Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка платформи для сервісу в сфері харчування на основі мікросервісної архітектури

Виконав(ла): студент(ка) 6 курсу, групи СПм-61
спеціальності Інженерія програмного забезпечення
(шифр і назва спеціальності)

	(підпис)	Коваль А.З (прізвище та ініціали)
Керівник	(підпис)	Петрик М. Р. (прізвище та ініціали)
Нормоконтроль	(підпис)	Стоянов Ю. М. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Петрик М. Р. (прізвище та ініціали)
Рецензент	(підпис)	Дуда О. М. (прізвище та ініціали)

Тернопіль
2022

РЕФЕРАТ

Кваліфікаційна робота на тему «Розробка платформи для сервісу в сфері харчування на основі мікросервісної архітектури» Коваля Андрія Зеновійовича, студента Тернопільського Національного Технічного Університету імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, група СПм-61.

Відомості про обсяг: сторінок – __, рисунків – 47, таблиць – 0, частин – 4, додатків – 4, посилань – 16.

Метою роботи є розробка платформи для оптимізації послуг в сервісі в сфери харчування та на основі мікросервісної архітектури.

Дана розробка покликана покращити сервіс обслуговування клієнтів закладів харчування шляхом оптимізації надання послуг закладами та економії часу клієнтів за допомогою попереднього замовлення.

Практичне її застосування полягає у наданні можливості власникам закладів харчування публікувати заклади, керувати ними, продуктами та замовленнями, користувачам здійснювати зручне замовлення. Також отримані результати мають цінність як взірець реалізації архітектури мікросервісів.

Ключові слова: ПЛАТФОРМА, МІКРОСЕРВІС, ПОПЕРЕДНЄ ЗАМОВЛЕННЯ, АРХІТЕКТУРА.

ABSTRACT

Qualification work on the topic "Development of a platform for service in the food sector based on microservice architecture" by Koval Andriy Zenoviyovych, a student of Ternopil National Technical University named after Ivan Pulyu, faculty of computer and information systems and software engineering, group SPm-61.

Information about the volume: pages - __, figures - 47, tables - 0, parts - 4, appendices - 4, references - 16.

The purpose of the work is to develop a platform for optimizing services in the food service and based on microservice architecture.

This development is designed to improve the customer service of catering establishments by optimizing the provision of services by establishments and saving customers' time by pre-ordering.

Its practical application consists in providing the opportunity for the owners of food establishments to publish establishments, manage them, products and orders, and for users to make a convenient order. Also, the obtained results have value as a model for the implementation of microservices architecture.

Keywords: PLATFORM, MICROSERVICE, PRE-ORDER, ARCHITECTURE.

ЗМІСТ

ВСТУП

Сфера з надання послуг харчування широко застосовує програмне забезпечення та Інтернет задля реклами, просування бізнесу, або задля безпосереднього надання послуг, як це робиться в багатьох інших сферах. Власники ресторанів, кафе, кав'ярень та інших закладів доставляють замовлення додому, приймають оплату телефоном чи резервують столики онлайн. При цьому все ще існують послуги які потребують покращення, але ринок розвивається і відкриваються нові можливості для оптимізації.

Так, у сфері харчування є недолік в системі обслуговування клієнтів. Він полягає у тому, що клієнтам закладів харчування часто доводиться довго очікувати на обслуговування та на приготування замовлення. Очікування на замовлення відвідувачу може завдавати незручностей, а самому закладу спричинити зайву завантаженість персоналу.

Даний сервіс можна покращити впровадженням попереднього замовлення. Це практика замовлення товару чи послуги ще до його фактичного отримання. В даному випадку передбачається, що клієнт замовляє щось з меню закладу на вказаний в час і отримує замовлене при прибутті в заклад.

Перевага попереднього замовлення полягає не лише у заздалегідь забронюваному місці у ресторані, а і у тому, що відвідувач може замовити щось із запропонованого рестораном на сайті меню і у бажаний для себе час при відвідуванні зразу його отримати, не витрачаючи час на обслуговування офіціантами та приготування. Також такий сервіс значно прискорить потік клієнтів та допоможе закладам швидко виконувати замовлення у пікові години.

Актуальність даної роботи зумовлена тим, що сфера харчування є однією з найпопулярніших на ринку. Щодня заклади харчування відвідують мільйони людей, для яких впровадження полегшення обслуговування є доречним. Сервіс дозволяє користувачам планувати свої прийоми їжі протягом дня та заощаджувати час на кожному візиті у ресторан. Також в нас час карантинні обмеження роблять дану роботу особливо актуальною.

Мікросервісна архітектура як сучасний шаблон проектування є хорошим рішенням для реалізації практичного застосування попереднього замовлення в сфері харчування і відповідно покращення надання послуг в цілому. Мікросервіси виникли в світі автоматизації інфраструктури та масштабованих систем та мають ряд переваг, які полягають в гнучкості структури та децентралізації.

Платформа, реалізована на мікросервісах, буде придатна для майбутній змін, впровадження нових послуг та оптимізації вже наявних на ринку. Практичне її застосування полягає у наданні можливості власника харчування публікувати заклади, керувати ними, продуктами та замовленнями. Користувачі, які є клієнтами, можуть шукати ресторани, кафе чи інші подібні заклади поблизу, переглядати меню, бронювати місце та здійснювати зручне замовлення.

Дана розробка покликана покращити сервіс обслуговування клієнтів закладів харчування. Буде корисна як для підприємців, що власниками ресторанів, так і для їх клієнтів.

Метою та завдання виконання кваліфікаційної роботи є проектування та реалізація платформи, яка допомагає власникам закладів харчування публікувати їх в Інтернеті, а клієнтам здійснювати попереднє замовлення. Дане завдання зумовлене доцільністю розробки у сфері, її практичним застосуванням, актуальністю в даний час та необхідність для потенційних користувачів.

1 АНАЛІЗ ВИМОГ ДО ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд сфери харчування

В даний час на ринку сфери харчування власники закладів можуть запропонувати клієнтам різні варіанти обслуговування та можливості для їх оптимізації, включно ті, що здійснюються за допомогою інформаційних технологій. Серед основних можна виділити такі:

- обслуговування столика
- бронювання столика;
- оплата замовлення електронними платіжними засобами;
- замовлення доставки до дому чи інше вказане місце;
- видача з собою.

Використання програмних продуктів в сфері обслуговування вже стало невід'ємною частиною для здійснення обслуговування клієнтів або керування закладом.

Наприклад, бронювання столика в закладах різних міст України можна здійснити онлайн на сайті RestOn. Сайт дозволяє знайти і вибрати бажаний ресторан чи інших бажаний заклад (див. рис. 1.1).

RestOn пропонує вибрати заклад вказаного міста з представлених в каталозі, або ж знайти бажаний на карті. Можливість виконати пошук закладів харчування на карті допомагає користувачам вибирати незнайомі для себе заклади, враховуючи своє місце знаходження. Користувач здійснює бронювання столика, вказавши кількість місць та час.

Також послуги закладам харчування допомагає надавати сервіс під іменем Mister.Am. Його основна задача – це доставка їжі до дому з ресторанів або ж здійснити вивіз замовленого самостійно. Сервіс працює в декількох обласних центрах України, серед яких немає великих міст.

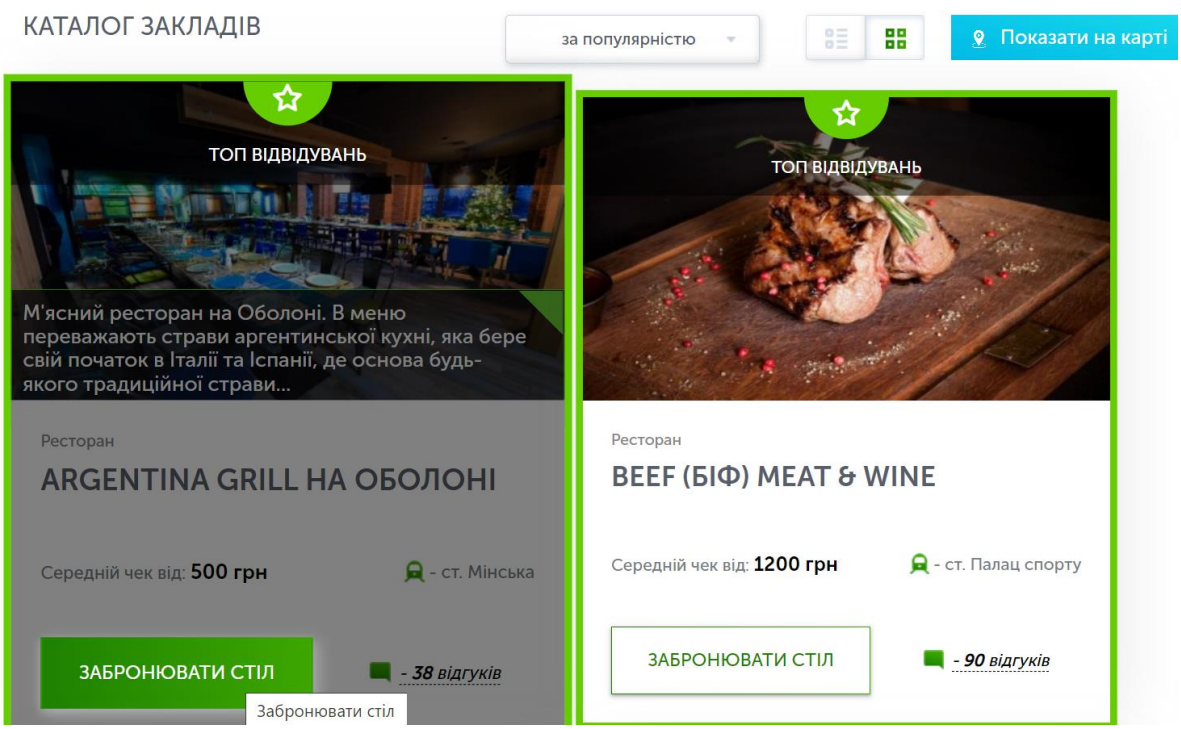

Рисунок 1.1 – Бронювання столика в RestOn

На сайті клієнт здійснює пошук та вибирає заклад, в яких хоче навідатися, а бо з якого хоче отримати доставку до дому. З меню вибирає страви, які бажає замовити, та переходить до здійснення замовлення, що наведено на рисунку 1.2

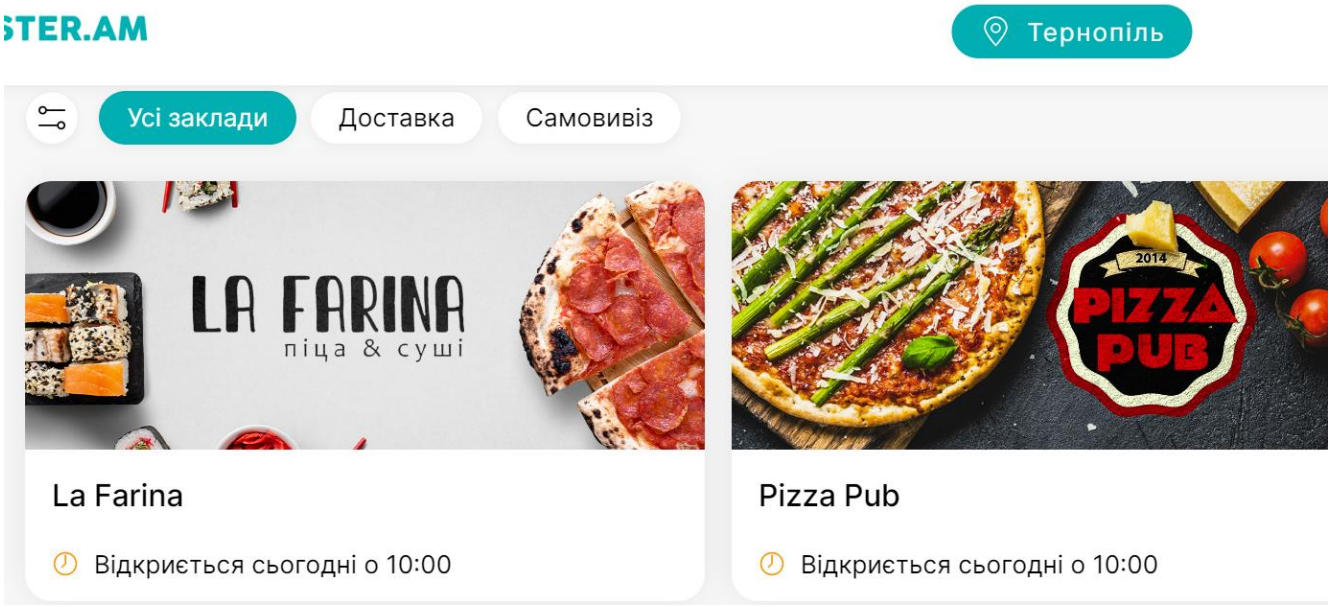

Рисунок 1.2 – Сайт замовлення їжі Mister.Am

Проте заклади харчування практично на надають послугу попереднього замовлення, яка є досить поширеною на світовому ринку в різних сферах і реалізується в тому числі різними сервісами на кшталт згаданих вище.

Практика попереднього замовлення передбачає завчасне замовлення товару чи послуги до моменту їх реалізації. Так, часто виробники надають можливість попередньо оформити замовлення свого товару ще до того як товар буде виготовленим та стане доступним для продажу. Покупцям при цьому може пропонуватися повністю або частково оплати замовлення, чи не оплачувати взагалі. Наприклад, автор може використати кошти, отримані за попереднє замовлення його, ще не надрукованої книги, для друку і видавництва тієї ж книги.

Попередні замовлення дозволяють споживачам гарантовано отримати товар, при чому зразу після його випуску. Виробники в свою чергу можуть зрозуміти, яким буде попит, і відповідно до цього оцінити початкову кількість товарів для продажу чи ресурси для здійснення послуги [1].

В закладах харчування послуга попереднього замовлення може реалізовуватися можливістю для клієнтів замовити щось з меню ще до приходу клієнта в сам заклад.

Це схоже на звичайне бронювання столика на вказаний час, але крім цього клієнт робить замовлення страв з меню ресторану, а персонал готує замовлення на вказаний клієнтом час. Таким чином клієнт отримує бажане зразу при відвідуванні закладу та економить свій завдяки відсутності необхідності чекати на обслуговування та на приготування їжі.

Така послуга може надаватися закладам які обслуговують столики, коли клієнт замовляє їжу за столом і вона подається офіціантами. А таке обслуговування поширене в більшості ресторанів та кафе. А також в різних закладах швидкого харчування, в яких столики не обслуговуються офіціантами, або їжу беруть на виніс.

Звичні для закладів харчування особливі дії в обслуговуванні не надто впливають на роботу закладу в цілому. Наприклад, для ресторану не має значення, здійснить оплату клієнт готівкою чи скористається картою. А от

реалізація попереднього замовлення на практиці має декілька нюансів, які закладам харчування потрібно враховувати.

Так, в будь якого закладу повинно бути можливість в принципі прийняти попереднє замовлення, адже це не найпростіша задача при відсутності клієнта в самому закладі. Це може здійснюватися за допомогою веб-платформ чи мобільних застосунків. Як це вже реалізовано на ринку багатьма програмними продуктами для, наприклад, бронювання столиків чи замовлення доставки до дому. В даному випадку вони надають можливість власникам публікувати свої заклади, а клієнтам виконувати пошук цих закладів, переглядати їх меню та замовляти бажане.

Також заклади харчування можуть просити попередню оплату замовлення, що є виправданим у випадку попереднього замовлення. Можливість такої оплати може здійснюватися практично лише за допомогою електронний платіжних засобів онлайн на відповідних сервісах.

Заклади можуть підтвердити попереднє замовлення клієнта, або ж відмовити у його виконанні. Такі дії необхідні, адже клієнт має пересвідчитись чи його замовлення готується, і він його отримає в запланований час, інакше він може здійснити замовлення в іншому закладі.

1.2 Постановка задач розробки

Основна мета розроблюваної програмної системи полягає у забезпеченні функціоналу для здійснення споживачами попереднього замовлення в закладах харчування, а також можливість керування виконанням замовлення закладом. Це можна реалізувати за допомогою платформи, яка може надавати інтерфейс веб-сайту.

Програмна система повинна надавати можливість власникам ресторанів, кафе, кав’ярень чи інших закладів публікувати їх в Інтернеті. Власники закладів,

або їх призначені менеджери чи адміністратори, повинні мати змогу керувати своїми закладами, стравами та замовленнями.

Користувач сайту, який є відвідувачем, клієнтом закладів може шукати бажані ресторани на відстані поблизу чи просто наявні в своєму місті та переглядати доступні елементи меню з корисною інформацією. Кінцевою метою клієнтів є здійснення замовлення.

Передбачалося, що на сайті користувачі можуть шукати ресторани, кафе чи інші подібні заклади поблизу, вибираючи їх на карті чи з запропонованих у своєму місті. Далі користувач може переглядати меню, вибирати страви та здійснювати замовлення онлайн.

Ресторан у свою чергу завдяки функціоналу системи та інтерфейсу сайту отримує замовлення та вирішує, виконувати його чи відмовити. Користувач повинен отримувати сповіщення, коли його замовлення будуть підтверджені або готові до обслуговування.

Для власників закладів передбачається такий функціонал:

- розміщення закладу на сайті, щоб його могли знайти відвідувачі;
- надання інформації про заклад;
- отримання замовлень від користувачів сайту;
- прийняття чи відмова замовлення;
- сповіщення клієнта щодо замовлення.

Власники повинні мати змогу розміщувати на сайті всю необхідну інформацію про свій заклад, зокрема меню, адресу закладу, та час роботи. Для відвідувачів закладу передбачається такий функціонал сайту:

- пошук та вибір закладу;
- ознайомлення з інформацією про заклад;
- перегляд меню;
- здійснення замовлення.

Для користувачам потрібно реалізувати інтерфейс який безпосередньо буде здійснюватися пошук закладів. Також для нього мають відображатися

рекомендації, згідно його попередніх запитів пошуку на сайті. Такі функції мають бути доступними і не зареєстрованим користувачам.

Для доступу до системи користувачу необхідно виконати авторизацію на яка підтверджуватиме його особу. Користувач може увійти в систему, використовуючи своє ім’я користувача та пароль.

Зрозуміло, що увійти може попередньо зареєстрований користувач. Також це надає можливість повністю використовувати сайт власника закладів та клієнтам. Не зареєстровані користувачі можуть лише знаходити заклади та переглядати меню, але для здійснення замовлень потрібно зареєструватися. Реєстрація здійснюється із зазначенням електронної пошти.

Вибравши заклад користувач здійснює замовлення та отримує всю необхідну інформацію. Також передбачається відповідний функціонал для прийняття отримання замовлень закладами та можливістю ними керувати.

1.3 Визначення варіантів використання системи

З визначеного функціоналу системи було зрозуміло, що передбачено два типи користувачів системи в залежності від вимог:

- Клієнт закладу;
- Менеджер закладу.

Клієнт вибирає заклад, переглядає меню, здійснює замовлення у вибраному закладі та відвідує його у вказаний час. Діаграма варіантів використання для даного користувача наведена на рисунку 1.3.

Менеджер є користувачем системою від закладу. В дійсності це може бути власник закладу, призначений адміністратор, чи інша особа. Діаграма варіантів використання для даного користувача наведена на рисунку 1.4. Для обох користувачів передбачено здійснення реєстрації в системі та виконання входу.

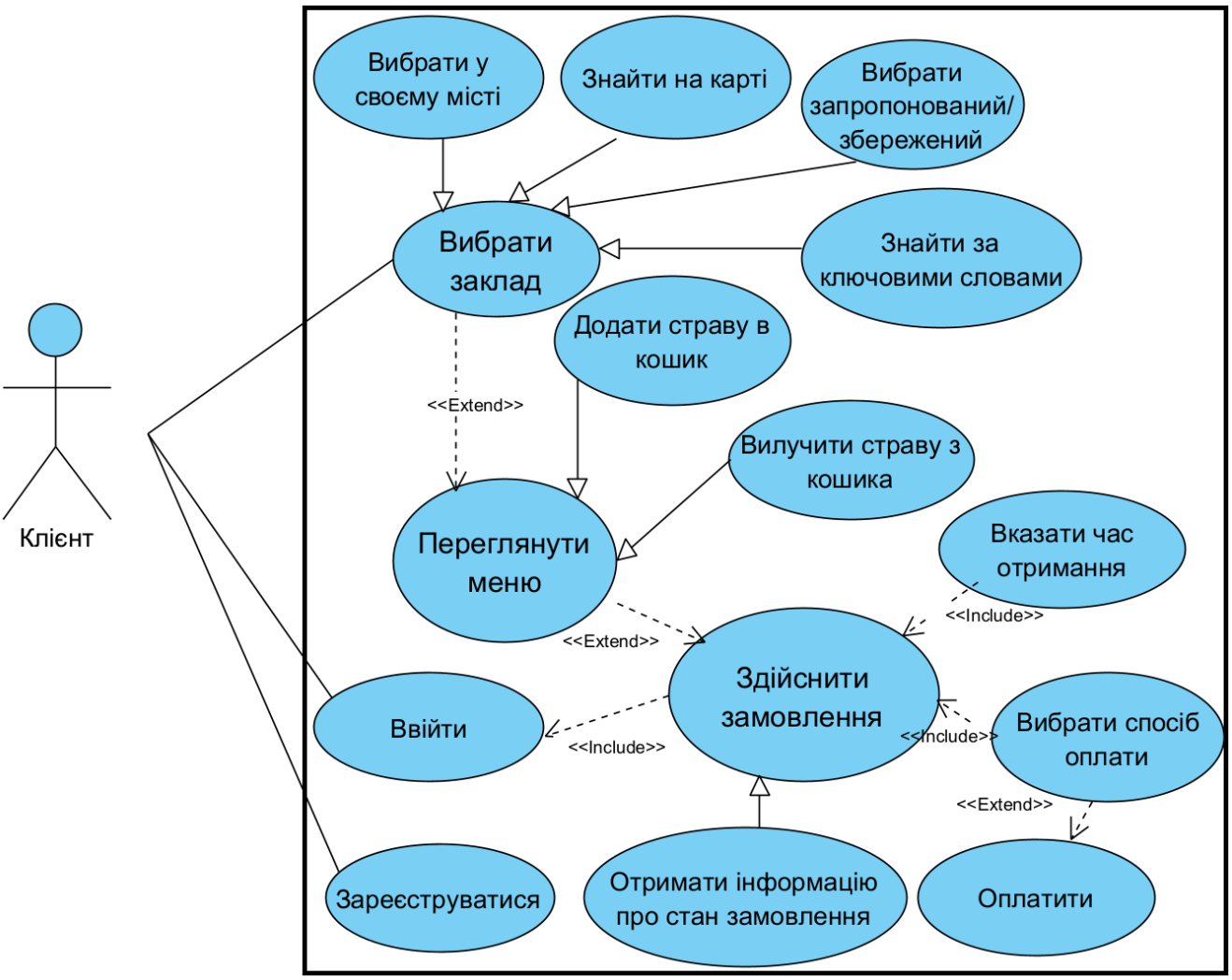

Рисунок 1.3 – Діаграма варіантів використання Клієнта

Для Клієнта визначено такі основні варіанти використання (див. рис. 1.3):

- Вибрати заклад – вибір закладу із знайдених та наданих платформою за одним із способів: знайдені на карті, запропоновані, знайдені поблизу, заклади вказаного міста.
- Переглянути меню – перегляд страв меню вибраного закладу з можливістю додавати та видаляти елемент в кошик замовлення.
- Здійснити замовлення – передбачає зазначення бажаного часу отримання та при оплати при необхідності.

Всі варіанти використання системи для Менеджера закладу доступні йому лише при виконанні входу (див. рис. 1.4).

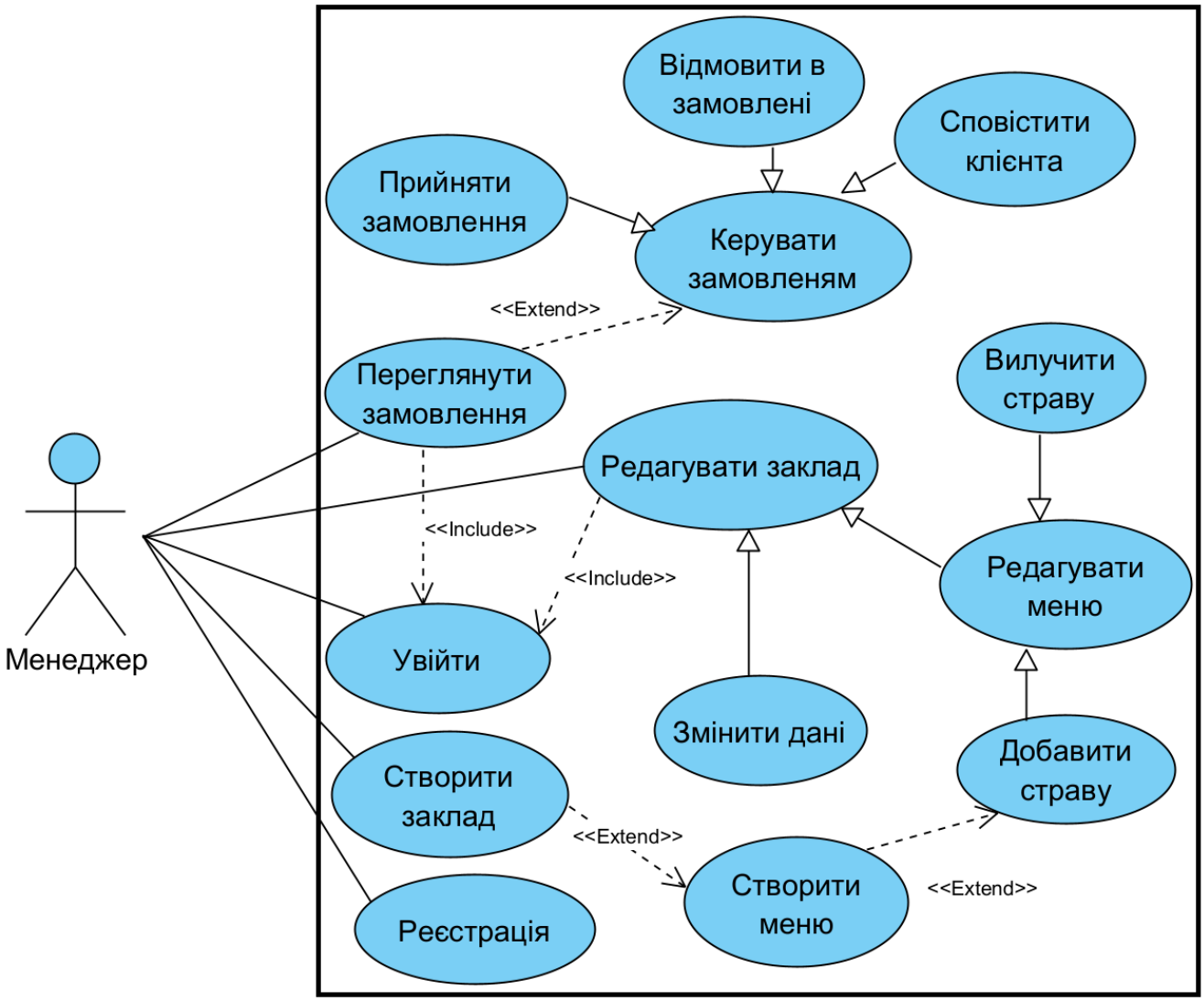

Рисунок 1.4 – Діаграма варіантів використання Менеджера

Для Менеджера визначено такі основні варіанти використання:

- Переглянути замовлення – перегляд списку отриманих закладом замовлень, що передбачай вибір певного для виконання чи відмови.
- Створення закладу – добавляння свого закладу для майбутньої роботи, що також передбачає створення меню та наповнення його стравами.
- Редагувати заклад – зміна інформації про заклад та редагування меню.

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ МІКРОСЕРВІСІВ

2.1 Мікросервісна архітектура в розробці проекту

Програмну систему було вирішено розробляти на основі мікросервісної архітектури. Така побудова добре підходить для даної системи, адже, в ході аналізу вимог до розробки було встановлено, що систему можна поділити на різні компоненти, що реалізують певну бізнес логіку. Це відповідає концепції архітектури, заснованої на мікросервісах.

Мікросервіси розділяють великі програмні системи на менші частини, які працюють незалежно. Кожен сервіс свої обов’язки та може виконувати їх незалежно від того, що роблять інші компоненти. Мікросервіси невеликі, незалежні та слабко пов’язані. Кожен відповідає за збереження власних даних або зовнішнього стану, а деталі внутрішньої реалізації кожного сервісу приховані від інших сервісів. Також вони не потребують спільного використання одного стеку технологій, бібліотек або фреймворків, за потреби сервіси можуть зберігати та обробляти дані за допомогою різних методів і можуть бути написані на інших мовах програмування [2].

Архітектура мікросервісів складається з набору таких невеликих автономних сервісів. Структуру архітектури мікросервісів наведено на рисунку 2.1. Вона ідеально підходить для сучасних цифрових компаній, які не завжди можуть врахувати всі різні типи пристроїв, які матимуть доступ до їх інфраструктури. Численні додатки, які починалися як моноліт, були повільно перероблені для використання мікросервісів, оскільки у світі після пандемії з’явилися непередбачені вимоги [3]. Тож навіть ситуація на світовому ринку обґрунтовує застосування мікросервісної архітектури ще з початку розробки.

Реалізація архітектури мікросервісів призводить до створення систем, які є гнучкими та масштабованими. Це дозволить уникнути складнощів, які можуть виникнути, як от при реалізації монолітної архітектури, у якій всі процеси значною мірою залежать один від одного та працюють як єдиний сервіс.

Кожен мікросервіс має власну базу даних для повної незалежності від інших. При необхідності виконати певну дію з користувацького інтерфейсу виконується запит до відповідного сервісу (див. рис. 2.1).

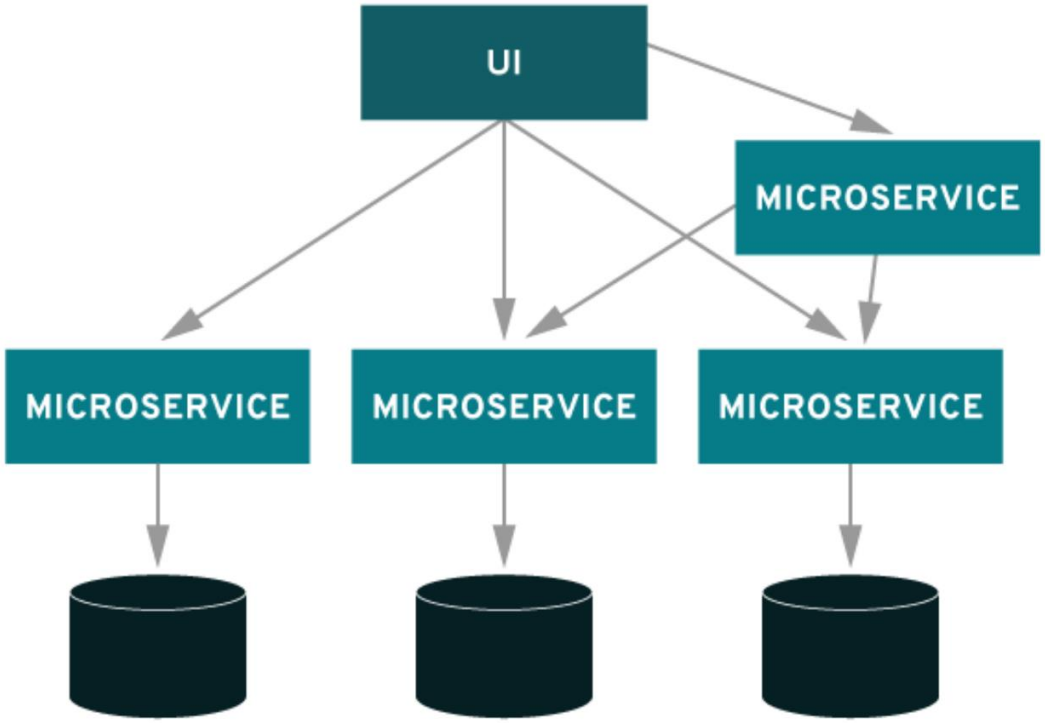

Рисунок 2.1 – Архітектура мікросервісів

Мікросервісні програми можна без зайвих проблем змінювати, оскільки вся система слабо взаємопов’язана. Таку архітектуру легко масштабувати, оскільки масштабування конкретної функції вимагатиме масштабування лише одного сервісу. Наприклад, монолітні програми можуть бути вразливими до збоїв, що пояснюється тим, що на тісно пов’язані, по суті взаємозалежні процеси важко впливати, якщо один процес виходить з ладу. В мікросервісах таких проблем не виникає [4].

Таким чином реалізація даної розробки на основі мікросервісів дозволить уникнути проблем з майбутніми змінами в наявному функціоналі чи при реалізації нового.

Отже, для даної розробки було визначено та спроектовано декілька мікросервісів в залежності від необхідного функціоналу:

- Identity – сервіс керування авторизацією користувачів;
- Restaurant – сервіс керування даними закладів;
- MenuCart – сервіс меню та керування кошиком клієнтів для замовлення;
- Order – сервіс керування замовленнями клієнтів;
- Payment – сервіс для здійснення оплати замовлення;
- Notification – сервіс сповіщення клієнта про стан замовлення.

Кожен із наведених сервісів було визначено відповідно до вимог системи, та враховано, що мікросервіси повинні охоплювати єдиний спеціалізований процес, мати високу згуртованість і демонструвати мінімальну залежність один від одного. Деякі сервіси мають більше завдань у системі, ніж інші, як наприклад сервіс Restaurant, що вимагало його детальнішого планування.

Враховано, що мікросервіси не класифікуються виключно за бізнес-функціями, оскільки це може створити мікросервіси, які є занадто малими або занадто великими. Перше може призвести до непотрібного збільшення операційних витрат, тоді як друге мінімізує вигоду від використання мікросервісів, які вже не є такими малими незалежними компонентами. Це було зауважено при визначені сервісу Cart, функціонал якого при гіршому, але при цьому неочевидному плануванні, міг би виконувати Restaurant.

Для кожного сервісу спроектовано різні бази даних і засоби керування даними для інших мікросервісів, які задовольнити конкретні вимоги кожного мікросервісу. Це також є важливим аспектом, який дає змогу створити надійну архітектурну структуру мікросервісів, де кожен керуватиметься незалежно, працюючи узгоджено з іншими.

Хорошою практикою вважається, що клієнт не повинен мати прямого зв’язку з мікросервісами. Замість безпосереднього виклику сервісу користувачі викликають шлюз API, який перенаправляє виклик до відповідних сервісів на сервері.

Таким чином, виходячи з принципів проектування мікросервісної архітектури, визначено загальну структуру архітектури даного проекту. На рисунку 2.1. наведено діаграму вигляду архітектури проекту.

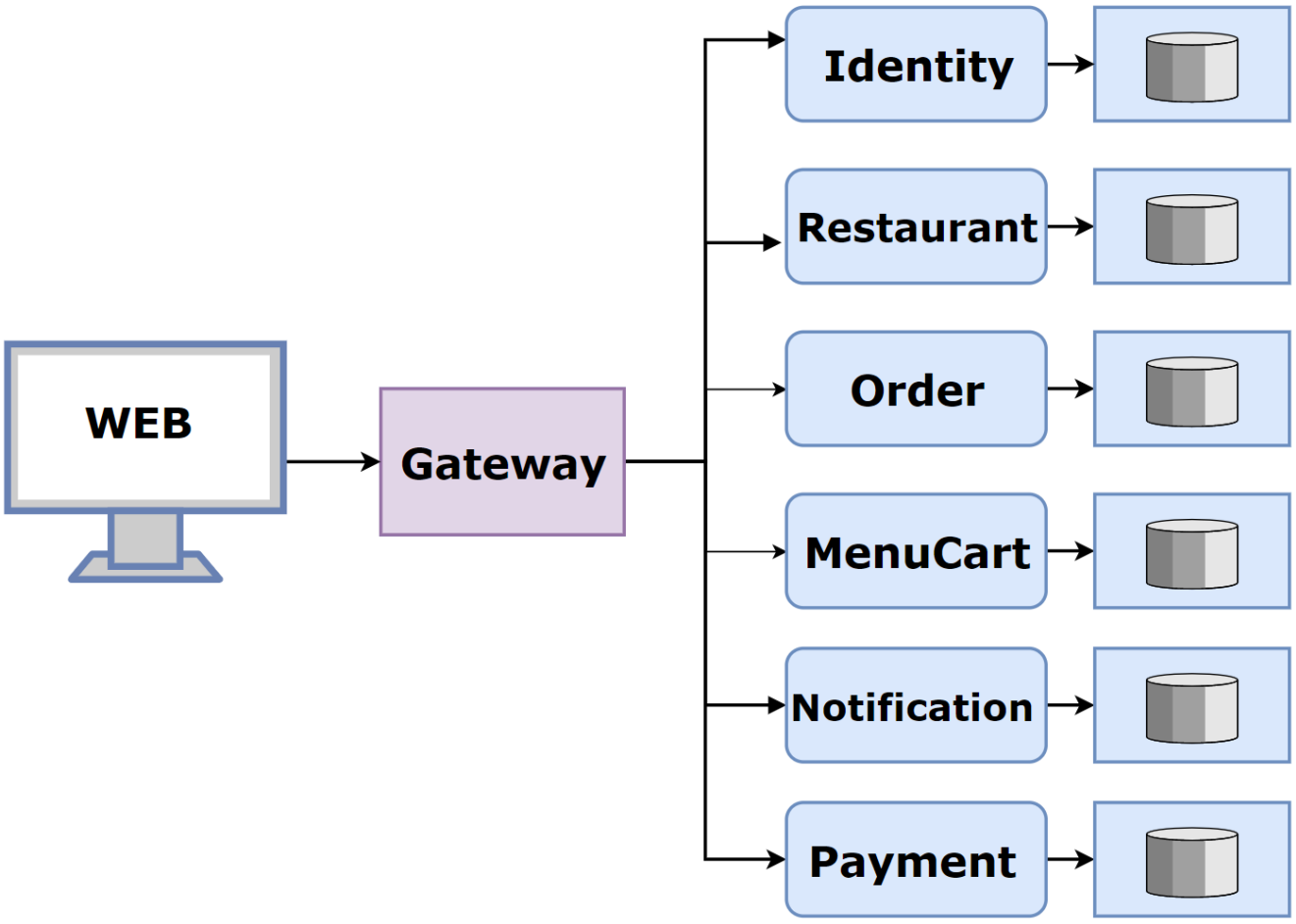

Рисунок 2.1 – Архітектура проекту

В даній розробці в якості клієнта передбачається веб-сайт, який надаватиме графічний інтерфейс.

Також сервіси взаємодіють зі шлюзом API, який є точкою входу для користувачів. При створені систем на основі мікросервісів безпосередній зв’язок з ними може створити проблеми. Наприклад, якщо потрібно передбачити можливість добавити нові мікросервіси, то керувати ними з клієнтського інтерфейсу дуже важко.

Це може спричинити велику кількість запитів до сервісів та створити можливу затримку та ускладнити інтерфейс користувача. В ідеалі відповіді мають бути агреговані на стороні сервера. Тому замість безпосереднього виклику сервісу користувачі викликають так званий шлюз API, який перенаправляє виклик до відповідних сервісів на сервері Крім того, шлюз API керує маршрутизацією до внутрішніх мікросервісів і може об’єднувати кілька запитів в одну відповідь [5].

2.2 Мікросервіс Identity

Сервіс Identity відповідає за дані користувачів та керує їх авторизацією та реєстрацією. Мікросервіс містить в своїй базі даних інформацію про власників закладів та їх клієнтів, а саме ім’я користувача, електронну пошту та пароль. Відповідно і використовується при реєстрації чи вході в систему клієнта закладу або його власника.

При здійсненні входу і виконується створення точену, шляхом перевірки даних сервісу при звертанні до власної бази даних. Сервіс використовує систему автентифікації на основі точена – це частина інформації, яка підтверджує особу користувача на веб-сайті, сервері або будь-кому, хто запитує підтвердження.

Користувач запитує доступ до сервера або захищеного ресурсу. Сервер визначає, чи користувач повинен мати доступ, виконує перевірку пароля на ім’я користувача, а далі видає токен і передає його користувачеві. Під час роботи маркер зберігається в браузері користувача. Якщо користувач намагається відвідати іншу частину сервера, маркер знову зв’язується з сервером [6].

Послідовність виконання входу користувача наведено в діаграмі послідовностей на рисунку 2.3.

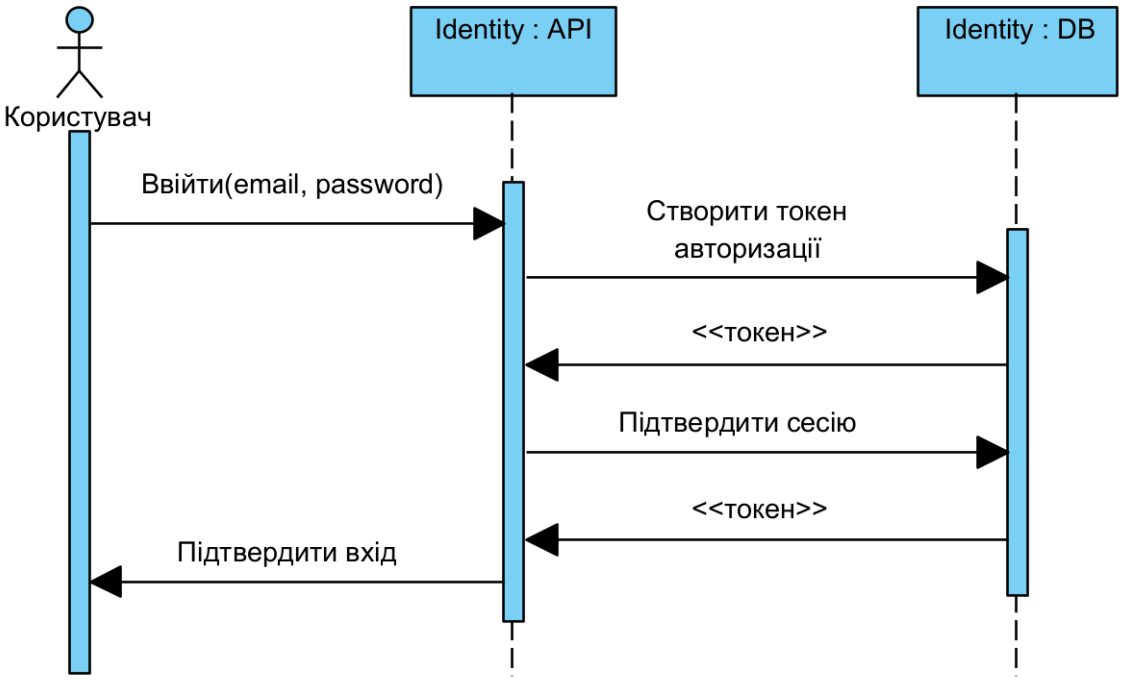

Рисунок 2.3 – Діаграма послідовностей входу користувача

При здійсненній входу сервіс Identity звертається в свою базу даних для отримання і перевірки даних наданих користувачем та з метою створити токен. При підтверджені даних повертає створений токен та дозволяє вхід.

2.3 Мікросервіс Restaurant

Сервіс Restaurant працює з даними закладів харчування, відповідає за функціонал для керування закладами, а також для здійснення пошуку закладів. Використовується власниками закладів для таких варіантів використання:

- створення закладу;
- створення страв для меню;
- додавання елементів меню;
- видалення елементів меню;
- редагування інформації про заклад.

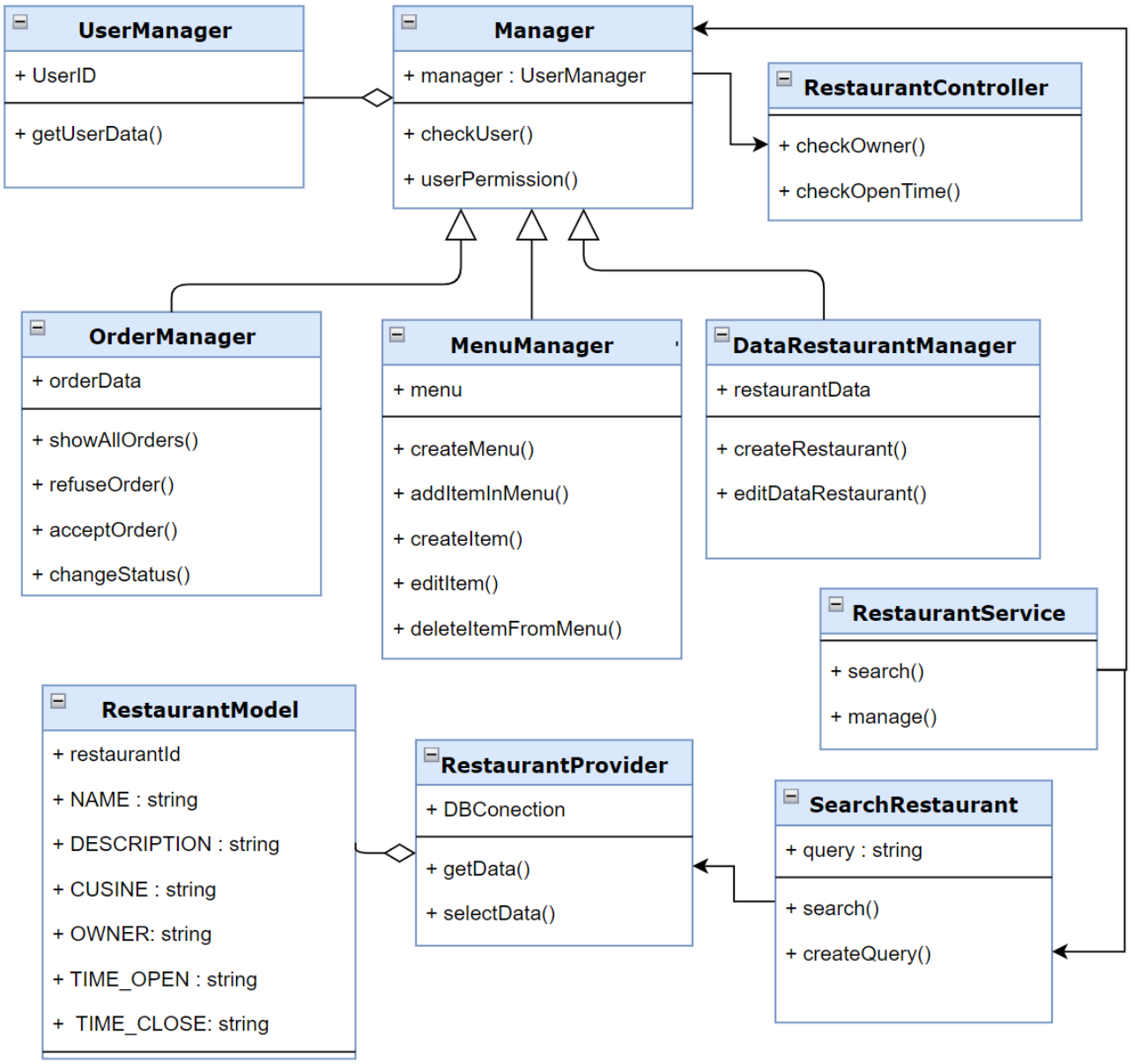

Рисунок 2.4 – Діаграма класів сервісу Restaurant

В системі класів мікросервісу Restaurant присутній клас Manager, від якого є похідним класи для керування закладом, а саме клас для керування даними ресторанами і клас керування елементами меню закладу.

Оскільки сервіс також виконує пошук закладів в своїй базі даних, то для розподілу дії введено клас RestaurantService. Клас SearchRestaurant виконує пошук в базі даних, формуючи відповідний запит, в залежності від типу пошуку.

В базі даних даний мікросервіс містить зареєстровані заклади, інформацію про них та страви з меню закладу. На рисунку 2.5 наведено ER – діаграму сервісу Restaurant.

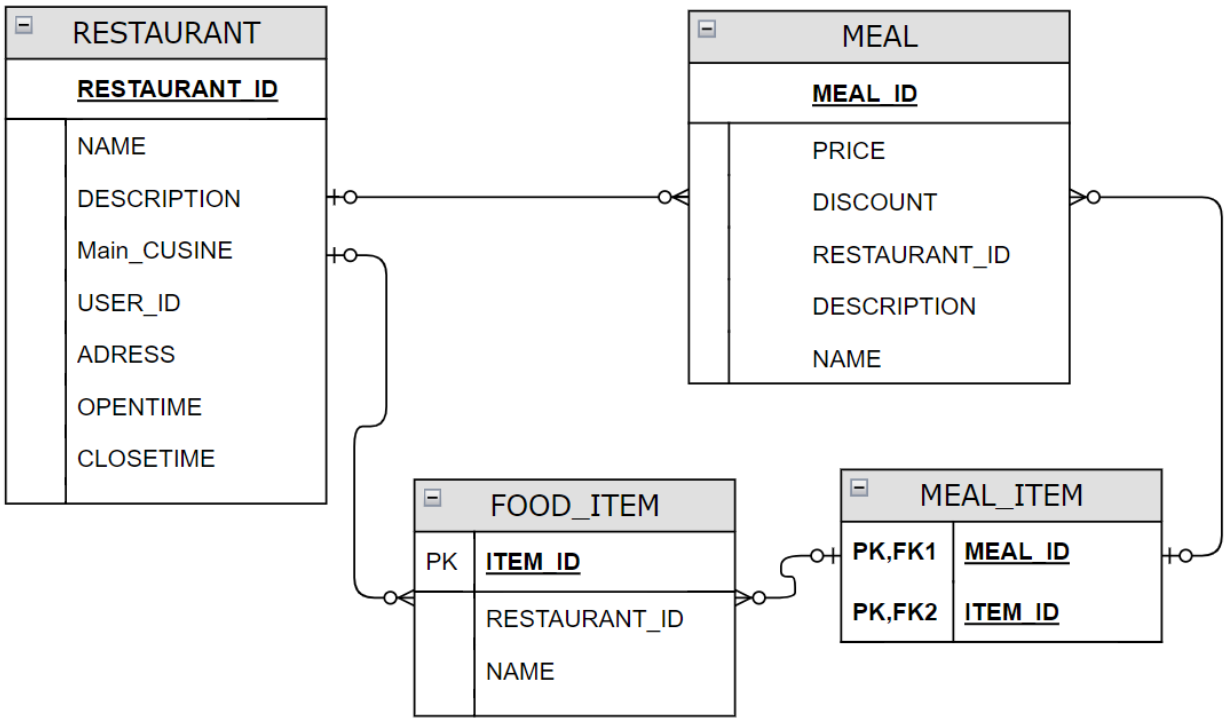

Рисунок 2.5 – ER-діаграма бази даних сервісу Restaurant

База даних містить сутність Restaurant, яка і відповідає суті закладу харчування. Сутність містить такі атрибути, що характеризують заклад та необхідні для надання інформації про нього клієнтам:

- NAME – назва закладу;
- DESCRIPTION – загальний опис, який показується клієнтам;
- Main_CUSINE – вид основної кухні закладу;
- USER_ID – ідентифікатор власника закладу;
- ADRESS – адреса місця розташування закладу;
- OPEN_TIME – час відкриття;
- CLOSE_TIME – час закриття закладу.

Також в базі даних міститься елементи меню заклад які в свою чергу містять інгредієнти зазначеної кількості.

Сервіс не використовується безпосередньо клієнтом закладів, але він приймає запит від користувачів на пошук закладів (див. рис. 5).

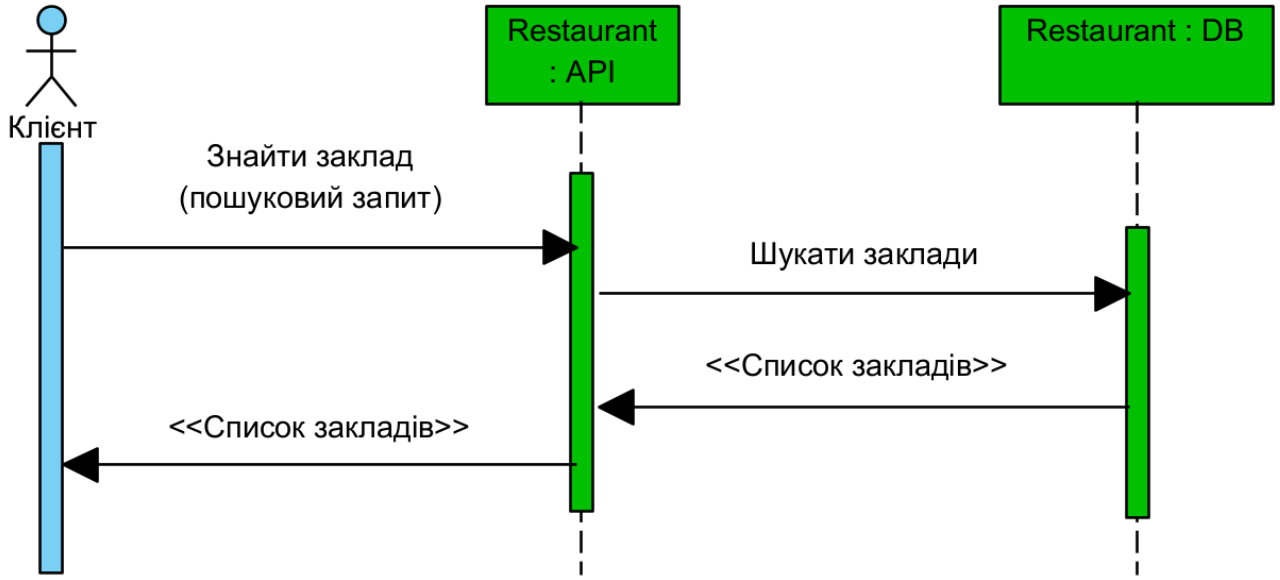

Рисунок 2.6 – Діаграма послідовні пошуку закладів

Клієнт шукає заклади за своїм містом, вибирає запропоновані, або намагається знайти за ключовим словом, яке може бути назвою закладу, видом його кухні, чи стравою з меню. Сервіс формує відповідний запис до своєї бази даних та повертає список знайдених закладів. При виборі певного закладу клієнт починає взаємодіяти зі сервісом MenuCart.

2.4 Мікросервіс MenuCart

Сервіс MenuCart працює лише клієнтом, надаючи йому інтерфейс роботи з меню закладу. Він надсилає інформацію в інтерфейс користувача, коли

користувач вибирає певний заклад і хоче побачити всі пункти меню цього закладу.

Оскільки даний мікросервіс не містить даних закладів, але при цьому працює з їх меню, він отримує необхідну інформацію від сервісу Restaurant (див. рис. 2.7).

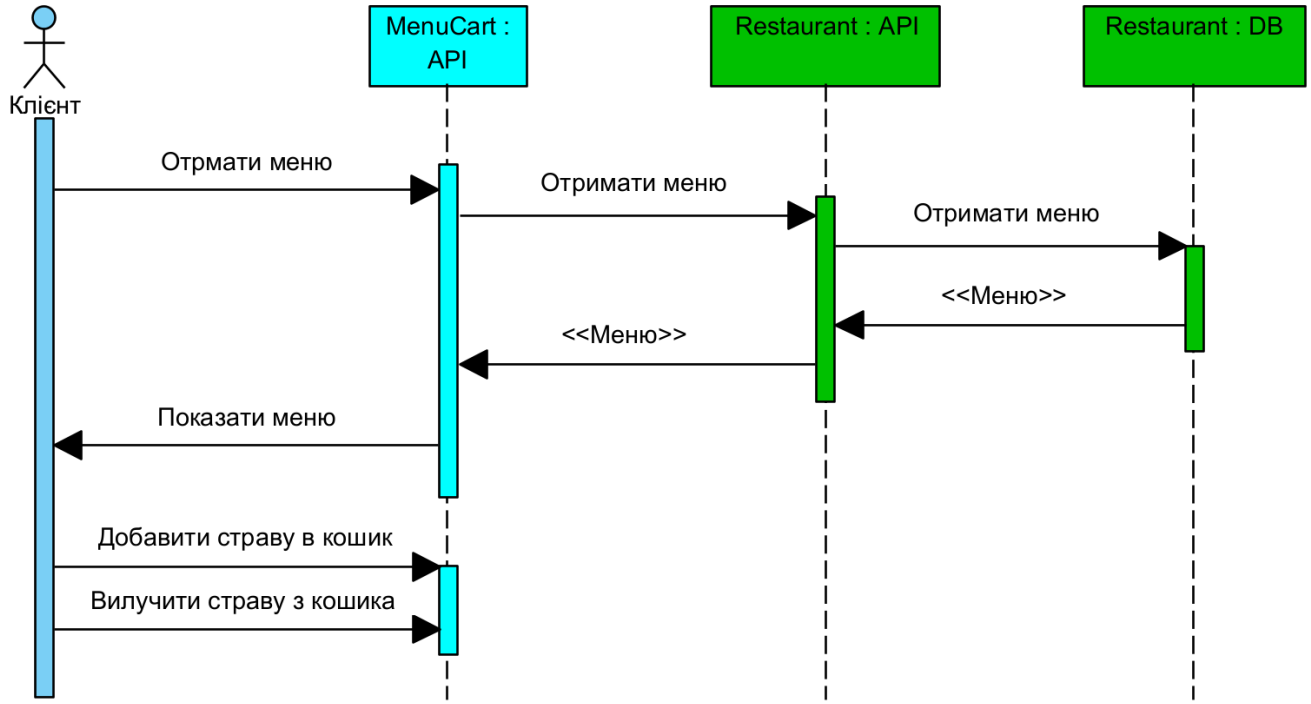

Рисунок 2.7 – Діаграма послідовності перегляду меню

Також мікросервіс виконує функції кошика замовлення. В кошик клієнт додає елементи меню, які бажає замовити, але ще не замовляє. Так, в кошик можна додавати страви та видаляти, якщо клієнт вирішить замовити щось інше. В кошику введеться підрахунок сумарної вартості вибраних на даний момент страв.

2.5 Мікросервіс Order

Сервіс Order використовується обома користувачами. Він відповідальний за керування замовленнями. Клієнт використовує даний сервіс коли хоче здійснити замовлення вибраних страв. Власник закладу керує отриманими замовленнями.

Коли клієнт визначився з вибором та бажає замовити вибране, сервіс Order звертається до сервісу MenuCart, який передає йому дані з кошика клієнта (див. рис. 2.8).

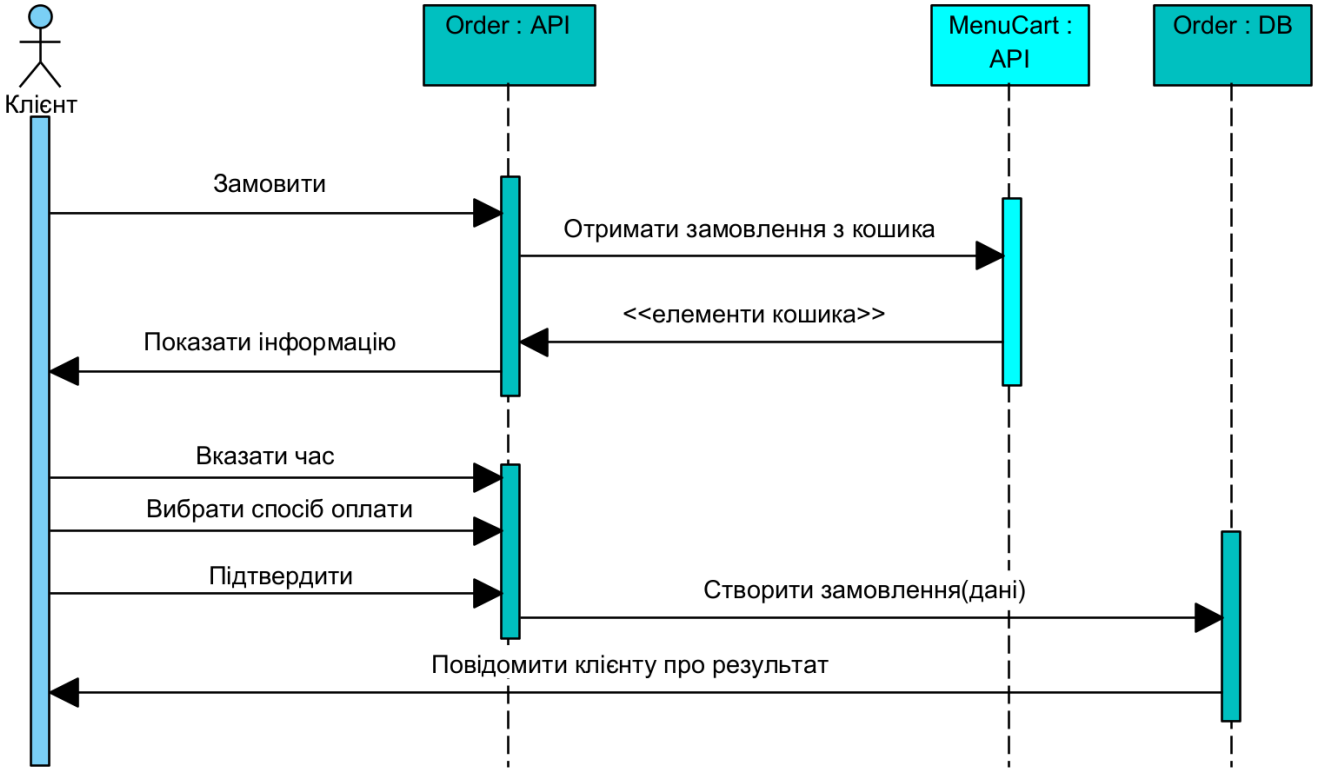

Рисунок 2.9 – Діаграма послідовності здійснення замовлення

Коли клієнт підтверджує замовлення сервіс створює відповідний запис у своїй базі даних, що видно на наведеній діаграмі. База даних мікросервісу містить інформацію про замовлення та список замовлених страв. ER-діаграму бази даних сервісу Order наведено на рисунку 2.9.

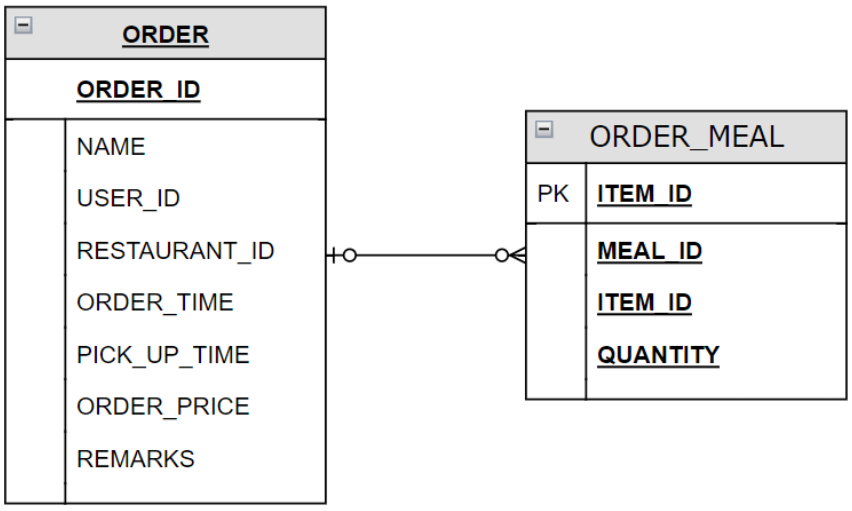

Рисунок 2.10 – ER-діаграма сервісу Order

Таблиця Order даного мікросервісу містить основну інформацію про замовлення, а саме:

- NAME – ім’я клієнта, що здійснює замовлення;
- USER_ID – ідентифікатор користувача;
- RESTAURANT_ID – ідентифікатор закладу, в якому замовляє клієнт;
- ORDER_TIME – час здійснення замовлення;
- PICK_UP_TIME – час отримання замовлення, вказаний клієнтом;
- ORDER_PRICE – сума до оплати замовлення.

Власник (менеджер, адміністратор) використовує мікросервіс Order для керування замовленнями, які отримує його заклад. Відповідно для власників сервіс надає такий функціонал:

- перегляд отриманих закладом замовлень;
- вибір замовлення;
- прийняття замовлення для виконання;
- відхилення замовлення.

Зі сторони закладу отриманому замовленню встановлюється статус відповідно до того, що було вирішено робити з даним замовленням.

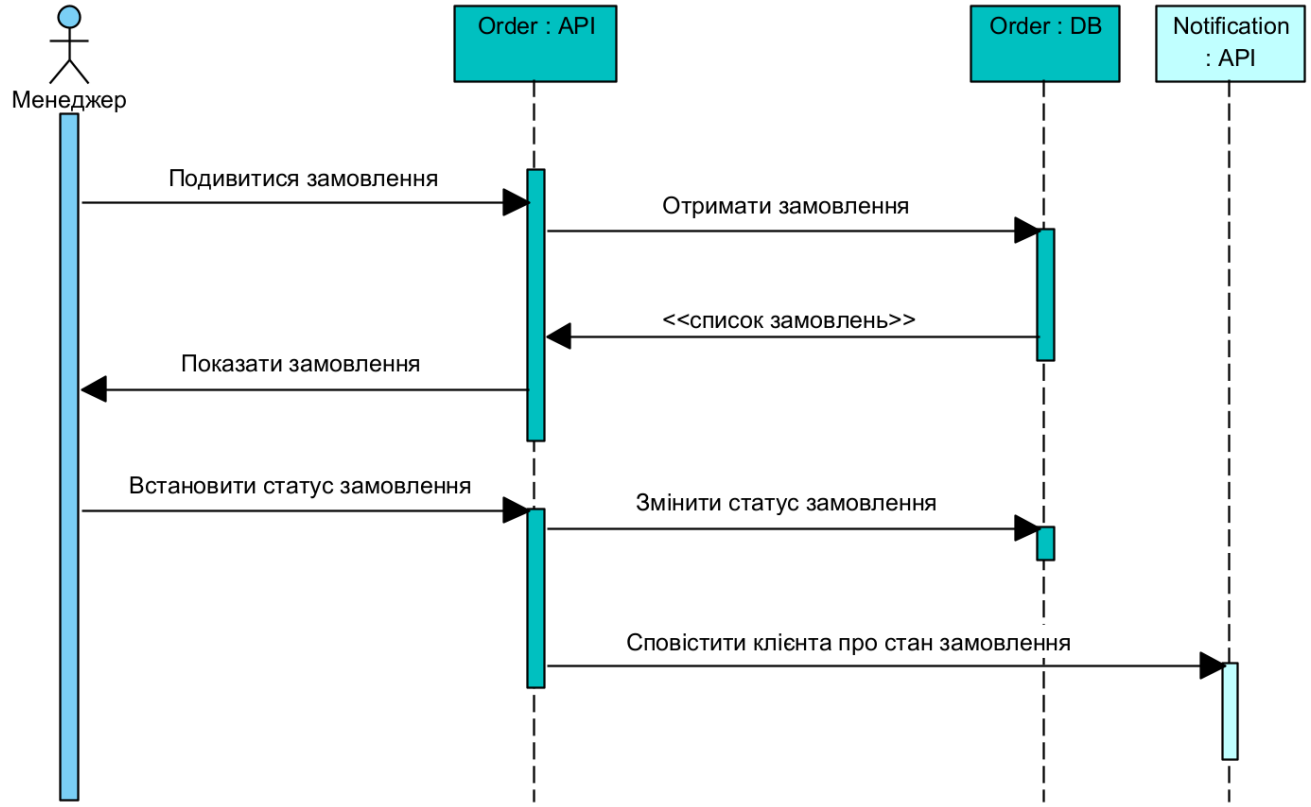

Рисунок 2.11 – Діаграма послідовності керування замовленням

На діаграмі послідовності керування замовленням (див. рис. 2.9) в якості користувача, який використовує систему наведено «Менеджера». Отримавши замовлення, він може встановити відповідний статус: «Прийнято», «Відмовлено», «Готується», «Виконано».

Зміна статусу замовлення супроводжується сповіщенням клієнта шляхом відправлення йому листа на електронну пошту. Це здійснює сервіс Notification.

2.6 Мікросервіс Notification

Сервіс Notification відповідає за відправлення повідомлень клієнтам щодо їх замовлень, а також для сповіщень про результат оплати замовлення. Виходячи з

вимог, до розробки така функція необхідна задля оптимальної реалізації замовлення за допомогою даної платформи в дійсності.

Даний мікросервіс надсилає повідомлення на електронну пошту клієнту коли стан його замовлення змінюється закладом. В залежності від статусу замовлення клієнт може отримати одне з таких повідомлень:

- «Ваше замовлення підтверджено, ми повідомимо вас, коли воно буде готове до отримання»;
- «Заклад відхилив ваше замовлення, спробуйте інший заклад»;
- «Ваше замовлення готове до отримання».

Створення для цього окремого сервісу доцільне, адже різні сервіси часто виконують сповіщення користувачів. Відокремлення такого сервісу від інших дозволить в майбутньому легко створити інші сервіси, яким не доведеться реалізовувати механізм повідомлень, а використовувати вже наявний функціонал для цього. Наприклад в даній розробці можна реалізувати мікросервіс, який відповідальний за акційні пропозиції закладів. Такий сервіс використовуватиме Notification для надсилання пропозицій клієнтам на електрону пошту.

2.7 Мікросервіс Payment

Сервіс Payment надає API для здійснення попередньої оплати за замовлення. Оплату клієнт здійснює при бажанню або у випадку, якщо заклад прийматиме замовлення лише після попередньої оплати за нього.

Даний сервіс використовуватиметься клієнтом не завжди, адже деякі заклади можуть не вимагати оплату заздалегідь і встановлювати клієнту можливість оплатити замовлення на місці при отриманні.

На рисунку 2.12 наведено діаграму послідовності оплати замовлення при сплаті. Даний сервіс отримує інформацію про замовлення від сервісу Order, зокрема ціну всього замовленого клієнтом.

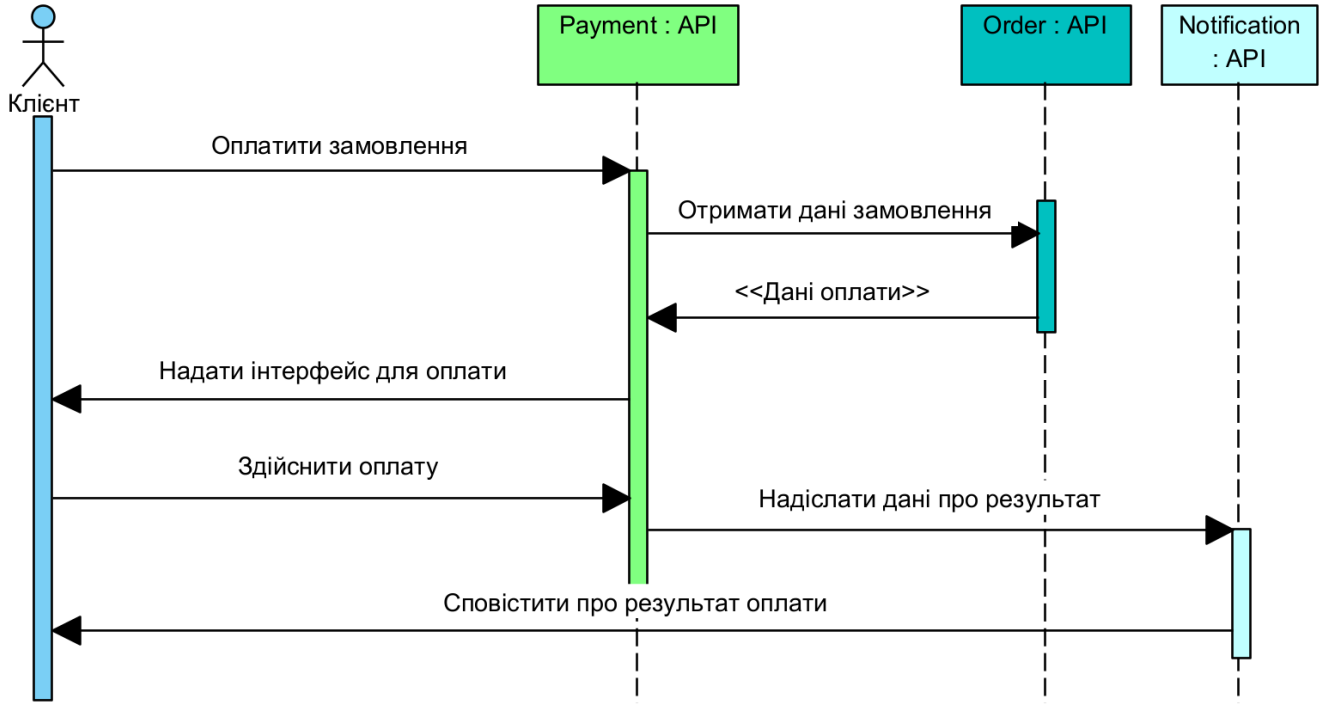

Рисунок 2.12 – Діаграма послідовності виконання оплати

Для оплати сервіс Payment надає клієнту сторонній інтерфейс, який надає можливість оплатити замовлення банківською карткою чи іншим платіжним способом. Також даний сервіс використовує Notification для повідомлення користувача про результат оплати.

3 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Технології розробки

Для реалізації проекту застосовано технології, які допомагають розробляти мікросервісну архітектуру та створювати веб застосунки. Також враховано висновки з проведеного аналізу та сформульованої задачі розробки програмної системи вибрано відповідні інструменти, що задовольняють виконання поставлених цілей.

Розробка мікросервісної архітектури є головним аспектом, який слід було враховувати при виборі технології. Саме тому, для виконання завдання було використано інструмент Docker.

Docker полегшує створення, розгортання та запуск програм на основі мікросервісів за допомогою підходу контейнеризації. Так званні контейнери легкі та потребують менше часу для запуску, ніж традиційні сервери. Контейнери також підвищують продуктивність і знижують вартість, водночас пропонуючи належне керування ресурсами [7].

Переваги системи Docker для швидкої доставки, тестування та розгортання коду дозволяють значно скоротити затримку між написанням коду та його запуском у виробництві. Docker забезпечує автоматизацію розгортання додатків як портативних самодостатніх контейнерів, які можуть працювати в хмарі або локально. Контейнери Docker можна запускати де завгодно: на локальному комп’ютері, або хмарі. В даній розробці вони запускалися локально.

Контейнер являє собою стандартну одиницю програмного забезпечення, яка упаковує код і всі його залежності, щоб програма швидко й надійно запускалася з одного обчислювального середовища в іншому. Образ контейнера Docker містить усе необхідне для запуску проектів. Під час використання Docker розробник розробляє програму та упаковує її разом із залежностями в образ контейнера. Образ – це статичне представлення програми з її конфігурацією та залежностями [8].

Також для створення проекту було обрано технологію .NET, адже вона дозволяє створювати програми для Інтернету, мобільних пристроїв, настільних комп'ютерів, хмарних сервісів тощо. Це одна з найпопулярніших технологій, яка використовує поширену мову програмування C#. Технологія має велику підтримку та потужний набір інструментів, тому є однією найпродуктивнішою платформою для розробників.

.NET дозволяє легко створювати API, які стають мікросервісами . ASP.NET має вбудовану підтримку для розробки та розгортання мікросервісів за допомогою контейнерів Docker, а також містить API для легкого використання мікросервісів із будь-якої створеної програми, включаючи мобільні пристрої, комп'ютери.

Написання коду проводилося в середовищі Microsoft Visual Studio яке включає низку інших інструментальних засобів, зокрема інструменти, що дозволяють розробляти веб-сайти та веб-застосунки. Перевагами обраного середовища є надійність, зручність та простота використання, можливість розгортання мікросервісних систем з Docker та можливість відладки програм, що спрощує пошук логічних помилок в коді. Також середовище надає зручний інтерфейс для роботи з базою даних SQL Server.

SQL Server є реляційною система керування базами даних, яка також розроблена компанією Microsoft. SQL Server побудовано на основі SQL, стандартної мови програмування для взаємодії з реляційними базами даних. Безкоштовна версія ідеально підходить для розробки та виробництва настільних, веб-додатків і невеликих серверних програм, тому використана і в даній розробці.

Для реалізації механізму зв'язку між сервісами було розглянуто протоколи HTTP та підхід REST, який є найпоширенішим способом розробки API [9]. Якщо ми спілкуємося між сервісами всередині нашого кластера мікросервісів, можна також використовувати механізми зв'язку у двійковому форматі, такий як gRPC. Він є одним із найкращих способів спілкування для внутрішньої комунікації мікросервісів.

У GRPC клієнтська програма може безпосередньо викликати метод серверної програми на іншій машині, як локальний об’єкт, що полегшує вам створення розподілених програм і служб. Це забезпечує на 30–40% більшу продуктивність [10]. Оскільки gRPC активно використовує HTTP/2, його неможливо викликати безпосередньо з веб-браузера. Сучасні браузери не забезпечують необхідний контроль над запитами для підтримки клієнта. Отже, для реалізації взаємодії між мікросервісами було обрано gRPC, а для взаємодіє з користувацьким інтерфейсом – REST.

3.2 Реалізація мікросервісів

Кожен мікросервіс даної розробки статично представлений образом Docker. Щоб реалізувати можливість розгортання системи за допомогою нього можна використовувати Dockerfile – файл, який визначає набір інструкцій, які створюють образ [11].

```
FROM mcr.microsoft.com/dotnet/aspnet:3.1 AS base
WORKDIR /app
EXPOSE 80

FROM mcr.microsoft.com/dotnet/sdk:3.1 AS build
WORKDIR /src
COPY ["Restaurant/Restaurant/Restaurant.csproj", "Restaurant/Restaurant/"]
RUN dotnet restore "Restaurant/Restaurant/Restaurant.csproj"
COPY . .
WORKDIR "/src/Restaurant/Restaurant"
RUN dotnet build "Restaurant.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "Restaurant.csproj" -c Release -o /app/publish
```

Рисунок 3.1 – Лістинг коду файлу Dockerfile для ініціалізації сервісу Restaurant

Аналогічно наведеному коду (див. рис. 3.1) виконується ініціалізації усіх інших мікросервісів системи. Це підхід, який реалізовується в платформі .Net. Метод dotnet publish використовується для створення та розгортання образу.

Важливим класом сервісу Identity є клас AccountController. Він виконує перевірку даних користувачів, визначає тип користувача, керує автентифікацією користувачів при вході та реєстрації. Частину його реалізації наведено в лістингу рисунка 3.2.

```
public class AccountController : Controller
{
    private Model db = new Model();

    // GET: USERs
    public ActionResult Index()
    {
        if (User.Identity.IsAuthenticated)
        {
            string userId = User.Identity.GetUserId();
            USER theUser = db.USERs.Find(userId);
            return View("Details",theUser);
        }

        return View("../Account/Login");
    }

    // GET: USERs/Details/5
    public ActionResult Details(string id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        USER uSER = db.USERs.Find(id);
        if (uSER == null)
        {
            return HttpNotFound();
        }
        return View(uSER);
    }
```

Рисунок 3.2 – Клас AccountController сервісу Identity

Одним з головних завдань є реалізації пошуку закладів сервісу Restaurant. Так, на рисунку 3.3 наведено код формування запиту в базу даних для пошуку закладів.

```
results = db.Restaurants.SqlQuery("Select * from Restaurant where latitude <="+
    latRange[1]+" and latitude >="+latRange[0]+" and longitude <="+lngRange[1]+
    " and longitude >="+lngRange[0]+";" );
```

Рисунок 3.3 – Приклад формування запиту пошуку закладів

В лістингу, зображеному на рисунку 3.4, наведено код створення замовлення, яке здійснює клієнт. Даний метод сервісу Order перевіряє отримані дані замовлення. Коли все зроблено правильно, в базу даних заноситься новий запис з даними замовлення. Якщо клієнт намагається вказати час отримання замовлення, коли заклад метод поверне повідомлення про помилку.

```
if (User.Identity.IsAuthenticated)           {
    var userID = User.Identity.GetUserId();
    var newOrder = new RESERVE_PICK_UP();
    OrderData data = JsonConvert.DeserializeObject<OrderData>(json);
    newOrder.PICK_UP_TIME = DateTime.ParseExact(data.pickTime, "dd/MM/yyyy HH:mm
    newOrder.ORDER_TIME = DateTime.ParseExact(data.orderTime, "dd/MM/yyyy HH:mm:
    newOrder.RESTAURANT_ID = data.restaurantID;
    newOrder.ORDER_PRICE = data.totalPrice;
    newOrder.TOTAL_ENERGY = data.totalEnergy;
    newOrder.USER_ID = userID;
    newOrder.STATE = "waiting";

    var restTime = db.Restaurants.Find(newOrder.RESTAURANT_ID);
    if (newOrder.PICK_UP_TIME.TimeOfDay < restTime.OPENTIME || newOrder.PICK_UP_
    {

        return Json(new { result = "failed",reason="Invalid Pick-up Time" });
    }

    db.RESERVE_PICK_UP.Add(newOrder);
    db.SaveChanges();
```

Рисунок 3.4 – Код створення замовлення

В першій стрічці коду створення замовлень (див. рис. 3.4) наведено роботу класу Identity. Метод GetUserId() виконав запит сервісу ідентифікації користувачів задля перевірки надання доступу до системи.

Для сервіс Order надає функціонал для керування замовленнями. На рисунку 3.5 наведено код керування замовленням.

```
var order = db.RESERVE_PICK_UP.Find(id);
var subject = "";
var content = "";
if (action=="confirm")
{
    order.STATE = "confirmed";
    subject = "Order conformed";
    content = "Your order had been conformed, we will notice you when it is ready for pick up";
}
else if (action == "refuse")
{
    order.STATE = "refused";
    subject = "Order Refused";
    content = "Your order had been refused by the restaurant, please try another one";
}
else if (action == "ready")
{
    order.STATE = "ready";
    subject = "Order Ready";
    content = "Your orderis ready for pick up";
}
else if (action == "done")
{
    order.STATE = "close";
}
db.Entry(order).State = EntityState.Modified;
db.SaveChanges();

if (action != "done") {
    var email = new Email();
    email.Send(id, subject, content);
}
```

Рисунок 3.5 – Лістинг коду керування замовленням

Метод сервісу Order реалізує функцію встановлення статусу виконання замовлення, вносить відповідні зміни в базу даних та викликає сервіс для сповіщення клієнтів.

На рисунку 3.6 наведено клас сервісу Notification, яких вико описує функціонал відправлення повідомлень клієнтам про стан замовлення.

```
public class Email
    {
        private Model db = new PreOrderModel();
        private const String API_KEY = "SG.0MqVjLkKRUuRkv2GxzXMug.YjENL0ZU-wbgh

        public Boolean Send(int orderID, String subject, String contents)
        {
            var userID = db.RESERVE_PICK_UP.Find(orderID).USER_ID;
            var toEmail = db.USERs.Find(userID).EMAIL;
            var client = new SendGridClient(API_KEY);
            var from = new EmailAddress("koval.andriy2015@gmail.com", "Pre Orde
            var to = new EmailAddress(toEmail, "");
            var plainTextContent = contents;
            var htmlContent = "<p>" + contents + "</p>";
            var msg = MailHelper.CreateSingleEmail(from, to, subject, plainText
            var response = client.SendEmailAsync(msg);

            return true;
```

Рисунок 3.6 – Код класу для сповіщення

Мікросервіс Notification використовує стороннє API для реалізації можливості надсилання повідомлень на електронну пошту, а саме API Gmail від Google. Це інтерфейс, який дозволяє надсилати електронні листи із простою конфігурацією і який можна впроваджувати у веб-проекти .NET. Для цього потрібно отримати у компанії Google особистий ключ (API_KEY), яких дозволяє реалізовувати інтерфейс у власних розробках.

Роботу з базою даних було виконано інтерфейсі середовища Microsoft Visual Studio (див. рис. 3.7), яка надає зручний інтерфейс Server Explorer для роботи з базою даних SQL Server.

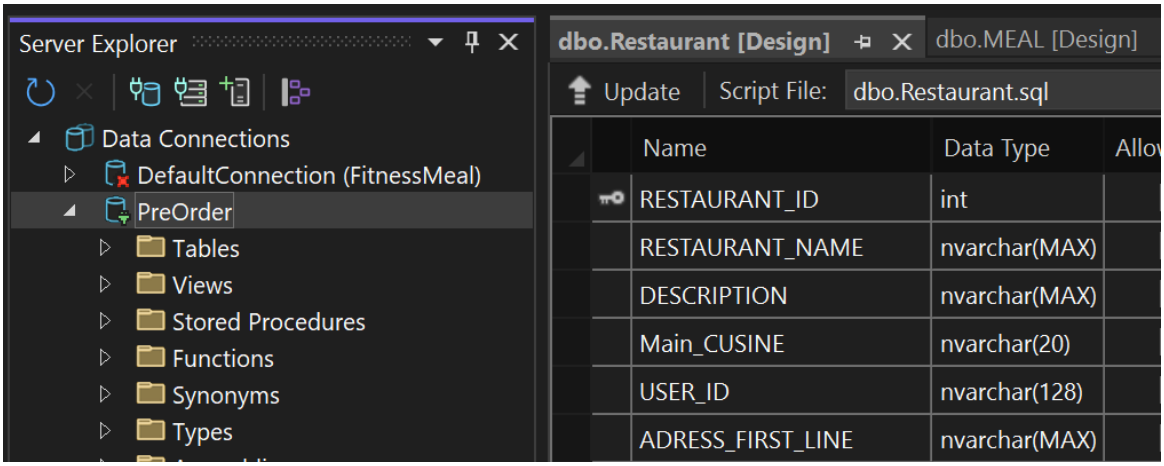

Рисунок 3.7 – Робота за базою даних в MVS

Server Explorer надає можливість створювати з’єднання з базою даних, а також виконувати всі необхідні дії для роботи з нею, які включають створення та редагування таблиць і наповнення їх записами.

3.3 Зв'язок з мікросервісами

В архітектурі мікросервісів мікросервіси не використовують спільну кодову базу та сховища даних. Замість цього вони спілкуються через API для операцій з даними. Оскільки мікросервіси розподілені і взаємодіють один з одним за допомогою міжсервісного зв’язку на рівні мережі, кожен мікросервіс має власний екземпляр і процес, то вони повинні взаємодіяти за допомогою протоколів зв’язку між службами, таких як HTTP , gRPC [12].

Перш ніж розробляти комунікації мікросервісів, потрібно визначити типи комунікації, які можна класифікувати на синхронний і асинхронний. Синхронне спілкування використовує протокол HTTP або gRPC для запиту клієнта. Клієнт – це частина програми, яку використовує користувач, наприклад веб-сайт. Клієнт відправляє запит і чекає відповіді від сервісу. Це означає, що код клієнта блокує

потік, доки відповідь не надійде від сервера. Код клієнта продовжить виконання завдання, коли він отримає відповідь сервера.

В асинхронному зв’язку клієнт надсилає запит, але не чекає відповіді від сервісу. Отже, ключовим моментом тут є те, що клієнт не повинен бути заблокований в очікуванні відповіді, а здійснював інші запити. Така реалізація не підходить для даної розробки, адже тут всі запити відбуваються поступово і вимагають негайної відповіді [13]. Тому реалізовано синхронний підхід.

Веб-сервіси, які використовують архітектуру REST, називаються сервісами RESTful. Системи RESTful зазвичай спілкуються через протокол HTTP за допомогою методів HTTP [14], які використовуються веб-браузерами для передачі сторінок:

- GET – використовується для отримання ресурсу, зазначеного в URI для переліку даних і операцій відображення.
- POST – створення нового ресурсу на сервері. Запит надсилається, щоб запустити ресурси контролера та надіслати вхідні дані форми.
- PUT – має область використання, подібну до методу POST. Ресурс надсилається разом із тілом, і якщо URI вказує на існуючий ресурс, цей ресурс оновлюється. Використовується для оновлення даних.
- DELETE – це запит, надісланий для видалення ресурсу, зазначеного в URI. Він використовується лише для того, щоб видалити відповідний ресурс і більше ніколи не мати доступу до нього.
- PATCH – використовується для оновлення окремої частини даних.

Повідомлення передаються у форматі JSON.

Таким чином і були реалізовані запити до сервісу даної розробки. На рисунку 3.8 наведено приклад запиту до сервісу Restaurant.

```
curl -X POST -H "Content-Type: application/json" \
    -d '{"query": "risotto"}' \
```

Рисунок 3.8 – Запит на пошук ресторанів

При здійсненні запиту на отримання закладів відповідно до ключового слова (див. рис. 3.8) повертається результат, наведений на рисунку 3.9.


```
{
    "_id": "<some id>,
    "nome": "ristorante buono",
    "indirizzo": "via Roma 1",
    "rating": "4",
    "menu": [
        {
            "category": "antipasti",
            "meals": [
                {"item": "prosciutto", "price": "5.5\u20ac"},
                {"item": "suppl\u00ec", "price": "1.5\u20ac"},
            ],
        },
        {
            "category": "Primi",
            "meals": [
                {"item": "pasta al sugo", "price": "7\u20ac"},
                {"item": "risotto", "price": "9\u20ac"},
            ],
        },
        {
            "category": "Secondi",
            "meals": [
                {"item": "carne", "price": "12\u20ac"},
                {"item": "pesce", "price": "15\u20ac"},
            ],
        },
    ],
},
```


Рисунок 3.9 – Результат запиту до сервісу Restaurant

На рисунку 3.10 наведено приклад запиту до сервісу Order з метою створити нове замовлення.

```
curl -X POST -H "Content-Type: application/json" \
    -d '{"restaurantName": "ristorante buono","date": "22/03","service": "lunch",
    "time": "13","notes": "","email": "a@a.t","status": "pending","authToken": "token"}' \
```

Рисунок 3.10 – Запит до сервісу Order

При успішному створені нового замовлення сервіс Order повертає ідентифікатор нового запису з бази даних замовлень, та повідомлення про стан замовлення (див. рис. 3.11).

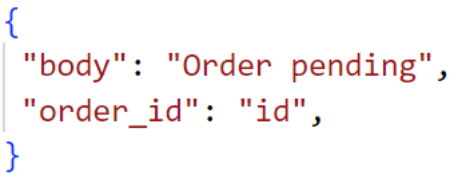

Рисунок 3.11 – Відповідь на створення замовлення

Частими є запити до сервісу Identity з метою перевірити дані користувача при вході чи виконанні користувачем певних дії, які може виконати лише зареєстрований користувач. При підтверджені даних сервісом повертаються електронна пошта користувача, а також токен авторизації.

3.4 Створення GUI

В даній розробці в якості графічного інтерфейсу для користувачів надано веб-сайт. Сторінки створювалися за допомогою представлень ASP.NET, які є файлами файли з розширенням cshtml. Вони описують зовнішній вигляд та структуру сторінок сайту [15]. У Visual Studio наявний зручний інтерфейс для створення сторінок веб-застосунків, який і було використано (див. рис. 3.12).

Створено декілька сторінок сайту розробленої платформи. Оновою кожного сайту є домашня сторінка, також розроблено строфіки які надають інтерфейс клієнтам здійснювати замовлення, а для власників закладів панель керування своїми закладами та замовленнями.

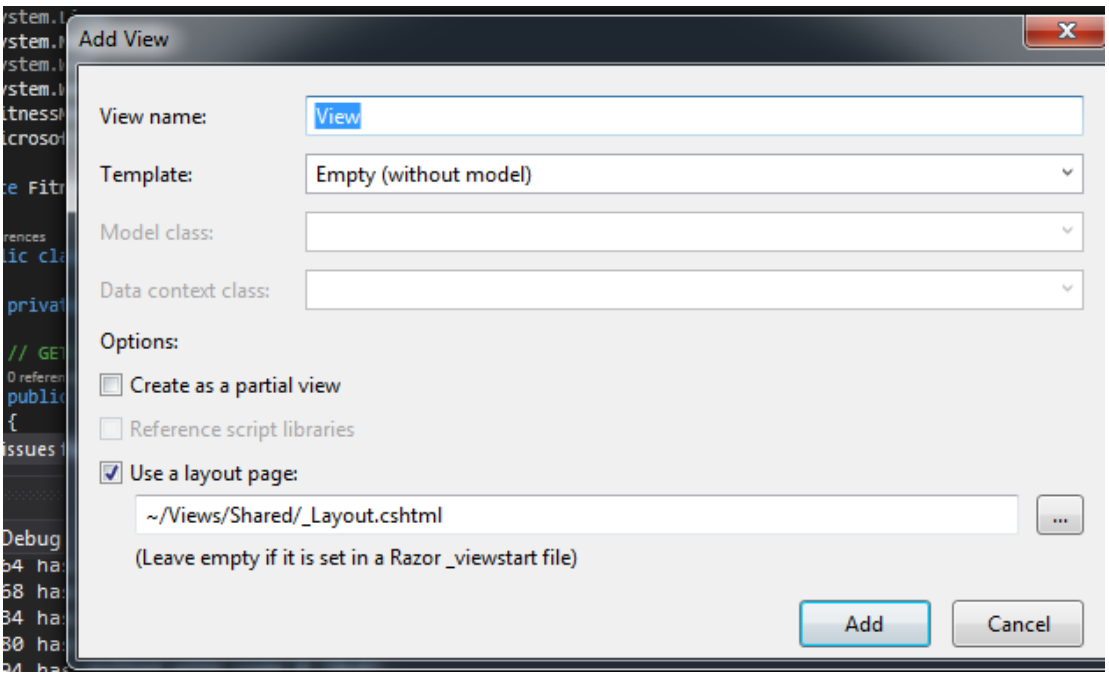

Рисунок 3.12 – Створення сторінок сайту

Для прикладу наведено головну сторінка, яка містить поле пошуку закладів, рекомендації для клієнтів та меню, що надає інтерфейс для дії, в залежності від користувача (див. рис. 3.13).

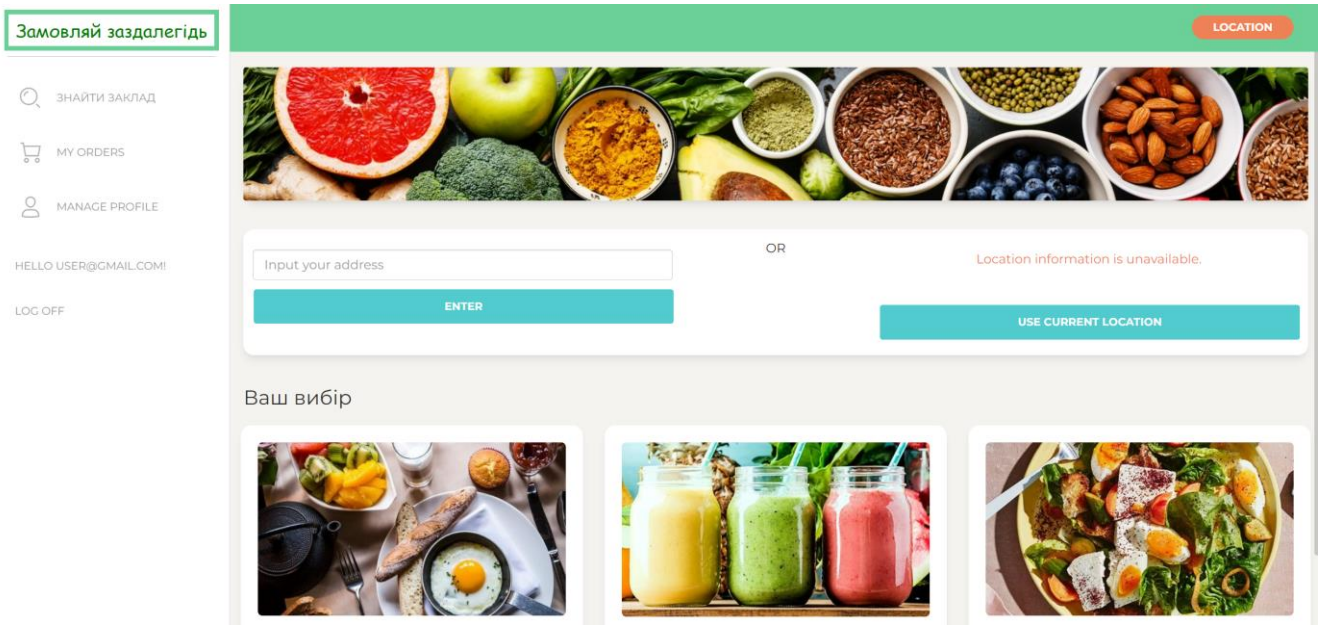

Рисунок 3.13 – Головна сторінка сайту

В розробці веб-сайту було застосовано сторонній інтерфейс Google MAPS API, який реалізує роботу з картами в власних розробках [16]. Це дозволило добавити на сайт карту, за допомогою якої клієнти можуть здійснювати зручний пошук закладів.

3.5 Тестування роботи платформи

При тестуванні справності роботи розробленої платформи виконано всі можливі діє користувачів системи, а також всі можливі випадки контролю даних та продемонстровано їх наглядно.

Тож на рисунку 3.14 наведено вигляд сторінки реєстрації. Власники для публікації свого закладу та отримання замовлень повинні бути зареєстровані.

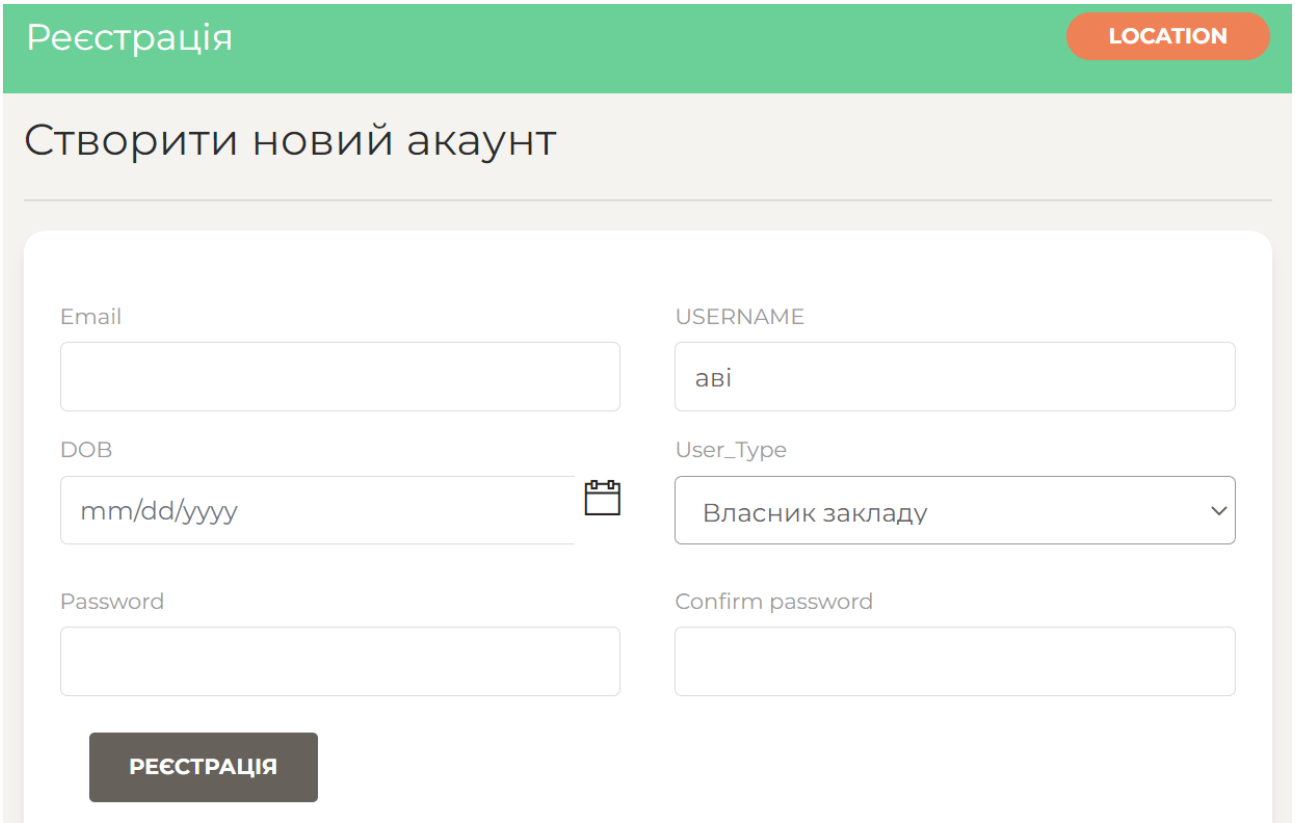

Рисунок 3.14 – Вигляд сторінки реєстрації користувача

Також реєстрація потрібна, щоб користувач мав змогу користуватися усіма функціями, що надає сайт, адже не зареєстрований користувач має можливість лише шукати та переглядати ресторани чи кафе без змоги виконувати замовлення.

Для реєстрації користувачу необхідно ввести деякі свої дані, зокрема електронну пошту, придумати ім’я користувача і пароль, а також вибрати тип користувача.

Якщо користувач звичайний відвідувач і збирається використовувати систему задля пошуку закладів і виконання замовлення то вибирає тип «Клієнт». Власники закладів реєструються, вибираючи відповідно тип «Власник закладу», та після реєстрації отримують функціонал сайту, що дозволяє публікувати заклад та приймати замовлення (див. рис. 3.15).

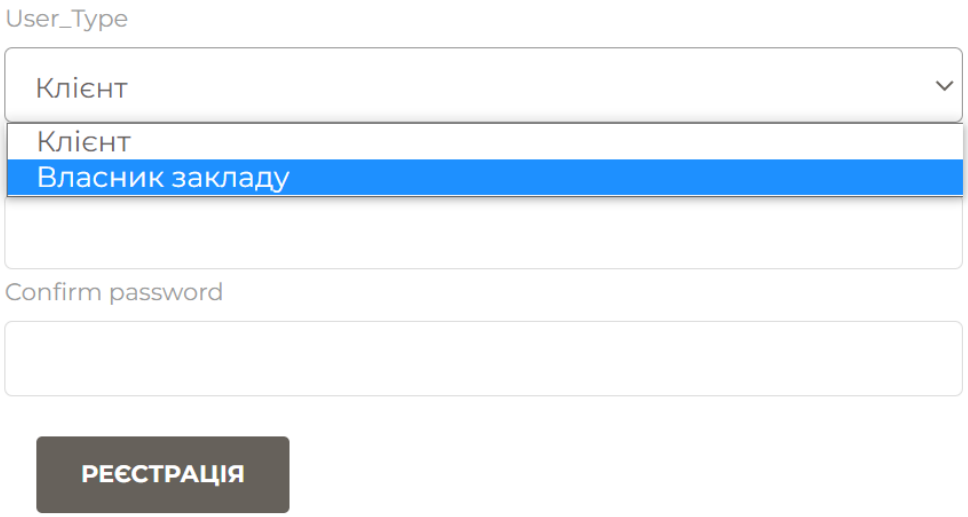

Рисунок 3.15 – Вибір типу користувача

Для того, щоб ввійти користувач використовує електронну пошту та пароль. Це видно на рисунку 3.16, де наведено форму для входу на сайт. Клієнт, увійшовши, отримує доступ до своїх замовлень.

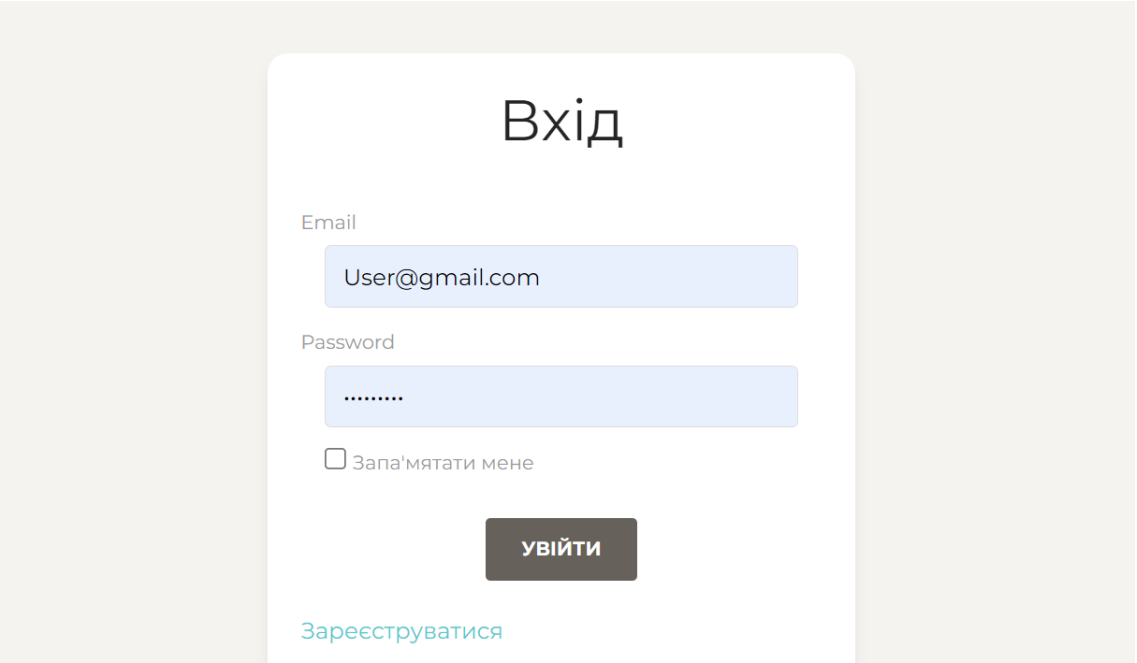

Рисунок 3.16 – Вхід в систему

Залогіненому клієнту закладів доступне меню керування своїм профілем та перегляд своїх замовлень (див. рис. 3.17).

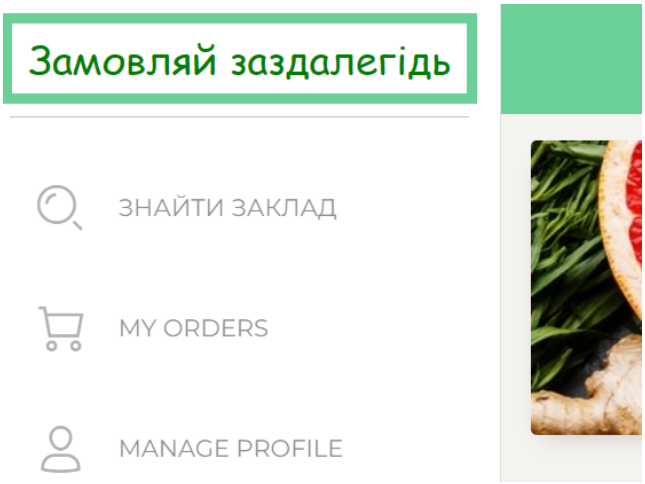

Рисунок 3.17 – Інтерфейс для клієнта

Для того, щоб здійснювати пошук закладів поблизу сайт повинен визначити поточне місцезнаходження користувача. Користувач може сам його вказати, а це передбачає, що можна вказати будь-яке місце, незалежно від того де користувач

знаходиться. Або використає поточне місцезнаходження, яке легко визначить система за допомогою системи навігації Google Map, яка працює на сайті.

Щоб визначитися з місцем знаходження користувач натискає на кнопку «Location», яка міститься на домашній сторінці, і йому надається невеличкий інтерфейс для цього, який наведено на рисунку 3.18.

Ця дія не є обов'язковою, адже за замовчуванням система сама визначає поточне місце знаходження користувача. Користувачу доведеться самому вказувати адресу в тому випадку, якщо я якихось технічних причин сайт не зміг її визначити, або якщо користувач бажає знайти ресторани поблизу іншої адреси, де він в даний момент не знаходиться.

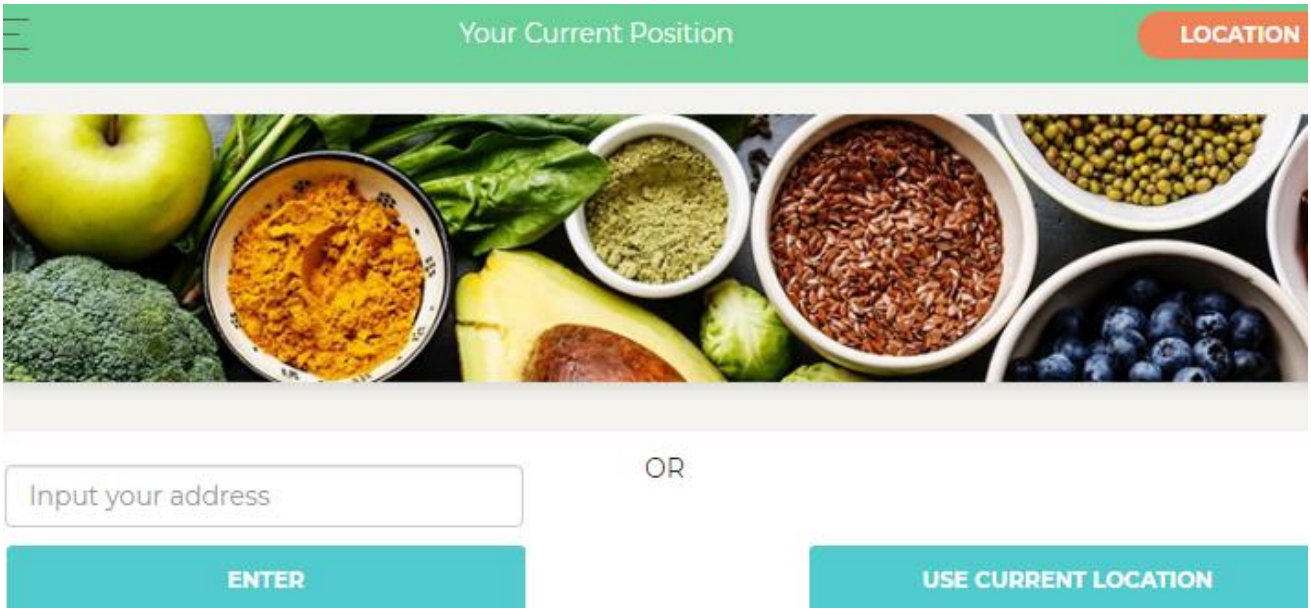

Рисунок 3.18 – Встановлення адреси місця знаходження користувача

Також користувач може вибрати заклад з рекомендацій, які видаються йому на домашній сторінці на основі раніше здійсненних запитів пошуку. Вигляд рекомендації на домашній сторінці наведено на рисунку 3.19.

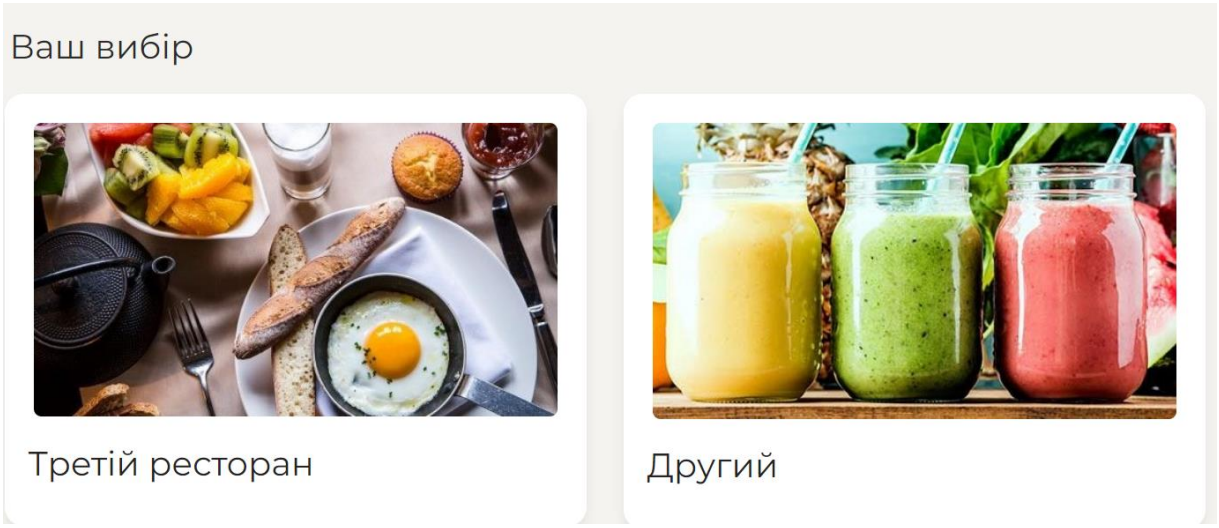

Рисунок 3.19 – Рекомендації на домашній сторінці

Щоб виконати пошук закладів поблизу, що власне є однією з основних цілей користувача, йому потрібно лише натиснути кнопку пошуку на домашній сторінці (див. рис. 3.20).

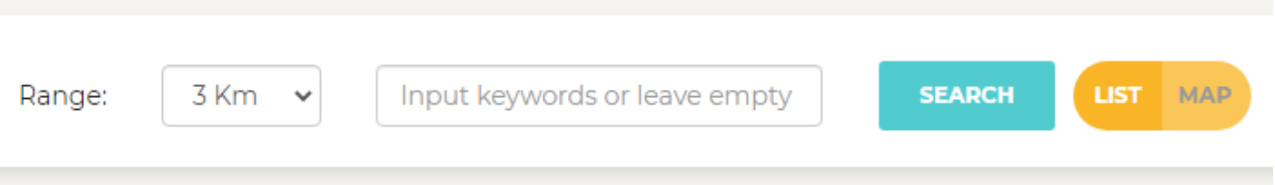

Рисунок 3.20 – Інтерфейс для пошуку закладів

До того ж інтерфейс пошуку закладів містить поле для пошуку, де користувач може ввести якесь ключове слово, щоб звузити пошук, чи отримати більш підходящі варіанти, або вказати конкретний заклад. Користувач може залишити поле для введення інформації пустим, тоді сайт видасть усі зареєстровані в його базі заклади поблизу.

Сайт видає всі заклади в радіусі 1-го, 3-х, 5-х чи 10-х кілометрів, залежно від того який діапазон вибере користувач на інтерфейсі пошуку.

Також у користувача є можливість знайти і вибрати заклад ще одним способом, а саме вибравши заклад на карті. У сайт влаштовано систему Google Map, за допомогою якої можна переглядати місцевість та вибрати бажаний заклад.

Для цього користувачу в інтерфейсі пошуку потрібно вибрати відповідний спосіб натиснувши відповідну кнопку. Після чого відкриється карта Google з поточним місцем знаходженням користувача, або з тим яке він вказав. Користувач може вказати у полі для пошуку заклад і сайт покаже його на карті. Як відбувається пошук закладів за допомогою карти наведено на рисунку 3.21.

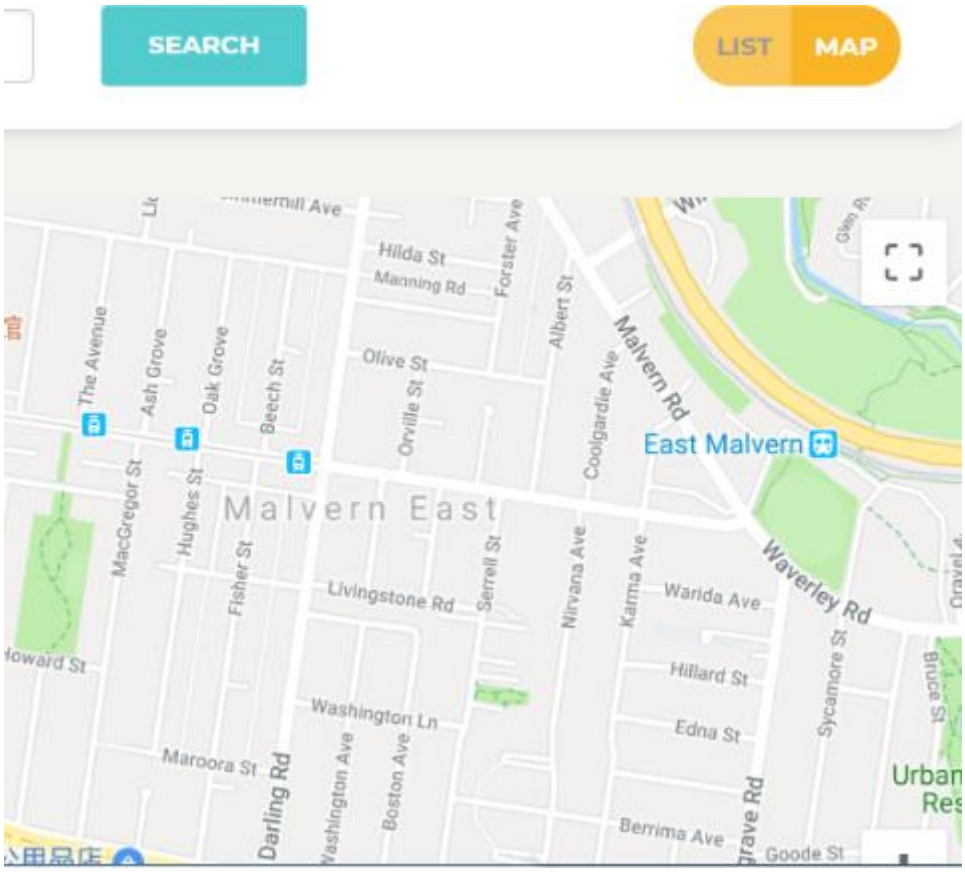

Рисунок 3.21 – Пошук закладів на карті Google

Після того як користувач знайшов та вибрав ресторан, кафе чи інший заклад харчування, зареєстрований на сайті власником чи адміністратором закладу, він може переглянути інформацію про заклад та здійснювати замовлення з меню.

Власники обов’язково розміщують таку інформацію про свій заклад:

- назву та опис місця;
- час роботи;
- меню, та опис страв;
- іншу інформацію, яка характеризує заклад.

Вигляд сторінки для оформлення замовлення наведено на рисунку 3.22. Вона стилізована так само як всі інші сторінки сайту. На сторінці замовлення міститься меню, в якому користувач може вибрати страву для замовлення, та меню оформлення замовлення, в якому міститься вся необхідна інформація про здійснюване замовлення.

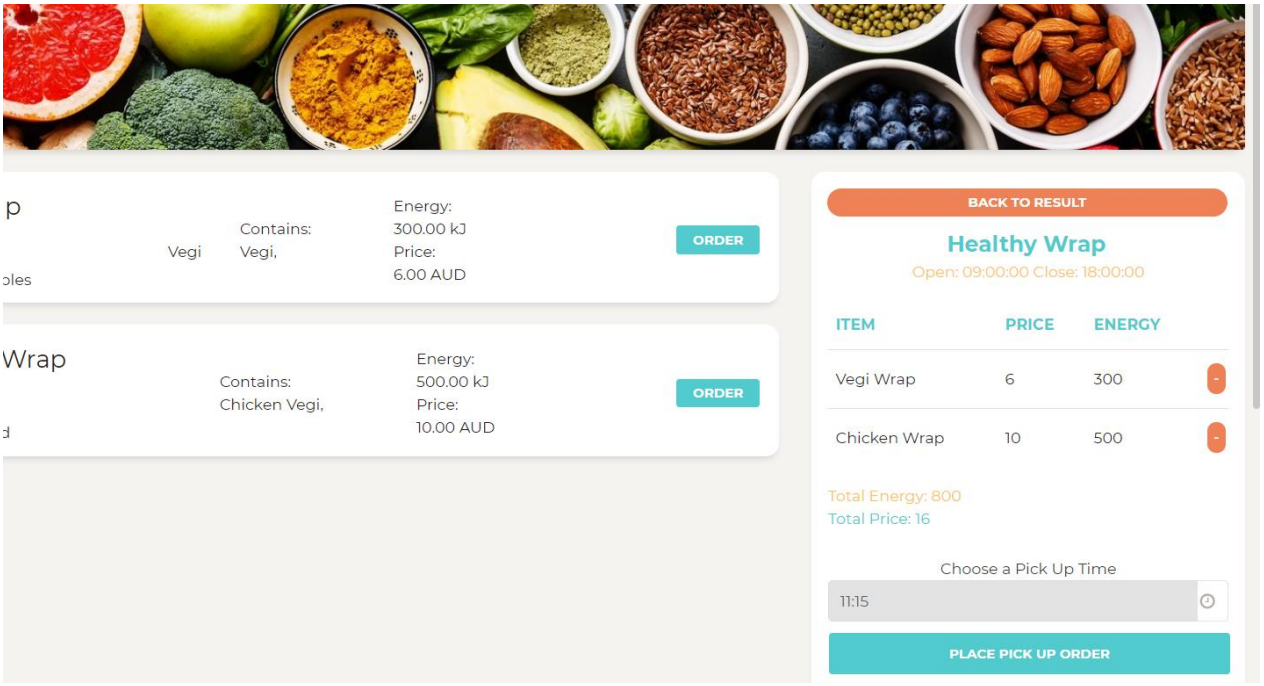

Рисунок 3.22 – Сторінка оформлення замовлення

На рис. 3.23 детальніше наведено меню закладу. Кожна страва меню ресторану може містити додаткову інформацію, яку заклад вважає за потрібне вказати, наприклад, що страва вегетеріанська, або, що страва містить лактозу, адже така інформація для деяких клієнтів є необхідною.

Меню закладу містить такі характеристики: назва страви; короткий опис; поживна цінність; вага порції; ціна страви;. Користувач вибирає з меню страву для замовлення, натиснувши на кнопку «Order» (див. рис. 3.23), після чого страва додаються в список усіх, що бажає замовити клієнт, у меню на тій ж сторінці.

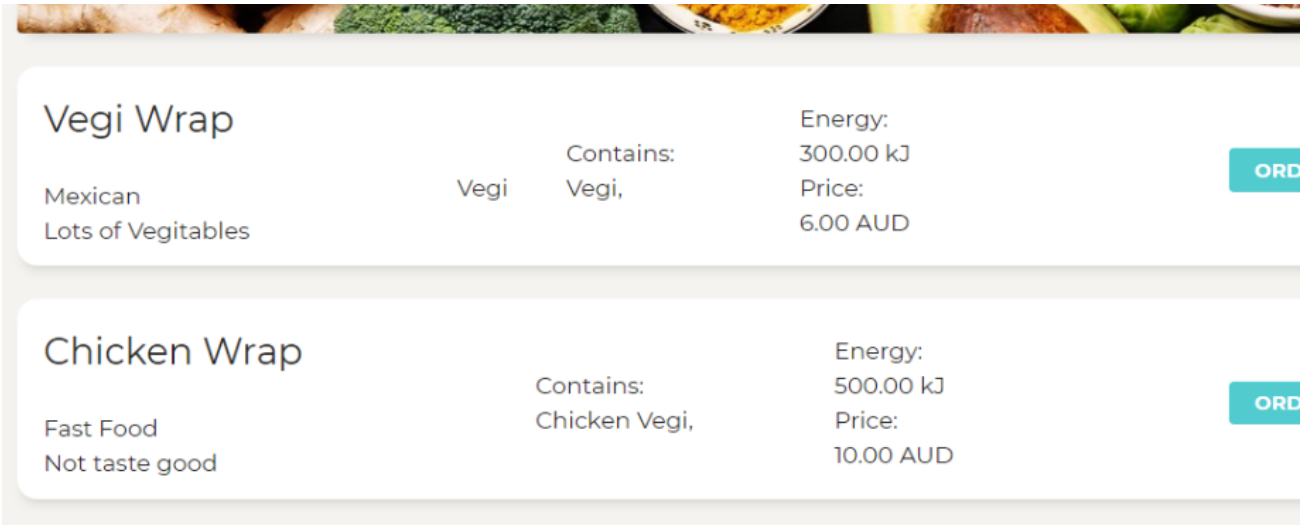

Рисунок 3.23 – Страви у меню закладу

В меню для здійснення оформлення замовлення міститься назва закладу, час його роботи та список вибраних для замовлення користувачам страв з меню. Також в меню підсумовується ціна кожної страви та видається сума всього замовлення.

Клієнту закладу потрібно вибрати час прийому замовлення, тобто час, на який йому підготують все замовлене та коли він з’явиться в ресторані. Після цього користувач підтверджує замовлення, натиснувши на відповідну кнопку та замовлення відправляється на розгляд закладом. Користувачу залишається очікувати на сповіщення про підтвердження прийому замовлення закладом, і якщо заклад береться за його виконання, користувач отримує сповіщення на електронну пошту та приходить в вказаний час.

Також, оформляючи замовлення, користувач може видалити з нього страви, які передумав замовляти, або навпаки ще додати в замовлення будь-яку кількість

страв, повернувшись для цього в меню ресторану. Виглядає меню оформлення замовлення наведено на рисунку 3.24.

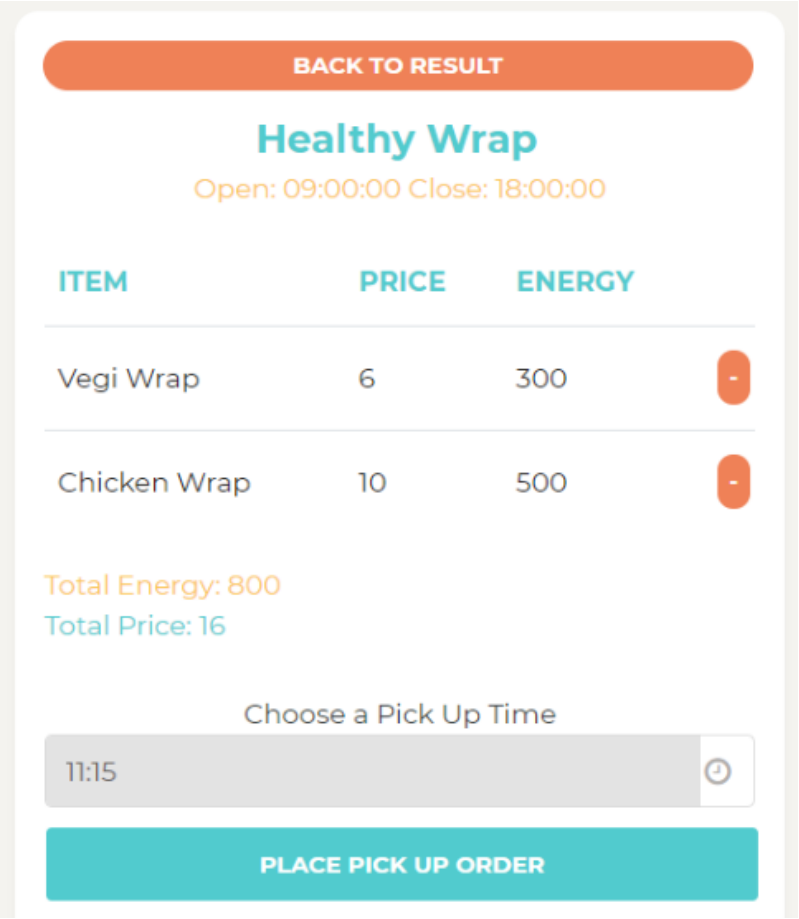

Рисунок 3.24 – Оформлення замовлення

В свою чергу власники приймають замовлення. Для цього адміністратору чи іншій особі яка працює в закладі потрібно виконати вхід на сайт та зайти на необхідну сторінку де публікуються замовлення клієнтів.

Меню, яке доступне власникам, наведено на рис. 3.25, воно містить кнопки для переходу на сторінку керування закладок, на якій можна вносити інформацію про заклад, а також кнопку для переходу на сторінку прийому замовлень від клієнтів.

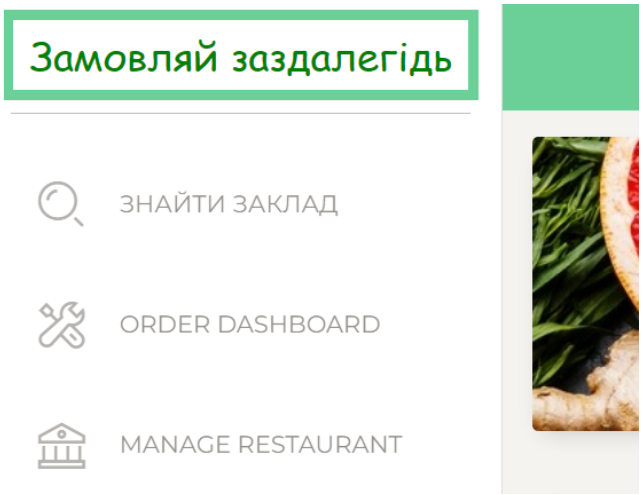

Рисунок 3.25 – Меню сайту для власників закладу

На сторінці для прийняття замовлення міститься спеціальна панель, на якій і відображаються замовлення клієнтів. Власники ресторанів, кафе та інших закладів чи їх адміністратори отримують замовлення та працюють з ними за допомогою інтерфейсу цієї панелі, який наведено на рисунку 3.26.

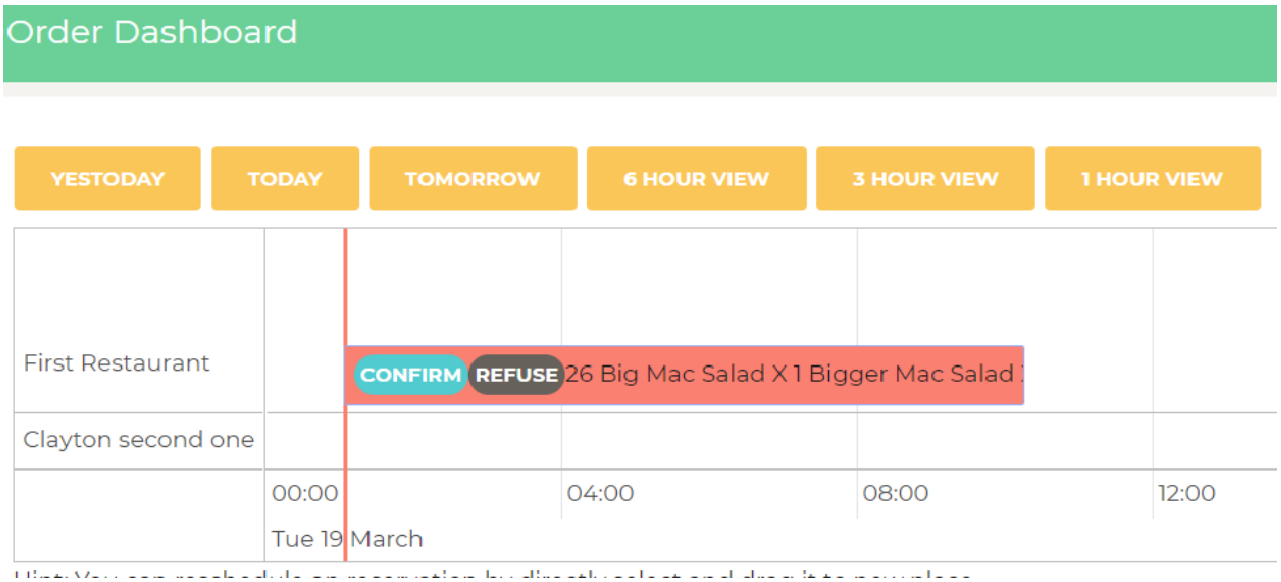

Рисунок 3.26 – Інтерфейс прийому замовлень для власників закладів

Особа із закладу яка приймає на сайті замовлення бачить, що саме замовив клієнт і на який час та вирішує прийняти його чи ні. Після цього клієнт отримує

сповіщення про прийом його замовлення, а заклад приступає до приготування страв та підготовки до обслуговування на вказаний клієнтом час. Ресторан може одночасно приймати декілька замовлень, як це видно на рис 3.27.

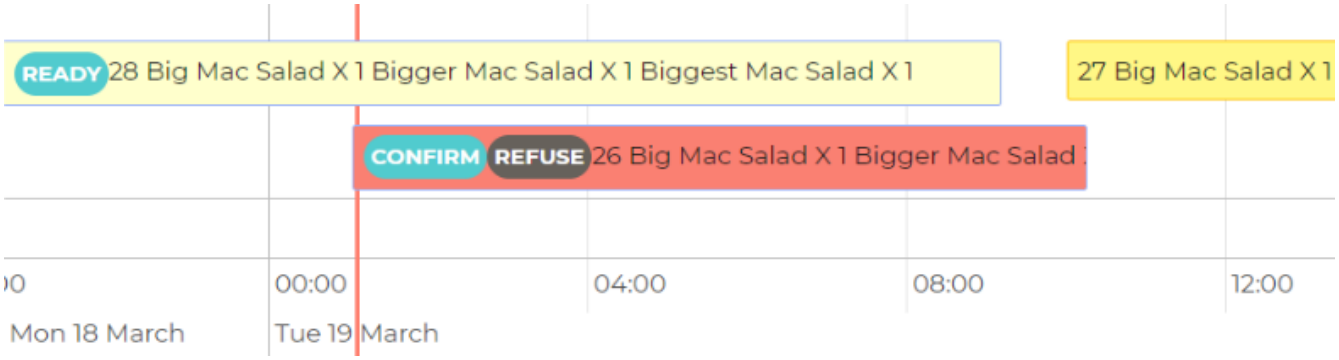

Рисунок 3.27 – Отримання декількох замовлень

На рисунку 3.28 показано приклад неправильної реєстрації та контроль введених даних в такому випадку системою.

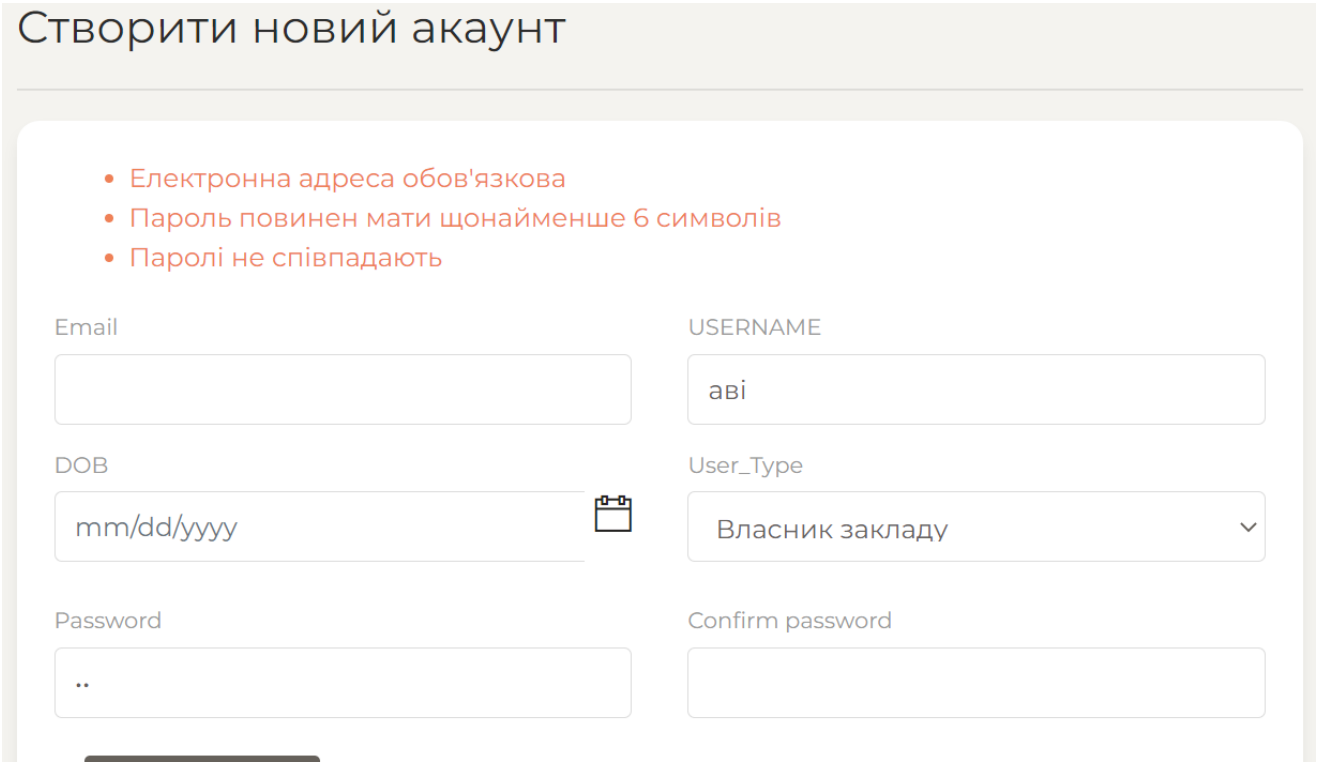

Рисунок 3.28 – Перевірка введених даних

Виконується перевірка на валідність даних (див. рис. 3.28), адже для того що зареєструватися потрібно вводити дані згідно з деякими правилами. Наприклад пароль задля надійності повинен складатися з букв, цифр, символьних знаків, та містити великі літери. Також повинні бути заповненні всі дані, електронна адреса повинна бути коректна, а ім’я користувача унікальне.

Аналогічно як при реєстрації відбувається контроль ведення даних і при неправильній електронній адресі чи паролю на сторінці з’явиться відповідне повідомлення, завдяки прописаним для цього випадку контролерами.

Якщо користувач при оформленні замовлення вказав час його прийняття такий, в який заклад закритий то система видасть відповідне повідомлення, і здійснити замовлення не вийде (див. рис. 2.29).

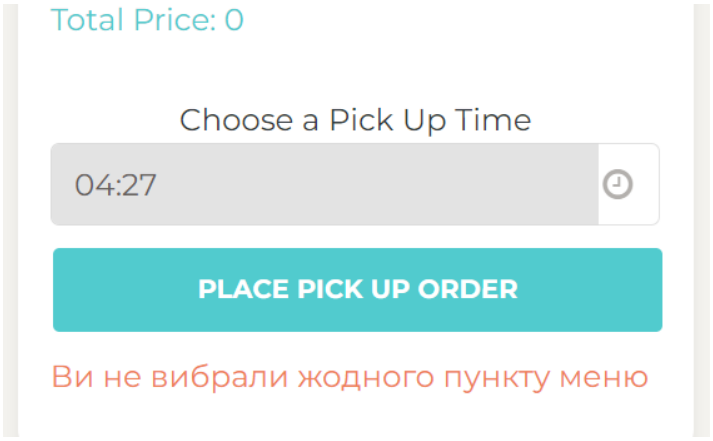

Рисунок 2.29 – Повідомлення про помилку при оформленні замовлення

Сайт було розроблено з елементами адаптивності до вікна браузера. При зміні вікна сайт зберігає свою естетичність та правильну структуру. На сторінках сайту при певному розмірі вікна приховується праве меню, щоб зберегти естетичність та масштаб, але з’являється кнопка для його показу при потребі, що видно на рисунку 2.30.

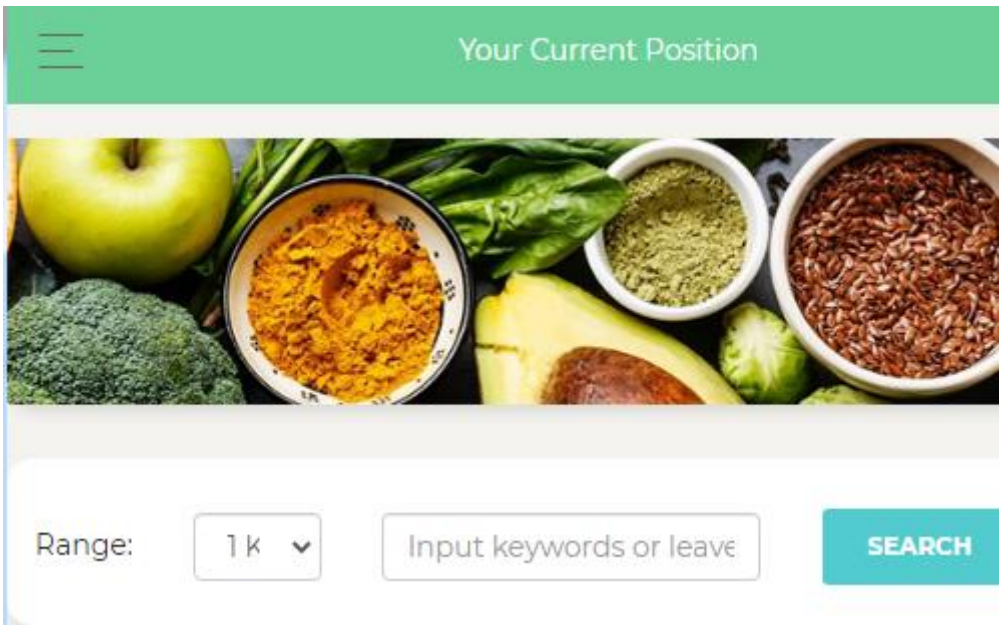

Рисунок 2.30 – Приховання правого меню сайту при зменшені вікна браузера

Праве меню з’являється при натисканні на кнопку, яка після його появи слугуватиме кнопкою для його закриття (див. рис. 2.31).

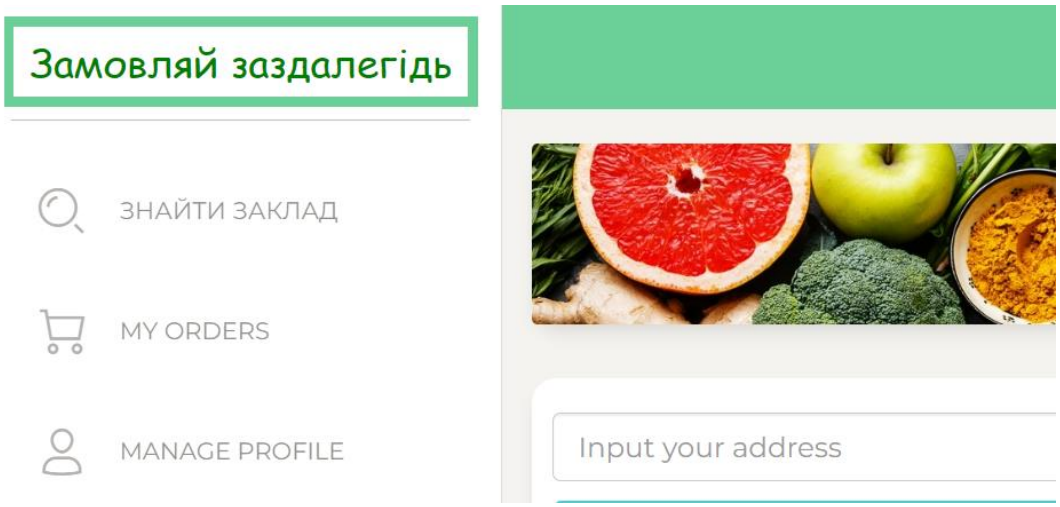

Рисунок 2.31 – Поява меню сайту у зменшенім вікні браузера

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Результатом кваліфікаційної роботи є веб-платформа, користування якою здійснюється за допомогою комп’ютера, та зокрема використовується для роботи в закладах харчування, які є приміщенням підприємницької діяльності. Тому слід зазначити, що при роботі з даною розробкою за комп’ютерною технікою необхідно дотримуватися вимог охорони праці задля запобігання проблем із здоров'ям.

При роботі з комп’ютером значним чином відбувається вплив на нервово-емоційний стан користувачів, така робота характеризується великим навантаженням на м’язи рук при роботі з клавіатурою комп'ютера, високою інтенсивністю зорової роботи та значним розумовим перенапруженням.

Отже, раціональне планування робочого місця повинно забезпечити: зменшення втоми працівників та підвищення продуктивності праці, уникнення загального дискомфорту, якнайкраще розташування інструментів та предметів праці.

Для того, щоб виявити та проаналізувати шкідливі і небезпечні виробничі фактори необхідно почати з аналізу дотримання вимог, встановлених санітарними правилами і нормами [ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»] для виробничих приміщень та робочих місць.

На виробництві для дійсної оцінки умов праці відбувається сертифікація робочих місць відповідно до умов праці та використовується «Гігієнічна класифікація праці» за показниками небезпечності та шкідливості факторів виробничого середовища, напруженості та тяжкості трудового процесу.

Веб-платформа для здійсненні замовлення та керування ними в закладах харчування відноситься до першого класу згаданої сертифікації. Згідно з ним

створюються оптимальні умови праці, для підтримки високого рівня працездатності.

На підставі сертифікації робочого місця необхідно охарактеризувати інтенсивність робіт за такими напрямами:

- відповідність обладнання нормативно-технічним вимогам, документації, а

- відповідність площі та обсягу займаного робочого місця чинним нормам;

- спеціалізоване устаткування робочого місця (засоби захисту пристроїв та їх технічний стан);

- відповідність технологічного процесу, інструментів, устаткування, засобів контролю вимогам стандартів безпеки і нормам охорони праці.

Обладнання та організація робочого місця ВДТ ЕОМ повинні задовольняти відповідність конструкції всіх елементів робочого місця та їх відносного розташування ергономічним вимогам з урахуванням особливостей та характеру трудової діяльності [ДСТУ 7299:2013 «Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки»].

Раціональне планування робочого місця повинно забезпечити: найкраще розміщення інструментів та предметів праці, уникати загального дискомфорту, зменшити втому працівників та підвищити продуктивність праці. Заходи щодо усунення ризику ураження електричним струмом зводяться до правильного розміщення обладнання та електричних кабелів.

Також інші заходи для забезпечення електробезпеки збігаються із загальними заходами пожежної та електробезпеки. В якості заходів профілактики для того, щоб забезпечити пожежну безпеку необхідно у приміщеннях використовувати приховану електромережу, ввімкнення та вимкнення живлення виконувати обладнанням за допомогою стандартних вимикачів, надійні розетки з важкозаймистих матеріалів.

Необхідно регулярно чистити внутрішні частини комп'ютерів та інше обладнання від пилу, комп'ютери розміщувати на окремих вогнестійких столах.

Для запобігання іскроутворення необхідно рідше вставляти і виймати вилки з розеток.

Робочі місця мають бути розміщені на відстані не є менше 1,5 м від стіни з вікнами, та від інших стін на відстані близько 1 м, між бічними поверхнями ВДТ – 1,2 м; від задньої площини одного ВДТ до іншого екрану – 2,5 м. Конструкція робочої поверхні користувача ВДТ повинна забезпечувати підтримку оптимального робочого положення. Сприятливе робоче положення в процесі роботи за комп'ютером забезпечується налаштуванням висоти робочого столу, підставки для ніг і стільця.

Важливою є форма спинки стільця, яка повинна повторювати форму спини працівника. Висота стільця має бути такою, щоб користувач не відчував тиску на стегна чи куприк. Бажано обладнати крісло підлокітниками. Потрібно встановити їх так, щоб не довелося тягнутися до клавіатури. Доцільно розмістити стіл таким чином, щоб на нього природне світло потрапляло з бокової частини, переважно зліва. Пряме світло не має потрапляти в очі. Рекомендовано розміщувати джерела світла по обидва боки екрану, паралельно напрямку погляду.

Для того, щоб уникнути відблисків на екрані, клавіатурі в напрямку очей користувача, від загальних освітлювальних приладів або сонячного світла, потрібно використовувати спеціалізовані фільтри для екранів, захисні навіси, антипроблискові сітки та жалюзі на вікнах.

Враховано, що дизайн графічного інтерфейсу повинен відповідати нормі при якій, екран монітора повинен бути розташований на оптимальній відстані від очей користувача, яка становить від 600 до 700 мм, але не ближче 600 мм. Для дотримання цієї норми враховуючи розмір буквено-цифрових символів та знаків. При обладнанні робочого місця ВДТ лазерним принтером параметри лазерного випромінювання повинні відповідати вимогам [ДСанПІН 3.3.2.007-98].

Дотримання всіх необхідних заходів з охорони праці забезпечує комфортні умови праці та відсутність шкоди для здоров'я, що сприяє підвищенню продуктивності праці, а також меншому виснаженню при роботі за персональним комп'ютером.

4.2 Безпека в надзвичайних ситуаціях

Фактори що впливають на функціональний стан користувачів комп'ютерів.

Трудова діяльність користувачів комп'ютерів (ВДТ) відбувається у певному виробничому середовищі, яке впливає на їх функціональний стан. Психофізіологічні та емоційні перенапруження, втома можуть призвести у комп'ютеризованих системах керування до помилок і як наслідок – до значних економічних втрат. Визначення та вивчення факторів, що впливають на функціональний стан користувачів комп'ютерів дозволить виділити основні причини виникнення станів напруженості, стомлення, стресу і здійснити відповідні профілактичні заходи.

До основних факторів, що впливають на функціональний стан користувачів комп’ютера належать:

1) виробниче середовище – характеризується такими шкідливими факторами:

- фізичні: електромагнітні хвилі різних частотних діапазонів, електростатичні поля, шум, параметри мікроклімату та ряд світлотехнічних показників;
- хімічні: пил, шкідливі хімічні речовини, які виділяються при роботі принтера і копіювальної техніки;
- біологічні: підвищений вміст в повітрі патогенних мікроорганізмів, особливо у приміщенні з великою кількістю працюючих, при недостатній вентиляції, особливо у період епідемій;
- психофізіологічні: напруження зору та уваги, інтелектуальні
- емоційні навантаження, тривалі статичні навантаження і монотонність праці.

2) трудовий процес - характеризується значними статичними фізичними навантаженнями; недостатньою руховою активністю; напруженнями сенсорного апарату, вищих нервових центрів, які забезпечують функції уваги, мислення, регуляції рухів. Окрім того, трудовий процес користувачів комп'ютерів відзначається значними інформаційними навантаженнями;

3) внутрішні засоби діяльності – це професійні риси та виробничий досвід, які обумовлюють надійну та безпомилкову діяльність користувачів комп'ютерів, дозволяють знаходити безпечні методи розв'язання виробничих завдань навіть у нестандартних ситуаціях;

4) зовнішні засоби діяльності - визначаються ергономічними показниками щодо організації робочого місця, форми та параметрів його елементів, просторового розташування основного і допоміжного устаткування, які можуть суттєво 98 знизити фізичні та психофізіологічні навантаження, що діють на користувачів комп'ютерів;

5) соціально-психологічні фактори трудових взаємовідносин. На функціональний стан людини та на її здоров’я під час роботи з ПК впливає комплекс чинників, зокрема, зміст і обсяг інформації, інтенсивність і тривалість роботи за ПК, якість і досконалість використовуваних програмних продуктів, їхні ергономічні, педагогічні, психогігієнічні властивості.

Окрім того, об’єктивними також вважають чинники які випливають з внутрішнього середовища приміщення, які виникають під час роботи комп’ютерів: показники мікроклімату, освітленість, яскравість, контрастність і колір зображення на екрані дисплея, іонізуюче та неіонізуюче опромінення, шум тощо.

У повітрі зовнішнього природного середовища, як і в повітрі приміщень, наявна певна кількість заряджених частинок, що називаються іонами. У приміщеннях, де працюють з комп'ютерами, концентрація легких негативних іонів зменшується (за 5 хвилин у вісім разів, а за 3 години – до нуля). Така зміна складу повітря призводить до несприятливого впливу на здоров'я користувачів ПЕОМ, на їх розумову та фізичну діяльність.

Гранично допустимі норми рівнів іонізації повітря приміщень при роботі з ЕОМ повинні відповідати ДНАОП 0.03-3.06-80.

Засоби забезпечення необхідної концентрації іонізації повітря такі:

- кондиціювання повітря;
- генератори негативних іонів (іонізатори);
- збільшення вологості повітря у приміщеннях.

Накопичення електричного заряду на поверхні обладнання може досягати кількох тисяч вольт (переважно на електронно-променевій трубці відеотерміналу, зокрема, на екрані). При дотику до такого обладнання може статись електричний удар.

Для захисту від статичної електрики необхідно:

- кілька разів протягом робочого дня мити руки і обличчя водою
- після закінчення роботи вимити руки й лице з милом;
- щоденно протирати екран монітора, клавіатуру, пристрій “миша”, а якщо є приекранний фільтр то і його антистатичною серветкою;
- щоденно у приміщенні з ПК проводити вологе прибирання;
- установити нейтралізатори статичної електрики;
- підтримувати у приміщенні відносну вологість повітря за значенням не нижче 45-50 %.

Разом з тим недопустима вологість повітря більше 75%; – виконати у відповідності з ДНАОП 0.00-1.21-98 “Правила безпечної експлуатації електроустановок споживачів” заземлення ВДТ;

– користувачу ПК бажано носити одяг з природних (льняних) волокон.

Відповідно до [ДСТУ 7299:2013 «Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки»] робочі місця з ВДТ повинні:

- розміщуватися в окремих приміщеннях; - площа одного робочого місця - не менше 6,0 м;

- об'єм приміщення – не менше 20,0 $м^3$;

- обладнуватись аптечками першої медичної допомоги;

- розміщуватись на відстані не менше 1 м від стін з вікнами;

- прохід між рядами робочих місць – не менше 1 м;

- бути обладнані системою автоматичної пожежної сигналізації з димовими оповіщувачами та вуглекислотними вогнегасниками з розрахунку два вогнегасника на кожні 20 $м^3$ площі приміщення та з урахуванням гранично допустимих концентрацій вогнегасної речовини;

- не рідше, як раз на квартал очищувати від пилу агрегати та вузли, кабельні канали та підлогу.

Під час роботи з веб-платформою для замовлення в закладах харчування головними факторами що будуть впливати на роботу користувача будуть виробниче середовище та зовнішні засоби діяльності. Тому правильна організація робочого місця водночас може позбавити одразу від двох цих факторів, а значить допоможе уникнути будь яких шкідливих наслідків для користувача.

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи є розроблена платформа, яка надає сервіс для послуг в сфері харчування. Платформа реалізована на онові мікросервісної архітектури, що є сучасним та прогресивним підходом до проектування програмного забезпечення.

Практичне застосування результатів роботи полягає у наданні можливості власникам харчування публікувати заклади, керувати ними, а клієнтам здійснювати зручне замовлення. Це виправляє недолік в системі обслуговування клієнтів, який полягає у довгому очікувані на обслуговування та на приготування замовлення. Для клієнтів сервіс надає можливість планувати свої прийоми їжі протягом дня та заощаджувати час на кожному візиті у заклад, а для власників платформа прискорить потік клієнтів та допоможе швидко виконувати замовлення.

Також розробка є цінним прикладом реалізації мікросервісної архітектури. В роботі наведено основні принципи такого підходу до проектування та детально сплановано власну систему мікросервісів. Показано механізм взаємодіє незалежних сервісів в одній платформі для реалізації певних вимог.

Мікросервісна архітектура розробленої платформи робить її придатною для майбутній змін, впровадження нових послуг та оптимізації вже наявних, тому виявилася хорошим рішенням для реалізації поставлених завдань.

Поставлене завдання та мету роботи виконано та виправдано доцільність їх постановки, адже результати роботи можна підкріпити практичним застосуванням, актуальністю в даний час та необхідність для користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Практика попереднього замовлення. – [Електронний ресурс] : https://uk.wikipedia .org/wiki/Попереднє_замовлення
2. Мікросервісна архітектура. – [Електронний ресурс] : https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices
3. Впровадження мікросервісів. – [Електронний ресурс] : https://microservices.io/patterns/microservices.html
4. Переваги мікросервісів. – [Електронний ресурс] : https://martinfowler.com/articles/microservices.html
5. Шлюз API. – [Електронний ресурс] : https://www.redhat.com/en/topics/api/what-does-an-api-gateway-do
6. Токен авторизації. – [Електронний ресурс] : https://www.okta.com/identity-101/what-is-token-based-authentication
7. Офіційна документація Docker. – [Електронний ресурс] : https://docs.docker.com/get-started/overview
8. Контейнер Docker. – [Електронний ресурс] : https://www.docker.com/resources/what-container
9. REST API. – [Електронний ресурс] : https://www.techtarget.com/searchapparchitecture/definition/RESTful-API
10. Офіційна документація GRPS. – [Електронний ресурс] : https://grpc.io/docs/what-is-grpc/introduction
11. Робота Dockerfile. – [Електронний ресурс] : https://docs.docker.com/engine/reference/builder
12. Взаємодія мікросервісів. – [Електронний ресурс] : https://blog.logrocket.com/methods-for-microservice-communication
13. Синхронна взаємодія. – [Електронний ресурс] : https://www.ringcentral.com/gb/en/blog/definitions/synchronous-communication

14. Методи HTTP. – [Електронний ресурс] : https://www.tutorialspoint.com/http/http_methods.htm
15. Створення представлення у Visual Studio. – [Електронний ресурс] : https://learn.microsoft.com/en-us/aspnet/core/mvc/views/overview?view
16. Документація Google MAPS API. – [Електронний ресурс] : https://developers.google.com/maps

ДОДАТКИ

ДОДАТОК А

Технічне завдання

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТІМЕНІ ІВАНА ПУЛЮЯ

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

ТЕХНІЧНЕ ЗАВДАННЯ

на кваліфікаційну роботу

«Розробка платформи для сервісу в сфері харчування на основі мікросервісної архітектури»

Розробники:

виконавець ст. гр. СПм-61

Коваль Андрій Зеновійович

_____

(підпис)

керівник роботи:

докт. фіз.-мат. наук, проф.

Петрик Михайло Романович

_____

(підпис)

Тернопіль 2022

ЗМІСТ

1. ПІДСТАВИ ДО РОЗРОБКИ

Розробка проводиться у відповідності до графіку навчального плану підготовки бакалаврів за спеціальністю 121 «Інженерія програмного забезпечення».

Тема кваліфікаційної роботи: Розробка платформи для сервісу в сфері харчування на основі мікросервісної архітектури».

Термін виконання: до «___»_____2021р.

2. ПРИЗНАЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Програмний продукт призначений для надання послуг в сервісі сфери харчування.

Програма буде корисною в сфері харчування.

Програма дозволить прогнозувати здійснювати замовлення в закладах харчування та керувати ними із закладу.

3. ВИМОГИ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Функціональні вимоги

Власники можуть публікувати свої заклади харчування в Інтернеті.

Клієнти можуть робити замовлення заздалегідь для його отримання, при прибутті у заклад, без подальшого очікування.

3.2 Технічні вимоги

Вимоги до клієнтської частини: підтримка в ОС мови програмування C++, та інтуїтивно зрозумілий, неперевантажений інтерфейс.

Додаткові вимоги: не менше 8ГБ ОЗП, не менше 4ГБ місця на жорсткому диску.

3.3 Програмні вимоги

Розробка серверної та клієнтської частини:

- C#/.NET
- Microsoft Visual Studio
- Docker

4. ЕТАПИ РОЗРОБКИ

Розробка інформаційної системи проводиться в наступному порядку:

1. Аналіз предметної області, аналіз конкурентів та основних алгоритмів програмної системи;
2. Вибір засобів розробки.
3. Розробка математичної моделі та складових програмного комплексу оформлення супровідної документації.
4. здача проекту.
5. Результати виконання кожного етапу проекту погоджуються з керівником проекту.

5. СУПРОВІДНА ДОКУМЕНТАЦІЯ

Для інформаційної системи повинні бути розроблені наступні документи: завдання

- пояснювальна записка до кваліфікаційної роботи;
- презентація проекту;
- рецензія на проект;
- диск з проектом.

Пояснювальна записка до проекту оформляється згідно діючих вимог донормоконтролю проектів.

6. ПОРЯДОК ЗДАЧІ ПРОЕКТУ

Розроблена інформаційна системи повинна відповідати вимогами, щоскладаються з перерахованих у п.3.1 цього документу характеристик.

Для здачі проекту необхідно підготувати весь перелік документів зазначенийу п.5 цього документу.

Приймання проекту проводиться спеціально створеною комісією в термін зазначені в п.1 цього документу.

7. ВІДМІТКИ ПРО ВИКОНАННЯ ЕТАПІВ ТА ЗМІНИ В ПРОЕКТІ

Назва етапу	Відмітка*
Аналіз предметної області	
Архітектура системи	
Проектування системи	
Використання системи	
Супровідна документація	

* відмітки про виконання етапу ставляться керівником проекту

ДОДАТОК Б

Апробація результатів роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

X НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

7–8 грудня 2022 року

ТЕРНОПІЛЬ

2022

УДК 004.41
А. Коваль, М. Петрик
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

МІКРОСЕРВІСНА АРХІТЕКТУРА В РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

UDC 004.41
A. Koval, M. Petryk

MICROSERVICE ARCHITECTURE IN SOFTWARE DEVELOPMENT

Мікросервісна архітектура є відносно новим рішенням для побудови програмних продуктів. Вона складається з набору невеликих автономних сервісів, кожен з яких може реалізовувати окрему бізнес модель. Мікросервіси невеликі, незалежні та слабко пов'язані. Кожна служба є окремою кодовою базою, якою може керувати невелика команда розробників. Сервіси відповідають за збереження власних даних або зовнішнього стану. Це відрізняється від традиційної моделі, де окремий рівень даних обробляє збереження даних.

Сервіси взаємодіють одне з одним за допомогою чітко визначених API. Деталі внутрішньої реалізації кожної служби приховані від інших сервісів. Також вони підтримують поліглотне програмування і не потребують спільного використання одного стеку технологій, бібліотек або фреймворків.

Таке архітектурне рішення має очевидні переваги. Так, оскільки мікросервіси розгортаються незалежно, керувати виправленнями помилок і випусками функцій легше. Розробники можуть вибрати технологію, яка найкраще підходить для їхніх послуг, використовуючи відповідну комбінацію стеків технологій. Якщо сервіс стає недоступним, це не призведе до переривання роботи всієї програми. Сервіси можна масштабувати незалежно, дозволяючи вам масштабувати підсистеми, які вимагають більше ресурсів, без масштабування всієї програми. Також вони ізолюють дані.

Також мікросервісна архітектура має ряд недоліків, які є важливими факторами, для вибору її як основу для проектів. Кожний мікросервіс простий, але вся система в цілому складніша. Написання невеликого сервісу, який покладається на інші залежні служби, вимагає іншого підходу, ніж написання традиційної монолітної або багаторівневої програми. Також складно тестувати залежності сервіси, особливо коли проектування у вас можуть виникнути проблеми із зворотною чи прямою сумі програма швидко розвивається. Може відбутися перевантаження мережі та затримка, через використання великої кількості сервісів. Кожний мікросервіс відповідальний за власне збереження даних. У результаті узгодженість даних може бути проблемою, як і узгодженість версій проекту.

Література

1. Мікросервісна архітектура. URL: learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices.

ДОДАТОК В

Диск із кваліфікаційною роботою