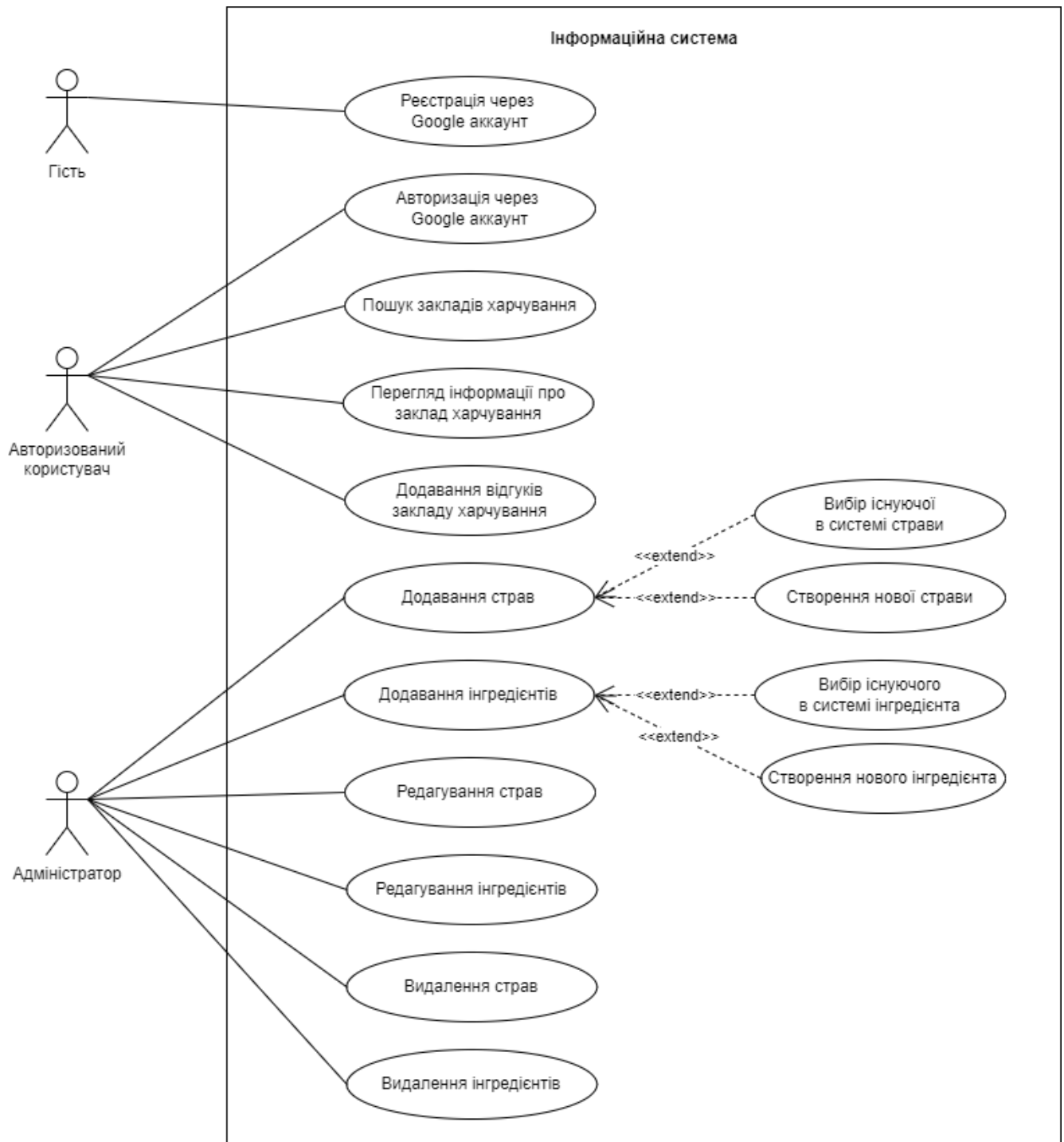


Рисунок 4.1 – Діаграма прецедентів повної інформаційної системи

4.2 Представлення процесів

На рисунку 4.2, представлено процес пошуку закладів харчування. Було розглянуто три різні варіанти пошуку за допомогою фільтрів, а саме пошук за

категорією: звичайний пошук в базі даних, розташуванням, використовуючи сервіс Routes API, а також за ключовими словами у відгуках, завдяки індексованому сховищу Elasticsearch.

Варто зазначити, що в даній послідовності не продемонстровано надання запитів в Places API, для отримання даних про ресторани, тому що при запуску програми всі необхідні дані з прикладного інтерфейсу автоматично додаються в бд та готові для майбутнього використання користувачем. Дану особливість було реалізовано спираючись на поточні вимоги, функціонал програмного продукту та можливості надані самим сервісом Places API.

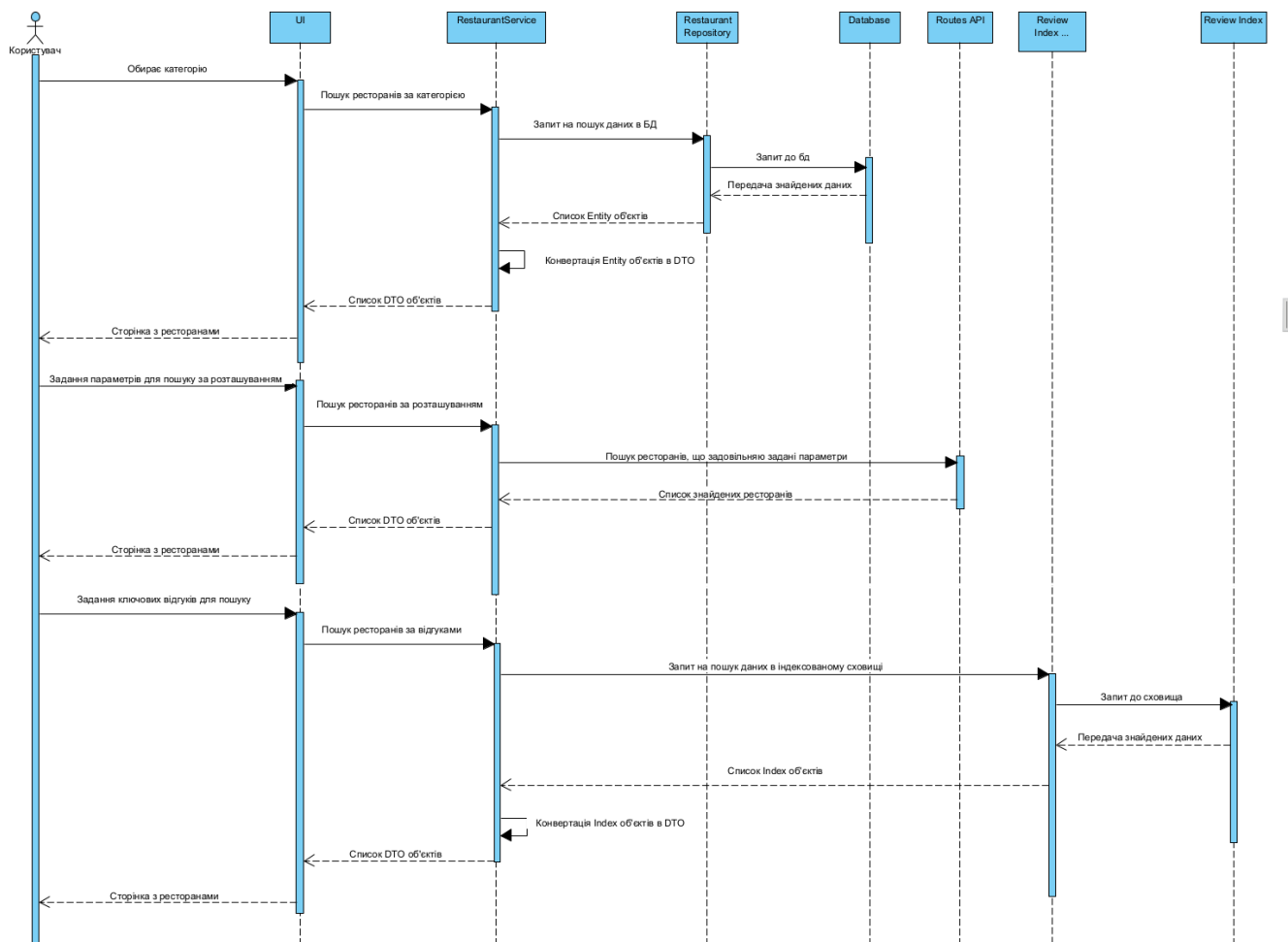


Рисунок 4.2 – Діаграма послідовностей для процесу пошуку закладів харчування за смаковими вподобаннями користувача

Одним з головних функціональних можливостей адміністратора системи є додавання страв та інгредієнтів закладам харчування. Для цього у нього є два

варіанти дій: вибір вже існуючих компонентів, страв чи інгредієнтів, або додавання нових з наданням всієї необхідної інформації. На рисунку 4.3 продемонстровано процес додавання страв та інгредієнтів з переглядом двох можливих варіантів.

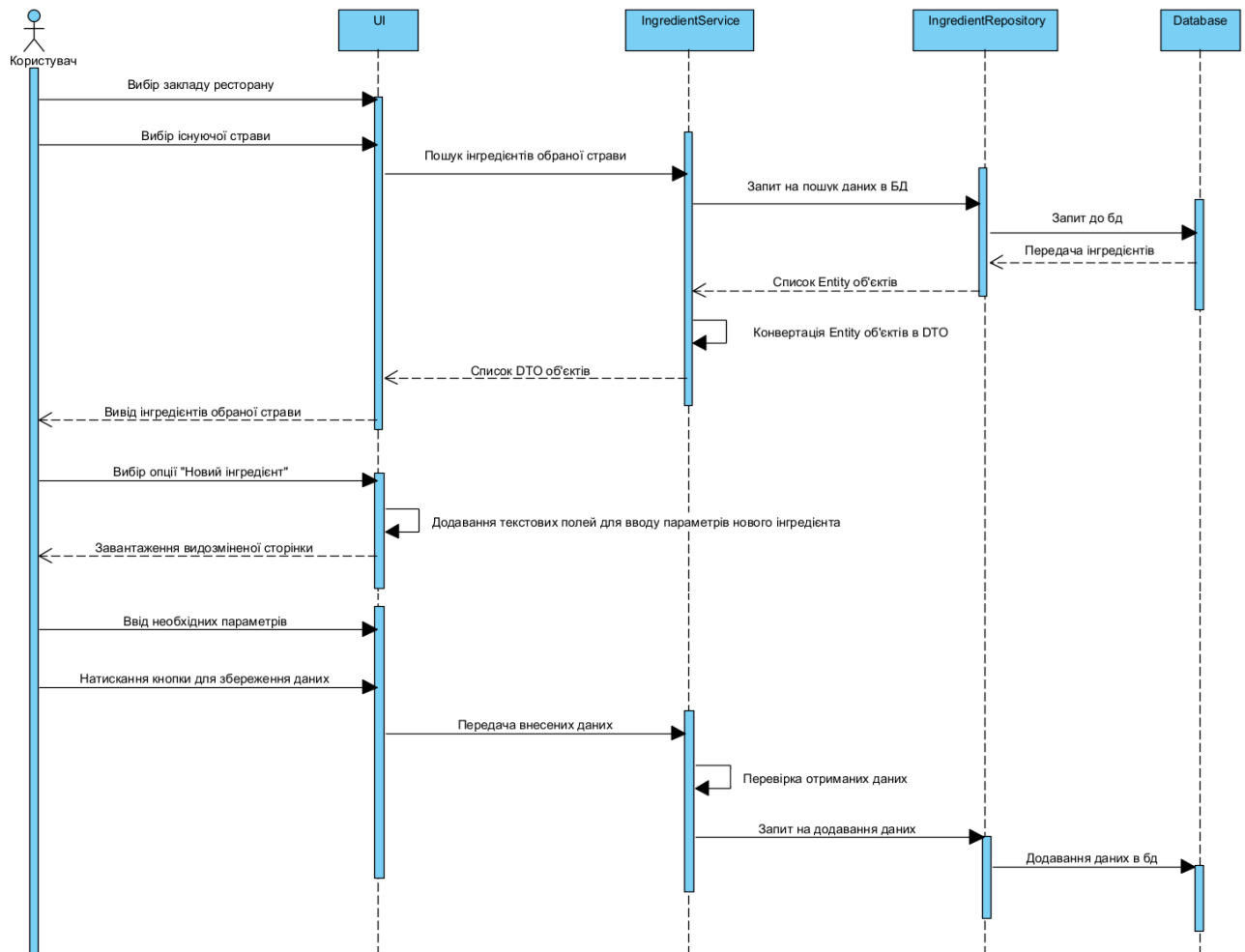


Рисунок 4.3 – Діаграма послідовностей для процесу додавання страв та інгредієнтів

Окрім можливості додавання страв та інгредієнтів, у адміністратора є також функція змінювати та видаляти дані в меню. Для детальнішого розуміння функціоналу зміни даних про страву було створено діаграму діяльності (рис. 4.4).

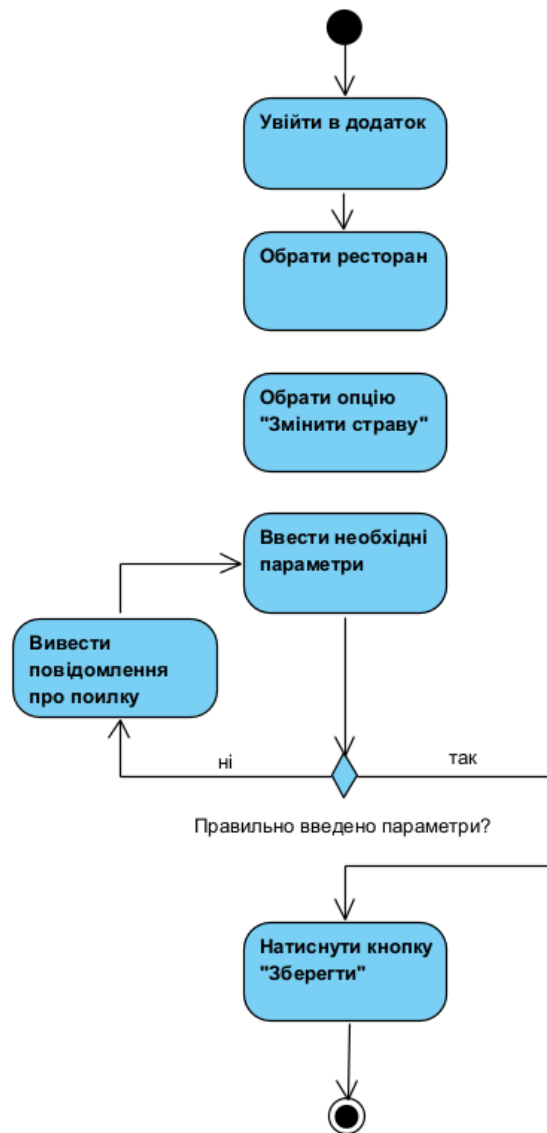


Рисунок 4.4 – Діаграма активності зміни даних про страву

4.3 Архітектура програмної системи

Основними компонентами системи (рис. 4.4) є: Rest API — веб-служби; Foodie Finds — багатосторінковий веб-застосунок; Google Maps — картографічний сервіс; Places API — інтерфейс програмування, що надає можливість інтегрувати інформацію про місця (наприклад, ресторани, кафе, готелі тощо); Routes API - Агент Google Карти API — компонент взаємодії з Places API та Routes API - це інтерфейс програмування, який надає можливість взаємодії з інформацією про

маршрути; сховище Elasticsearch — індексоване сховище даних закладів харчування; сховище MySQL — сховище даних закладів харчування.

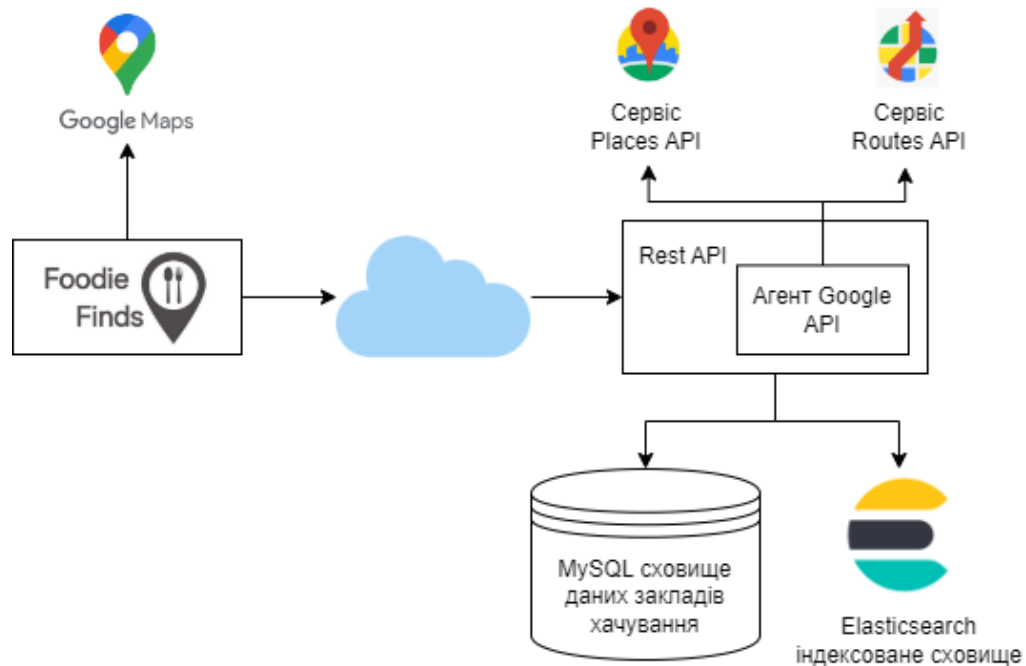


Рисунок 4.4 – Архітектурна схема додатку

4.4 Структура проєкту

Розроблена система є багат шаровою, тому її можна поділити на 3 шари: представлення (Presentation Layer), бізнес-логіки (Business Layer) та доступу до даних (Persistence Layer). Кожен з цих шарів відповідає за конкретний аспект системи. Детальна діаграма шарів системи розташована в додатку Б.

Шар доступу до даних відповідає за доступ до даних, збережених у системі. Він містить компоненти, які взаємодіють з базами даних або іншими джерелами даних, проводять операції збереження, отримання, оновлення та видалення даних. Даний шар складається з п'яти основних пакетів.

1. Пакет “config” – це пакет, що відповідає за конфігурацію програми. Він містить налаштування клієнта, безпеку та загальні властивості системи.

2. Пакет “entity” – це пакет, що містить класи об'єктів, які використовуються для передачі даних між програмного та базою даних..

3. Пакет “efrepository” – це пакет, що відповідає за взаємодію з базою даних, а саме Entity об’єктами.

4. Пакет “elasticmodel” – це пакет, що містить індекси Elasticsearch .

5. Пакет “elasticrepository” – це пакет, що відповідає за взаємодію з індексами Elasticsearch.

Шар бізнес-логіки містить компоненти, які визначають всю функціональність програми. В ньому описані правила та процеси, які визначають основні функції додатка. Розглянемо пакети, з яких складається даний шар.

1. Пакет “service” – це пакет, що містить класи, які реалізують всю бізнес-логіку системи.

2. Пакет “dto” – це пакет, що використовується для передачі даних між серверною та клієнтською частинами програми.

3. Пакет “utils” – це пакет, де реалізовані функції для конвертації даних та задання параметрів запиту сторінки.

Шар представлення, це клієнтська частина програми, що зазвичай, описує ту частину додатка, яка відповідає за інтерфейс користувача та безпосередню взаємодію з ним. Цю частину додатка формують компоненти представлення (view package), які містять інтерфейси всіх сторінок, що використовуються в застосунку для відображення її функціоналу.

1. Компонент “Filters” – це елемент сторінки Menu View, де реалізовано всі фільтри для пошуку закладів харчування.

2. Компонент “RestaurantItem” – це елемент сторінки Menu View, що відповідає за відображення списку ресторанів.

3. Компонент “Logout View” – це сторінка, яка дозволяє користувачеві вийти з системи.

4. Компонент “Main View” – це основний елемент усіх наявних в програмі сторінок. Він містить заголовок (header) та навігаційне меню (navbar).

5. Компонент “RestaurantInfo View” – це сторінка, яка виводить коротку інформацію про конкретний ресторан та всі його відгуки.

6. Компонент “Menu View” – це головна сторінка, що містить елементи Filters та RestaurantItem. Вона демонструє користувачу всі доступні функції пошуку та

окрім цього, містить поле для повнотекстового пошуку та можливість переходу на карту.

4.3 Опис моделі даних

Для створення бази даних необхідно визначити найважливіші таблиці, зв'язки між ними та атрибути кожної сутності. Для цього створимо логічну модель даних (рис. 4.5).

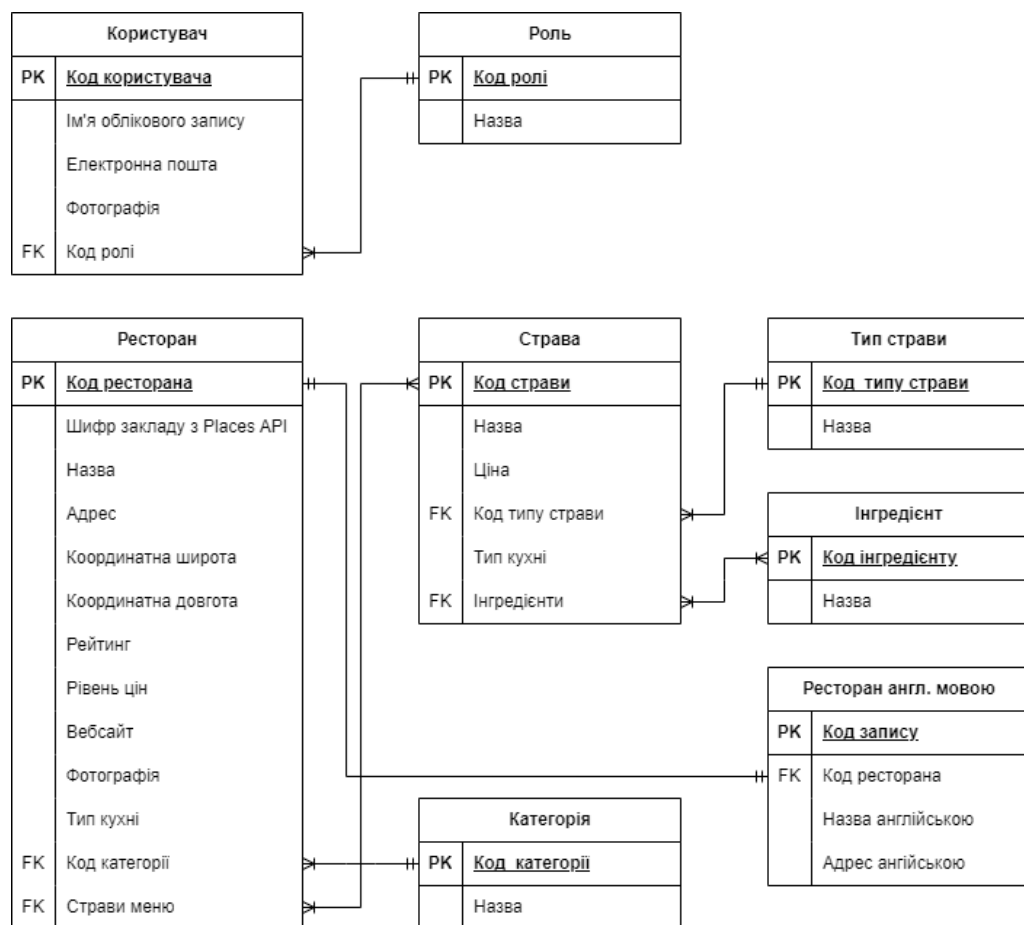


Рисунок 4.5 – Логічна модель бази даних

Після аналізу логічної моделі бази даних буде створено фізичну модель бази даних (рис. 4.6). Дана схема містить 12 таблиць:

– 9 основних: role, user, category, ingredient, dish_type, dish, restaurant, review та restaurant_info_en;

	role_id	name
1	1	user
2	2	admin

Рисунок 4.7 – Приклад заповнення таблиці “role”

Таблиця “user” містить 4 поля даних: user_id, username, email, picture та role (рис. 4.8). Стовпець user_id є первинним ключем та використовується для забезпечення унікальності кожного запису таблиці. Поле username відповідає за назву облікового запису користувача (найчастіше ім’я та прізвище), email зберігає зареєстровану пошту, а picture містить шлях до файлу з іконкою акаунту. Всі ці дані отримуються за допомогою облікового запису Google, що був зареєстрований цим користувачем в системі. Останнє поле, role_id, є зовнішнім ключем, що посилається на стовпець role_id в таблиці role, і вказує на роль, яка призначена цьому користувачу.

	user_id	username	email	picture	role_id
1	0	googleUser	google.user@gma...	<null>	1
2	1	Ірина Гурин	irynka.huryn@gm...	https://lh3.googleuse...	2
3	2	Iryna Huryn	huryn.irynd@lll...	https://lh3.googleuse...	1

Рисунок 4.8 – Приклад заповнення таблиці “user”

Таблиця “category” містить 2 поля даних: category_id та name (рис. 4.9). Стовпець category_id є первинним ключем та використовується для забезпечення унікальності кожного запису таблиці. Поле name відповідає за назву категорії.

	category_id	name
1	1	restaurant
2	2	food
3	3	point_of_interest
4	4	establishment
5	5	meal_delivery
6	6	bar

Рисунок 4.9 – Приклад заповнення таблиці “category”

Таблиця “ingredient” містить 2 поля даних: ingredient_id та name (рис. 4.10). Стовпець ingredient_id є первинним ключем та використовується для забезпечення унікальності кожного запису таблиці. Поле name відповідає за назву інгредієнта.

	ingredient_id	name
1	1	Картопля
2	2	М'ясо
3	3	Тісто
4	4	Капуста
5	5	Гірчиця
6	6	Кетчуп

Рисунок 4.10 – Приклад заповнення таблиці “ingredient”

Таблиця “dish_type” містить 2 поля даних: type_id та name (рис. 4.11). Стовпець type_id є первинним ключем та використовується для забезпечення унікальності кожного запису таблиці. Поле name відповідає за назву типу страви.

	type_id	name
1	1	Салат
2	2	Суп
3	3	Піца
4	4	Основна страва
5	5	Десерт
6	6	Напій

Рисунок 4.11 – Приклад заповнення таблиці “dish_type”

Таблиця “dish” містить 5 полів даних: dish_id, name, price, dish_type та cuisine_type (рис. 4.12). Стовпець dish_id є первинним ключем та використовується для забезпечення унікальності кожного запису таблиці. Поле name відповідає за назву страви, а price – ціна блюда. Поле dish_type є зовнішнім ключем, що посилається на стовпець type_id в таблиці dish_type, і вказує на тип даної страви. Стовпець cuisine_type вказує на приналежність страви до заданої кухні.

	dish_id	name	price	dish_type	cuisine_type
1	1	Вареники з картоплею	80	4	Українська
2	2	Голубці з м'ясом	90	4	Українська
3	3	Гамбургер	150	4	Американська

Рисунок 4.12 – Приклад заповнення таблиці “dish”

Таблиця “restaurant” містить 11 полів даних: restaurant_id, place_id, name, address, latitude, longitude, rating, price_level, website, photo_ref та cuisine_type (рис. 4.13). Дана таблиця відіграє найважливішу роль в розробленій програмі, адже відповідає за обробку даних про ресторани. Щоб гарантувати, що система максимально реалістично відповідає вимогам, всі наведені нижче дані в цій таблиці, окрім restaurant_id, було визначено за допомогою Places API. Стовпець restaurant_id є первинним ключем та використовується для забезпечення унікальності кожного запису таблиці. Поле place_id є унікальним ідентифікатором даного місця у використаному API. Стовпець name відповідає за назву ресторану, address визначає адрес закладу харчування, latitude та longitude задають розташування місця в системі географічних координат, rating позначає рейтинг закладу (від 0 до 4), price_level – це рівень цін (від 0 до 4), website містить посилання на офіційну сторінку ресторану в Інтернеті. Поле photo_ref містить шлях до файлу із зображенням закладу харчування, cuisine_type – це тип кухні, до якої відноситься заклад.

	place_id	name	address	lat...	long...	rat...	pri...	webs...	photo_ref	cuisine_type
1	ChIJG-u0FnXP1EAR...	Царське село	вулиця Лаврська...	50.430675699...	30.5568319	4.5		3 http://tsarsk...	/db_data/restaura...	Українська
2	ChIJuR7a01f01EAR...	Корчма Тарас...	вулиця Євгена Ч...	50.4473507	30.5188238	4.5		2 https://www.i...	/db_data/restaura...	Українська
3	ChIJES0e4z301EAR...	Хуторець на ...	вул. Набережно-...	50.4699476	30.5243313	4.6999998092...		0 <null>	/db_data/restaura...	Українська
4	ChIJY0Qe_QDP1EAR...	BARVY	вулиця Мечников...	50.4376708	30.5291614	4.5999999046...		0 <null>	/db_data/restaura...	Українська
5	ChIJWxt4ALH01EAR...	Первак	вулиця Рогнідин...	50.439659	30.519268	4.5999999046...		2 http://www.pe...	/db_data/restaura...	Українська
6	ChIJC-BGyhrP1EAR...	О'Панас	вулиця Терещенк...	50.4423658	30.5138406	4.5		3 http://opanas...	/db_data/restaura...	Українська
7	ChIJbWYVLKP01EAR...	За двома зай...	Андріївський уз...	50.4593396	30.5171894	4.5999999046...		3 http://zaytsi...	/db_data/restaura...	Українська
8	ChIJIIdZio_P01EAR...	Рідна хата	вулиця Жиянськ...	50.442578	30.4956174	4.3000001907...		2 http://www.r.i...	/db_data/restaura...	Українська
9	ChIJk1HmrV_01EAR...	Petrus-ь	вулиця Еспланад...	50.437118200...	30.5203268	4.4000000953...		2 http://petrus...	/db_data/restaura...	Українська
10	ChIJ3Yf84Fn01EAR...	Мітла	вулиця Прорізна...	50.4485487	30.5162415	4.5		2 http://www.m1...	/db_data/restaura...	Українська

Рисунок 4.13 – Приклад заповнення таблиці “restaurant”

Таблиця “review” містить 5 полів даних: review_id, text, time, restaurant_id та user_id (рис. 4.14). Стовпець review_id є первинним ключем та використовується

для забезпечення унікальності кожного запису таблиці. Щодо поля `text`, що відповідає за текстове наповнення відгуку, та `time`, що вказує на час, коли цей відгук було написано, то початкова інформація про них була також взята з Places API. Поле `restaurant_id` є зовнішнім ключем, що посилається на стовпець `restaurant_id` в таблиці `restaurant`, і вказує до якого ресторану відноситься даний запис. Поле `user_id` є зовнішнім ключем, що посилається на стовпець `user_id` в таблиці `user`, і вказує на те, хто додав цей відгук.

	review_id ▼	text ▼	time ▼	restaurant_id ▼	user_id ▼
1	1	Гарно стилізований заклад, п...	2024-04-15 16:11:46.000000	1	0
2	2	Мабуть найгірший сервіс не т...	2024-05-23 13:08:09.000000	1	0
3	3	Чудовий заклад та атмосфера,...	2024-01-13 17:49:11.000000	1	0
4	4	Дуже гарне атмосферне україн...	2023-11-04 21:07:03.000000	1	0
5	5	Якщо ви бажаєте посмакувати ...	2023-11-02 17:36:00.000000	1	0
6	6	Ресторан з традиційною украї...	2024-03-26 13:34:12.000000	2	0
7	7	Ви скучили за справжніми укр...	2024-05-05 17:08:41.000000	2	0
8	8	Як я скучила після Іспанії з...	2023-12-30 21:33:37.000000	2	0

Рисунок 4.14 – Приклад заповнення таблиці “review”

Таблиця “`restaurant_info_en`” містить 4 поля даних: `info_id`, `restaurant_id`, `name_en` та `address_en` (рис. 4.15). Дане сховище створене для збереження інформації про ресторан англійською мовою та заповнюється також завдяки Places API. Стовпець `review_id` є первинним ключем та використовується для забезпечення унікальності кожного запису таблиці. Поле `restaurant_id` є зовнішнім ключем, що посилається на стовпець `restaurant_id` в таблиці `restaurant`, і вказує до якого ресторану відноситься даний запис. Щодо поля `name_en` та `address_en`, то вони містять дані про ресторан англійською мовою.

	info_id ▼	restaurant_id ▼	name_en ▼	address_en ▼
18	18	21	Prynada Ukrainian C...	проспект Володимира Івасюка...
19	19	22	American bar	вулиця Костьольна, 11, Київ
20	20	23	Cheesesteak	вулиця Січових Стрільців, 1...
21	21	24	Sam's Steak House	вулиця Володимирська, 97/37...

Рисунок 4.15 – Приклад заповнення таблиці “restaurant_info_en”

Окрім цього, як було зазначено раніше, в даній системі існує 3 суміжні таблиці, що поєднують в собі ідентифікатори двох інших таблиць. Зв'язок в усіх цих таблиць може бути описаний як MANY-TO-MANY. Це вказує на те, що кілька записів у таблиці пов'язані з кількома записами в іншій таблиці. Нижче описані наявні таблиці.

1. “restaurant_category” – це таблиця, що зв'язує дві інші, а саме “restaurant” та “category”. Вона містить такі поля, як restaurant_id та category_id. Дана таблиця вказує на те, що кожен ресторан може відноситися до багатьох категорій, так само як і категорія до багатьох ресторанів.

2. “restaurant_dish” – це таблиця, що зв'язує дві інші, а саме “restaurant” та “dish”. Вона містить такі поля, як restaurant_id та dish_id. Дана таблиця вказує на те, що кожен ресторан може мати багато страв, а страва може бути в меню різних ресторанів.

3. “dish_ingredient” – це таблиця, що зв'язує дві інші, а саме “dish” та “ingredient”. Вона містить такі поля, як dish_id та ingredient_id. Дана таблиця вказує на те, що кожна страва може мати багато інгредієнтів, а інгредієнт може бути елементом багатьох страв.

5 ВЗАЄМОДІЯ КОРИСТУВАЧА З СИСТЕМОЮ

У даному розділі буде розглянуто основні принципи та методи взаємодії користувача з системою. Буде продемонстровано, як користувачі можуть ефективно працювати з програмним продуктом, які інструменти доступні для полегшення цієї взаємодії та які рекомендації слід враховувати для забезпечення зручності та продуктивності.

5.1 Функціональні можливості для авторизованого користувача

Головний функціонал додатка доступний авторизованому користувачу. У нього є можливість пошуку закладів харчування за смаковими вподобаннями, а також перегляд та додавання відгуків.

Для того, щоб отримати доступ до функцій застосунку, користувачу потрібно зареєструватися в системі за допомогою акаунту Google. Для цього йому потрібно обрати наявний обліковий запис Google та надати дозвіл для обробки особистих даних (рис. 5.1).

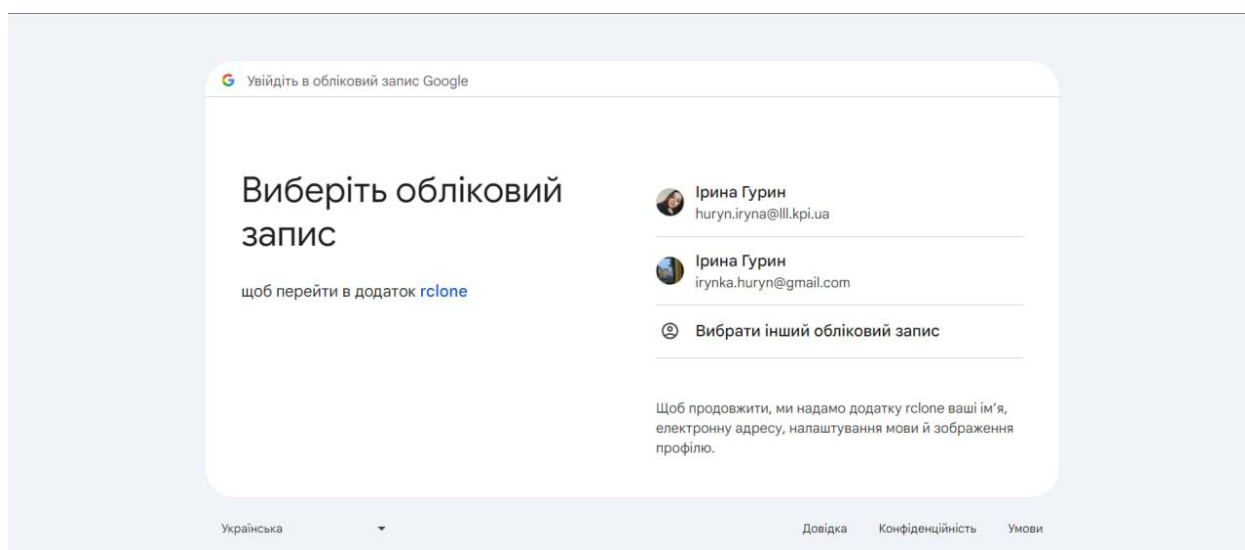


Рисунок 5.1 – Реєстрація за допомогою облікового запису Google

5.1.1 Пошук закладів харчування за смаковими вподобаннями користувача

Після входу в систему, клієнт буде перенаправлений на головну сторінку, де йому буде продемонстровано різноманітні фільтри для пошуку та список ресторанів, що буде змінюватися залежно від обраних фільтрів (рис. 5.2). Перейдемо до розгляду функціоналу, що представлено на поточній сторінці.

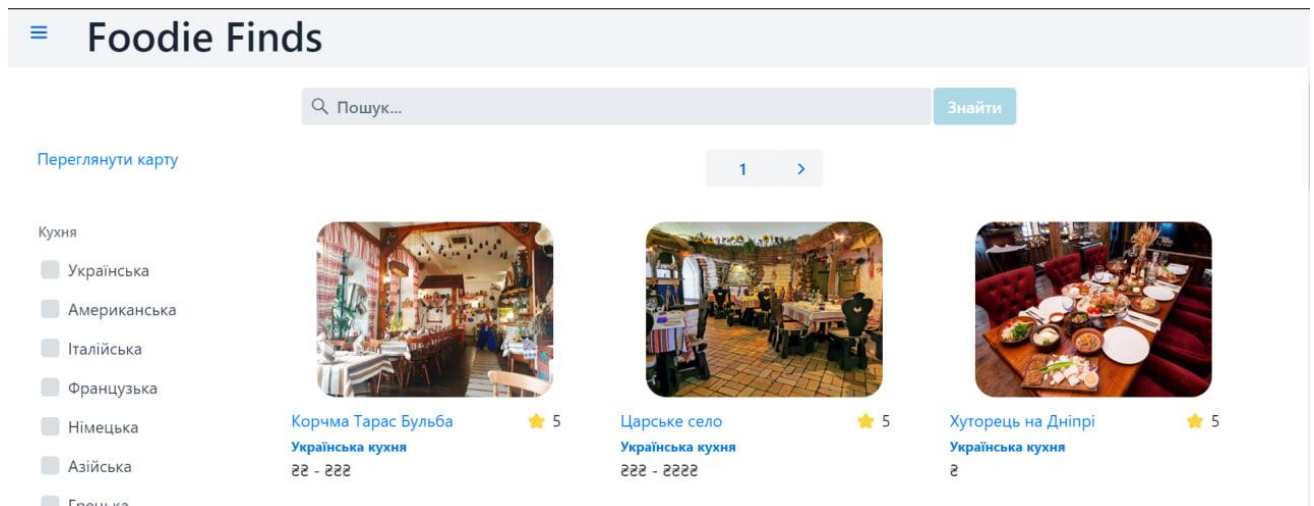


Рисунок 5.2 – Головна сторінка

При натисканні на "Переглянути карту" користувач буде перенаправлений на Google Карти, де він може ввести назву закладу харчування та ознайомитися з його розташуванням (рис 5.3). Це важливо у тому випадку, якщо людині важливе розташування закладу харчування і вона бажає проаналізувати його локацію.

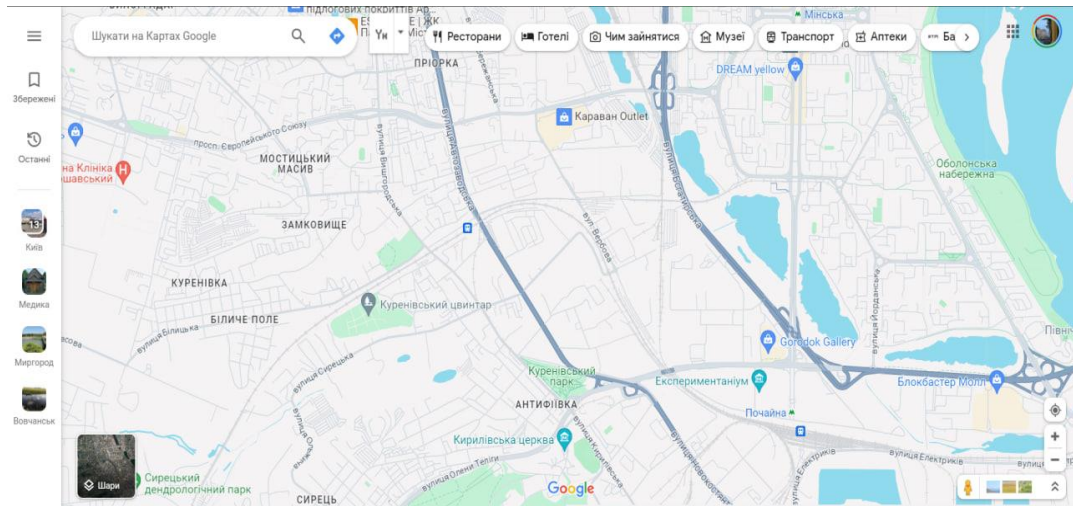


Рисунок 5.3 – Перегляд Google Карт

Отже, щодо наявних в системі фільтрів, всього їх 8 (включаючи повнотекстовий пошук) :

- пошук за кухнею;
- пошук за рейтингом закладу (на основі оцінок Google);
- пошук за відстанню;
- пошук відносно рівню цін;
- пошук за конкретними стравами;
- пошук виключаючи задані інгредієнти;
- пошук за ключовими словами у відгуках.

Фільтри системи реалізовані так, що їх можна поєднувати та використовувати в різних комбінаціях. Це спрощує пошук та допомагає отримати більш точні результати.

Основне завдання користувача при виборі фільтрів для кухні, рейтингу та цінового рівня — зазначити ті, які він хоче бачити у результатах списку закладів. Для цього йому потрібно активувати checkbox, та отримати результат, зображений на рисунку 5.4.

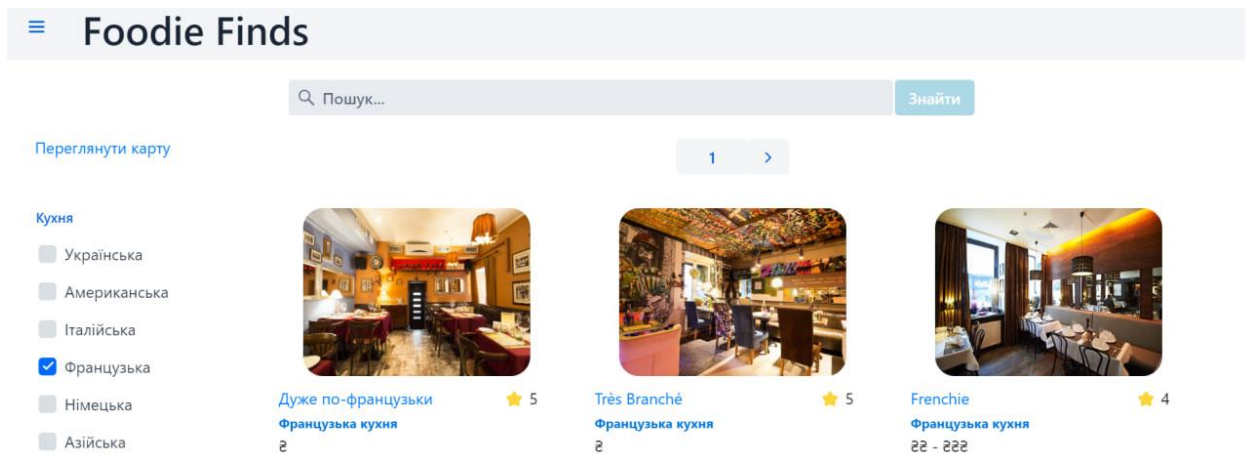


Рисунок 5.4 – Фільтрування за кухнею

Фільтр відстані (рис. 5.5) працює за схожим принципом, що й попередні, але для нього спочатку потрібно ввести адресу місця відправлення та числове значення через кому в текстове поле. Цей фільтр визначає заклади, розташовані в межах вказаних хвилин ходьби або їзди на автомобілі. Можна також використовувати відстань у кілометрах як параметр пошуку. Числове значення в полі буде границею для визначення часу та відстані. Даний фільтр дозволяє користувачеві легко знаходити заклади на зручній відстані, враховуючи його можливості переміщення та індивідуальні потреби.

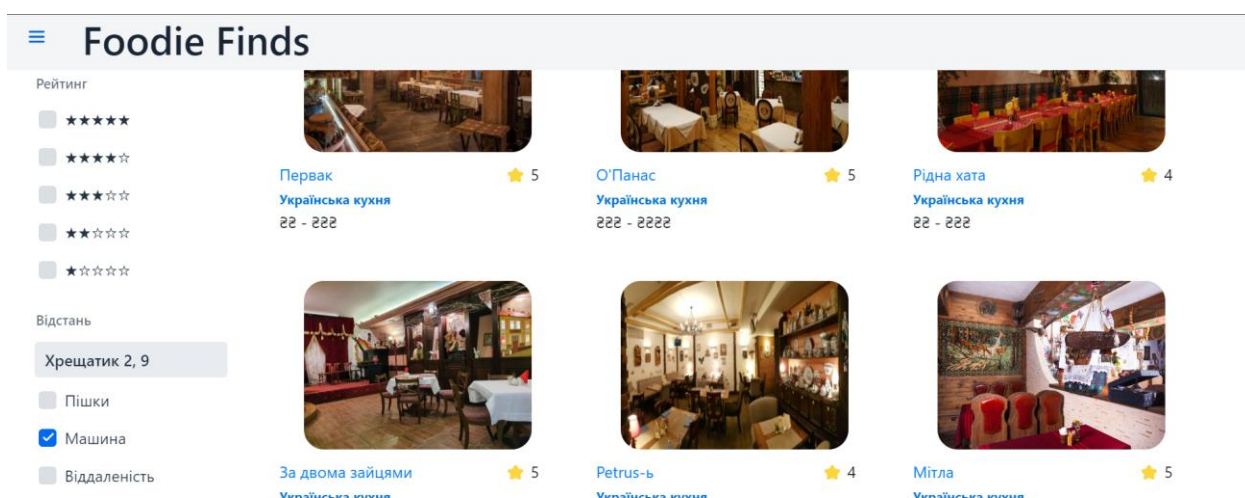


Рисунок 5.5 – Задання параметрів фільтру відстані

За допомогою фільтру страв (рис. 5.6) користувач має можливість отримати список закладів харчування, які містять в своєму меню задану страву. Даний фільтр, як і наступний, пов'язаний з інгредієнтами, реалізований за допомогою автодоповнення, що використовує технологію Elasticsearch.

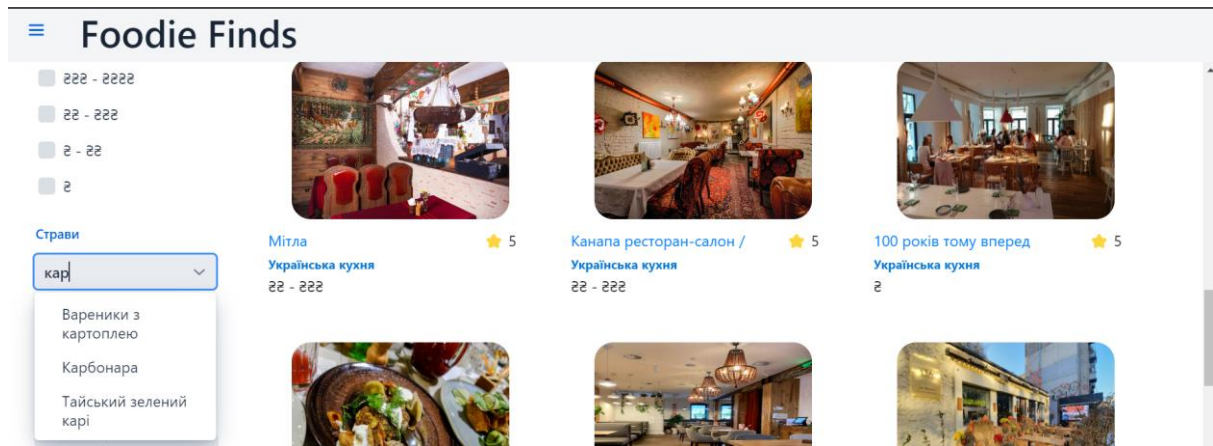


Рисунок 5.6 – Фільтр страв

Фільтр інгредієнтів працює як виключення деяких змінних (рис. 5.7). Він надає можливість отримати лише ті заклади харчування, які не будуть містити обрані користувачем інгредієнти. Це може бути корисно для людей, які мають алергію на деякий продукт, або просто не хочуть його бачити в своїх стравах.

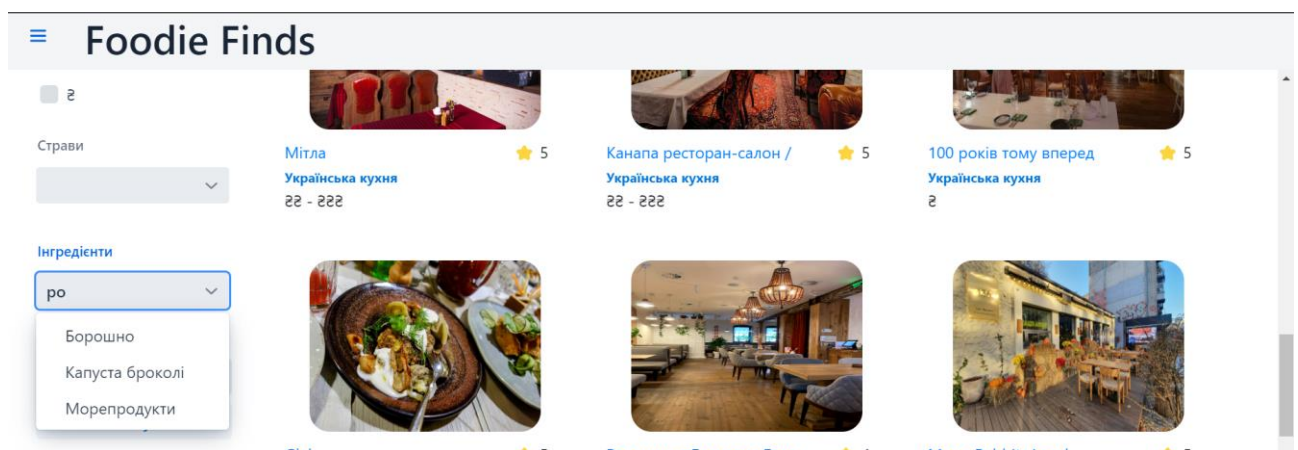


Рисунок 5.7 – Фільтр інгредієнтів

Можливість пошуку ресторанів за ключовими словами (рис. 5.8), може допомогти з вибором людям, у яких є особливі вподобання щодо закладів. Даний пошук реалізований з використанням технології Elasticsearch. В даному випадку відбувається пошук лише тих результатів, які містять повну задану фразу. Наприклад, якщо людина обожнює пиво, то вона може організувати пошук за словом "Beer". Деяким індивідуумам важливо, щоб сервіс був на висоті, тому вони можуть скористатись цим фільтром задавши запит на пошук фрази "good service".

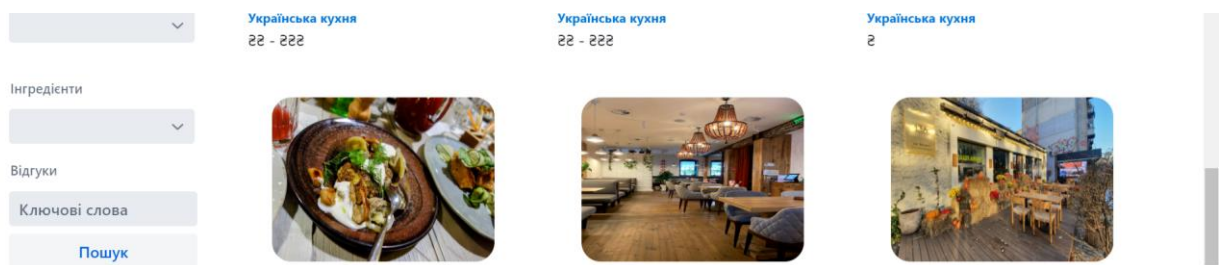


Рисунок 5.8 – Пошук за ключовими словами у відгуках

Останній фільтр, який важливо розглянути - це повнотекстовий пошук (рис. 5.9). Він реалізований для тих випадків, коли людина хоче віднайти один або декілька закладів харчування за назвою або адресом. Варто врахувати той факт, що пошук в даному фільтр реалізований в англійській та українській морфології. Саме тому, якщо людина знає назву закладу лише англійською мовою, це не викличе у неї проблем під час пошуку. Пошук відбувається так, що система шукає всі ресторани в яких хоча б раз згадувалося одне зі слів запиту.

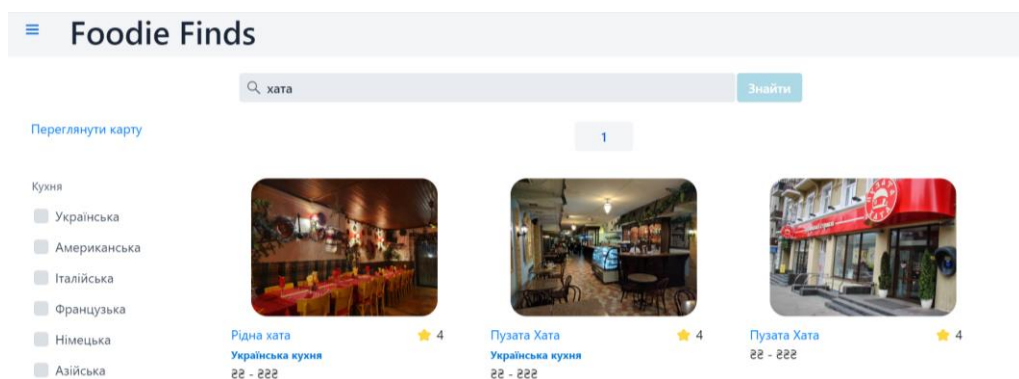


Рисунок 5.9 – Повнотекстовий пошук

5.1.2 Перегляд та додавання відгуків

У користувача також є можливість переглянути коротку інформацію про заклад харчування (рис. 5.10), його відгуки та навіть додати нові, завдяки вбудованій системі додавання відгуків. Для цього користувачу необхідно натиснути на заклад харчування і його автоматично буде перенаправлено на сторінку з інформацією про обраний ресторан.

Щодо самостійного додавання відгуків, то людині потрібно вписати бажану інформацію в текстове поле та натиснути на кнопку. Відгук буде автоматично додано до відгуків даного користувача та відображено на сторінці.

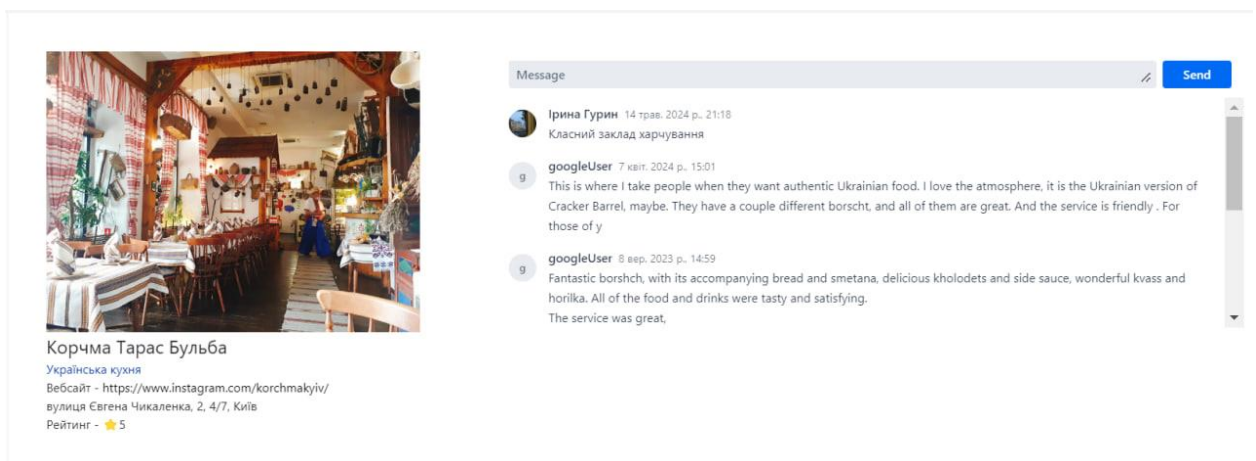


Рисунок 5.10 – Сторінка з інформацією про обраний заклад харчування

5.2 Функціональні можливості для адміністратора системи

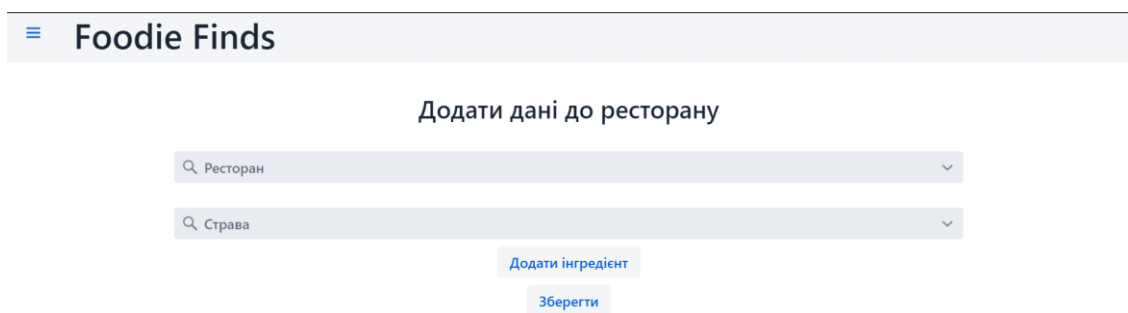
Адміністратор – це друга роль, що наявна в даній системі. Його головними задачами є додавання страв та інгредієнтів до ресторанів, зміна їх параметрів, а також їх видалення (рис. 5.11).

Хотілось би зазначити, що пошук значень в даній частині програми повністю реалізована за допомогою автодоповнення реалізованого з використанням технології Elasticsearch. Для цього користувачу необхідно почати вписувати

бажане слово, а система почне швидко знаходити співпадіння в даних та подавати найбільш схожі.

5.2.1 Додавання страв та інгредієнтів в меню

Для додавання існуючих або нових страв та інгредієнтів до меню ресторанів, користувачу необхідно перейти на головну сторінку свого облікового запису (рис. 5.11).



Foodie Finds

Додати дані до ресторану

Ресторан

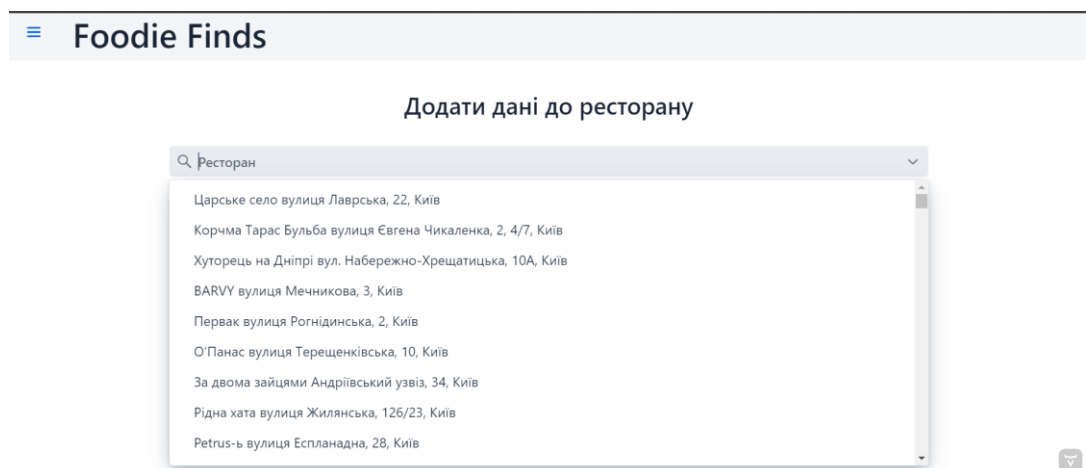
Страва

Додати інгредієнт

Зберегти

Рисунок 5.11 – Головна сторінка для додавання інформації до закладу харчування

На даній сторінці адміну необхідно обрати бажаний ресторан, вписавши його назву або адрес (рис. 5.12).



Foodie Finds

Додати дані до ресторану

Ресторан

- Царське село вулиця Лаврська, 22, Київ
- Корчма Тарас Бульба вулиця Євгена Чикаленка, 2, 4/7, Київ
- Хуторець на Дніпрі вул. Набережно-Хрещатицька, 10А, Київ
- BARVY вулиця Мечникова, 3, Київ
- Первак вулиця Рогнідинська, 2, Київ
- О'Панас вулиця Терещенківська, 10, Київ
- За двома зайцями Андріївський узвіз, 34, Київ
- Рідна хата вулиця Жилинська, 126/23, Київ
- Petrus-ь вулиця Еспланадна, 28, Київ

Рисунок 5.12 – Вибір закладу харчування

Наступним етапом є вибір страви, яку хочеться змінити (рис. 5.13). В даному випадку, можна обрати вже існуюче блюдо, або додати нове.

Foodie Finds

Додати дані до ресторану

Першак вулиця Рогнідинська, 2, Київ

Страва

- Нова страва
- Вареники з картоплею 80.0 грн.
- Голубці з м'ясом 90.0 грн.
- Гамбургер 150.0 грн.
- Хот-дог 70.0 грн.
- Карбонара 160.0 грн.
- Капрезе 120.0 грн.

Рисунок 5.13 – Вибір страви

У випадку, якщо адміністратор обрав опцію «Нова страва», у нього з'являється можливість додати нову страву в систему. Для цього після вибору на екрані виводяться текстові поля, які необхідні для додавання нового елемента, а саме: його назва, ціна, тип страви та кухня, яка додається автоматично в залежності від кухні, до якої відноситься обраний ресторан (рис. 5.14).

Foodie Finds

Додати дані до ресторану

Першак вулиця Рогнідинська, 2, Київ

Нова страва

Назва

Ціна

Тип страви

Кухня

Українська

Додати інгредієнт

Зберегти

Рисунок 5.14 – Створення нової страви

Після заповнення всіх необхідних полей необхідно додати інгредієнти до цієї страви. Для цього адміністратору потрібно натиснути на кнопку «Додати інгредієнт». Вона активує подію та додає на сторінку нове поле, з допомогою якого можна обрати інгредієнт (рис. 5.15).

Foodie Finds

Додати дані до ресторану

Первак вулиця Рогнідинська, 2, Київ

Нова страва

Назва	Ціна	Тип страви	Кухня
Апельсиновий сік	60	Напій	Українська

Додати інгредієнт

Інгредієнт

Зберегти

Рисунок 5.15 – Додавання інгредієнту

Як і у випадку з стравою, можна також створити новий інгредієнт та додати його в систему. Для цього у випадаючому списку треба обрати варіант «Новий інгредієнт» (рис. 5.16).

Foodie Finds

Новий інгредієнт

Картопля

М'ясо

Тісто

Капуста

Гірчиця

Кетчуп

Курка

Сир Пармезан

Сало

Новий інгредієнт

Рисунок 5.16 – Випадаючий список для вибору інгредієнта

Для задання нового інгредієнту потрібно вписати його назву. Окрім цього, хотілось би зазначити, що можна додавати декілька інгредієнтів. Для цього необхідно натиснути кнопку «Додати інгредієнт» стільки раз, скільки елементів хочемо додати до страви. Після вибору та введення всіх необхідних даних потрібно натиснути кнопку «Зберегти».

Рисунок 5.17 – Додавання декількох інгредієнтів

У випадку, якщо адміністратор вирішить обрати вже наявну в системі страву, що автоматично на екран буде виведено всі її інгредієнти з можливістю додавання нових.

Рисунок 5.18 – Вибір наявної в системі страви

5.2.2 Зміна інформації про страви на інгредієнти

Для зміни інформації про страви та інгредієнти, користувачу необхідно перемістити на сторінку для зміни даних в меню (рис. 5.19). Тут він також може вибрати необхідний ресторан та страву, яку він хоче змінити.

Foodie Finds

Змінити дані в меню

Первак вулиця Рогнідинська, 2, Київ

Апельсиновий сік

Назва	Ціна	Тип страви	Кухня
Апельсиновий сік	60	Напій	Українська

Змінити страву

Інгредієнт

Зберегти

Рисунок 5.19 – Сторінка для зміни даних в меню

Все що потрібно зробити користувачу для зміни інформації про страву це вписати необхідну інформацію та натиснути на кнопку «Змінити страву».

Приклад зміни ціни представлено на рисунку 5.20.

Foodie Finds

Змінити дані в меню

Первак вулиця Рогнідинська, 2, Київ

Апельсиновий сік

Назва	Ціна	Тип страви	Кухня
Апельсиновий сік	66	Напій	Українська

Змінити страву

Інгредієнт

Зберегти

Рисунок 5.20 – Зміна ціни для страви

5.2.3 Видалення інформації про страви на інгредієнти

Для доступу до функції видалення, користувачу потрібно перейти на відповідну сторінку. Функціонал представлений в даному вікні легкий для розуміння. Користувач обирає ресторан та страву, після чого біля страви та її інгредієнтів появляться червоні кнопки (рис. 5.21). Саме з їх допомогою і відбувається видалення. Користувач повинен натиснути на кнопку, що відповідає бажаному запису.

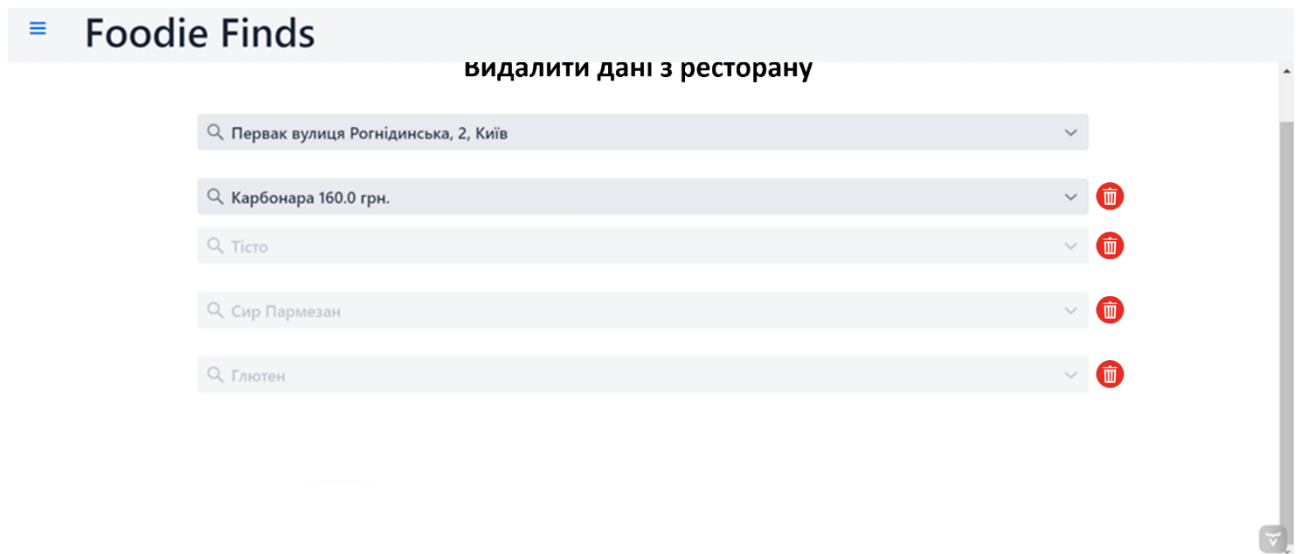


Рисунок 5.21 – Видалення страв на інгредієнтів

ВИСНОВКИ

1. На основі аналізу аналогічних систем, визначення їх переваг та недоліків, дослідження популярних пошукових запитів та їх структури, обґрунтовано використання технології Elasticsearch для повнотекстового пошуку та пошуку за ключовими словами. Вибір цієї технології допомагає організувати швидкий пошук необхідних ранжованих результатів та відповідає усім вимогам розроблюваної системи.

2. На основі аналізу програмних засобів обґрунтовано використання мови програмування Java, середовища розробки IntelliJ IDEA, таких фреймворків та бібліотек як Spring, Vaadin та Lombok, а також бази даних MySQL. Вибір цих технологій базується на їх високій продуктивності, гнучкості, масштабованості, швидкості та відповідності вимогам до розробки системи пошуку закладів харчування.

3. На основі вимог для розроблюваного продукту, було створено та продемонстровано архітектурну схему системи, структуру проєкту, модель даних та діаграми варіантів використання (Use case). Дані схеми допомагають краще зрозуміти загальну структуру проєкту, його функціональні можливості та взаємодію різних системних елементів між собою.

4. Взаємодія користувача з системою реалізована відповідно до основних принципів UX/UI дизайну. Це робить програмний продукт зручним та легким в користуванні, а розроблений інтерфейс забезпечує зрозумілу навігацію та доступ до основного функціоналу даної системи.

5. Розроблено веб-додаток "FoodieFinds", що дозволяє користувачам шукати пошук заклади громадського харчування на основі їх смакових вподобань. Система надає можливість швидкого отримання ранжованих результатів пошуку та підтримує гнучке налаштування пошукових параметрів, які можна комбінувати між собою.

6. Можливі перспективи вдосконалення програмної системи включають додавання нових функцій, таких як широку інтеграція з різноманітними сервісами для покращення візуалізації результатів пошуку і способу їх представлення,

можливість додавати заклади в обране, а також використання машинного навчання для надання можливості користувачу отримувати рекомендації на основі обраних закладів харчування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сучасні проблеми наукового забезпечення енергетики. У 2-х т. : Матеріали XXI Міжнар. наук.-практ. конф. молод. вчених і студ., м. Київ, 23–26 квіт. 2024 р. – Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2024. – Т. 2. – С. 274–275.
2. Home - Glovo Corporate Site. *Home - Glovo Corporate Site*: веб-сайт. URL: <https://about.glovoapp.com/> (дата звернення: 01.06.2024).
3. *Trip Advisor*: веб-сайт. URL: <https://www.tripadvisor.com/> (дата звернення: 01.06.2024).
4. *Open Table*: веб-сайт. URL: <https://www.opentable.com/> (дата звернення: 01.06.2024).
5. Web Languages-What language are most commonly used for web development - Dotnetlanguages. *Dotnetlanguages*: веб-сайт. URL: <https://www.dotnetlanguages.net/web-languages-what-language-are-most-commonly-used-for-web-development/> (дата звернення: 01.06.2024).
6. Introduction to Java - GeeksforGeeks. *GeeksforGeeks*: веб-сайт. URL: <https://www.geeksforgeeks.org/introduction-to-java/> (дата звернення: 01.06.2024).
7. Most Popular Java IDEs in 2024 | JRebel by Perforce. *JRebel by Perforce*: веб-сайт. URL: <https://www.jrebel.com/blog/best-java-ide> (дата звернення: 01.06.2024).
8. IntelliJ IDEA: Why is it the best IDE for you?. *Data Science Courses | DataScientest*: веб-сайт. URL: <https://datascientest.com/en/intellij-idea-why-is-it-the-best-ide-for-you> (дата звернення: 01.06.2024).
9. Thkaur S. Top 5 Java Frameworks for Web Application Development - DZone. *dzone.com*: веб-сайт. URL: <https://dzone.com/articles/top-5-java-frameworks-for-web-application-develomp> (дата звернення: 01.06.2024).
10. Mertens R. Java Spring - Was ist das?. *lernprogrammieren.com*: веб-сайт. URL: <https://lernprogrammieren.com/java-spring-was-ist-das/> (дата звернення: 01.06.2024).

11. Stack Overflow Developer Survey 2022. *Stack Overflow*: веб-сайт.
URL: <https://survey.stackoverflow.co/2022/#most-popular-technologies-database-prof> (дата звернення: 01.06.2024).

12. MySQL: What It Is, How It Works, And What It's Used For. *Knowi*: веб-сайт.
URL: <https://www.knowi.com/blog/mysql-what-it-is-how-it-works-and-what-its-used-for/> (дата звернення: 01.06.2024).

13. Elasticsearch: What it is, How it works, and it's usage. *Knowi*: веб-сайт.
URL: <https://www.knowi.com/blog/what-is-elastic-search/> (дата звернення: 01.06.2024).

ДОДАТОК А

Програмні засоби:

- мова програмування – Java;
- технологія повнотекстового пошуку – Elasticsearch;
- база даних - MySQL

```
@Override
public List<ReviewModel> searchByReviewKeywords(List<String> parts) {
    BoolQueryBuilder boolQuery = QueryBuilders.boolQuery();

    if (parts != null && !parts.isEmpty()) {
        for (String part : parts) {
            if (!part.isBlank()) {
                boolQuery.must(QueryBuilders.matchQuery("text", part));
            }
        }
    }

    String queryString = boolQuery.toString();
    StringQuery stringQuery = new StringQuery(queryString);

    stringQuery.setPageable(PageRequest.of(0, 3000));

    SearchHits<ReviewModel> searchHits = elasticsearchOperations.search(
        stringQuery, ReviewModel.class);

    return searchHits.stream()
        .map(SearchHit::getContent)
        .collect(Collectors.toList());
}

@Override
public RestaurantDTO searchByRestaurantInfo(String info) {
    List<RestaurantDTO> restaurantDTOS = getAll();
    for (RestaurantDTO restaurant: restaurantDTOS){
        String string = restaurant.getName() + " " + restaurant.getAddress();
        if(string.equals(info)){
            return restaurant;
        }
    }
    return null;
}

public List<RestaurantDTO> getRestaurantsByReviewKeywords(String keywords,
List<RestaurantDTO> filteredData){
    if(keywords != null && !keywords.isBlank()){
        List<ReviewDTO> foundReviews = getReviewsDTOByKeywords(keywords);

        Set<Long> restaurantsId = new HashSet<>();

        for(ReviewDTO review: foundReviews){
            restaurantsId.add(review.getRestaurantDTO().getRestaurantId());
        }
        List<RestaurantDTO> result = new ArrayList<>();

        for (RestaurantDTO restaurantDTO: filteredData){
```

```

        for (Long id: restaurantsId) {
            if (restaurantDTO.getRestaurantId().equals(id)) {
                result.add(restaurantDTO);
            }
        }
        return result;
    }
    return filteredData;
}

@Override
public List<ReviewDTO> getReviewsDTOByKeywords(String searchString) {
    searchString = searchString.toLowerCase();
    String[] words = searchString.split("\\s+");
    List<String> wordList = Arrays.asList(words);

    java.util.List<ReviewModel> list = searchByReviewKeywords(wordList);
    Set<Long> reviewsId = new HashSet<>();
    for (ReviewModel reviewModel: list) {
        reviewsId.add(reviewModel.getReviewId());
    }
    List<Review> entityList = new ArrayList<>();
    for (Long id : reviewsId) {
        entityList.add(reviewDbRepository.findById(id).orElse(null));
    }
    return Convertor.convertReviewEntityListToDTO(entityList);
}

@Override
public List<RestaurantModel> fullTextSearchInEngAndUkr(List<String> parts) {
    BoolQueryBuilder boolQuery = QueryBuilders.boolQuery();

    if (parts != null && !parts.isEmpty()) {
        for (String part : parts) {
            if (!part.isBlank()) {
                boolQuery.should(QueryBuilders.wildcardQuery("search_string_ukr",
                    "*" + part + "*"))
                    .should(QueryBuilders.wildcardQuery("search_string_eng",
                        "*" + part + "*"));
            }
        }
    }

    String queryString = boolQuery.toString();
    StringQuery stringQuery = new StringQuery(queryString);
    stringQuery.setPageable(PageRequest.of(0, 3000));

    SearchHits<RestaurantModel> searchHits = elasticSearchOperations.search(
        stringQuery, RestaurantModel.class);

    return searchHits.stream()
        .map(SearchHit::getContent)
        .collect(Collectors.toList());
}

@Override
public List<RestaurantDTO> getRestaurantsDTOBySearchInEngAndUkr(String
searchString, List<RestaurantDTO> restaurantList) {
    if (searchString != null && !searchString.isBlank()) {
        searchString = searchString.toLowerCase();
        String[] words = searchString.split("\\s+");
        List<String> wordList = Arrays.asList(words);

        java.util.List<RestaurantModel> list =

```

```

fullTextSearchInEngAndUkr(wordList);
    List<Restaurant> entityList = new ArrayList<>();
    for (RestaurantModel restaurantModel : list) {

entityList.add(restaurantDbRepository.findById(restaurantModel.getRestaurantId()))
;
        }
        return Convertor.convertRestaurantEntityListToDTO(entityList);
    }
    return restaurantList;
}

private void getRestaurantsDataFromResponseWithNextPageToken(LatLng loc,
PlacesSearchResponse response, GeoApiContext context, String cuisine) throws
ApiException, InterruptedException, IOException {
    String nextPageToken = response.nextPageToken;
    while (nextPageToken != null) {
        try {
            response = PlacesApi.nearbySearchQuery(context, loc)
                .radius(1500)
                .type(PlaceType.RESTAURANT)
                .language("uk")
                .pageToken(nextPageToken)
                .await();
            saveDataFromResponseToDb(response.results, context, cuisine);
            nextPageToken = response.nextPageToken;
        } catch (InvalidRequestException e) {
            break;
        }
    }
}

private void saveDataFromResponseToDb(PlacesSearchResult[] arrayOfResults,
GeoApiContext context, String cuisine) {
    log.info("Received info of {} restaurants", arrayOfResults.length);
    int countOfProcessedRestaurants = 0;
    int countOfSavedRestaurants = 0;
    for (PlacesSearchResult result : arrayOfResults) {
        log.debug("one result from search - {}", result);
        Restaurant existsObject =
restaurantDbRepository.findByPlaceId(result.placeId);
        if (existsObject == null) {
            log.debug("The restaurant wasn't find. Let's add it");
            Restaurant restaurant = getRestaurantData(result, context, cuisine);
            log.debug("Add restaurant: {}", restaurant);
            restaurantDbRepository.save(restaurant);

            saveRestaurantReviews(restaurant, restaurant.getPlaceId(), context);
            countOfSavedRestaurants++;
        }
        countOfProcessedRestaurants++;
        if(countOfProcessedRestaurants%100 == 0) {
            log.info("Processed {} restaurants, {} were saved",
countOfProcessedRestaurants, countOfSavedRestaurants);
        }
    }
}

@Override
public List<RestaurantDTO> getRestaurantInGivenDistance(String firstPoint,
List<Integer> routes, List<RestaurantDTO> filtered) {
    if(routes != null && !routes.isEmpty() && firstPoint != null &&
!firstPoint.isEmpty() && !firstPoint.isBlank()) {

        List<TravelMode> modes = new ArrayList<>();
        for (Integer route: routes){
            if(route == 2){
                modes.add(TravelMode.WALKING);
            }
        }
    }
}

```

```

        }
        if (route == 1) {
            modes.add(TravelMode.DRIVING);
        }
        if (route == 0) {
            modes.add(TravelMode.UNKNOWN);
        }
    }

    try {
        String[] parts = firstPoint.split(",");
        GeoApiContext context = getGeoApiContext();
        String location = parts[0];
        String numValue = parts[1].replace(" ", "");
        filtered = getRestaurantsByAddress(context, location, numValue,
filtered, modes);
    } catch (IOException e) {
        throw new RuntimeException(e);
    }

    }
    return filtered;
}

private List<RestaurantDTO> filteredByDishes(List<String> dishes,
List<RestaurantDTO> filteredData) {
    if (dishes != null && !dishes.isEmpty()) {
        List<RestaurantDTO> filtered = new ArrayList<>();
        Set<Long> addedRestaurantIds = new HashSet<>();

        for (String dish : dishes) {
            List<Restaurant> restaurantsByDish =
restaurantDbRepository.findByDishName(dish);
            for (Restaurant restaurant : restaurantsByDish) {
                Long restaurantId = restaurant.getId();
                addedRestaurantIds.add(restaurantId);
            }
        }
        for (RestaurantDTO restaurantDTO : filteredData) {
            for (Long id : addedRestaurantIds) {
                if (restaurantDTO.getRestaurantId().equals(id)) {
                    filtered.add(restaurantDTO);
                }
            }
        }
        return filtered;
    } else {
        return filteredData;
    }
}

private List<RestaurantDTO> filteredByIngredients(List<String> ingredients,
List<RestaurantDTO> filteredData) {
    if (ingredients != null && !ingredients.isEmpty()) {
        List<RestaurantDTO> filtered = new ArrayList<>();
        Set<Long> addedRestaurantIds = new HashSet<>();

        for (String ingredient : ingredients) {
            List<Restaurant> restaurantsByDish =
restaurantDbRepository.findRestaurantByIngredientName(ingredient).orElse(null);
            if (restaurantsByDish != null) {
                for (Restaurant restaurant : restaurantsByDish) {
                    addedRestaurantIds.add(restaurant.getId());
                }
            }
        }
    }
}

```

```

//          log.info("set size - {}", addedRestaurantIds.size());
        for (RestaurantDTO restaurantDTO : filteredData) {
            if(!addedRestaurantIds.contains(restaurantDTO.getRestaurantId())){
                filtered.add(restaurantDTO);
            }
        }
        return filtered;
    }else{
        return filteredData;
    }
}

@Override
public void setAllRestaurantListsData(Restaurant restaurant) {
    List<Dish> dishes = dishDbRepository.findByRestaurantId(restaurant.getId());
    for (Dish dish: dishes){
        List<Ingredient> ingredients =
ingredientDbRepository.findIngredientsByDishId(dish.getId()).orElse(null);
        dish.setIngredients(ingredients);
    }
    restaurant.setDishes(dishes);
}

private GeoApiContext getGeoApiContext() throws IOException {
    return new GeoApiContext.Builder()
        .apiKey(generalProperties.getGoogleApiKey())
        .build();
}

private List<RestaurantDTO> getRestaurantsByAddress(GeoApiContext context, String
address, String numValue, List<RestaurantDTO> filtered, List<TravelMode> modes){
    List<RestaurantDTO> restaurantModelList = new ArrayList<>();
    try {
        for (TravelMode mode: modes){
            for (RestaurantDTO restaurant : filtered) {
                if(mode.equals(TravelMode.UNKNOWN)){
                    mode = TravelMode.WALKING;
                    DistanceMatrix distanceMatrix = new
DistanceMatrixApiRequest(context)
                        .origins(address)
                        .mode(mode)
                        .destinations(restaurant.getAddress())
                        .await();

                    long distanceInMeters =
((distanceMatrix.rows[0]).elements[0]).distance.inMeters;
                    int distanceInKm = (int) Math.round(distanceInMeters * 0.001);
                    int givenValue = Integer.parseInt(numValue);
                    if (distanceInKm <= givenValue) {
                        restaurantModelList.add(restaurant);
                    }
                }else {
                    DistanceMatrix distanceMatrix = new
DistanceMatrixApiRequest(context)
                        .origins(address)
                        .mode(mode)
                        .destinations(restaurant.getAddress())
                        .await();

                    long timeInSec =
((distanceMatrix.rows[0]).elements[0]).duration.inSeconds;
                    int timeInMin = Math.round((float) timeInSec /60);
                    if (timeInMin <= Integer.parseInt(numValue)) {
                        restaurantModelList.add(restaurant);
                    }
                }
            }
        }
    }
}

```

```
        return restaurantModelList;
    } catch (ApiException | InterruptedException | IOException e) {
        throw new RuntimeException(e);
    }
}
```

ДОДАТОК Б

Діаграма шарів системи пошуку закладів харчування
за смаковими вподобаннями користувача

