МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ПРОГРАМУВАННЯ

До захисту допускається:
завідувач кафедри ІТП
Лифар В.О
«_____» _____2023 р.

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА
ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Мобільний додаток з
з пошуку кафе та ресторанів

Освітній ступінь – «бакалавр»
Спеціальність 126 «Інформаційні системи та технології»
(шифр і назва спеціальності)

Керівник проекту: _____ Марченко Д.М.
(підпис) (прізвище, ініціали)

Здобувач вищої освіти: _____ Григор`єв А.В.
(підпис) (прізвище, ініціали)

Група: ІСТ-19д

Київ 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Інформаційних технологій та програмування
Освітній ступінь бакалавр
Спеціальність 126 Інформаційні системи та технології
(шифр і назва)

ЗАТВЕРДЖУЮ:
завідувач кафедри ІТП
Лифар В.О
«_____» _____ 2023 р.

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Григор'єва Андрія Володимировича
(прізвище, ім'я, по батькові)

1.Тема роботи: Мобільний додаток з пошуку кафе та ресторанів

Керівник проекту (роботи) Марченко Д.М. доктор технічних наук, професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "26 " 04 2023 р. № 240/15-ОД

2. Термін подання здобувач вищої освіти роботи 05.06.2023

3. Вихідні дані до роботи матеріали переддипломної практики.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): а) аналіз методів і засобів створення додатку;
б)проектування інформаційної системи;
в) розробка продукту;
г)висновки

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

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посадаконсультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 24.03.2023

Керівник

(підпис)

Завдання прийняв до виконання

(підпис)

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Ознайомлення з проблематикою досліджуваної галузі	25.03.2023-04.04.2023	
2	Аналіз існуючих сервісів	05.04.2023-10.04.2023	
3	Проектування інформаційної системи	11.04.2023-19.04.2023	
4	Розробка сайту	20.04.2023-07.05.2023	
5	Розробка додатку	08.05.2023-20.05.2023	
6	Тестування фінальної версії продукту	20.05.2023-26.05.2023	
7	Оформлення пояснювальної записки	27.05.2023-04.06.2023	

Здобувач вищої освіти _____ Григор’єв А.В
(підпис) (прізвище, ініціали)

Керівник _____ Марченко Д.М.
(підпис) (прізвище, ініціали)

Р Е Ф Е Р А Т

Пояснювальна записка до дипломного проекту (роботи) бакалавра: 91 с., 27 мал., 20 бібліографічних джерел, 1 додаток.

Об'єкт розробки: мобільний додаток з пошуку кафе та ресторанів.

Мета роботи: створити зручний сервіс для пошуку кафе та ресторанів з використанням мобільного додатку.

В проекті виконано:

1. Аналіз методів і засобів створення додатку.

2. Аналіз існуючих проектів з пошуку закладів харчування.

3. Реалізовано сайт для сервісу.

4. Реалізовано додаток для сайту сервісу з використанням технології Progressive Web Application.

5. Додана адаптивність дизайну інтерфейсу сервісу.

JAVASCRIPT, HTML, CSS, PWA, ВЕБ-СЕРВІС, БРАУЗЕР, МЕДІА ЗАПИТ, ФРЕЙМВОРК, SQL, ДОМЕН

Зміст

Вступ

Темою дипломної роботи було обрано розробка додатку для сервісу з пошуку кафе та ресторанів. На це є багато причин, серед яких:

Сервіси з пошуку зараз стають дедалі популярнішими серед бізнесменів та звичайних користувачів. Ресторанний сегмент є особливо популярним серед людей віком до 50 років, які активно користуються можливостями пристроїв. Тому створення сервісу для пошуку кафе та ресторанів є актуальною та перспективною темою.

По-друге, такий сервіс може значно полегшити процес вибору закладу. З урахуванням великої кількості кафе та ресторанів, користувачі потребують зручного і ефективного інструменту, який допоможе знайти найкращі варіанти відповідно до їхніх вимог. Ця розробка може надати різні функції, такі як фільтрація за різними параметрами, відгуки та рейтинги, а також показати розташування закладів на мапі.

Крім того, створення мобільного додатка для пошуку кафе та ресторанів є цікавим і актуальним напрямом дослідження. На сьогоднішній день, коли люди все більше прагнуть отримувати інформацію та послуги на мобільних пристроях, розробка додатку стає необхідністю для багатьох компаній та бізнесів у галузі харчування. Такий додаток може стати потужним інструментом для залучення нових клієнтів, покращення їх взаємодії з закладом та підвищення загальної якості обслуговування.

Обрана тема має практичну значимість і потенціал для впровадження в реальному бізнес-середовищі. Вона може сприяти покращенню взаємодії між клієнтами та закладами громадського харчування, забезпечуючи зручний і швидкий доступ до інформації про ресторани, їхні пропозиції та відгуки користувачів. Крім того, розробка може виявитися цікавою в рамках

дослідницьких проектів з використання сучасних технологій та інноваційних рішень у сфері програмування та розробки.

Отже, обрана тема є актуальною, перспективною та корисною для користувачів. Вона поєднує в собі зручність вибору, різноманітний каталог закладів та сучасний підхід у розробці.

Об'єкт дослідження моєї дипломної роботи - мобільний додаток для сервісу з пошуку кафе та ресторанів. Об'єкт дослідження визначає ту сферу або явище, на якій спрямована увага дослідника. У даному випадку, об'єктом дослідження є сам додаток, який розробляється для спрощення процесу пошуку кафе та ресторанів для користувачів.

Предмет дослідження - функціональні можливості та користувацький досвід мобільного додатку з пошуку кафе та ресторанів. Предмет дослідження визначає конкретний аспект об'єкта, який буде досліджуватися. У даному випадку, предметом дослідження є те, які функції та можливості надає додаток користувачам, а також яким чином він забезпечує зручний та задовільний досвід використання.

Таким чином, об'єктом дослідження є мобільний додаток з пошуку кафе та ресторанів, а предметом дослідження є функціональні можливості та користувацький досвід цього додатку.

Мета моєї дипломної роботи полягає у розробці мобільного додатка з пошуку кафе та ресторанів, який надає користувачам зручний та ефективний спосіб знаходження та вибору закладів громадського харчування. Для досягнення цієї мети поставлені наступні задачі:

1. Аналіз потреб та очікувань користувачів щодо пошуку кафе та ресторанів. Вивчення їхніх пріоритетів, вимог та переваг, щоб створити додаток, який відповідає їхнім потребам.

2. Аналіз ринку закладів харчування. Аналіз особливостей роботи бізнесу.

3. Розробка зручного та інтуїтивно зрозумілого інтерфейсу сервісу. Під час розробки необхідно врахувати простоту використання, зрозумілість навігації та ефективність взаємодії з користувачем.

4. Розробка функціональності сервісу, яка включає пошук та фільтрацію кафе та ресторанів за різними параметрами, відображення інформації про заклади, рейтинги та відгуки користувачів.

Для розробки мобільного додатка пошуку кафе та ресторанів будуть використані наступні методи дослідження:

1. Аналіз потреб користувачів: Проведення опитувань або інтерв'ю з потенційними користувачами для визначення їх потреб та очікувань від додатка. Це допоможе з'ясувати головні функції та особливості, які треба врахувати при розробці.

2. Аналіз конкурентного ринку: Вивчення існуючих додатків для пошуку кафе та ресторанів, аналіз їх функціоналу, дизайну та переваг. Це допоможе виявити унікальні можливості та особливості, які можна впровадити у розроблюваному додатку.

3. Створення прототипу та тестування: Розробка прототипу додатка, який буде перевірятися та тестуватися серед невеликої групи користувачів. Це дозволить оцінити його зручність використання, ефективність функцій та виявити можливі проблеми чи покращення.

4. Збір зворотного зв'язку від користувачів: Отримання відгуків та коментарів від користувачів, які вже використовують подібні додатки. Це допоможе виявити та виправити можливі проблеми, а також дізнатися про додаткові потреби користувачів для майбутніх оновлень.

5. Тестування продуктивності: Оцінка продуктивності додатка в різних умовах, таких як різні типи пристроїв або з'єднання з Інтернетом. Це дозволить оптимізувати додаток для досягнення оптимальної продуктивності.

Застосування цих методів дослідження допоможе створити функціональний та зручний мобільний додаток для пошуку кафе та ресторанів, який задовольнить потреби та очікування користувачів.

1. АНАЛІЗ МЕТОДІВ І ЗАСОБІВ СТВОРЕННЯ ДОДАТКУ

1.1 Аналіз потреб користувачів щодо пошуку кафе та ресторанів.

Дослідження спрямоване на створення сервісу, який повністю задовольняє потреби користувачів.

Під час аналізу було взято до уваги думки та досвід різних користувачів. Були проведені опитування, інтерв'ю та огляди існуючих ресурсів для збору даних. В ході цього дослідження були виявлені наступні аспекти:

1. Вимоги щодо зручного та швидкого пошуку: багато користувачів вказали на важливість простоти та ефективності пошуку кафе та ресторанів. Вони хотіли мати можливість швидко знайти потрібне місце за категоріями, місцезнаходженням, ціною та іншими параметрами.
2. Інформація про рейтинги та відгуки: багато користувачів висловлювали бажання мати доступ до відгуків та оцінок інших клієнтів про кафе та ресторани. Це допомагає їм зробити об'єктивний вибір та отримати якісні послуги.
3. Інтеграція з картами та навігацією: багато користувачів виразили бажання мати можливість переглядати місцезнаходження кафе та ресторанів на карті, а також використовувати навігацію для зручного досягнення цих місць.
4. Мобільна доступність: більшість користувачів вказали, що бажають мати мобільний додаток для пошуку кафе та ресторанів, який буде зручно працювати на їхніх смартфонах.

На основі аналізу вимог та очікувань будуть розроблені функціональні вимоги та дизайн сервісу з пошуку кафе та ресторанів, щоб відповідати потребам користувачів.

1.2 Аналіз існуючих сервісів пошуку.

У цьому розділі розглянемо приклади вже працюючих сервісів з пошуку. Дослідження допоможе вивчити функціональність та особливості цих ресурсів з метою з'ясування їх переваг та недоліків.

Під час аналізу будуть розглянуті різні агрегатори, додатки та сайти, які надають можливість здійснювати пошук. Оцінка буде здійснюватися на основі таких факторів:

1. Зручність інтерфейсу: буде оцінено, наскільки просто та зрозуміло користуватися додатком чи сайтом. Це включає навігацію, пошукові функції, фільтри та інші елементи, які впливають на зручність використання.

2. Розмаїття інформації: буде досліджено, яку кількість та які види інформації надаються про кафе та ресторани. Це можуть бути рейтинги, відгуки, фотографії, меню та інші дані, які допомагають користувачам приймати рішення.

3. Функціональні можливості: будуть проаналізовані функції, доступні в додатках та на сайтах, такі як бронювання столиків, створення списку улюблених закладів, інтеграція з картами та інші.

4. Наявність мобільного додатку: буде досліджено, чи є мобільний додаток для зручного пошуку на смартфонах. Це може бути важливим фактором для користувачів, які шукають мобільні рішення.

Після аналізу наявних додатків та сайтів будуть визначені їх переваги та недоліки, що дозволить врахувати ці фактори при розробці власного додатку з пошуку кафе та ресторанів.

Для аналізу оберемо декілька найпопулярніших сервісів з пошуку закладів харчування та оцінимо їх:

Сервіс Yelp (https://www.yelp.com/) має деякі переваги і недоліки, які були розглянуті.

Переваги:

1. Наявність додатку для смартфонів.

2. Розмаїтість інформації: На сайті Yelp доступна широка база даних про різні кафе, ресторани та заклади громадського харчування. Користувачі можуть знайти відгуки, фотографії та інформацію про рейтинги та ціни.

3. Відгуки користувачів: Yelp дозволяє користувачам залишати відгуки та оцінки про кафе та ресторани, що допомагає іншим користувачам зробити інформований вибір. Відгуки можуть містити корисну інформацію про якість обслуговування, смак страв та загальний досвід.

4. Можливість пошуку за різними критеріями: Yelp дозволяє користувачам шукати кафе та ресторани за різними критеріями, такими як місцезнаходження, тип кухні, цінова категорія та інші. Це дозволяє знайти варіанти, що відповідають конкретним потребам користувача.

Недоліки:

1. Незалежність та достовірність відгуків: Оскільки відгуки на Yelp залишаються користувачами, існує ризик, що деякі відгуки можуть бути необ'єктивними або недостовірними. Користувачам слід бути уважними при оцінці інформації та приймати рішення на основі власного розсуду.

2. Обмежена кількість інформації: Незважаючи на широкий обсяг інформації, доступної на Yelp, іноді може бути обмежена кількість деталей про певні кафе та ресторани. Деяка інформація, наприклад, про актуальні ціни або розклад роботи, може бути не завжди точною або повною.

3. Залежність від активності користувачів: Якщо певне кафе або ресторан має невелику кількість відгуків або низький рейтинг на Yelp, це може бути пов'язано з низькою активністю користувачів. Це може вплинути на представлену на сайті інформацію та об'єктивність оцінки.

4. Відсутність можливості користування без підключення до інтернету.

Сервіс OpenTable (https://www.opentable.com/) має такі переваги і недоліки:

Переваги:

1. Зручне бронювання: Сервіс OpenTable надає можливість зручно бронювати столи у ресторанах онлайн. Користувачі можуть обирати бажаний час та кількість осіб, щоб забезпечити собі місце у ресторані без зайвих зусиль.
2. Наявність додатку для смартфонів.
3. Широкий вибір ресторанів: OpenTable пропонує велику базу даних ресторанів, серед яких можна знайти різні типи кухні та атмосферу. Це дає можливість користувачам знайти ресторан, що відповідає їхнім перевагам та настрою.

Недоліки:

1. Залежність від доступності столів: Деякі популярні ресторани можуть мати обмежену доступність столів через високий попит. Це може ускладнити заведення бронювання через OpenTable та вимагати додаткового планування або резервування заздалегідь.
2. Інформаційна обмеженість: Відомості про ресторани на OpenTable можуть бути обмеженими в порівнянні з іншими джерелами. Деякі деталі, такі як меню, можуть бути недоступними або не повністю оновлені. Це може обмежити користувачів у отриманні повної інформації про ресторан перед відвідуванням.
3. Не завжди коректна робота бронювань.
4. Відсутність можливості користування без підключення до інтернету.

Аналіз було проведено на основі відгуків користувачів цих сервісів. За результатами аналізу були визначені аспекти які треба ураховувати при створенні сервісу для пошуку кафе та ресторанів. Основні аспекти: зручний та інтуїтивно зрозумілий інтерфейс, детальний опис закладів, наявність фільтрації з

урахуванням потреб користувачів, надійний спосіб бронювання, відображення місцезнаходження закладу на мапі, можливість користування сервісом на різних операційних системах, зручний додаток для смартфонів, можливість користування сервісом без підключення до Інтернету.

1.3 Засоби розробки веб-сервісів

Для розробки сервісів з пошуку кафе та ресторанів існує широкий спектр засобів та технологій, які допомагають реалізувати функціональність та забезпечити зручний і ефективний досвід користувачам. У цьому контексті, окрім популярних баз даних, таких як MySQL, PostgreSQL та MongoDB, також варто згадати про можливість використання спеціалізованих API, які надають доступ до зовнішніх сервісів, таких як Google Maps або Yelp. Ці API надають широкі можливості для отримання географічних даних, рейтингів, відгуків та іншої інформації про кафе та ресторани.

Для розробки інтерфейсу сервісу також використовуються різні технології та інструменти. Наприклад, HTML, CSS та JavaScript є основою для створення веб-інтерфейсу. Крім того, популярними варіантами є використання фреймворків, таких як React або Angular, які дозволяють розробляти реактивні та інтерактивні користувацькі інтерфейси. Ці фреймворки надають широкий набір інструментів та компонентів для зручної та ефективної розробки.

При реалізації функціональності пошуку місць та їх відображення на мапі, можна скористатися спеціалізованими API та інструментами. Один з таких інструментів - це фреймворк Leaflet, який надає потужні засоби для роботи з географічними даними та мапами. З його допомогою можна легко відображати місцезнаходження кафе та ресторанів на інтерактивній мапі, налаштовувати стилізацію та додавати різноманітні елементи управління.

Для взаємодії з сервером та отримання даних про кафе та ресторани також використовуються спеціалізовані інструменти. Наприклад, Axios є одним з таких

інструментів і є простим та потужним клієнтом HTTP. Він дозволяє зручно здійснювати запити до серверу для отримання списку кафе та ресторанів, а також додаткової інформації про них, наприклад, рейтинги та відгуки.

У сучасних додатках з пошуку кафе та ресторанів мобільні технології відіграють важливу роль. При розробці мобільних додатків використовуються різні підходи та фреймворки, які допомагають створювати кросплатформні додатки, що працюють як на Android, так і на iOS.

Один з таких фреймворків - Flutter. Він надає можливість розробляти кросплатформенні мобільні додатки за допомогою одного коду. Це означає, що розробник може написати код один раз і використовувати його для створення додатків для різних мобільних платформ. Flutter дозволяє швидко розробляти додатки зі зручним інтерфейсом користувача та високою продуктивністю.

Крім того, важливо згадати про технологію Progressive Web Apps (PWA), яка набуває все більшої популярності в розробці мобільних додатків. PWA поєднує переваги веб-сайтів та мобільних додатків, дозволяючи створювати додатки, які можна встановлювати на пристрої, подібно до звичайних мобільних додатків. PWA можуть працювати в автономному режимі, мати доступ до апаратних можливостей пристрою та надавати зручний інтерфейс для користувача.

Таким чином, розробка мобільних додатків з використанням фреймворків, таких як Flutter, або використання технології PWA, дозволяє створювати зручні та функціональні додатки, які працюють на різних мобільних платформах та надають зручний користувацький інтерфейс.

Проект також може включати використання api для відображення інформації про заклади. Прикладом такої api є Yelp- цей сервіс має велику базу закладів з інформацією про них. Іншим варіантом є використання власної бази даних закладів. Ця база даних забезпечить збереження та організацію інформації

про кафе та ресторани, включаючи їхні адреси, рейтинги, відгуки та інші характеристики.

Наша власна база даних закладів дозволить нам зберігати актуальну та специфічну інформацію про кафе та ресторани, яку ми можемо оновлювати та розширювати за потребою. Це дає нам більшу гнучкість і контроль над даними, а також забезпечує більшу точність й персоналізувати результати пошуку для користувачів.

Для відображення карт з пошуку кафе та ресторанів можна використовувати різні API, які надають доступ до картографічних сервісів та геоданих. Найпопулярнішими API для цих цілей є Google Maps API, Mapbox API та OpenStreetMap API.

Google Maps API є одним з найбільш розповсюджених та повнофункціональних API для роботи з картами. Він надає доступ до широкого спектру функцій, включаючи відображення мап, маршрутне планування, маркери, інформаційні вікна та інші елементи для взаємодії з картою. Google Maps API також має детальні геодані, які дозволяють отримувати інформацію про місця, включаючи кафе, ресторани, їх рейтинги та відгуки.

Mapbox API є ще одним популярним варіантом для відображення карт. Це набір інструментів та сервісів, які дозволяють створювати кастомні картографічні рішення. Mapbox API надає гнучкість у відображенні та налаштуванні карт, а також має можливості для розробки інтерактивних елементів та стилізації карт. Також Mapbox API надає доступ до геоданих, які можуть бути використані для отримання інформації про кафе, ресторани та їх характеристики.

OpenStreetMap API заснований на проекті OpenStreetMap, який є вільною та відкритою картографічною системою. Це означає, що дані про місця та геомапу доступні для використання та редагування спільнотою користувачів. OpenStreetMap API дозволяє відображати карти, додавати маркери, шари та інші

елементи. Крім того, цей API надає можливість отримувати геодані, які можуть бути використані для отримання інформації про кафе та ресторани.

Кожен з цих API має свої особливості та можливості. Вибір конкретного API залежить від потреб проекту та вимог до функціональності та доступності даних. Ці API дозволяють розробникам створювати потужні та інтерактивні карти, які полегшують користувачам пошук кафе та ресторанів та надають багато корисних функцій для навігації та взаємодії з мапою.

У цьому розділі ми розглянули деякі засоби та технології, що використовуються при розробці сервісів з пошуку кафе та ресторанів.

1.4 Постановка задачі

У сучасному світі додатки та сервіси, що надають інформацію та допомагають користувачам знаходити та вибирати кафе та ресторани для прийому їжі, стали невід'ємною частиною нашого повсякденного життя. Популярні сервіси, такі як Yelp та OpenTable, здатні задовольнити деякі потреби користувачів, проте кожен з них має свої переваги та недоліки.

Отже, метою даного дипломного проекту є розробка нового сервісу для пошуку кафе та ресторанів, який об'єднає найкращі характеристики, врахує унікальні потреби та вимоги користувачів, а також використає сучасні підходи і засоби при створенні сервісу. Цей проект має на меті створити зручний та ефективний інструмент для пошуку та вибору місць харчування, що відповідають вимогам користувачів.

Основні задачі даного проекту включають розробку зручного пошуку, надання інформації про рейтинги та відгуки, навігацію та маршрутизацію, зручне бронювання столиків та розробку мобільного додатку.

Пошуковий функціонал дозволить користувачам швидко та зручно знаходити кафе та ресторани, враховуючи різні критерії, такі як

місцезнаходження, тип кухні, цінова категорія, рейтинг та інші. Завдяки сучасним алгоритмам пошуку та фільтрації даних, користувачі зможуть легко знайти варіанти, що відповідають їхнім потребам та вимогам.

Для зручної навігації користувачів до обраного закладу будуть використовуватися дані місцезнаходження. Це забезпечить точну та оптимальну навігацію, допомагаючи знайти адресу призначення.

Сервіс також має надати можливість зручного бронювання столиків у ресторанах. Користувачі матимуть можливість обрати бажаний час та кількість осіб, щоб забезпечити собі місце без зайвих зусиль. Механізми бронювання та календарні функції забезпечать зручність та ефективність процесу бронювання.

Окрім того, додаток буде доступний для використання на мобільних пристроях, що дозволить користувачам зручно користуватися ним навіть під час поїздок чи відвідування нових місць. Застосування сучасних мобільних технологій та інструментів розробки забезпечить оптимальну роботу додатку на різних пристроях і забезпечить зручне взаємодію з користувачами.

Цей проект має на меті створення інноваційного сервісу, що відповідає сучасним вимогам та потребам користувачів. Використання сучасних технологій, методів аналізу даних та зручного інтерфейсу дозволить поліпшити гастрономічний досвід користувачів і залучити активну спільноту любителів гастрономії до обміну враженнями та рекомендаціями.

У процесі реалізації даного проекту будуть використані сучасні технології розробки додатків, такі як мови програмування, фреймворки та інструменти. Для забезпечення зручного та ефективного використання додатку буде проведено тестування та оптимізацію його функціональності.

Очікується, що розроблений додаток буде допомагати користувачам знаходити та вибирати найкращі місця для прийому їжі, враховуючи їхні потреби

та вимоги. Крім того, він надасть можливість користувачам ділитися своїми враженнями та рекомендаціями, сприяючи розвитку активної спільноти любителів гастрономії.

В цілому, розробка цього сервісу відповідає сучасним вимогам та потребам користувачів, що шукають зручні та ефективні інструменти для пошуку та вибору місць. Цей дипломний проект спрямований на створення інноваційного сервісу, який відповідає вимогам сьогодення та сприяє поліпшенню користувацького.

2.Проектування інформаційної системи

2.1 Вибір методу проектування

У проектуванні сервісу пошуку кафе та ресторанів можна використовувати різні методології, що сприяють ефективній організації процесу розробки та досягненню успішних результатів. Далі ознайомимося з кожною методологією детальніше.

2.1.1Огляд різних методологій проектування

1. Водопадна модель: це послідовна структура, де кожен етап розробки починається після завершення попереднього. Спочатку проводиться аналіз вимог, потім проектування, реалізація, тестування та впровадження. Цей підхід дозволяє чітко структурувати роботу та забезпечити деталізацію вимог до системи.
2. Гнучка (агільна) методологія: це ітеративний підхід, де розробка поділяється на короткі ітерації. Кожна ітерація включає планування, розробку, тестування та оцінку результатів. Гнучка методологія акцентується на співпраці між замовником та розробниками, швидкому реагуванні на зміни та постійному вдосконаленні системи.
3. Методологія прототипування: вона передбачає створення прототипу системи для оцінки функціональності та взаємодії з користувачами. Прототипи дозволяють швидко отримувати фідбек та вносити зміни до проекту перед фінальною реалізацією.
4. Каскадна модель: цей підхід передбачає послідовну розробку, де кожен етап залежить від завершення попереднього. Вона включає аналіз, проектування, реалізацію, тестування та впровадження. Цей метод дозволяє систематично пройти крок за кроком по розробці системи.
5. Ітеративне уточнення: цей підхід передбачає постійні зміни та уточнення системи протягом розробки. Розробники працюють над декількома ітераціями, кожна з яких додає нові функції або вдосконалення до системи.

6. Спіральна модель: цей підхід поєднує елементи водопадної моделі та ітеративного уточнення. Він передбачає послідовність етапів, проте на кожному етапі можливі ітерації, що дозволяє уточнювати вимоги та ризики проекту.
7. Методологія швидкої розробки (RAD): це підхід, спрямований на швидку розробку системи. Він використовує готові компоненти, шаблони та інструменти для прискорення процесу розробки.

Вибір методології для проектування сервісу з пошуку кафе та ресторанів залежить від конкретних вимог проекту, розміру команди, термінів та бюджету. Також можливо комбінувати різні методології або адаптувати їх до потреб проекту для досягнення успішних результатів.

2.1.2 Вибір методології, яка найкраще підходить для даного проекту

При виборі методології для розробки сервісу пошуку кафе та ресторанів важливо враховувати особливості проекту та його вимоги. На основі розглянутих раніше методологій можна визначити найбільш підходящий підхід для даного проекту.

Один із можливих варіантів - гнучка (агільна) методологія, яка сприяє швидкому реагуванню на зміни та постійному вдосконаленню системи. Завдяки ітеративному підходу, розробка буде проводитись у коротких циклах, що дозволить швидко отримувати фідбек та вносити необхідні зміни. Це особливо корисно в проектах, де вимоги можуть змінюватися або уточнюватися з часом.

Також можливо використати методологію прототипування, яка передбачає створення прототипів системи для оцінки функціональності та взаємодії з користувачами. Це дозволить швидко перевірити та вдосконалити інтерфейс користувача, а також отримати фідбек від потенційних користувачів.

Для проектів з обмеженими термінами та бюджетом можливо розглянути методологію швидкої розробки (RAD). Цей підхід передбачає використання

готових компонентів та інструментів для прискорення розробки. Завдяки цьому можна зекономити час і ресурси, зосередившись на основних функціях та потребах користувачів.

При виборі методології необхідно також враховувати розмір команди та можливість співпраці між розробниками та замовником. Важливо, щоб обрана методологія забезпечувала зручну комунікацію та співпрацю, що сприятиме успішному завершенню проекту.

Однією з розглянутих методологій проектування є спіральна модель. Ця модель використовує ітераційний підхід до розробки, де процес розбивається на послідовні ітерації або фази. Кожна ітерація складається з планування, аналізу ризиків, розробки та оцінки.

У контексті дипломної роботи з проектування сервісу пошуку кафе та ресторанів, спіральна модель може бути використана для ефективного впровадження та розробки системи. Основна перевага спіральної моделі полягає в тому, що вона дозволяє зосередитись на аналізі ризиків та виявленні проблем на ранніх етапах проекту. Це дозволяє уникнути потенційних проблем, зменшити витрати та забезпечити якість розробки.

У контексті дипломної роботи, використання спіральної моделі може допомогти виявити можливі ризики та проблеми, пов'язані з розробкою сервісу пошуку. Наприклад, аналізуючи можливі ризики, можна заздалегідь спланувати стратегії їх управління та запобігання. Крім того, спіральна модель дає можливість проводити експерименти та вносити зміни на ранніх етапах розробки, що сприяє швидкому прототипуванню та вдосконаленню системи.

Іншою перевагою спіральної моделі є її гнучкість та адаптивність. Ця модель дозволяє змінювати пріоритети та вимоги проекту в процесі розробки, що особливо важливо в дипломній роботі, де вимоги можуть змінюватися з часом.

Таким чином, застосування спіральної моделі у дипломній роботі з проектування сервісу пошуку кафе та ресторанів дозволить ефективно впроваджувати систему, виявляти й управляти ризиками, проводити експерименти та забезпечити гнучкість у виробництві.

Наступним кроком після вибору методології буде детальне проектування структури системи, схеми бази даних та клієнтської частини. Комбінація вибраної методології та правильно спроектованих компонентів допоможе створити функціональний та зручний сервіс пошуку кафе та ресторанів, що задовольнятиме потреби користувачів та забезпечуватиме їм зручну інтерактивну платформу.

2.2 Проектування структури системи

Для розробки системи пошуку кафе та ресторанів необхідно визначити основні компоненти та модулі, які будуть входити до складу системи. Основними компонентами такої системи можуть бути. Далі розглянемо основні компоненти та модулі майбутньої системи.

2.2.1 Визначення основних компонентів та модулів системи

1. Користувацький інтерфейс: цей компонент включає в себе всі елементи, які користувач буде бачити та з якими буде взаємодіяти при використанні сервісу. Сюди можуть входити екрани пошуку, списки результатів, детальна інформація про кафе та ресторани, фільтри, картки вибору тощо.
2. База даних: цей компонент відповідає за зберігання та управління інформацією про кафе та ресторани. В базі даних можуть зберігатися дані про назви, адреси, рейтинги, типи кухні, розклад роботи тощо. Це дозволяє швидко та ефективно отримувати потрібну інформацію при пошуку.
3. Модуль пошуку: цей компонент відповідає за здійснення пошуку кафе та ресторанів на основі введених користувачем параметрів. Він використовує базу даних для знаходження відповідних записів та відображення результатів пошуку.

4. Модуль відгуків та рейтингу: цей компонент дозволяє користувачам залишати відгуки та оцінювати кафе та ресторани. Відгуки та рейтинги можуть бути використані для покращення вибору користувачів та надання інформації про якість закладів.
5. Модуль мапи: цей компонент відображає географічне розташування кафе та ресторанів на карті. Користувачі можуть використовувати цей модуль для локалізації та зручного вибору закладів у певній області.

Враховуючи потреби дипломної роботи, визначені компоненти та модулі відповідають основним функціональним вимогам системи пошуку кафе та ресторанів. Запропонована архітектура системи дозволяє ефективно організувати роботу з користувачем, зберігати та обробляти інформацію про заклади, а також забезпечувати зручний пошук та вибір.

2.2.2 Розгляд можливих взаємозв'язків та взаємодій між компонентами.

У системі пошуку кафе та ресторанів існують різні компоненти, які взаємодіють між собою для забезпечення функціональності та зручного користувацького досвіду. Один з головних взаємозв'язків - між модулем пошуку та модулем відображення результатів. Модуль пошуку отримує вхідні дані від користувача, такі як розташування або критерії пошуку, і виконує відповідні запити до бази даних для отримання результатів. Після цього модуль відображення результатів отримує ці дані та відображає їх у зручному форматі для користувача, наприклад, у вигляді списку або на мапі.

Інший важливий взаємозв'язок - між модулем відгуків та модулем рейтингу. Модуль відгуків дозволяє користувачам залишати свої відгуки про заклади, а також переглядати відгуки інших користувачів. Ці відгуки можуть містити інформацію про якість обслуговування, страви та загальну атмосферу закладу. Модуль рейтингу використовує ці відгуки для обчислення загального рейтингу

кожного закладу. Це дозволяє користувачам швидко оцінювати якість різних закладів та знаходити найкращі варіанти.

Крім того, модуль авторизації взаємодіє з іншими компонентами системи. Після успішної авторизації користувач має доступ до свого особистого облікового запису, де може зберігати улюблені заклади або залишати власні відгуки. Модуль авторизації також забезпечує безпеку та конфіденційність даних користувача, запобігаючи несанкціонованому доступу.

У системі пошуку кафе та ресторанів також присутній модуль адміністрування. Цей модуль відповідає за управління та контроль системи з боку адміністратора.

Модуль адміністрування забезпечує доступ до адміністративних функцій, таких як додавання та видалення закладів, керування користувачами, модерація відгуків та управління іншими аспектами системи. Адміністратор може мати спеціальні права доступу, які дозволяють йому здійснювати ці дії та забезпечувати безпеку та якість системи.

Модуль адміністрування дозволяє адміністратору ефективно керувати системою та забезпечувати її надійну роботу. Він забезпечує можливість швидкого реагування на зміни, вирішення проблем та встановлення необхідних параметрів для оптимальної роботи системи.

Ці взаємозв'язки та взаємодії між компонентами системи допомагають забезпечити зручну та функціональну роботу сервісу пошуку кафе та ресторанів. Користувачі зможуть швидко знаходити потрібні заклади, переглядати відгуки інших користувачів та здійснювати персоналізовану взаємодію з системою.

2.2.3 Створення блок-схеми архітектури системи

Архітектура системи може бути організована наступним чином:

1. Клієнтська частина: Користувачі взаємодіють з системою через мобільний додаток або веб-інтерфейс. Вони можуть вводити пошукові запити, переглядати інформацію про заклади, залишати відгуки та оцінки.
2. Серверна частина: Це центральний обчислювальний модуль, який обробляє запити від клієнтів і забезпечує взаємодію з базою даних. Він містить основну логіку системи, включаючи модулі пошуку, авторизації, відгуків та адміністрування.
3. База даних: Це сховище, де зберігаються дані про кафе та ресторани, включаючи їх атрибути, відгуки, оцінки та іншу інформацію. Вона використовується для збереження та організації даних, які потрібні для ефективного виконання пошукових запитів та надання інформації користувачам.
4. Модуль пошуку: Цей модуль відповідає за обробку пошукових запитів користувачів і повертає відповідні результати. Він використовує дані з бази даних та алгоритми пошуку для знаходження найбільш підходящих закладів відповідно до введеного користувачем запиту.
5. Модуль авторизації: Цей модуль відповідає за аутентифікацію користувачів і забезпечення безпеки доступу до системи. Він перевіряє облікові дані користувача та забезпечує доступ до відповідних функцій системи залежно від прав доступу.
6. Модуль відгуків: Цей модуль дозволяє користувачам залишати відгуки та оцінки про заклади. Він забезпечує можливість відображення та взаємодії з відгуками, а також оновлення рейтингу закладів на основі цих оцінок.
7. Модуль адміністрування: Цей модуль призначений для управління системою та закладами. Адміністратори мають доступ до додаткових функцій, таких як додавання та видалення закладів, редагування інформації та керування користувачами.

Така архітектура системи забезпечує ефективну взаємодію між компонентами та забезпечує користувачам зручний інтерфейс для пошуку кафе та ресторанів, адміністрування та надання відгуків.

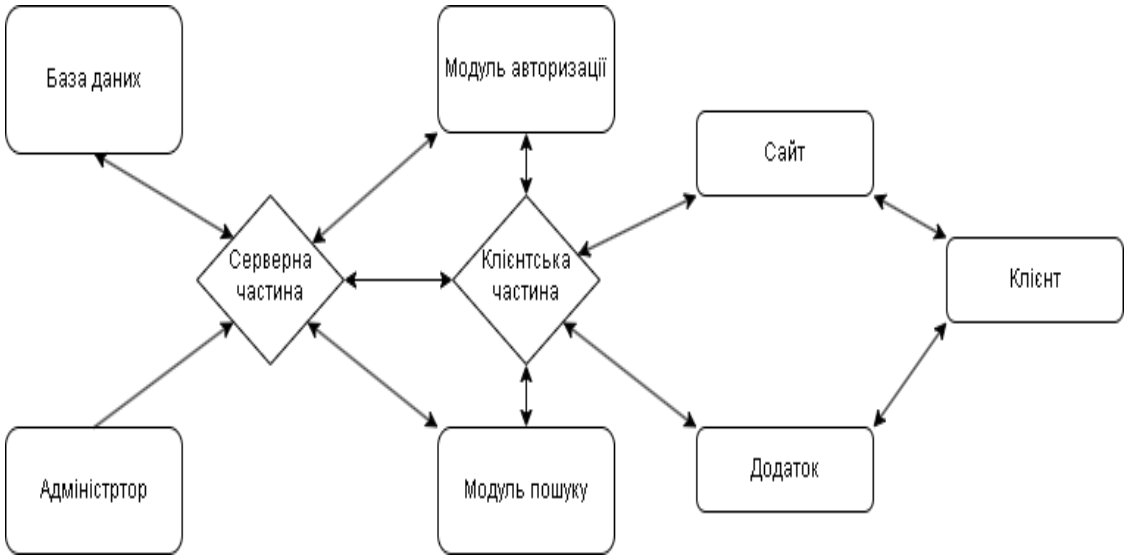

Рис. 2.2.3.1 – Схема архітектури системи

2.3 Проектування схеми та опису бази даних

2.3.1 Визначення необхідних сутностей та атрибутів для зберігання даних.

Визначення необхідних сутностей та атрибутів для зберігання даних полягає у визначенні основних складових системи, які використовуються для організації та збереження інформації. Сутності визначаються як основні об'єкти, про які ми хочемо зберігати дані, наприклад, заклади харчування. Атрибути вказують на конкретні характеристики цих сутностей, такі як назва закладу, адреса, тип кухні тощо.

Однією з основних сутностей є "Заклад". Вона включає інформацію про кафе та ресторани, таку як назва, адреса, тип кухні, рейтинг, середня оцінка та інші важливі характеристики. Кожен заклад може мати такі атрибути, як назва і адреса

Додатковою сутністю може бути "Користувач". Вона включатиме інформацію про користувачів системи, таку як ім'я, прізвище, електронна пошта,

пароль та інші персональні дані. Атрибути цієї сутності можуть включати ім'я та електронну пошту.

Для зв'язку між сутностями "Заклад" та "Користувач" можна визначити додаткову сутність "Відгук". Вона включатиме інформацію про відгуки, залишені користувачами про певний заклад, такі як текст відгуку, оцінка та дата. Атрибути цієї сутності можуть включати текст відгуку, оцінку та дату.

Також для реалізації функції бронювання закладу необхідно створити сутність бронювання, яка буде містити інформацію про користувача який виконав бронювання, ресторан, кількість місць, коментар, час на який потрібен стіл, дату створення запиту, телефонний номер клієнту та статус для підтвердження.

Ще було відмічено, що для хорошого функціоналу треба реалізувати список улюблених закладів для цього теж створимо окрему сутність "улюблене". У якій помістимо унікальний ідентифікатор користувача та список унікальних ідентифікаторів закладів.

Визначення цих сутностей та атрибутів дозволить ефективно зберігати та організовувати дані, необхідні для роботи системи пошуку кафе та ресторанів.

Для ефективного зберігання та організації даних необхідно визначити основні сутності, їх взаємозв'язки та відповідні атрибути.

2.3.2 Розробка схеми бази даних, включаючи таблиці, зв'язки та індекси

Розробка схеми бази даних включає в себе створення таблиць, визначення зв'язків між ними та встановлення індексів для поліпшення продуктивності запитів.

Таблиці в базі даних використовуються для збереження структурованої інформації. Кожна таблиця представляє собою сукупність стовпців, які

визначають типи даних та атрибути для кожного запису. Наприклад, таблиця "Заклади харчування" може містити стовпці для назви, адреси, типу кухні тощо.

Зв'язки в базі даних використовуються для встановлення зв'язку між таблицями. Зв'язок визначається за допомогою спеціальних ключів, які пов'язують записи в одній таблиці з записами в іншій таблиці. Наприклад, в таблиці "Відгуки" може бути зв'язок з таблицею "Заклади харчування", що дозволяє пов'язати відгук з відповідним закладом.

Індекси в базі даних використовуються для покращення продуктивності запитів. Вони створюються для певних стовпців таблиць і дозволяють швидко знаходити записи, використовуючи ці стовпці як ключі. Наприклад, індекс може бути створений для стовпця "Назва закладу" у таблиці "Заклади харчування", що дозволить швидко здійснювати пошук закладу за назвою.

Створення схеми бази даних з таблицями, визначенням зв'язків та встановленням індексів допомагає організувати та забезпечити ефективний доступ до даних. Це дозволяє виконувати різноманітні запити та операції з даними, забезпечуючи продуктивну роботу з системою бази даних.

Інфологічне проектування

Етап інфологічного проектування є важливою частиною процесу розробки інформаційної системи. На цьому етапі відбувається аналіз та визначення вимог до системи, а також створення інфологічної моделі.

Починаючи з аналізу вимог, вивчаються потреби користувачів та бізнес-процеси, що пов'язані з системою. Збираються вимоги до функціональності, ефективності, безпеки та інших аспектів системи.

Далі проводиться моделювання інфологічної структури системи. Це включає створення схеми бази даних, опис сутностей та їх атрибутів, визначення

зв'язків між сутностями. Модель може бути представлена у вигляді діаграми, що відображає структуру і залежності між сутностями.

Після створення інфологічної моделі виконується її аналіз та перевірка на відповідність вимогам. Результати аналізу використовуються для внесення необхідних змін та вдосконалення моделі.

Етап інфологічного проектування є фундаментальним кроком у розробці інформаційної системи, оскільки він визначає основні концептуальні аспекти системи. Вірне інфологічне проектування допомагає забезпечити ефективну та гнучку систему, яка відповідає потребам користувачів і сприяє успішній реалізації проекту.

Сутності які можна виділити у даному проекті: ресторан, користувач, адміністратор. Додаткові сутності: відгук, бронювання, улюблене.

Сутність "Ресторан": Ця сутність представляє собою ресторан або заклад громадського харчування. Вона має атрибути, такі як назва ресторану, адреса, тип кухні, рейтинг тощо. Ресторан може мати багато відгуків, бронювань і може бути доданий до списку "улюблених" користувачів.

Сутність "Користувач": Ця сутність відображає користувача системи. Вона має атрибути, такі як ім'я, прізвище, електронна пошта, пароль тощо. Користувач може залишати відгуки про ресторани, робити бронювання та додавати ресторани до списку "улюблених". Він також може мати роль адміністратора.

Сутність "Адміністратор": Ця сутність відображає адміністратора системи, який має розширені права доступу та можливості управління. Адміністратор може додавати та видаляти ресторани, керувати відгуками, управляти бронюваннями та здійснювати інші адміністративні дії в системі.

Сутність "Відгук": Ця сутність представляє відгук користувача про певний ресторан. Вона містить атрибути, такі як текст відгуку, оцінка, дата тощо. Кожен відгук пов'язаний з конкретним користувачем та рестораном.

Сутність "Бронювання": Ця сутність відображає бронювання столика або послуги у ресторані. Вона містить атрибути, такі як дата та час бронювання, кількість осіб, підтвердження тощо. Кожне бронювання пов'язане з конкретним користувачем та рестораном.

Сутність "Улюблене": Ця сутність відображає список "улюблених" ресторанів користувача. Вона містить атрибути, що посилаються на конкретного користувача та ресторан.

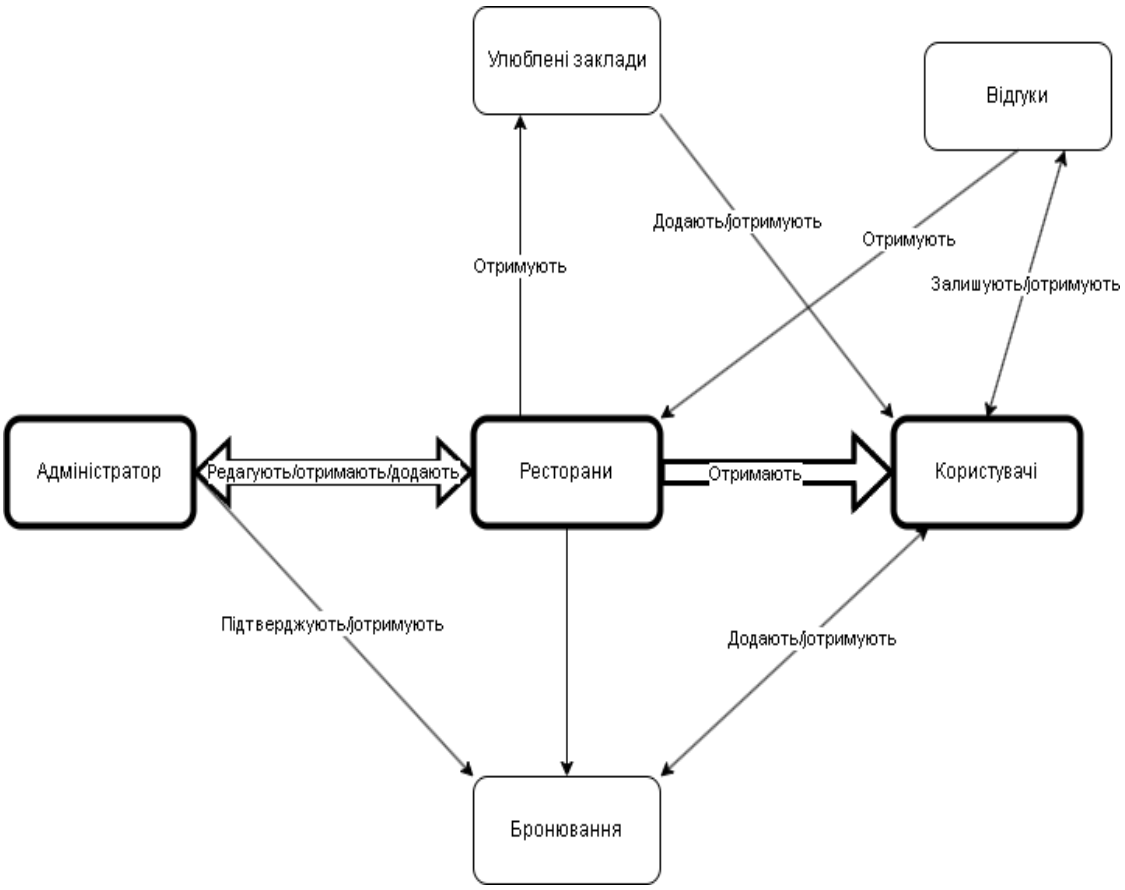

Рис. 2.3.2.1 – Схема зв’язків між таблицями

Ці сутності та їх взаємозв'язки утворюють основу інфологічної моделі для проекту. Вона дозволяє зберігати та управляти даними про ресторани, користувачів, відгуки, бронювання та список улюблених закладів.

Даталогічне проектування

При даталогічному проектуванні для цього проекту використовується база даних MySQL разом з програмою phpMyAdmin для зручного керування базою даних. MySQL використовує реляційну модель даних, що є одним з найпоширеніших та найбільш використовуваних типів структури даних у базах даних. Реляційна модель дозволяє організовувати дані у вигляді таблиць, які мають рядки (кортежі) і стовпці (атрибути), що спрощує роботу з даними та виконання різноманітних запитів.

MySQL є реляційною системою керування базами даних (СКБД), яка підтримує реляційну модель даних. Вона надає ефективні методи зберігання, організації та обробки даних, що відповідають принципам реляційної моделі.

Для розробки прикладного програмного забезпечення з використанням MySQL можна використовувати різні технології та засоби програмування, зокрема PHP, Python, Java, C# та інші. MySQL має драйвери та інтерфейси для багатьох популярних мов програмування, що дозволяє зручно взаємодіяти з базою даних у контексті розробки програм.

Одним з таких засобів прикладного програмування для роботи з MySQL є phpMyAdmin. Це веб-прикладний інтерфейс, який надає зручні можливості для адміністрування бази даних MySQL через веб-браузер.

Для розробки проекту також можна використовувати openServer, яке є одним з популярних комп'ютерних середовищ для розробки та тестування веб-додатків. openServer надає зручне середовище, що включає в себе веб-сервер Apache, мови програмування PHP та інші компоненти, включаючи MySQL.

Використання openServer дозволяє швидко налаштувати та запустити локальне середовище, де можна розробляти та тестувати веб-додатки, включаючи роботу з базою даних MySQL. Це зручно, оскільки дозволяє вести розробку на власному комп'ютері без необхідності підключення до зовнішнього сервера.

Під час проектування було створено таблиці для кожної з сутностей, що були ідентифіковані на інфологічному рівні.

Таблиця "Ресторан" містить стовпці для зберігання назви ресторану, адреси, типу кухні та іншої інформації про ресторан. Також в таблиці "Ресторан" є зовнішні ключі, що посилаються на інші таблиці, наприклад, для зв'язку з відгуками та бронюваннями.

Таблиця "Користувач" зберігає дані про користувачів, такі як ім'я, прізвище, електронна пошта та пароль. Ця таблиця також містить додаткові стовпці для зберігання інформації про роль користувача, наприклад, чи є він адміністратором.

Таблиця "Відгук" містить дані про відгуки користувачів про ресторани. Вона включає стовпці для тексту відгуку, оцінки та дати відгуку. Кожен відгук пов'язаний з конкретним користувачем та рестораном за допомогою зовнішніх ключів.

Таблиця "Бронювання" використовується для зберігання даних про бронювання столиків або послуг у ресторані. Вона містить стовпці для дати та часу бронювання, кількості осіб та підтвердження бронювання.

Таблиця "Улюблене" відображає список "улюблених" ресторанів користувача. Вона містить стовпці, які зв'язують конкретного користувача з його улюбленими ресторанами.

Кожна таблиця має відповідні стовпці та обмеження, що допомагають зберігати та організовувати дані в базі даних. Це дозволяє зручно управляти

інформацією про ресторани, користувачів, відгуки, бронювання та улюблені ресторани.

Після розгляду сутностей та їх взаємозв'язків формується наступна даталогічна модель, яка відображає основну бізнес-логіку системи:

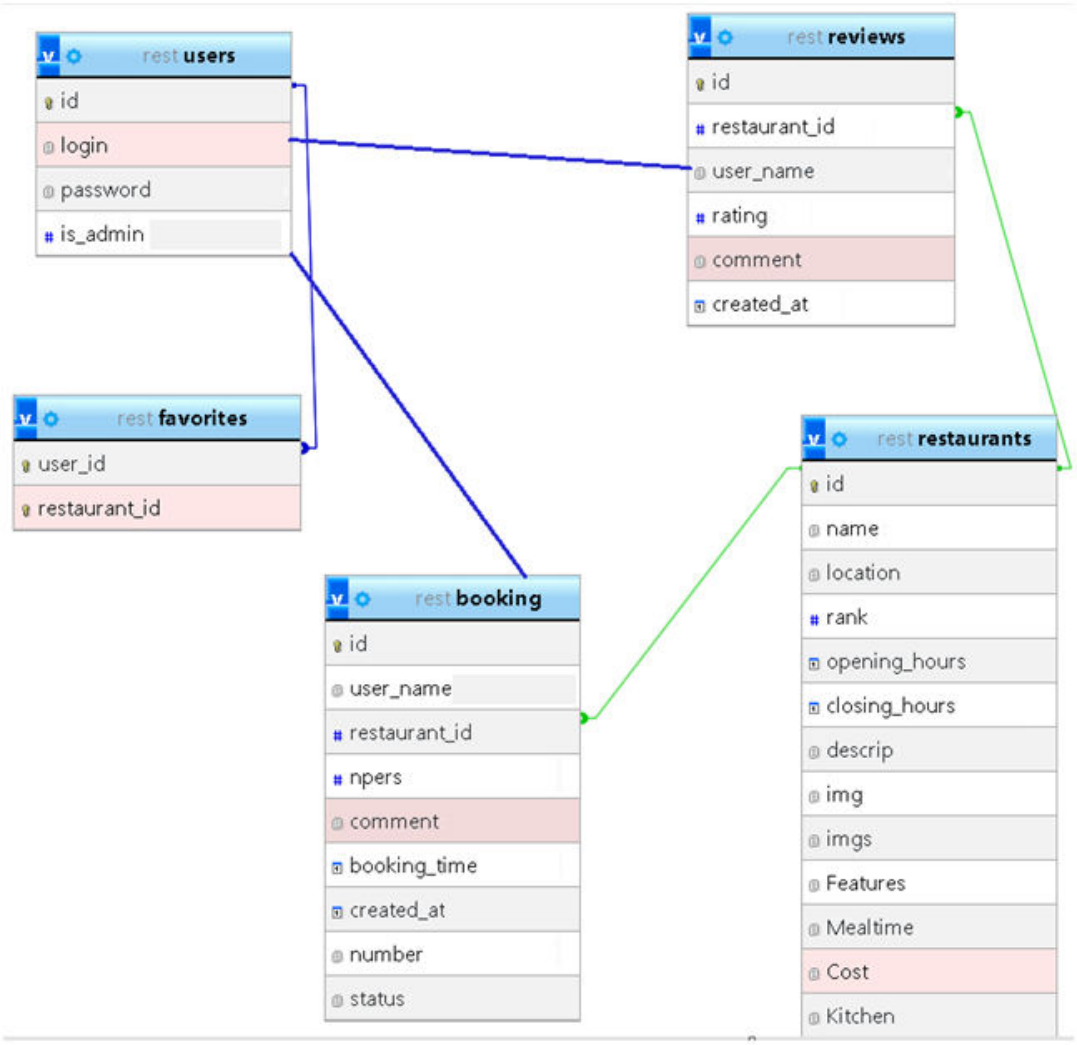

Рис. 2.3.2.2 – Дизайн бази даних у phpMyAdmin

Фізичне проектування

Фізичне проектування бази даних включає розробку конкретної структури та організацію даних для забезпечення ефективного зберігання, доступу та обробки інформації. Під час фізичного проектування вирішуються питання щодо

вибору платформи, налаштування параметрів бази даних та оптимізації роботи з нею.

Одним з перших кроків у фізичному проектуванні є вибір типу системи управління базами даних (СУБД). У даному проекті було обрано СУБД MySQL, оскільки вона є потужною, надійною та широко використовується в галузі веб-розробки. MySQL підтримує реляційну модель даних, що дозволяє зручно організувати дані у вигляді таблиць зі зв'язками між ними.

Для роботи з базою даних MySQL використовується програма phpMyAdmin, яка надає зручний інтерфейс для адміністрування бази даних. phpMyAdmin дозволяє створювати таблиці, визначати їх структуру, виконувати запити та керувати даними.

Технологія та засоби прикладного програмування, що використовуються у проекті, залежать від конкретних вимог та веб-фреймворку, який використовується для розробки. Наприклад, для створення динамічних веб-сторінок можуть використовуватись мови програмування, такі як PHP, Python або JavaScript, разом з фреймворками, такими як Laravel, Django або Node.js.

Для забезпечення фізичної реалізації бази даних необхідно також вибрати конкретне комп'ютерне середовище, на якому буде працювати СУБД і програми розробки. У даному проекті в якості прикладу може використовуватись комп'ютерне середовище OpenServer, яке надає зручну конфігурацію серверного середовища для розробки та тестування веб-додатків.

Цей этап включає й вибір та налаштування типів даних, які будуть використовуватись для збереження інформації. Важливо обрати відповідні типи даних для кожного поля таблиці з урахуванням характеру даних, обсягу і вимог до продуктивності системи.

Наприклад, для збереження текстової інформації можуть використовуватись типи даних, такі як VARCHAR або TEXT, залежно від очікуваної довжини тексту. Для збереження числових значень можуть бути використані типи INT, FLOAT або DECIMAL, в залежності від точності та масштабу чисел. Для збереження дат та часу можуть бути використані типи DATE, TIME або DATETIME.

При виборі типів даних також слід враховувати вимоги до ефективності та економії ресурсів. Наприклад, використання оптимальних розмірів для полів може зменшити обсяг пам'яті, необхідний для збереження даних, а використання відповідних типів для операцій порівняння та обчислень може покращити продуктивність запитів до бази даних.

Таким чином, під час фізичного проектування бази даних важливо ретельно вибирати та налаштовувати типи даних, щоб забезпечити ефективне збереження та обробку інформації у відповідності до вимог проекту.

Фізичне проектування бази даних вимагає ретельного планування та налаштування, щоб забезпечити оптимальну продуктивність та надійність системи. Враховуючи особливості проекту та використовувані технології, можна створити ефективну базу даних, яка забезпечить потрібний функціонал та швидкодію веб-сервісу.

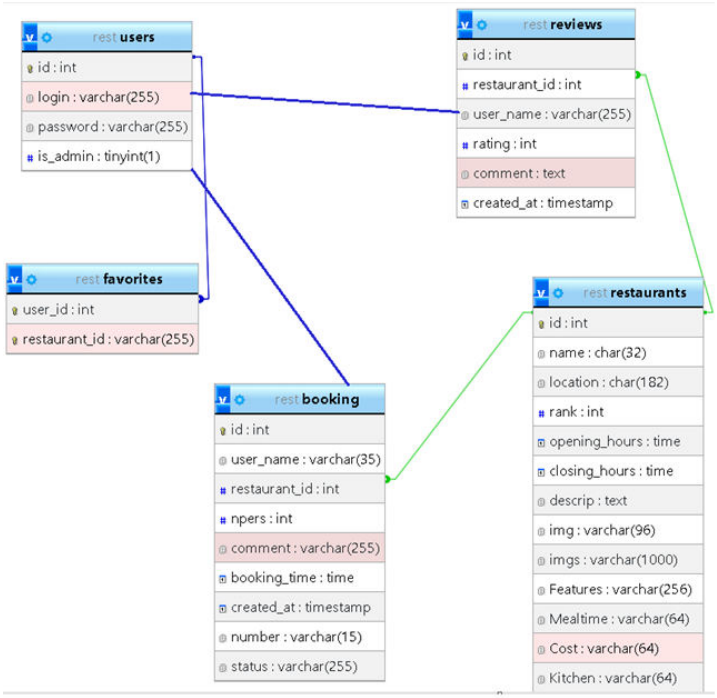

Рис. 2.3.2.1 – Дизайн таблиць з вказаними типами даних

2.3.3 Опис основних операцій з базою даних (створення, читання, оновлення, видалення)

Важливо розуміти, що база даних є центральним елементом системи, який забезпечує збереження та управління даними. Операції, такі як створення, читання, оновлення та видалення, є фундаментальними для роботи з базою даних. Кожна з цих операцій виконує певну функцію і має свої особливості. Детальний опис кожної операції надасть зрозуміння процесу взаємодії з базою даних у контексті даного проекту.

Для роботи сервісу з базою даних виконуються такі операції:

Користувач який є адміністратором має можливість читати, заповнювати та редагувати таблицю з інформацією про ресторани. У синтаксисі mySQL це операція читання (READ) створення (CREATE) та оновлення (UPDATE). Також повинна бути можливість отримувати дані з бронювань операція читання (READ)

та редагувати операція редагування (UPDATE) для підтвердження чи відміни заявки на бронювання.

Користувач який не має прав адміністратора може читати дані з таблиць ресторани, відгуки, бронювання та улюблених закладів, операція (READ). Також для реалізації функцій створення списку улюблених закладів виконувати операції створення (CREATE) та видалення (DELETE) у таблиці улюблених закладів. Залишення відгуків, бронювання відбуватиметься за допомогою (CREATE).

Використання цих операцій дозволяє забезпечити зручний та ефективний доступ до бази даних проекту.

2.4 Проектування клієнтської частини

Розглянемо процес створення і проектування інтерфейсу користувача для нашого сервісу. Мета цього розділу полягає в розробці зручного, естетичного та інтуїтивно зрозумілого інтерфейсу, який дозволить користувачам легко взаємодіяти з нашим сервісом та виконувати необхідні операції з базою даних.

Зосередимось на створенні зручного та привабливого інтерфейсу для нашого сервісу ресторанної резервації, з урахуванням вимог і потреб наших користувачів.

Однією з вимог є доступність додатку на мобільних пристроях. Ми розуміємо, що користувачі хочуть мати можливість зручно взаємодіяти з нашим сервісом навіть під час поїздок або відвідування нових місць. Тому планується реалізація сайту, а також розробка мобільного додатку на базі сайту. Це дозволить користувачам використовувати наш сервіс на будь-якому пристрої, що має доступ до Інтернету, і забезпечить зручну та безперебійну роботу.

Для реалізації мобільного додатку звернемося до сучасних мобільних технологій та інструментів розробки. Один з варіантів - використання технологія PWA (Progressive Web Application). PWA поєднує переваги веб-сайту та

мобільного додатку, дозволяючи створити додаток, який можна встановити на головний екран мобільного пристрою та працювати офлайн. Використання PWA дозволить нам забезпечити швидку завантаження, плавну анімацію та зручну навігацію, що позитивно позначиться на враженнях користувачів.

Крім технології PWA, для розробки мобільного додатку на базі сайту також можна розглянути інші варіанти, такі як нативний додаток і гібридний додаток.

Нативний додаток розробляється спеціально для певної мобільної платформи, такої як iOS або Android, використовуючи мови програмування та інструменти, що підтримуються цією платформою. Він забезпечує глибоку інтеграцію з операційною системою пристрою та надає доступ до всіх його функцій. Проте, розробка нативних додатків вимагає великої кількості ресурсів та часу, оскільки потрібно створювати окремий додаток для кожної платформи.

Гібридний додаток поєднує в собі переваги веб-технологій (HTML, CSS, JavaScript) та мобільних фреймворків, таких як Apache Cordova або React Native. Він може працювати на різних платформах, оскільки базується на веб-технологіях, але забезпечує доступ до нативних функцій пристрою. Гібридні додатки розробляються швидше і витрачають менше ресурсів порівняно з нативними додатками. Однак, вони можуть бути менш оптимізованими і мати обмежену функціональність порівняно з нативними додатками.

У порівнянні з цими технологіями, PWA виходить вперед завдяки своїм перевагам. Вона дозволяє створити додаток, який може працювати на будь-якій платформі, без необхідності розробляти окремі версії для кожної платформи. PWA забезпечує велику швидкість завантаження, можливість роботи в офлайн-режимі та легку інсталяцію на пристроях користувачів. Вона також надає користувачам зручний спосіб взаємодії з додатком, адаптивний дизайн та можливість отримувати сповіщення. Враховуючи вимоги та потреби

користувачів, PWA є прекрасним варіантом для розробки мобільного додатку на базі сайту.

Виходячи з вимог, пов'язаних з поліпшенням користувацького досвіду також необхідно відмітити бібліотеку axios, що використовується для асинхронних запросів. Бібліотека Axios є популярним інструментом для виконання HTTP-запитів у середовищі JavaScript. Вона дозволяє легко відправляти HTTP-запити та обробляти відповіді в зручному форматі. Axios підтримує виконання запитів як у браузері, так і на сервері з використанням Node.js.

В процесі розробки інтерфейсу ми будемо враховувати вимоги та потреби наших користувачів, забезпечуючи простоту використання, зрозумілість інтерфейсу та зручну навігацію. Ми будемо використовувати сучасні дизайнерські та інтерактивні елементи, які сприятимуть легкості взаємодії та покращенню загального досвіду користувачів.

Завдяки застосуванню сучасних мобільних технологій та інструментів розробки, включаючи використання PWA, ми забезпечимо оптимальну роботу додатку на різних пристроях і забезпечимо зручну взаємодію з користувачами, задовольняючи їхні вимоги та потреби.

У розділі проектування клієнтської частини велику увагу слід приділити адаптації інтерфейсу до різних пристроїв та розмірів екранів. Для того, щоб наш сервіс був зручним у використанні на мобільних телефонах, планшетах та комп'ютерах, ми можемо використати спеціальні CSS правила, які змінюють вигляд елементів на основі розміру екрану. Наприклад, ми можемо змінювати розмір шрифтів, зображень та розміщення блоків в залежності від розміру пристрою.

Це дозволяє створити адаптивний дизайн, який забезпечить зручне користування сервісом незалежно від того, який пристрій вони використовують.

Наприклад, коли користувач відкриває наш сервіс на мобільному телефоні, ми можемо змінювати вигляд меню, щоб воно було зручним для навігації на маленькому екрані.

При проектуванні клієнтської частини важливо враховувати різні розміри екранів та потреби користувачів у зручному використанні на різних пристроях. Використання спеціальних CSS правил для адаптації допомагає створити зручний інтерфейс, який буде оптимально відображатись на різних пристроях.

Розгляд можливих функцій та елементів управління

При розробці сервісів з пошуку часто виникає потреба у розгляді різних можливих функцій та елементів управління, які покращують користувацький досвід та роблять додаток більш функціональним. Для розробки цього проекту були використані такі функції та елементи управління:

- Пошукове поле: Пошукове поле дозволяє користувачам здійснювати пошук за ключовими словами або фразами. У випадку з цим проектом поле для пошуку буде використовуватись для пошуку закладів за місцезнаходженням (містом, районом, вулицею та інше). Зазвичай поле супроводжується кнопкою пошуку, яка ініціює пошуковий запит.
- Фільтри: Фільтри дозволяють користувачам обмежувати результати пошуку або відображати певну категорію контенту. Наприклад, у списку товарів фільтри можуть дозволяти користувачам вибрати конкретну цінову категорію, колір або розмір товару. Фільтри можуть бути представлені у вигляді прапорців, перемикачів або списків з випадаючими пунктами. Для сервісу з пошуку закладів за допомогою фільтрів буде здійснюватись пошук необхідного закладу за критеріями: час прийому їжі, ціна, вид кухні, блюда, харчові обмеження, для кого підходить, додаткові особливості.
- Карта: Карта є корисним елементом управління для відображення місць, розташувань або географічних об'єктів. Вона може бути інтерактивною,

дозволяючи користувачам масштабувати, перетягувати та отримувати детальні відомості про конкретне місце. Карти часто використовуються у послугах доставки, туристичних додатках або місцевих довідниках. Це хороша перевага, що ідеально підходить до випадку с сервісом з пошуку закладів.

3. Розробка сервісу з пошуку кафе та ресторанів

3.1 Створення сайту

3.1.1 Розробка клієнтської частини:

Каталог

Основою реалізації інтерфейсу буде сторінка каталогу на якій будуть відображатися заклади з основною інформацією про них, фільтрами та полем для пошуку. Сторінка буде розділена за допомогою технології CSS Flexbox. За допомогою Flexbox можна гнучко налаштовувати розміщення елементів на веб-сторінці, встановлювати їх порядок, вирівнювати їх по горизонталі та вертикалі, а також розтягувати чи стискувати елементи в залежності від доступного простору. Завдяки Flexbox стає можливим створення адаптивного дизайну, який забезпечує коректне відображення інтерфейсу на різних пристроях і розмірах екрану. Використання Flexbox дозволяє легко адаптувати макет інтерфейсу до різних розмірів екрану, забезпечуючи оптимальне використання доступного простору та зручну навігацію для користувачів.

Порівняно з іншими методами розташування елементів, Flexbox має декілька переваг. Він простий у використанні, забезпечує більшу гнучкість та контроль над розташуванням елементів, дозволяє автоматично адаптувати їх до змінних розмірів екрану. Крім того, використання Flexbox дозволяє писати менше CSS-коду, що полегшує розробку та підтримку інтерфейсу.

Таким чином, використання технології Flexbox дозволяє зручно та ефективно керувати розташуванням елементів на сторінці, створюючи адаптивний та гнучкий дизайн, що відповідає потребам та вимогам користувачів.

У таблиці стилів CSS блокам на сторінках буде надана властивість display:flex, що дасть змогу зробити гнучкий макет для більш привабливого відображення інформації. Приклад:

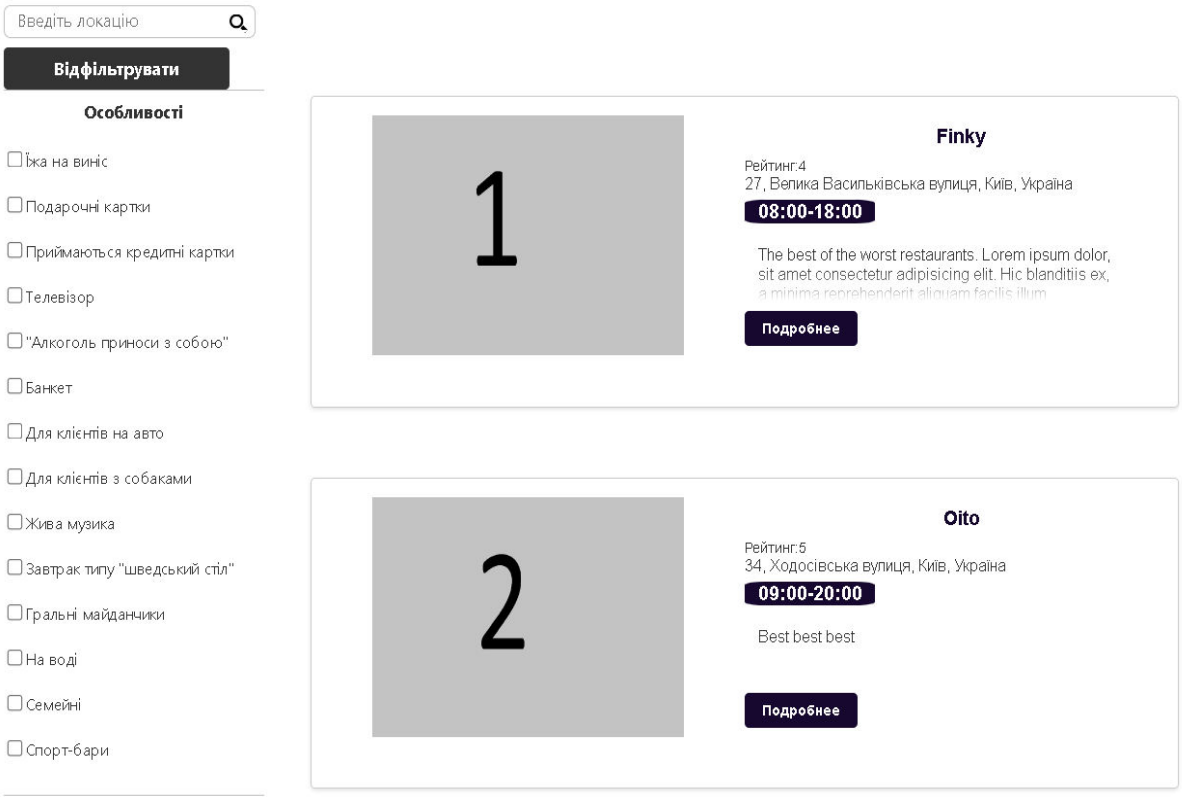

Рис. 3.1.1.1 – Інтерфейс сторінки каталогу закладів

Для поліпшення користувацького досвіду та надання зручного способу візуалізації розташування кафе та ресторанів в рамках реалізації клієнтської частини сервісу, було вирішено використовувати карту OpenStreetMap. OpenStreetMap - це відкрита та спільнотною розроблена мапа, яка надає доступ до детальної інформації про географічне положення об'єктів.

Застосування карти OpenStreetMap дозволяє відобразити розташування кафе та ресторанів на інтерактивному інтерфейсі, що дозволяє користувачам швидко знайти бажане заклад та отримати необхідну інформацію. Карта може бути інтегрована в веб-сторінку або мобільний додаток, забезпечуючи зручну навігацію та візуалізацію даних.

Один з переваг використання OpenStreetMap полягає у його відкритості та гнучкості. Він надає можливість користувачам внести власний внесок у картографічні дані та внести зміни, що робить його актуальним та точним

джерелом інформації. Крім того, OpenStreetMap надає розширений набір функцій, таких як маршрутизація, пошук об'єктів та налаштування вигляду карти.

Використання карти OpenStreetMap в реалізації клієнтської частини сервісу дозволяє забезпечити користувачам зручний спосіб взаємодії з інтерфейсом та отримання необхідних географічних даних. Ця технологія відкриває нові можливості для навігації та візуалізації даних, створюючи більш змістовний та ефективний досвід для користувачів.

Для використання мап openStreetMap необхідно підключити бібліотеку leaflet. Приклад коду для відображення мапи:

```
<script>
  const mymap = L.map('mapid').setView([49.167,31.553], 5.5);

  L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: 'Map data © <a href="https://openstreetmap.org">OpenStreetMap</a> contributors',
    maxZoom: 18
  }).addTo(mymap);
```

Рис. 3.1.1.2 – Реалізація додавання карти до інтерфейсу

Сторінка закладу

За допомогою подібних засобів створимо сторінку для відображення індивідуальної сторінки закладу.

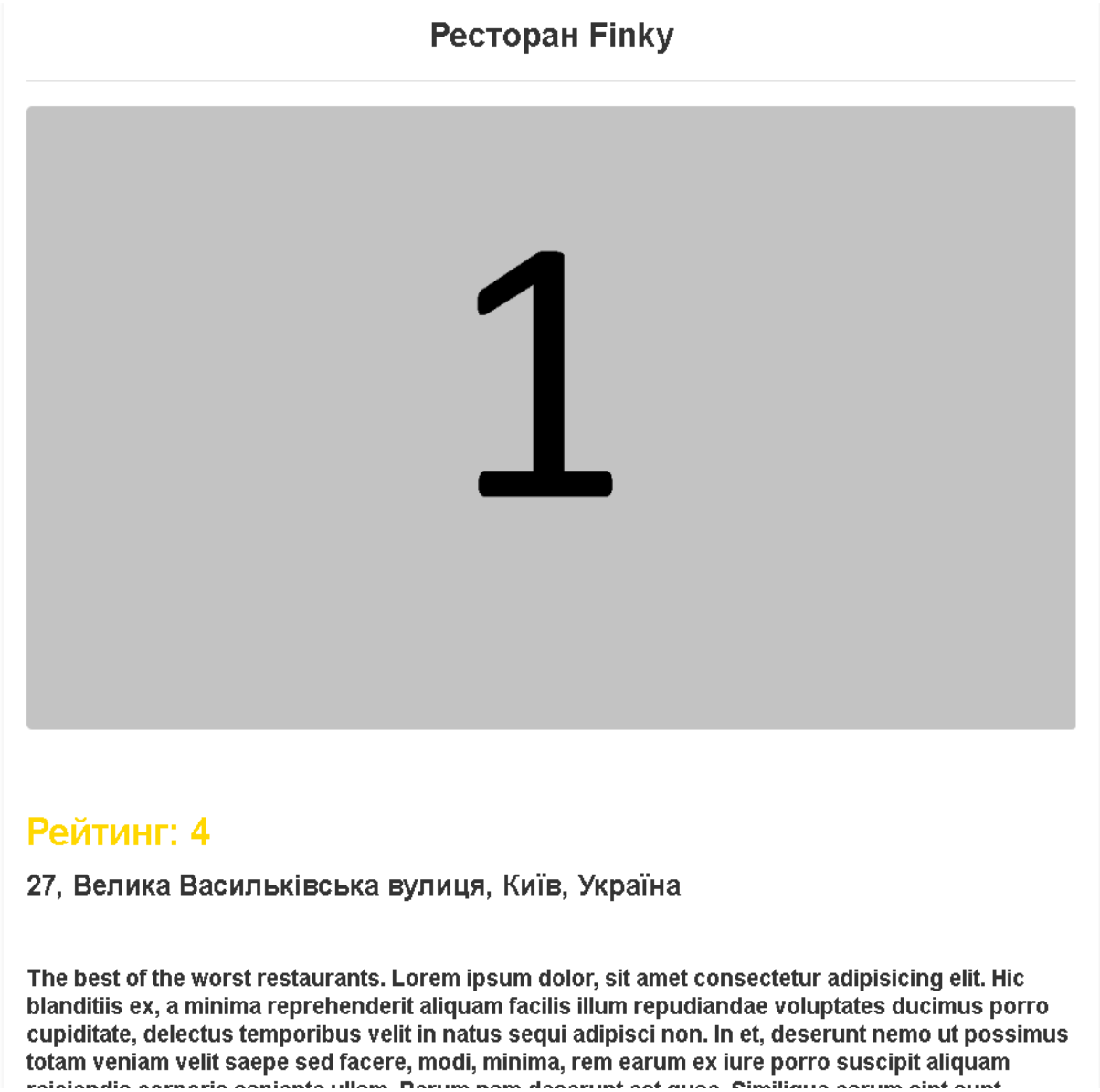

Рис. 3.1.1.2 – Приклад інтерфейсу сторінки закладу

Додамо на сторінку блок для відображення відгуків. За допомогою форми можна буде відправляти відгук на сервер. Відправка буде реалізована за допомогою методу POST. При використанні методу POST користувач буде мати можливість заповнити форму з відповідними полями, такими як ім'я, оцінка, коментар та інші релевантні дані про відгук. Після натискання кнопки "Надіслати" дані будуть передані на сервер у вигляді HTTP-запиту типу POST.

На сервері буде реалізована логіка для обробки цього запиту. Вона буде включати перевірку введених даних, їх збереження в базі даних та повернення

відповідного повідомлення користувачеві про успішне або невдале збереження відгуку.

Одним із переваг використання методу POST є можливість передачі значних обсягів даних і збереження їх на сервері. Цей метод також забезпечує безпеку, оскільки дані відправляються у тілі запиту, а не у URL-адресі, що робить їх менш доступними для зловмисників.

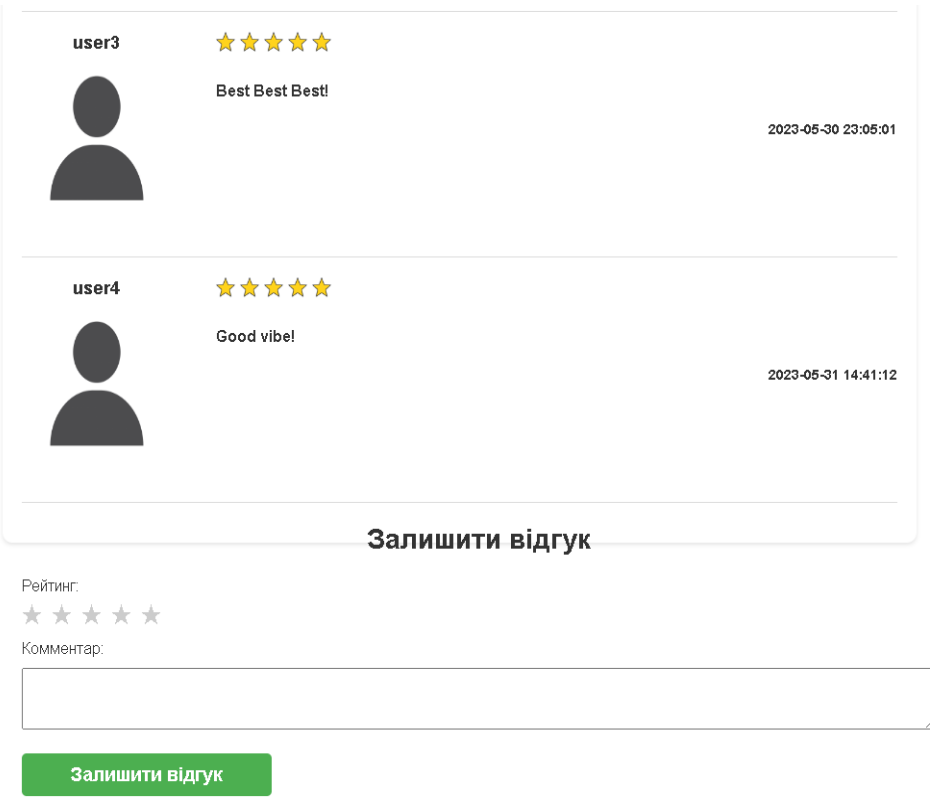

Рис. 3.1.1.3 – Інтерфейс для функції відгуків

```
  <form class="review-form" action="php/submit_review.php" method="POST">
    <input type="hidden" name="name" value="' . htmlspecialchars($username) . '" required>
    <h3>Залишити відгук</h3>
    <label for="rating">Рейтинг:</label>
    <div class="rating">
      <input type="hidden" name="rating" id="rating-value" required>
      <div class="rating-stars">
        <span class="star" data-value="1"></span>
        <span class="star" data-value="2"></span>
        <span class="star" data-value="3"></span>
        <span class="star" data-value="4"></span>
        <span class="star" data-value="5"></span>
      </div>
    </div>
<label for="comment">Комментар:</label>
<textarea name="comment" id="comment" rows="4" required></textarea>';
```

Рис. 3.1.1.3 – Код для створення форми залишення відгуку

Ін’єкції для інтерфейсу

Для вставки повторюваних елементів у інтерфейс використовується функція include в PHP. Цей метод дозволяє включити вміст іншого файлу у поточний файл, що спрощує керування та підтримку коду. Наприклад, якщо у нас є повторювані елементи, такі як заголовок, навігаційне меню або футер, ми можемо створити окремий файл для кожного з цих елементів і потім використовувати include, щоб вставити їх у потрібні місця на сторінці.

Цей підхід забезпечує зручне управління кодом, оскільки зміни у повторюваних елементах можна вносити лише в одному місці - у відповідних включених файлах. Крім того, це полегшує перевикористання коду, оскільки однакові елементи можна включати в різні сторінки і проекти.

Застосування функції include в PHP дозволяє забезпечити чистоту та структурованість коду, полегшує його розробку та підтримку. Крім того, цей підхід сприяє покращенню продуктивності та швидкості завантаження сторінок, оскільки деякі елементи можуть бути кешовані для подальшого використання.

Використання функції include є корисним інструментом при проектуванні інтерфейсу, оскільки він дозволяє розбити код на логічні частини та забезпечити

їх вставку у потрібні місця на сторінці. Це забезпечує зручну та ефективну розробку веб-додатків і сприяє підтримці та розширенню проектів у майбутньому.

Ін’єкція header.php

У файлі header.php розмістимо елементи для швидкої навігації у рамках сайту та кнопку для відкриття авторизації та реєстрації або відкриття блоку особистого кабінету користувача.

Рис. 3.1.1.4 – вигляд ін’єкції для верхньої частини сайту

Сторінка бронювання

На сторінці бронювання буде виводитись інформація про бронювання та його статус. Кращим рішенням для виводу такої інформації є html таблиці. Вид сторінки бронювань:

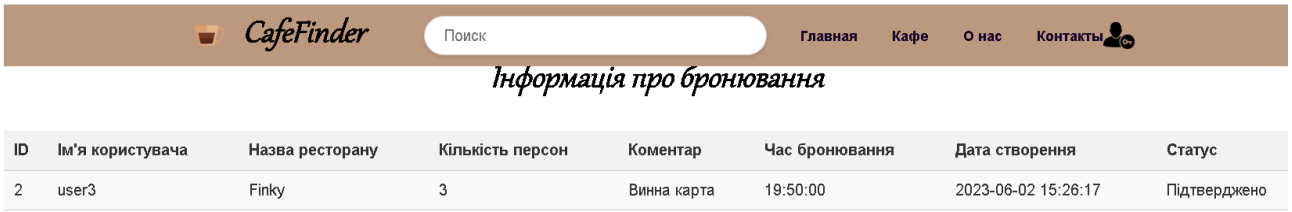

Рис. 3.1.1.5 – Сторінка з відображенням інформації про бронювання

```
<?php
    include 'php/header.php';
?>
<div style="position: absolute; top: 6%; width: 100%;">
    <h1>Інформація про бронювання</h1>
<?php
if ($bookingResult->num_rows > 0) {
?>
<table>
    <tr>
        <th>ID</th>
        <th>Ім'я користувача</th>
        <th>Назва ресторану</th>
        <th>Кількість персон</th>
        <th>Коментар</th>
        <th>Час бронювання</th>
        <th>Дата створення</th>
        <th>Статус</th>
    </tr>
    <tr>
        <td>2</td>
        <td>user3</td>
        <td>Finky</td>
        <td>3</td>
        <td>Винна карта</td>
        <td>19:50:00</td>
        <td>2023-06-02 15:26:17</td>
        <td>Підтверджено</td>
    </tr>
```

Рис. 3.1.1.6 – html код для відображення таблиць для інформації про бронювання

3.1.2 Розробка серверної частини

Серверна частина веб-сервісу відіграє ключову роль у забезпеченні обробки запитів від клієнтської частини та забезпеченні доступу до бази даних. Це компонент, який виконує логіку бізнес-процесів, обробляє запити, генерує відповіді та здійснює взаємодію з базою даних.

Для реалізації серверної частини можна використовувати різні технології та фреймворки, залежно від потреб проекту. Наприклад, одним з популярних варіантів є використання мови програмування PHP та веб-фреймворка Laravel. PHP є мовою з великою кількістю розширень та добре підходить для веб-

розробки, а Laravel надає зручні інструменти та структуру для швидкої розробки серверної частини.

Іншими популярними варіантами для розробки серверної частини є використання мов програмування, таких як Python (з фреймворком Django або Flask), Node.js (з фреймворком Express.js) або Ruby (з фреймворком Ruby on Rails). Кожна з цих мов та фреймворків має свої переваги та властивості, що робить їх відповідними для різних типів проектів.

Серверна частина також може включати в себе розробку API (інтерфейсу програмування додатків), який дозволяє взаємодіяти з веб-сервісом за допомогою стандартних HTTP-запитів. Це дозволяє розширити функціональність сервісу та надати доступ до його можливостей іншим додаткам або сервісам.

Реалізація серверної частини є важливим етапом у створенні функціонального та безпечного веб-сервісу. Правильний вибір технологій та фреймворків, а також дотримання кращих практик розробки, допоможуть забезпечити якісну та ефективну реалізацію серверної частини проекту.

Виведення

Для виведення всіх записів з таблиці спочатку встановлюються параметри підключення до бази даних, такі як назва серверу, ім'я користувача, пароль та назва бази даних, до якої потрібно підключитись. Після цього створюється з'єднання з базою даних за допомогою класу mysqli. Проводиться перевірка з'єднання на наявність помилок, і у разі їх виникнення програма завершує своє виконання та виводить повідомлення про помилку підключення. Далі формується SQL-запит для отримання всіх записів з таблиці "restaurants". Запит виконується за допомогою методу query() об'єкту з'єднання, і результати зберігаються в змінній $result. Ці кроки дозволяють успішно підключитись до бази даних, виконати SQL-запит та отримати результати, які можна подальше обробити або відобразити на сторінці веб-сервісу.

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "rest";

$conn = new mysqli($servername, $username, $password, $dbname);


if ($conn->connect_error) {
    die("Ошибка подключения: " . $conn->connect_error);
}

$sql = "SELECT * FROM restaurants";

$result = $conn->query($sql);
```

Рис. 3.1.2.1 – Приклад коду для збирання даних з таблиці

Для виводу зробимо цикл який буде створювати новий блок для кожного закладу та виводити записи з $result відповідно до необхідного елементу.

```php
<?php
while ($place = mysqli_fetch_assoc($result)){

<div class="product" onclick="location.href='product.php?id=<?php echo $place['id']; ?>'">
    <div class="dimg">
        <img src="<?php echo $place['img']; ?>" alt="Product 1">
    </div>
    <div class="des">
        <h3><?php echo $place['name']; ?></h3>
        <span class="raiting">Рейтинг:<?php echo $place['rank']; ?></span>
        <p class="ploc"><?php echo $place['location']; ?></p>
        <p class="ptime" style="display: inline-block; background-color: #16082F; padding:
          <?php echo substr($place['opening_hours'], 0, 5); ?>-<?php echo substr($place['cl
        </p>
        <p class="pdes"><?php echo $place['descrip']; ?></p>
        <button>Детальніше</button>
    </div>
</div>
```

Рис. 3.1.2.2 – Приклад коду для виводу даних з таблиці

За цим же принципом виконується вивід даних з будь-якої таблиці у сервісі.

Фільтрація

Для виконання фільтрації виводу результатів формується SQL-запит для вибірки даних з таблиці "restaurants". Запит починається з "SELECT * FROM restaurants WHERE 1", що означає вибрати всі записи з таблиці "restaurants". Далі додаються додаткові умови до запиту, які базуються на значеннях фільтрів. Умови формуються за допомогою операторів AND та LIKE, які дозволяють шукати відповідні значення в колонках таблиці.

Кроки повторюються для кожного фільтра, що дозволяє застосовувати багатофакторні фільтри до SQL-запиту. У кінці коду результати запиту зберігаються в змінній $result за допомогою функції mysqli_query(). Далі в коді є функція is_checked(), яка перевіряє, чи вибрано певне значення фільтру, і повертає рядок "checked" для відображення відповідного стану чекбоксів на сторінці.

Ці кроки дозволяють обробляти та використовувати фільтри для вибірки даних з бази даних, що дозволяє користувачам здійснювати точний пошук та налаштовувати відображення результатів на веб-сервісі.

```php
$sql = "SELECT * FROM restaurants WHERE 1";
if (!empty($city_name)) {
  $sql .= " AND (";
  $sql .= " location LIKE '%{$city_name}%' AND";
  $sql = rtrim($sql, "AND") . ")";
}

if (!empty($features)) {
  $sql .= " AND (";
  foreach ($features as $feature) {
    $sql .= " features LIKE '%{$feature}%' AND";
  }
  $sql = rtrim($sql, "AND") . ")";
}
if (!empty($mealtime)) {
  $sql .= " AND (";
  foreach ($mealtime as $time) {
    $sql .= " mealtime LIKE '%{$time}%' AND";
  }
  $sql = rtrim($sql, "AND") . ")";
}

if (!empty($cost)) {
  $sql .= " AND (";
  foreach ($cost as $price) {
    $sql .= " cost LIKE '%{$price}%' AND";
  }
  $sql = rtrim($sql, "AND") . ")";
}

$result = mysqli_query($induction, $sql);

function is_checked($value, $filter) {
    return in_array($value, $filter) ? 'checked' : '';
}
```

Рис. 3.1.2.3 – Приклад коду для виконання фільтрації

Пошук

Для виведення підказок у поле пошуку міст отримується значення змінної searchQuery, яке представляє введений користувачем рядок пошуку, після чого виконується перевірка на наявність значення. Якщо значення присутнє, виконується запит до сервісу OpenStreetMap за допомогою бібліотеки Axios. URL запиту формується з використанням введеного користувачем рядка пошуку та параметрів, що вказують на формат відповіді, мову, деталі адреси та обмеження результатів до 5.

Після виконання запиту, результати отримуються у відповідь (response). З отриманих даних створюється список ul (resultList), а контейнер результатів пошуку searchResults очищається. Для кожного результату створюється елемент li, до якого додається текст відображення результату. Також, для кожного елементу списку встановлюється обробник події click, який виконується при кліку на елементі. У обробнику події отримуються координати (lat та lon) та назва місця (displayName) з результату, після чого карта (mymap) позиціонується за координатами та контейнер результатів пошуку та поле введення очищаються.

Ці кроки дозволяють здійснювати пошук місць за допомогою сервісу OpenStreetMap та динамічно оновлювати результати пошуку на веб-сторінці з використанням JavaScript та бібліотеки Axios.

```
const searchInput = document.getElementById('myInput');
const searchResults = document.getElementById('suggestion');

function searchPlace() {
  const searchQuery = searchInput.value.trim();
  if (searchQuery) {
    axios
      .get(`https://nominatim.openstreetmap.org/search?q=${se
      .then(response => {
        const results = response.data;
        const resultList = document.createElement('ul');
        searchResults.innerHTML = '';
        results.forEach(result => {
          const listItem = document.createElement('li');
          listItem.textContent = result.display_name;
          listItem.addEventListener('click', () => {
            const lat = result.lat;
            const lon = result.lon;
            const displayName = result.display_name;
            mymap.fitBounds([[lat, lon]]);
            searchResults.innerHTML = '';
            searchInput.value = '';
```

Рис. 3.1.2.4 – Реалізація функції пошуку локації

```
      const url = `catalog.php?myInput=${encodeURIComponent(displayName)}`;
      window.location.href = url;
    });
    resultList.appendChild(listItem);
  });
  searchResults.appendChild(resultList);
})
            .catch(error => {
              console.error(error);
              alert('Error occurred while searching for the place.');
            });
        }
      }
      searchInput.addEventListener('input', searchPlace);
      document.addEventListener('click', event => {
        if (event.target !== searchInput && event.target !== searchResults) {
          searchResults.innerHTML = '';
        }
      });
      function showPlaceFromUrl() {
  const urlParams = new URLSearchParams(window.location.search);
  const searchQuery = urlParams.get('myInput');
  if (searchQuery) {
    axios
      .get(`https://nominatim.openstreetmap.org/search?q=${searchQuery}&forma
      .then(response => {
        const result = response.data[0];
        if (result) {
          const lat = result.lat;
          const lon = result.lon;
          const displayName = result.display_name;
          mymap.setView([lat, lon], 11);
        } else {
          alert('Place not found.');
        }
      })
```

Рис. 3.1.2.5 – Реалізація виводу результатів пошуку

Відображення локацій

Для відображення місцезнаходжень кафе та ресторанів на карті виконується запит до сервісу OpenStreetMap за допомогою функції axios.get(). URL запит формується з використанням коду PHP <?php echo $place['location']; ?>, який використовується для передачі значення розташування місця в запиті. Значення розташування передається у вигляді рядка, який конвертується за допомогою encodeURIComponent() для правильної обробки спеціальних символів.

Після виконання запиту, результати отримуються у відповідь (response). З отриманих даних вилучаються географічні координати першого результату (point). Якщо значення point присутнє, створюється маркер (marker) за допомогою

бібліотеки Leaflet (L.marker()) з використанням отриманих координат та додається на карту (mymap). Також, до маркера прив'язується спливаюче вікно (bindPopup()), в якому відображається інформація про місце, така як назва, рейтинг та розташування.

```
<script>
    axios.get(`https://nominatim.openstreetmap.org/search.php?q=${encodeURIComponent(
    '<?php echo $place['location']; ?>')}&format=jsonv2`)
    .then(response => {
        const point = response.data[0];
        if (point) {
            const marker = L.marker([point.lat, point.lon]).addTo(mymap);
            marker.bindPopup(`<b><h3><?php echo $place['name']; ?></h3><br><h3>Рейтинг:
            <?php echo $place['rank']; ?></h3><br><p><?php echo $place['location']; ?></p></b>`);
        }
    })
    .catch(error => console.error(error));
</script>
```

Рис. 3.1.2.6 – Функція для створення маркерів за адресою

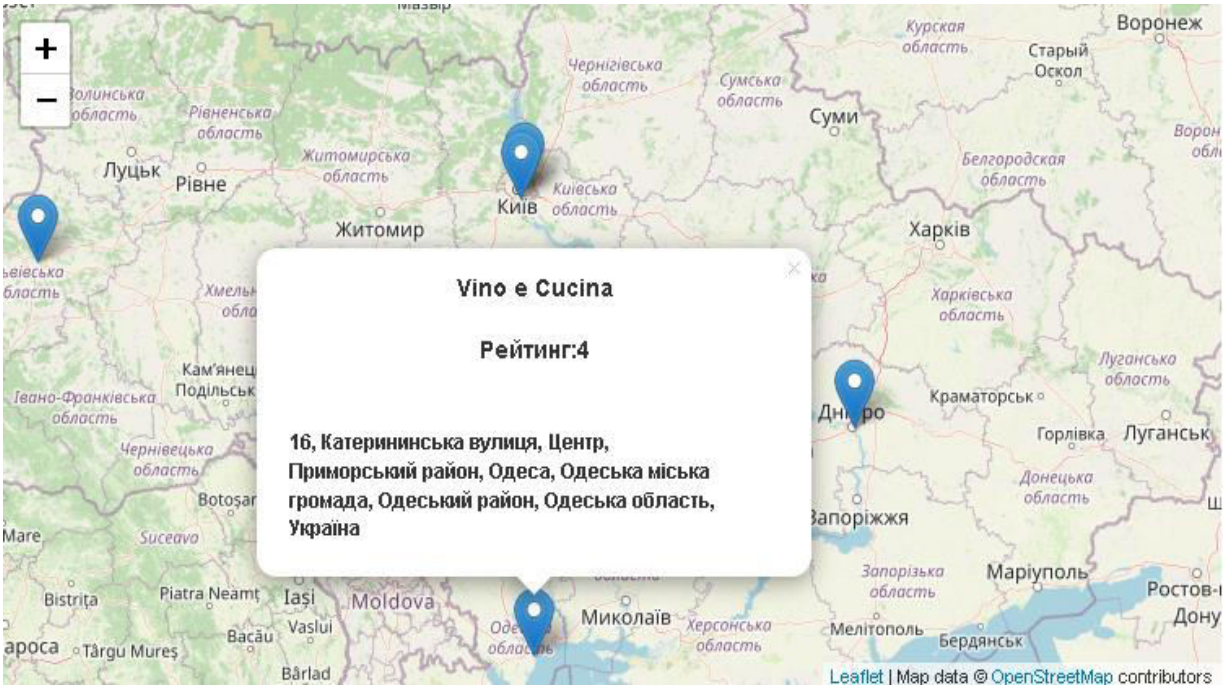

Рис. 3.1.2.7 – Інтерфейс відображення мапи з маркерами закладів

Авторизація та реєстрація

Для реєстрації відбувається отримання даних з форми (ім'я користувача і пароль) зберігаються у відповідних змінних.

Після цього виконується перевірка, що поля форми не є порожніми. Якщо обидва поля заповнені, виконується перевірка унікальності імені користувача. Запит до бази даних для перевірки унікальності формується з використанням підготовленого оператора (mysqli_prepare()), параметри запиту передаються за допомогою функції mysqli_stmt_bind_param(), а сам запит виконується за допомогою mysqli_stmt_execute(). Результат запиту перевіряється, і якщо знайдено існуючого користувача з таким іменем, виводиться відповідне повідомлення, і виконання скрипту завершується.

Якщо користувача з таким іменем не знайдено, пароль хешується за допомогою функції md5(), і виконується реєстрація нового користувача. Запит для вставки нового користувача в базу даних також формується з використанням підготовленого оператора, параметри запиту передаються за допомогою mysqli_stmt_bind_param(), а сам запит виконується за допомогою mysqli_stmt_execute(). Якщо реєстрація успішна, виводиться повідомлення про успішну реєстрацію.

Для авторизації з форми отримуються дані, такі як логін і пароль, які присвоюються відповідним змінним.

Далі виконується запит до бази даних для пошуку користувача з вказаним логіном. Запит формується з використанням отриманого логіна і обмежується вибіркою одного запису (LIMIT 1). Результат запиту зберігається в змінну $result, а дані про знайденого користувача вилучаються за допомогою функції mysqli_fetch_assoc() і зберігаються в змінну $row.

Далі відбувається перевірка наявності запису користувача і порівняння хешу пароля, переданого у формі, з хешем пароля користувача з бази даних. Якщо користувач існує і паролі збігаються, то відбувається успішна авторизація

користувача. Створюється куки з ім'ям користувача, встановлюється термін дії куки і відбувається перенаправлення на сторінку "index.php".

У випадку помилки авторизації (незбіг пароля або відсутність користувача) відбувається перенаправлення на сторінку "login.php" з вказанням параметра "error=1", щоб можна було відобразити відповідне повідомлення про помилку.

За допомогою кук на сайті можна реалізувати багато корисних функцій наприклад: за допомогою перевірки кук відкривається відповідний блок з авторизацією/реєстрацією або особистим кабінетом вже авторизованого користувача. Таким же чином виконується визначення адміністратора по логіну з кук знаходиться поле is_admin у таблиці users .

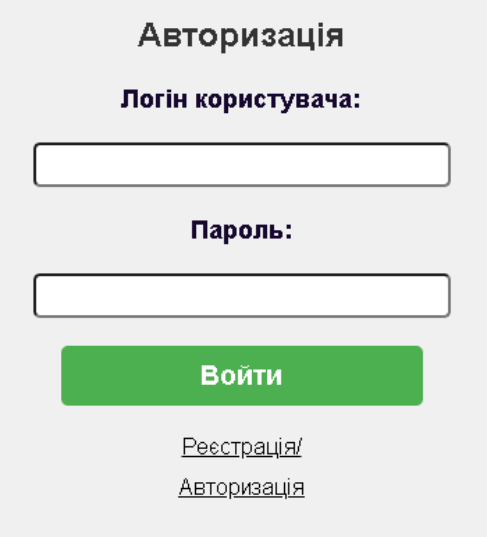

Рис. 3.1.2.8 – Інтерфейс форми авторизації

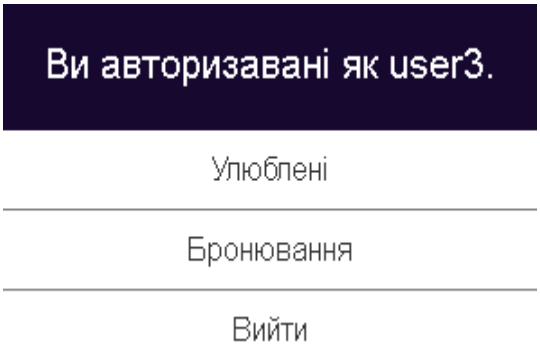

Рис. 3.1.2.9 – Інтерфейс особистого кабінету користувача

3.2 Розробка мобільного додатку

Розробка мобільних додатків, заснованих на технології PWA (прогресивні веб-додатки), стає все більш популярною. PWA поєднує в собі переваги веб-сайтів і традиційних мобільних додатків, надаючи користувачам зручність використання, швидкість і можливість доступу до функцій офлайн.

Розробка мобільних додатків на базі технології PWA відкриває широкі можливості для створення універсальних додатків, які можуть працювати на різних платформах і пристроях, включаючи смартфони, планшети і навіть комп'ютери. Це дозволяє підприємствам і розробникам зосередитися на створенні одного додатку замість розробки окремих версій для кожної платформи.

За допомогою технології PWA можна створювати додатки з респонсивним дизайном, які адаптуються до розміру екрану користувача і забезпечують оптимальний вигляд на різних пристроях. Також PWA дозволяє використовувати різноманітні функції, такі як сповіщення, офлайн-режим, доступ до камери і місцезнаходження, що забезпечує багатофункціональність додатка.

Розробка мобільних додатків на базі технології PWA також спрощує процес розгортання і оновлення додатків, оскільки вони можуть оновлюватися автоматично без необхідності встановлення оновлень через магазини додатків. Користувачі можуть отримувати оновлення безпосередньо з веб-сайту, що забезпечує швидкий доступ до нових функцій і виправлень.

Розробка мобільних додатків на базі технології PWA відкриває нові перспективи для бізнесу і розробників, дозволяючи створювати потужні, масштабовані і зручні у використанні додатки, які можуть працювати на різних платформах і надавати користувачам незабутній досвід.

Основні вимоги для реалізації PWA на базі сайту включають наступне:

1. Застосування SSL-сертифіката: Для забезпечення безпеки та захищеної передачі даних, веб-сайт повинен мати SSL-сертифікат, що дозволяє використовувати протокол HTTPS.
2. Адаптивний дизайн: PWA має бути розроблений з урахуванням різних розмірів екранів та пристроїв, забезпечуючи респонсивний дизайн.
3. Маніфест: Маніфест - це файл JSON, який містить метадані додатку, такі як назва, іконки, кольори теми та інші параметри. Він дозволяє додатку виглядати та працювати як нативна програма на мобільних пристроях.
4. Service Worker: Service Worker - це JavaScript-скрипт, який працює в фоновому режимі та керує кешуванням ресурсів, офлайн-режимом та пуш-сповіщеннями. Використання Service Worker дозволяє створити офлайн-доступ до додатку та покращити його продуктивність.
5. Кешування ресурсів: Використовуючи Service Worker, ви можете кешувати статичні ресурси, такі як HTML, CSS, JavaScript, зображення та інші, для швидшого завантаження та покращення реакції додатку навіть при відсутності Інтернет-з'єднання.

Розміщення сайту з використанням HTTPS протоколу

Для розміщення сайту сервісу оберемо хостинг провайдера, що буде відповідати вимогам підключення SSL-сертифікату та буде оптимальним за характеристиками. На мою думку буде «hosting Ukraine» буде хорошим варіантом, він має повністю автоматизовану систему всіх процесів розміщення, підтримку Apache, задовільну ціну та підтримку SSL-сертифікату.

Другим етапом буде реєстрація домену, головною вимогою якого буде унікальне ім’я сайту, яке буде використовуватися в URL-адресі. Отже оберемо: projectdip2. Зареєструємо це ім'я через сервіс ADM-tools, для більш зручної роботи з сайтом у майбутньому. Сервіс ADM-tools має широкий функціонал для будь яких операцій з сайтом.

Далі завантажимо файли сайту та бази даних на сервіс, враховуючи структуру каталогів та розміщення сайтів, щоб забезпечити коректне відображення.

Тепер сайт доступний у мережі інтернет за вказаною адресою.

Розробка адаптивного дизайну

Один з ключових інструментів, який допомагає нам досягти адаптивності, це медіа-запити.

Медіа-запити - це техніка, яка дозволяє нам застосовувати різні стилі до веб-елементів, залежно від характеристик пристрою, на якому відображається сайт. Вони дозволяють налаштувати зовнішній вигляд та поведінку елементів на різних пристроях, таких як комп'ютери, планшети та смартфони.

Медіа-запити використовують різні параметри для визначення характеристик пристрою, такі як ширина екрану, орієнтація, роздільна здатність та інші. За допомогою медіа-запитів ми можемо задати різні стилі для різних пристроїв, щоб забезпечити оптимальний вигляд та взаємодію з нашим сайтом на будь-якому пристрої.

Наприклад, ми можемо використовувати медіа-запити, щоб змінити розмір шрифту, розміщення елементів, кількість стовпців у макеті, відображення або приховування деяких елементів на основі розміру екрану. Це дозволяє нам створювати приємний та зручний інтерфейс для користувачів незалежно від пристрою, який вони використовують для перегляду сайту.

Під час розробки адаптивного дизайну ми використовуємо медіа-запити для створення різних "точок перелому" або "розмежовувальних точок", де ми задаємо спеціальні стилі для певних пристроїв. Наприклад, ми можемо встановити медіа-запит, що застосовується, коли ширина екрану менше 768 пікселів, і задати відповідні стилі для мобільних пристроїв. Приклад використання:

```css
@media only screen and (max-width: 600px) {
  .product {
    flex-direction: column; margin: 20px 0; padding: 20px;
  }

  .dimg {
    padding: 0;
    padding-bottom: 20px;
    text-align: center;
    width: 100%;
    height: 100%;
  }
  .dimg  img{
  }
  .des {
    margin-left: 0;
    text-align: center;
  }

  .product h3 {
    font-size: 1.2rem;
    margin-bottom: 10px;
  }
```

Рис. 3.2.1 – Приклад використання медіа-запитів

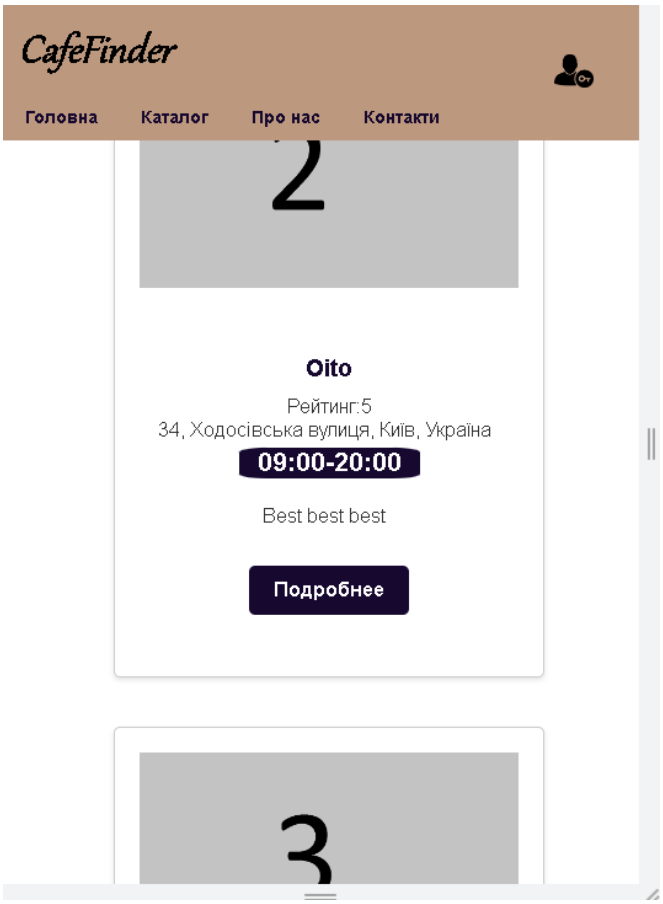

Рис. 3.2.2 – Приклад роботи сайту з використанням адаптивного дизайну

Маніфест файл

Створимо маніфест файл. Розробка маніфесту для PWA додатку є важливим кроком у процесі створення Progressive Web App. Маніфест - це JSON-файл, який містить інформацію про додаток, таку як його назва, іконки, кольори та налаштування поведінки.

Маніфест дозволяє додатку "повідомити" браузеру про свої характеристики та налаштування, щоб він міг коректно відображати та взаємодіяти з додатком на різних пристроях та в різних умовах використання.

При розробці маніфесту ми задаємо такі параметри, як "name" (назва додатку), "short_name" (скорочена назва), "start_url" (початкова URL-адреса), "icons" (іконки для різних пристроїв), "theme_color" (колір теми), "background_color" (колір фону) та багато іншого.

Маніфест також дозволяє визначити налаштування поведінки додатку, такі як "display" (спосіб відображення), "orientation" (орієнтація екрану), "scope" (область видимості) та інші параметри.

Після створення маніфесту ми розміщуємо його в кореневій директорії нашого веб-сайту та посилаємо на нього в HTML-коді за допомогою тегу link з вказанням відповідного rel атрибуту.

Створимо коректний та докладний маніфест файл, оскільки він допомагає забезпечити правильне відображення та функціональність додатку на різних пристроях та платформах.

Реалізація роботи без підключення до мережі інтернет

Для реалізації роботи додатку без підключення до інтернету необхідно підключити скрипт, що буде кешувати ресурси та давати доступ до них. Таким скриптом є service-worker.

Щоб підключити службового робітника до проекту необхідно додати service-worker.js файл до каталогу проекту та добавити посилання на нього за допомогою тегу script.

У файлі службового робітника ми можемо визначити кеші, додавати ресурси до кешу, налаштовувати стратегії кешування та багато іншого.

Підключення службового робітника дозволяє додатку працювати незалежно від наявності інтернет-з'єднання та забезпечує покращений досвід користувача шляхом швидкого завантаження сторінок та доступу до кешованих ресурсів.

Для підключення при завантаженні сторінки слід провірити підтримку service-worker браузером за допомогою if ('serviceWorker' in navigator), якщо функція підтримується, виконується реєстрація service-worker у системі за допомогою методу navigator.serviceWorker.register('sw.js'). Тільки після цього service-worker можна вважати працюючим.

Встановлення додатку

Щоб додати функцію зручного встановлення додатку створимо спеціальну кнопку, яка буде відповідати за цю дію.

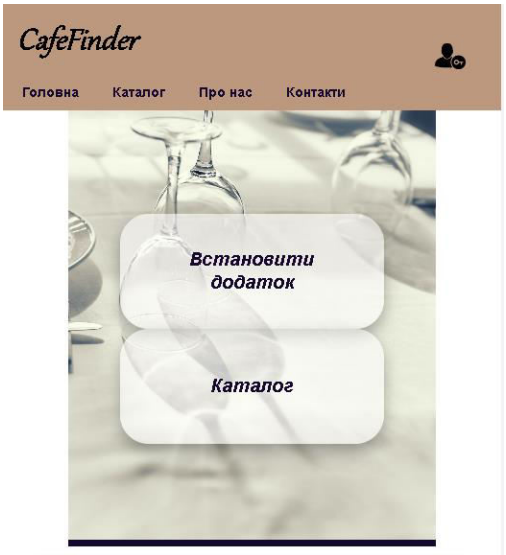

Рис. 3.2.3 – Відображення кнопки встановлення додатку

Спочатку при завантаженні сторінки слід провірити підтримку service-worker браузером та виконати реєстрацію service-worker у системі, тільки при виконанні цих умов за кнопкою закріпляється функція показу пропозиції з встановленням додатку, і якщо користувач дає згоди на встановлення то винонується функція install з самого service-worker.

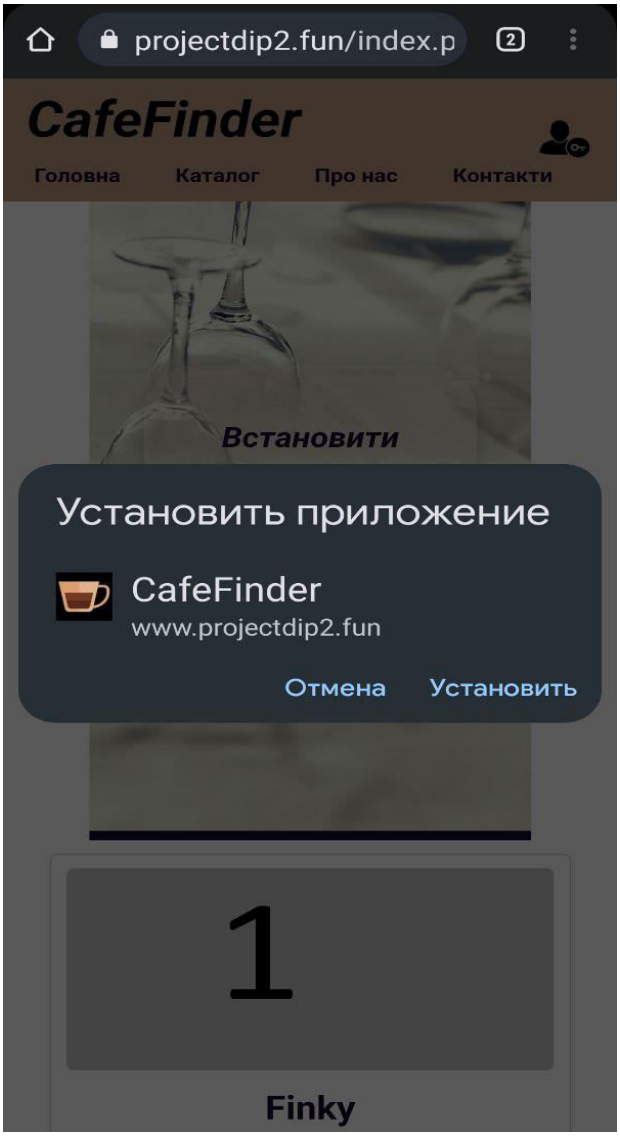

Рис. 3.2.4 – Вікно запиту для встановлення додатку

Після підтвердження встановлення додатку він з’явиться на робочому столі користувача з підписом «CafeFinder». Та матиме вид:

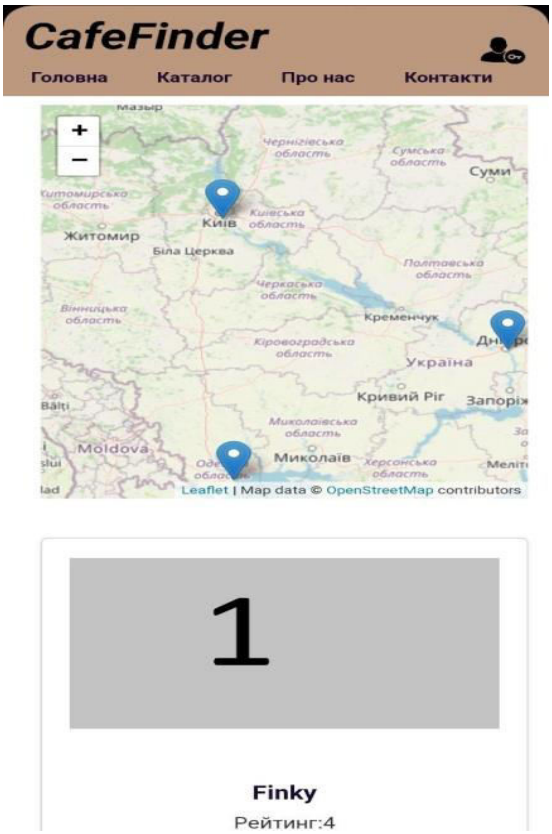

Рис. 3.2.5 – Інтерфейс каталогу у додатку

Рис. 3.2.2 – Інтерфейс сторінки закладу у додатку

Висновок

Даний проект досліджує тему розробки та реалізації додатку для сервісу, який поєднує в собі мету поліпшення користувацького досвіду в межах додатку та сайту. В результаті аналізу предметної області та дослідження аналогічних продуктів створено сервіс з пошуку кафе та ресторанів з підтримкою кросплатформного додатку.

У процесі розробки була проведена дослідницька робота, яка розглядає різні методики створення додатків. Мета моєї дипломної роботи була реалізована, а точніше було створено додаток для сервісу з пошуку кафе та ресторанів, який надає користувачам зручний та ефективний спосіб знаходження та вибору закладів громадського харчування. Для досягнення цієї мети було проаналізовано думки потенціальних користувачів сервісу та ринок закладів харчування, створено зручний інтерфейс, додано весь необхідний функціонал включаючи: пошук, фільтрацію за різними параметрами, відображення інформації та місцезнаходжень на мапі.

У дипломній роботі було використано технологію Progressive Web App (PWA), що дала можливість створити функціональний і зорієнтований на клієнта додаток, що надає зручний та інтуїтивно зрозумілий спосіб знаходження інформації про різноманітні заклади харчування.

Один з головних аспектів розробки створення адаптивного дизайну, який забезпечує правильне та оптимізоване відображення додатку на різних пристроях і екранах. Це важливо, оскільки користувачі використовують різні пристрої, такі як мобільні телефони, планшети та настільні комп'ютери, і очікують, що додаток буде зручно працювати на будь-якому з них.

Окремо варто відзначити використання сервісного працівника (service worker) у додатку. Спеціального скрипту, який працює в фоновому режимі і забезпечує офлайн-роботу та кешування контенту. Сервісний працівник покращує

продуктивність додатку та надає користувачам зручність використання, навіть при обмеженому або відсутньому інтернет-з'єднанні.

У підсумку, результати роботи підтверджують ефективність технології PWA у створенні сервісу пошуку кафе і ресторанів. Додаток надає зручність, швидкість і доступність для користувачів, відкриваючи нові можливості для взаємодії та знаходження необхідної інформації. Результати цієї дипломної роботи можуть бути використані для подальшого вдосконалення та розвитку подібних сервісів, сприяючи покращенню користувацького досвіду та задоволенню потреб користувачів.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Пітерсон, Л. К. Прогресивні веб-додатки: створення веб-додатків, що працюють на всіх пристроях [Електронний ресурс] / Л. К. Пітерсон. – Київ: Видавництво "ДУХ І ЛІТЕРА", 2018. – 352 с. – Режим доступу: електронний ресурс.

2. Головченко, О. В. Технологія Progressive Web App: створення і оптимізація [Текст] / О. В. Головченко // Молодий вчений. – 2019. – № 8(72). – С. 279-282.

3. Соколова, А. В. Розробка мобільних додатків на базі технології Progressive Web Apps (PWA) [Текст] / А. В. Соколова, В. Ю. Волошинов. // Проблеми інформатизації і управління. – 2020. – Том 4, Вип. 54. – С. 126-133.

4. Королюк, Д. М. Розробка мобільних додатків на базі технології Progressive Web Apps [Текст] / Д. М. Королюк, В. О. Підлесний, І. О. Літвінов. // Вісник Харківського національного університету Повітряних Сил імені Івана Кожедуба. – 2021. – № 1(62). – С. 93-100.

5. Google Developers. Progressive Web Apps [Електронний ресурс]. – Режим доступу: https://developers.google.com/web/progressive-web-apps/

6. Mozilla Developer Network. Progressive Web Apps [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps

7. PWA.rocks [Електронний ресурс]. – Режим доступу: https://pwa.rocks/

8. Medium. Progressive Web Apps [Електронний ресурс]. – Режим доступу: https://medium.com/topic/progressive-web-apps

9. Stack Overflow [Електронний ресурс]. – Режим доступу: https://stackoverflow.com/

10. Udacity. Progressive Web Apps Course [Електронний ресурс]. – Режим доступу: https://www.udacity.com/course/progressive-web-apps--ud811

11. Smashing Magazine. The Complete Guide to Progressive Web Apps [Електронний ресурс]. – Режим доступу: https://www.smashingmagazine.com/2018/11/complete-guide-pwa/

12. Sitepoint. Progressive Web Apps [Електронний ресурс]. – Режим доступу: https://www.sitepoint.com/progressive-web-apps/

13. Awwwards. Progressive Web Apps [Електронний ресурс]. – Режим доступу: https://www.awwwards.com/what-are-progressive-web-apps-pwas.html

14. W3Schools. Progressive Web Apps Tutorial [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/whatis/whatis_pwa.asp

15. Web.dev. Progressive Web Apps [Електронний ресурс]. – Режим доступу: https://web.dev/progressive-web-apps/

16. FreeCodeCamp. Progressive Web Apps [Електронний ресурс]. – Режим доступу: https://www.freecodecamp.org/news/what-are-progressive-web-apps/

17. The App Solutions. Progressive Web App Development [Електронний ресурс]. – Режим доступу: https://theappsolutions.com/blog/development/progressive-web-app-development/

18. CSS-Tricks. Progressive Web Apps [Електронний ресурс]. – Режим доступу: https://css-tricks.com/progressive-web-apps/

19. Microsoft Developer. Progressive Web Apps [Електронний ресурс]. – Режим доступу: https://developer.microsoft.com/en-us/windows/pwa/

20. ProIT. Що таке PWA і як воно працює? [Електронний ресурс]. – Режим доступу: https://proit.com.ua/blog/whats-new/chto-takoe-pwa-i-kak-ono-rabotaet

ДОДАТОК А

1. Файл product.php

```
<?php
  include 'php/1.php';
  $id = $_GET['id'];
  $result = mysqli_query($induction, "SELECT * FROM `restaurants` WHERE `id` =
'$id'");
  $place = mysqli_fetch_assoc($result);
  $isFavorite = false;
  if (isset($_COOKIE['username'])) {
      $username = $_COOKIE['username'];
      $userResult = mysqli_query($induction, "SELECT * FROM `users` WHERE
`login` = '$username'");
      $user = mysqli_fetch_assoc($userResult);
      $favoriteResult = mysqli_query($induction, "SELECT * FROM `favorites`
WHERE `user_id` = '$user[id]' AND `restaurant_id` = '$id'");
      $favorite = mysqli_fetch_assoc($favoriteResult);
      if ($favorite) {
          $isFavorite = true;
      }
  }
?>
```

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title><?php echo $place['name']; ?></title>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.7.1/leaflet.min.css" />
    <link rel="stylesheet" href="css/productStyle.css">
    <script src="https://cdn.jsdelivr.net/npm/leaflet@1.7.1/dist/leaflet.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.7.1/leaflet.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/slick-
carousel@1.8.1/slick/slick.min.js"></script>
    <script>
      $(document).ready(function() {
          var userId = '<?php echo isset($user['id']) ? $user['id'] : ''; ?>';
          var restaurantId = '<?php echo isset($id) ? $id : ''; ?>';
          function checkFavorite() {
            $.ajax({
```

```
        url: 'check_favorite.php',
        type: 'POST',
        data: {
          userId: userId,
          restaurantId: restaurantId
        },
        success: function(response) {
          if (response == 'exists') {
             $('#addFavoriteButton').empty();
            $('#addFavoriteButton').attr('alt', 'Удалить из избранного');
            var logoImage = $('<img>').attr('src', 'img/favdel.png').attr('alt', 'Удалить из избранного').css({
          width: '100%',
          height: '100%',
          objectFit: 'contain'
        });
        $('#addFavoriteButton').append(logoImage);
          } else {
             $('#addFavoriteButton').empty();
            $('#addFavoriteButton').attr('alt', 'Добавить в избранное');
            var logoImage = $('<img>').attr('src', 'img/logofav.png').attr('alt', 'Добавить в избранное').css({
```

```
            width: '100%',
            height: '100%',
            objectFit: 'contain'
          });
          $('#addFavoriteButton').append(logoImage);
            }
          },
          error: function(xhr, status, error) {
            console.error(error);
          }
        });
      }
      checkFavorite();
      $('#addFavoriteButton').click(function() {
  var altText = $('#addFavoriteButton').attr('alt');
  if (altText == 'Удалить из избранного') {
    removeFavorite();
  } else {
    addFavorite();
  }
});
```

```
function addFavorite() {
  $.ajax({
    url: 'add_favorite.php',
    type: 'POST',
    data: {
      userId: userId,
      restaurantId: restaurantId
    },
    success: function(response) {
      console.log(response);
      $('#addFavoriteButton').empty();
      $('#addFavoriteButton').attr('alt', 'Удалить из избранного');
        var logoImage = $('<img>').attr('src', 'img/favdel.png').attr('alt', 'Удалить из избранного').css({
      width: '100%',
      height: '100%',
      objectFit: 'contain'
    });
    $('#addFavoriteButton').append(logoImage);
      $('#addFavoriteButton').attr('alt', 'Удалить из избранного');
    },
```

```
      error: function(xhr, status, error) {
        console.error(error);
      }
    });
  }
  function removeFavorite() {
    $.ajax({
      url: 'remove_favorite.php',
      type: 'POST',
      data: {
        userId: userId,
        restaurantId: restaurantId
      },
      success: function(response) {
        console.log(response);
        $('#addFavoriteButton').empty();
        $('#addFavoriteButton').attr('alt', 'Добавить в избранное');
          var logoImage = $('<img>').attr('src', 'img/logofav.png').attr('alt', 'Удалить
из избранного').css({
        width: '100%',
        height: '100%',
```

```
            objectFit: 'contain'
          });
          $('#addFavoriteButton').append(logoImage);
            $('#addFavoriteButton').attr('alt', 'Добавить в избранное');
          },
          error: function(xhr, status, error) {
            console.error(error);
          }
        });
      }
    });
  </script>
  <link rel="stylesheet" type="text/css" href="https://cdn.jsdelivr.net/npm/slick-carousel@1.8.1/slick/slick.css"/>
</head>
<body>
  <?php
  include 'php/header.php';
  ?>
<main>
 <div class="container">
```

```
    <h3>Ресторан <?php echo $place['name']; ?></h3>
    <hr>
    <div class="des">
      <div class="slider">
        <?php
          $images = explode(',', $place['imgs']);
          foreach ($images as $image) {
            echo '<div><img src="' . $image . '"></div>';
          }
        ?>
      </div>
      <div class="button-box">
        <?php if (isset($_COOKIE['username'])) : ?>
          <button id="addFavoriteButton"></button>
          <button class="book-button"
onclick="openBookingForm()">Бронировать</button>
        <?php endif; ?>
      </div>
      <span class="rating">Рейтинг: <?php echo $place['rank']; ?></span>
      <p><?php echo $place['location']; ?></p>
      <div class="ddes"><p class="pdes" style="font-size: 2vh;"><?php echo
$place['descrip']; ?></p></div>
```

```
  </div>
  <div id="mapid"></div>
  <div class="reviews">
<h2>Відгуки:</h2>
<?php
  $servername = "ki500876.mysql.tools";
  $username = "ki500876_rest";
  $password = "B7+yz4_k4G";
  $dbname = "ki500876_rest";
  $conn = new mysqli($servername, $username, $password, $dbname);
  if ($conn->connect_error) {
    die("Ошибка подключения к базе данных: " . $conn->connect_error);
  }
  $restaurantId = $_GET['id'];
  $sql = "SELECT * FROM reviews WHERE restaurant_id = ?";
  $stmt = $conn->prepare($sql);
  if (!$stmt) {
    die("Ошибка подготовки запроса: " . $conn->error);
  }
  $stmt->bind_param("i", $restaurantId);
  if (!$stmt->execute()) {
```

```
  die("Ошибка выполнения запроса: " . $stmt->error);
}
$result = $stmt->get_result();
if (!$result) {
  die("Ошибка получения результата: " . $conn->error);
}
while ($row = $result->fetch_assoc()) {
  echo "<div class='review'>";
  echo "<div class='user-info'>";
  echo "<p class='nick'>" . $row['user_name'] . "</p>";
  echo "<img src='img/ukuser.png' alt='user' class='user-image'>";
  echo "</div>";
  echo "<div class='review-content'>";
  echo "<p class='rating'>";

  $rating = intval($row['rating']);
  for ($i = 1; $i <= 5; $i++) {
    if ($i <= $rating) {
      echo "<img src='img/starf.png' alt='star' class='star'>";
    } else {
      echo "<img src='img/stare.png' alt='star' class='star'>";
```

```
        }
      }

      echo "</p>";
      echo "<p class='comment'>" . $row['comment'] . "</p>";
      echo "</div>";
      echo "<div class='review-date'>";
      echo "<p>" . $row['created_at'] . "</p>";
      echo "</div>";
      echo "</div>";
      echo "<hr>";
    }
  ?>
</div>
 <?php
    $username = isset($_COOKIE['username']) ? $_COOKIE['username'] : '';
    if ($username) {
      echo '
      <form class="review-form" action="php/submit_review.php" method="POST">
        <input type="hidden" name="name" value="' . htmlspecialchars($username) . '"
required>
```

```
        <h3>Залишити відгук</h3>
        <label for="rating">Рейтинг:</label>
        <div class="rating">
          <input type="hidden" name="rating" id="rating-value" required>
          <div class="rating-stars">
            <span class="star" data-value="1"></span>
            <span class="star" data-value="2"></span>
            <span class="star" data-value="3"></span>
            <span class="star" data-value="4"></span>
            <span class="star" data-value="5"></span>
          </div>
        </div>
    <label for="comment">Комментар:</label>
    <textarea name="comment" id="comment" rows="4" required></textarea>';
  if (isset($_GET['id'])) {
    $restaurantId = $_GET['id'];
    echo '<input type="hidden" name="restaurant_id" value="' .
htmlspecialchars($restaurantId) . '">';
  }
  echo '<input type="submit" value="Залишити відгук">
  </form>';
```

```
}
?>
  </div>
</main>
    <?php
    include 'footer.php';
    ?>
    <script>
    $(document).ready(function(){
      $('.slider').slick({
        prevArrow: '<button class="slick-prev" aria-label="Previous"
type="button">&lt;</button>',
        nextArrow: '<button class="slick-next" aria-label="Next"
type="button">&gt;</button>'
      });
    });
  </script>
    <script
src="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v5.3.0/build/ol.js
"></script>
    <script src="js/slider.js"></script>
```

```
    <script
src="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v5.3.0/build/ol.js
"></script>
    <script>
        var address = "<?php echo $place['location']; ?>";
        const mymap = L.map('mapid').setView([49.167, 31.553], 5.5);
        L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
            attribution: 'Map data © <a
href="https://openstreetmap.org">OpenStreetMap</a> contributors',
            maxZoom: 18
        }).addTo(mymap);
axios.get(`https://nominatim.openstreetmap.org/search.php?q=${encodeURIComponent
('<?php echo $place['location']; ?>')}&format=jsonv2`)
            .then(response => {
                const point = response.data[0];
                if (point) {
                    const marker = L.marker([point.lat, point.lon]).addTo(mymap);
                    marker.bindPopup(`<b><h3><?php echo $place['name']; ?></h3></b>`);
                    mymap.setView([point.lat, point.lon], 12.5);
                }
            })
            .catch(error => console.error(error));
    </script>
```

```
<script>
const stars = document.querySelectorAll('.rating-stars .star');
const ratingValue = document.getElementById('rating-value');
stars.forEach((star, index) => {
    star.addEventListener('click', () => {
        ratingValue.value = index + 1;
        resetStars();
        highlightStars(index);
    });
    star.addEventListener('mouseover', () => {
        resetStars();
        highlightStars(index);
    });
    star.addEventListener('mouseout', () => {
        resetStars();
        const selectedRating = ratingValue.value;
        if (selectedRating !== '') {
            highlightStars(selectedRating - 1);
        }
    });
});
```

```
  function resetStars() {
      stars.forEach(star => {
          star.classList.remove('active');
      });
  }
  function highlightStars(index) {
      for (let i = 0; i <= index; i++) {
          stars[i].classList.add('active');
      }
  }
  </script>
  <script>
function openBookingForm() {
  var bookingForm = document.getElementById("bookingForm");
  var overlay = document.getElementById("overlay");
  bookingForm.style.display = "block";
  overlay.style.display = "block";
}

var overlay = document.getElementById("overlay");
overlay.onclick = function() {
```

```
    var bookingForm = document.getElementById("bookingForm");
    var overlay = document.getElementById("overlay");
    bookingForm.style.display = "none";
    overlay.style.display = "none";
  };
</script>
</body>
</html>
```

2. **Файл manifest.json**

```
{
  "background_color": "#ffffff",
  "description": "PWA description",
  "dir": "ltr",
  "display": "standalone",
  "id": "com.example.app1",
  "name": "CafeFinder",
  "orientation": "any",
  "scope": "/",
  "short_name": "CafeFinder",
  "start_url": "/catalog.php",
  "theme_color": "#BC987E",
```

```
"categories": [],
"screenshots": [],
"icons": [
  {
    "src": "logo/windows11/SmallTile.scale-100.png",
    "sizes": "71x71"
  },
  {
    "src": "logo/maskable_icon.png",
    "sizes": "1024x1024",
    "type": "image/png",
    "purpose": "maskable"
  },
  {
    "src": "logo/windows11/SmallTile.scale-125.png",
    "sizes": "89x89"
  },
  {
    "src": "logo/ios/1024.png",
    "sizes": "1024x1024"
  }
```

```
  ],
  "shortcuts": []
}
```