Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

РОЗРОБКА WEB-РЕСУРСУ «РЕСТОРАН»

Виконала: студентка 4 курсу, групи 2КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Діана ДЕМЮК
(ім'я ПРІЗВИЩЕ)
Керівник: к.т.н., доцент каф. КН
Руслан БЕЛЗЕЦЬКИЙ
(ім'я ПРІЗВИЩЕ)
«04» червня 2025 р.

Рецензент: к.т.н., доцент каф. САІТ
Ілона ВАРЧУК
(ім'я ПРІЗВИЩЕ)
«05» червня 2025 р.

Допущено до захисту
Зав. кафедри Яровий А. А.
«06» червня 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«05» травня 2025 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТА
Демюк Діани Ярославівни

1. Тема роботи: Розробка WEB-ресурсу «Ресторан».
керівник роботи: Белзецький Руслан Станіславович, к.т.н., доц.
затверджені наказом вищого навчального закладу «20» березня 2025 року
№ 97

2. Термін подання студентом роботи 30 травня 2025 р.

3. Вихідні дані до роботи:
Мова програмування – об'єктно-орієнтована;
Кількість підтримуваних платформ не менше 2;
Інтерфейс користувача має бути інтуїтивно зрозумілий.

4. Зміст текстової частини (перелік питань, які потрібно розробити):
вступ; обґрунтування доцільності розробки WEB-ресурсу «Ресторан»; проектування WEB-ресурсу «Ресторан»; програмна реалізація WEB-ресурсу «Ресторан»; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): схема алгоритму роботи модуля бронювання; схема алгоритму загальної роботи WEB-ресурсу «Ресторан»; UML-діаграма класів WEB-ресурсу «Ресторан»; UML-діаграма прецидентів WEB-ресурсу «Ресторан»; ER-модель для бази даних WEB-ресурсу «Ресторан»; приклад роботи програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийня
1–3	Белзецький Р. С., к.т.н., доц. каф. КН		

7. Дата видачі завдання 05 травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітк
		початок	закінчення	
1	Вступ. Обґрунтування доцільності розробки WEB-ресурсу «Ресторан»	05.05.25	07.05.25	
2	Проектування WEB-ресурсу «Ресторан»	08.05.25	11.05.25	
3	Програмна реалізація WEB-ресурсу «Ресторан»	12.05.25	15.05.25	
4	Висновки та аналіз отриманих результатів	16.05.25	26.05.25	
5	Оформлення додатків	27.05.25	29.05.25	
6	Розробка інструкції користувача	30.05.25	01.06.25	
7	Оформлення матеріалів до захисту БКР	02.06.25	04.06.25	

Студент _____ Діана ДЕМЮК
(підпис) (ім'я ПРІЗВИЩЕ)

Керівник роботи _____ Руслан БЕЛЗЕЦЬКИ
(підпис) (ім'я ПРІЗВИЩЕ)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 94 сторінок формату А4, які включають 29 рисунків і 4 таблиці. У списку використаних джерел зазначено 41 найменування.

Метою роботи є створення WEB-ресурсу для оптимізації процесу обслуговування клієнтів та покращення взаємодії із користувачами.

У роботі було проведено аналіз предметної області ресторанної галузі, розглянуто сучасні дослідження в цій сфері та програмні аналоги. Обґрунтовано необхідність створення WEB-платформи «Ресторан». Створено алгоритм функціонування системи і UML-діаграми, що відображають основні процеси.

Розробка програмного продукту проводилася в середовищі Visual Studio Code з використанням мови програмування JavaScript. Були реалізовані основні функціональні модулі WEB-ресурсу та проведено його тестування. Результати випробувань підтвердили успішну реалізацію поставлених завдань і правильну роботу системи.

Ключові слова: WEB-ресурс, ресторан, UML-діаграма, JavaScript.

ABSTRACT

The bachelor's thesis consists of 94 pages of A4 format, which include 29 illustrations and 4 tables. The list of references includes 41 titles.

The purpose of the work is to create a web resource for optimizing the customer service process and improving interaction with users.

Based on the analysis of subject area of the restaurant industry was conducted, modern research in this field and software analogues were considered. The necessity of creating a web platform "Restaurant" was justified. An algorithm for the system's operation and UML diagrams reflecting the main processes were created.

The development of the software product was carried out in the Visual Studio Code environment using the JavaScript programming language. The main functional modules of the web resource were implemented and its testing was conducted. The test results confirmed the successful implementation of the tasks set and the correct operation of the system.

Keywords: web resource, restaurant, UML diagram, JavaScript.

ЗМІСТ

ВСТУП 3
1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ WEB-РЕСУРСУ «РЕСТОРАН» 6
1.1 Аналіз галузі 6
1.2 Огляд сайтів-аналогів 8
1.3 Формулювання вимог та постановка задачі 11
1.4 Висновок 13
2 ПРОЕКТУВАННЯ WEB-РЕСУРСУ «РЕСТОРАН» 14
2.1 Вибір архітектури для розробки WEB-ресурсу «Ресторан» 14
2.2 Проектування WEB-ресурсу «Ресторан» із застосуванням UML-діаграм 17
2.3 Розробка схеми алгоритму роботи WEB-ресурсу «Ресторан» 23
2.4 Розробка ER-моделі бази даних 27
2.5 Висновок 28
3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-РЕСУРСУ «РЕСТОРАН» 30
3.1 Обґрунтування вибору мови програмування та бібліотек 30
3.2 Обґрунтування вибору мови програмування для управління реляційною базою даних 34
3.3 Обґрунтування вибору середовища розробки 35
3.4 Розробка клієнтської та серверної частин 38
3.5 Тестування роботи програми та аналіз результатів 44
3.6 Висновок 56
ВИСНОВКИ 57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 59
ДОДАТКИ 63
ДОДАТОК А (обов’язковий) Протокол перевірки кваліфікаційної роботи 64
ДОДАТОК Б (обов’язковий) Лістинг програми 65
ДОДАТОК В (обов’язковий) Графічна частина 73
ДОДАТОК Г (довідниковий) Інструкція користувача 91

ВСТУП

Актуальність досліджень.

З появою Інтернету та розвитком інформаційних технологій суттєво змінилися всі сфери людської діяльності, включаючи ресторанну галузь. Коли Інтернет став доступним на широкий загал та з'явились перші WEB-сайти, ресторанний бізнес почав поступово впроваджувати цифрові технології у свою діяльність.

Поява смартфонів та WEB-додатків істотно пришвидшила зміни, але пандемія Covid-19 ще дужче вплинула на цей процес, змусивши заклади харчування перейти у цифровий формат через карантинні обмеження. У людей закріпилася звичка замовляти їжу онлайн, підвищилися вимоги до гігієни та безконтактного обслуговування, значно зросла популярність доставки, що перетворило онлайн-присутність з бажаної опції на життєву необхідність для виживання ресторанного бізнесу. Потреба в таких змінах досі залишається актуальною та росте з кожним днем.

Споживачі все частіше шукають інформацію про ресторани в інтернеті перед відвідуванням, віддають перевагу ресторанам, які пропонують зручні сервіси онлайн-бронювання столиків, перегляд електронного меню, можливість замовлення страв на виніс або з доставкою, а також залишення відгуків безпосередньо через WEB-інтерфейс. Такі сервіси не лише підвищують рівень обслуговування чи спрощують надання послуг, а й значно покращують комунікацію між закладом та клієнтом, підвищуючи лояльність та задоволення від взаємодії.

Відбувається масштабна технологічна трансформація всієї галузі, коли великі ресторанні мережі змушені впроваджувати цифрові рішення заради збереження та успішного зростання бізнесу в конкурентних реаліях.

Економічні виклики також диктують необхідність цифровізації, оскільки зростають операційні витрати на персонал та оренду приміщень, що змушує

власників шукати способи оптимізації процесів для підвищення прибутковості та розвивати нові джерела доходу через диверсифікацію сервісів.

Соціальні фактори також підтверджують актуальність WEB-ресурсів, адже онлайн-репутація та відгуки клієнтів стають критично важливими для успіху ресторану, соціальні мережі перетворюються на основний канал просування закладів харчування, а персоналізований підхід до кожного клієнта стає обов'язковою умовою конкурентоспроможності.

Хоча великі ресторанні мережі активно використовують цифрові технології, малі та середні ресторанні бізнеси стикаються з серйозними проблемами. Головні труднощі – це відсутність простих та доступних WEB-сервісів. Існуючі рішення часто занадто складні, дорогі або потребують спеціальних технічних знань, а ресторанам потрібні прості, зрозумілі та недорогі рішення, які можна швидко реалізувати.

Таким чином, виникає реальна потреба в розробці доступних, функціональних та простих у використанні WEB-сервісів, які б дозволили навіть невеликим закладам громадського харчування стати більш конкурентноспроможними, покращити якість обслуговування, залучити нових клієнтів, зменшити навантаження на персонал, оптимізувати бізнес-процеси та підвищити рівень цифровізації. Впровадження подібного рішення дозволить ефективно поєднати традиційний підхід до ресторанного обслуговування з новітніми цифровими технологіями, що в результаті призведе до підвищення ефективності бізнесу, задоволення клієнтів і сталого розвитку закладів громадського харчування.

Отже, дослідження та розробка сучасного WEB-ресурсу для ресторану є актуальним і доцільним напрямом, який відповідає вимогам сьогодення.

Метою дослідження є розширення функціональних можливостей WEB-ресурсу у сфері ресторанного бізнесу.

Об’єктом дослідження є процес розробки WEB-ресурсу в сфері ресторанного бізнесу.

Предметом дослідження є програмні засоби та технології для реалізації WEB-ресурсу «Ресторан».

Задачі дослідження:

- обґрунтувати доцільність розробки WEB-ресурсу «Ресторан», враховуючи сучасні тенденції розвитку ресторанного бізнесу;
- обрати архітектуру для проектування WEB-ресурсу «Ресторан»;
- проаналізувати та обґрунтувати вибір інструментів для розробки;
- виконати програмну реалізацію клієнтської та серверної частин WEB-ресурсу «Ресторан»;
- виконати тестування WEB-ресурсу «Ресторан» та провести аналіз отриманих результатів.

Апробація роботи.

Результати досліджень було апробовано на Міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» м. Вінниці у 2025 р. [1].

Публікації: за основними результатами досліджень опубліковано тези доповіді на науково-технічній конференції [1].

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ WEB-РЕСУРСУ «РЕСТОРАН»

1.1 Аналіз галузі

Ресторанний бізнес – це сектор світової економіки, що постійно прогресує. Від закладів швидкого харчування до вишуканих ресторанів – клієнтам пропонується широкий вибір страв та напоїв. З розвитком ресторанної індустрії стає критично важливим розуміти тренди та виклики, що впливають на цю сферу.

Ресторанна індустрія робить значний внесок у світову економіку. Відповідно до даних Національної ресторанної асоціації, лише ресторанний бізнес Сполучених Штатів за 2019 рік згенерував 863 мільярди доларів продажів та забезпечив роботою 15,3 мільйона людей, а на момент 2025 року обсяги продажів прогнозовано зростуть до 1,5 трильйонів доларів. Крім того, ресторанна індустрія є ключовим роботодавцем у багатьох країнах світу [2].

Разом з цим, ресторанна галузь стикається з низкою викликів, в тому числі зі зростанням конкуренції і, як наслідок, боротьбою за увагу клієнтів. Така тенденція змушує ресторани шукати нові способи виділитися та привернути увагу споживачів, що, таким чином, спонукає індустрію ресторанного бізнесу постійно змінюватись та адаптуватись під сучасні тренди.

Впровадження цифрових технологій в процес обслуговування задовольняє клієнтів в естетичному, організаційному, технічному та інших аспектах. Для власників ресторанів WEB-сайт допомагає заощаджувати гроші. Коли клієнти замовляють через інтернет, не потрібно наймати багато працівників для прийняття замовлень по телефону. Створення WEB-сайту для ресторану дає багато переваг як власникам закладів, так і їхнім клієнтам. Це змінює спосіб роботи ресторанів та робить життя відвідувачів більш зручним [3].

Запровадження WEB-сервісів у ресторанній індустрії забезпечить рівні можливості для малого та середнього бізнесу та задовольнить потреби сучасних

споживачів у зручних цифрових послугах. Цифрова трансформація та впровадження WEB-ресурсу в ресторанну галузь допоможе бізнесу змінити бізнес-стратегії, маркетинговий підхід та цілі шляхом прийняття цифрових технологій, що прискорить продаж і допоможе йти з новітніми технологія нога в ногу [4].

Наявність таких сервісів не лише задовільнить очікування споживача, а й стане вагомим фактором у формуванні конкурентної переваги на ринку. З огляду на високу конкуренцію в галузі, саме ті заклади, які активно використовують цифрові технології, мають кращі шанси на успіх, сталий розвиток, збереження і збільшення постійних клієнтів.

WEB-ресурси представляють собою цифрові засоби та інструменти, що функціонують у мережі інтернет, зокрема інтернет сайти, інформаційні бази даних та онлайн платформи, які забезпечують користувачів інформацією, різноманітними послугами або інтерактивними можливостями. Ефективність та значущість таких ресурсів безпосередньо залежить від того, наскільки добре вони відповідають потребам користувачів, дотримуються організаційних цілей та можливості багаторазового використання.

WEB-ресурси у сучасному світі посідають важливе місце, що підтверджується численними статистичними даними та дослідженнями. Станом на 2025 рік, понад 5,5 мільярда людей у світі користуються Інтернетом, що становить близько 66% від загальної кількості населення планети. Кількість користувачів збільшилась на 2,4% у порівнянні із 2024. Загалом, кількість користувачів Інтернетом по всьому світу постійно зростала за останнє десятиліття [5].

Ресторанна індустрія приносить різноманітні переваги як клієнтам, так і працівникам. Для клієнтів ресторани пропонують зручність та приємну атмосферу для харчування. Клієнти можуть насолоджуватися різноманітною кухнею, смаками та атмосферою в одному місці. Ресторани також створюють соціальну атмосферу, дозволяючи клієнтам спілкуватися з друзями та родиною під час трапези.

Для працівників ресторанна індустрія пропонує широкі можливості працевлаштування. Від офіціантів та кухарів до менеджерів та посудомийників, індустрія громадського харчування надає різноманітні посади для людей з різним рівнем кваліфікації. Співробітники можуть також отримувати вигоду від гнучкого графіка, конкурентоздатної заробітної плати та можливості отримання нових навичок.

Ресторанна індустрія також приносить економічну вигоду місцевим громадам. Ресторани створюють робочі місця, забезпечують надходження податків та сприяють підтримці місцевого бізнесу. Більше того, ресторани можуть допомогти залучити туристів у регіон, що може позитивно вплинути на місцеву економіку.

Загалом, ресторанна індустрія пропонує різні переваги клієнтам, працівникам та місцевим громадам. Від смачної їжі до можливостей для працевлаштування – ресторанний бізнес є важливим сегментом економіки [6].

1.2 Огляд сайтів-аналогів

На сьогодні ресторанна галузь пропонує користувачам широкий вибір платформ, що значно варіюються в якості та ефективності. Для того, щоб створити конкурентоспроможний ресурс, важливо провести аналіз існуючих системи-аналогів, визначивши їх переваги, недоліки та можливості для вдосконалення.

Розглянемо сервіси, такі як «За двома зайцями», «EastWest Connection» та «Plates».

«За двома зайцями» – ресторан, який пропонує автентичну київську кухню, що поєднує традиції української, єврейської, болгарської та молдавської гастрономії. Заклад відомий своїм затишним інтер’єром, що відтворює атмосферу старого Києва, та стравами, приготованими з локальних сезонних продуктів. Особливу увагу приділено збереженню кулінарної спадщини та автентичності смаків [7].

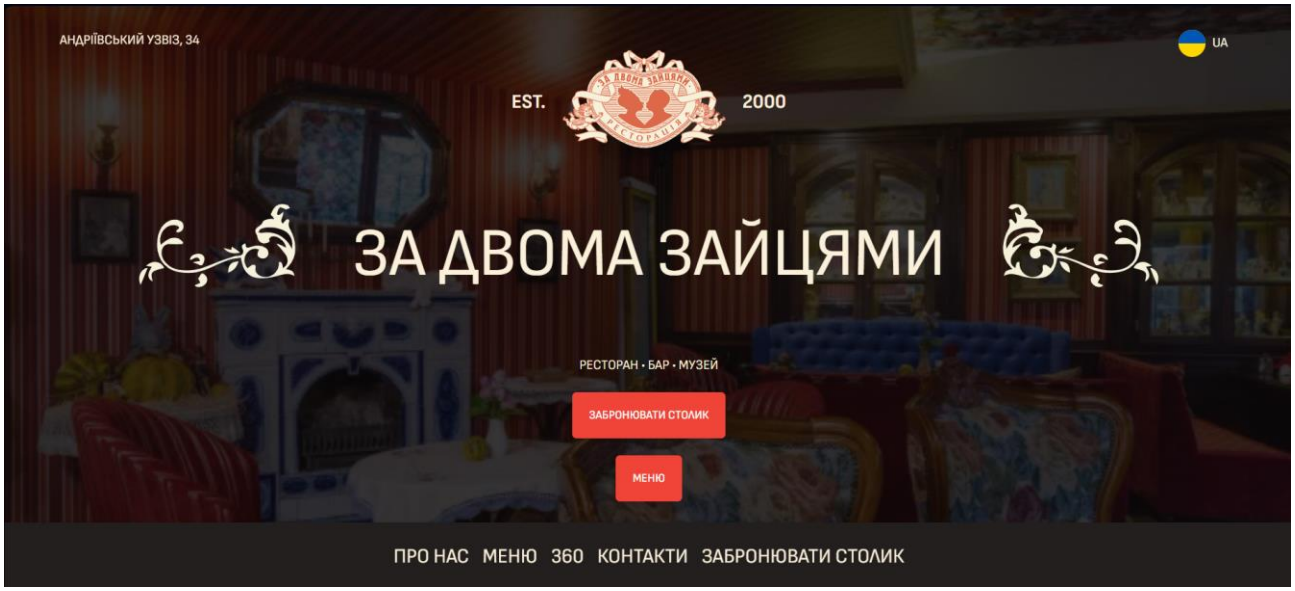

Рисунок 1.1 – Загальний вигляд WEB-ресурсу «За двома зайцями»

Переваги:

- наявність багатомовності, що забезпечує доступність для широкої аудиторії;
- можливість здійснити бронювання столиків безпосередньо через сайт;
- привабливий дизайн, який відображає історичну тематику ресторану, зберігаючи атмосферу старого Києва.

Але є такі недоліки:

- сайт не адаптований для відображення на мобільних пристроях, це ускладнює його використання для користувачів, що відвідують його через смартфони або планшети;
- меню доступне лише через зовнішнє посилання на PDF-файл, що може бути незручним для користувачів;
- немає можливості залишити відгук та оцінку про сайт.

«EastWest Connection» – це ресторан азійської ф'южн-кухні, розташований у Солт-Лейк-Сіті. Шеф-кухар Такеясу Нунокава, родом з Японії, пропонує поєднання класичних та сучасних інтерпретацій азійських страв. Меню включає різноманітні страви, такі як рамен, сукіякі, теріякі та інші, приготовані

з високоякісних інгредієнтів. Ресторан відомий своєю спокійною атмосферою та уважним обслуговуванням [8].

Переваги:

- зрозумілий інтерфейс, що дозволяє швидко знайти основну інформацію;
- оптимізований для мобільних пристроїв;
- є можливість залишення відгуків користувачами.

Недоліки:

- сайт виглядає застарілим, що не буде приваблювати користувачів;
- обмежене графічне подання меню, що може ускладнити вибір;
- не містить достатньої кількості інформації про ресторан та події.

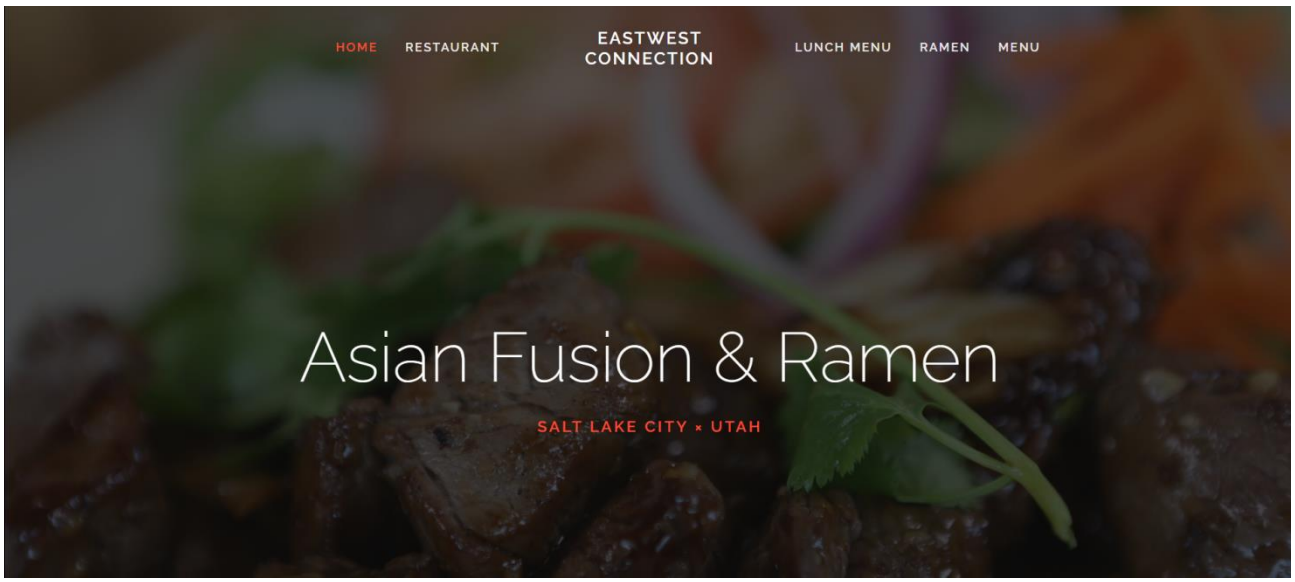

Рисунок 1.2 − Загальний вигляд WEB-ресурсу «EastWest Connection»

«Plates» – веганський ресторан, розташований на Old Street у Лондоні. Шеф-кухар Кірк Гоуорт, за рахунок великого досвіду та набутих навичків під час роботи на кухні, створює інноваційні страви, які поєднують сезонні інгредієнти та креативну подачу. Ресторан пропонує дегустаційне меню з кількох курсів, яке змінюється відповідно до сезону, та має обмежену кількість місць, що забезпечує інтимну атмосферу для гостей [9].

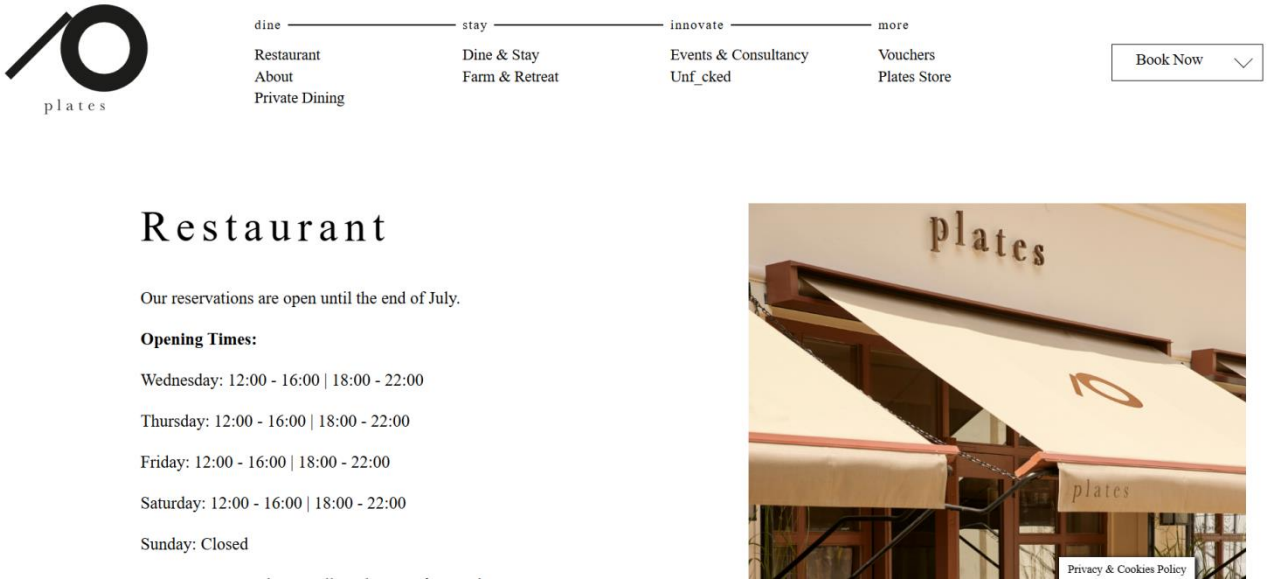

Рисунок 1.3 − Загальний вигляд WEB-ресурсу «Plates»

Переваги:

- привабливий та стильний дизайн сайту привертає увагу користувачів та залишає позитивне враження про ресторан;
- містить надзвичайно велику кількість різноманітної інформації.

Недоліки:

- сайт має проблеми зі стабільністю, часто виходить з ладу, створюється негативне враження для клієнтів;
- незрозумілий інтерфейс, що викликає проблеми з навігацією;
- недостатнє візуальне подання меню, що може ускладнювати вибір для користувачів.

1.3 Формулювання вимог та постановка задачі

Значення WEB-ресурсів у сучасному світі важко переоцінити, оскільки вони забезпечують швидкий та зручний доступ до широкого спектру інформації, що підтримує освітні процеси, бізнес-діяльність та прийняття обґрунтованих рішень. Для комерційних компаній, особливо тих, що працюють у сфері електронної торгівлі, якісно розроблені WEB-платформи відіграють критично

важливу роль у процесах залучення та утримання клієнтів, формування довіри споживачів та здійснення транзакцій.

Ефективність сайтів визначається кількома ключовими факторами, серед яких найважливішими є швидкість завантаження сторінок, інтуїтивно зрозумілий інтерфейс, зручна навігація, високий рівень інтерактивності, актуальний контент, надійні засоби безпеки та орієнтація на клієнта. WEB-ресурси, які дотримуються цих принципів, з більшою ймовірністю заохочують повторні відвідування та залучення нових користувачів. Водночас, ефективні WEB-платформи активно підтримують комунікацію, процеси обміну знаннями та управління якістю, що безпосередньо корелює з покращенням загальної ефективності та рівня задоволеності як співробітників, так і клієнтів [10, 11]. Але також існують певні недосконалості. Наприклад, часто сторінка може завантажуватись надто довго, що змушує відвідувачів чекати або взагалі залишати сторінку, меню незручно організоване, застарілий дизайн не передає атмосферу сучасного закладу, а іноді сайт взагалі не відображається на мобільних пристроях, що створює великі незручності, адже більшість на сьогодні виконують пошук інформації через мобільні пристрої.

Розробка ефективного WEB-ресурсу ресторану потребує комплексного підходу до проектування користувацького досвіду, який ґрунтується на сучасних принципах UI/UX дизайну. Інтерфейс розробленого WEB-ресурсу повинен бути інтуїтивно зрозумілим, простим в навігації і доступним, а дизайнерські рішення повинні ґрунтуватися на розумінні потреб і поведінки користувачів [12].

На основі розглянутих сайтів «За двома зайцями», «EastWest Connection» та «Plates» потрібно реалізувати рішення, яке дозволить клієнтам ознайомитися з асортиментом страв, їх складом, цінами та спеціальними пропозиціями, не звертаючись до персоналу, надасть можливість самостійно вибрати зручний час відвідування та зарезервувати столик, уникаючи черг або непорозумінь, а також оцінити сервіс, якість страв і загальне враження від закладу. Це значно підвищить ефективність обслуговування та створить позитивний досвід взаємодії з рестораном ще до його фізичного відвідування.

Для створення WEB-ресурсу «Ресторан» потрібно вирішити основні завдання:

- реалізувати онлайн-меню;
- реалізувати систему онлайн-бронювання;
- реалізувати систему відгуків та оцінок;
- здійснити тестування та оптимізацію WEB-ресурсу.

Отже, реалізація цих функцій дозволить створити зручний та ефективний інструмент для користувачів, який забезпечить простий взаємодію з рестораном та легкий доступ до його пропозицій, покращить взаємодію між персоналом і клієнтами, а також підвищить рівень задоволеності від використання сервісу.

1.4 Висновок

У першому розділі було проведено аналіз ресторанної галузі та розглянуто перспективи впровадження WEB-сайтів для розвитку ресторанного бізнесу.

В якості систем-аналогів були розглянуті WEB-сайти «За двома зайцями», «EastWest Connection» та «Plates». Однак огляд показав, що існуючі WEB-ресурси, незважаючи на свої переваги, не задовольняють всі потреби користувачів, маючи проблеми з швидкістю завантаження, стабільністю роботи, адаптивністю до мобільних пристроїв. Розробка WEB-сайту ресторану з урахуванням вимог користувачів та аналізу систем-аналогів є доцільною, оскільки дозволить створити зручний та ефективний інструмент для залучення нових клієнтів, покращення комунікації з ними, підвищення продажів.

Визначено завдання для подальшої розробки WEB-ресурсу, що має на меті забезпечити ефективність взаємодії клієнтами та допомогти оптимізувати процес їх обслуговування.

2 ПРОЕКТУВАННЯ WEB-РЕСУРСУ «РЕСТОРАН»

2.1 Вибір архітектури для розробки WEB-ресурсу «Ресторан»

Важливим етапом в розробці є вибір відповідної архітектури, яка забезпечить ефективну організацію коду, полегшить підтримку та розвиток проекту, а також дозволить масштабувати систему у майбутньому. Розробка WEB-ресурсу для ресторану – вагомий проект, в якому із масштабуванням та регулярними оновленнями інтерфейсу можуть виникати помилки, що цілком здатні порушити цілісність та принцип роботи системи. Виникає потреба надалі вносити зміни, опрацьовуючи лише конкретні аспекти, а не переробляючи все програмне забезпечення. Для оптимальної роботи програмного забезпечення доцільно розділити процеси на окремі структури, які відповідають за різні функціональні частини системи. Таке розділення дозволить підвищити зручність підтримки, забезпечить повторне використання коду та спростить тестування окремих компонентів. Крім того, воно створить чітку організацію проекту, що особливо важливо при роботі у команді або подальшому масштабуванні.

Архітектура MVC (Model-View-Controller) – це конструкційний шаблон архітектури програмного забезпечення, який розділяє програму на три основні компоненти: модель, вигляд і контролер, що полегшує керування та підтримку кодової бази. Це дозволяє модифікувати кожен елемент незалежно і також використовувати компоненти повторно, що забезпечить більш гнучку архітектуру, що легко адаптується під нові вимоги і технології, роблячи проект більш стійким до змін. Завдяки MVC розробники можуть паралельно працювати над різними частинами програми – один над логікою даних, інший – над візуальним оформленням і т.д. Це підвищує продуктивність команди і полегшує внесення змін чи доповнень у майбутньому [13, 14].

На рис. 2.1. представлено компоненти шаблону MVC та як вони між собою взаємодіють.

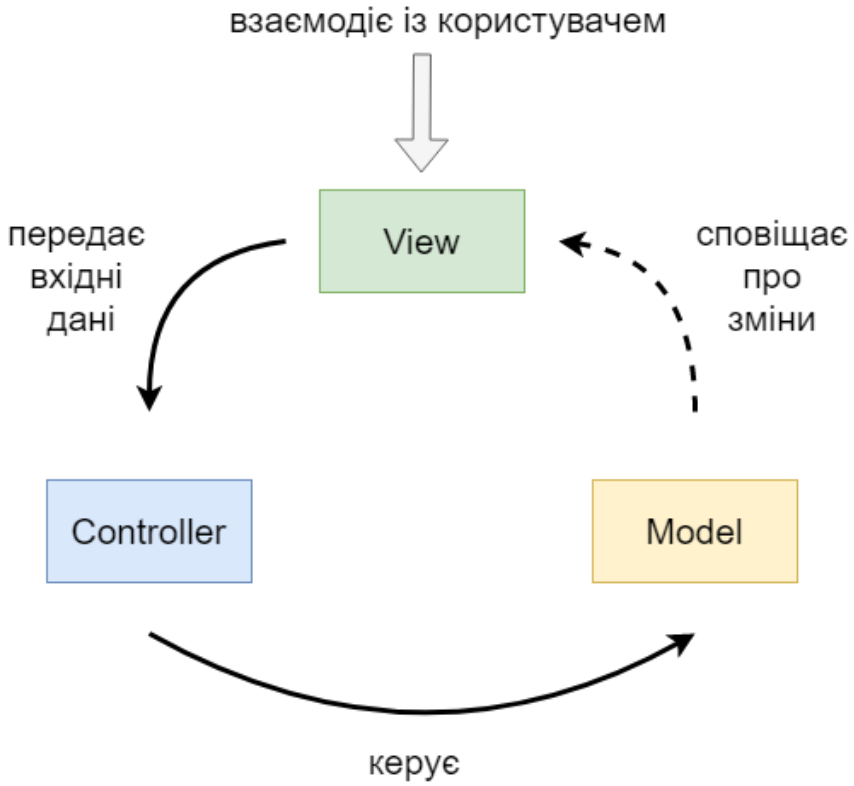

Рисунок 2.1 – Діаграма взаємодії між компонентами шаблону MVC

Наприкінці 1970-х Трюгве Ренскауг розробив концепцію MVC, прагнучи створити універсальний підхід до структурування програм, в яких користувачі працюють із великими і складними даними. Спочатку його модель включала чотири компоненти: Модель, Вид, Річ і Редактор. Однак після обговорень із колегами команда вирішила зосередитися лише на трьох основних елементах: моделі, виді та контролері [15].

Модель (Model) надає контролеру інформацію, яку запросив користувач, наприклад, повідомлення, сторінку книги або фотоальбом. Вона залишається незмінною незалежно від способу її представлення, що дозволяє використовувати різні варіанти відображення даних. Модель містить основну логіку застосунку, відповідальну за виконання поставлених задач, таких як робота форуму, магазину чи банківської системи.

Вид (View) відповідає за те, як дані з моделі будуть представлені користувачу. Він може бути шаблоном, наповненим отриманою інформацією.

Може бути декілька різних видів, серед яких контролер вибирає, який підходить найкраще для даної ситуації.

Контролер (Controller) займається організаційною логікою програми, забезпечуючи її коректне функціонування та управління взаємодією між користувачем і системою. Він керує запитами користувача, коли той взаємодіє із елементами інтерфейсу. Його головне завдання – викликати та узгодити роботу потрібних ресурсів і об’єктів, необхідних для виконання запитів користувача [16].

Переваги використання MVC архітектури:

- швидка розробка: MVC підтримує паралельну розробку, що значно прискорює процес створення WEB-ресурсу. Один програміст може працювати над представленням, а інший – над контролером для реалізації бізнес-логіки. Завдяки цьому розробка може проходити приблизно втричі швидше, ніж при використанні інших архітектурних підходів;

- підтримка асинхронного підходу: MVC легко інтегрується з JavaScript-фреймворками та підтримує асинхронне завантаження даних, що дозволяє додаткам працювати швидше та ефективніше. Він також може обробляти PDF-файли, спеціальні браузерні додатки та віджети на робочому столі;

- локальні модифікації без впливу на всю модель: інтерфейс WEB-ресурсу змінюється частіше, ніж його бізнес-логіка. MVC дозволяє оновлювати вигляд, не зачіпаючи модель. Наприклад, можна змінювати шрифти, кольори або макети без ризику порушити роботу основного функціоналу;

- чітке розділення даних та форматування: MVC повертає дані без форматування, що дозволяє використовувати їх у будь-якому інтерфейсі. Наприклад, один і той самий контент можна відобразити у вигляді HTML, Flash або іншими способами без необхідності його змінювати;

- оптимізація для пошукових систем (SEO): завдяки MVC можна легко створювати SEO-дружні WEB-сторінки та URL-адреси, що сприяє залученню більшої кількості користувачів. Крім того, архітектура MVC добре підходить для

розробки WEB-ресурсів із використанням тестування та інтеграції скриптових мов, таких як JavaScript і jQuery, для розширення можливостей [17].

2.2 Проектування WEB-ресурсу «Ресторан» із застосуванням UML-діаграм

Об’єктно-орієнтоване програмування(ООП) на сьогоднішній день є одним із найпопулярніших і найефективніших підходів до розробки програмного забезпечення. Для глибшого розуміння принципів цього підходу далі розглянемо основні поняття та складові ООП.

ООП – це метод розробки, у якому програми створюються на основі об’єктів, що містять дані (поля) та функціональність (методи). Головні концепції ООП включають інкапсуляцію, наслідування, поліморфізм і абстракцію, що дозволяє будувати гнучкі, масштабовані та прості у підтримці програмні рішення [18].

Клас – це шаблон, який визначає структуру майбутніх об’єктів. Він об’єднує пов’язані дані та методи, що дозволяє легко управляти ними. Коли створюється екземпляр класу, він стає конкретним об’єктом, побудованим за заданою моделлю. Класи містять в собі властивості та методи [19].

Об’єкт – це конкретний екземпляр класу, який формується в пам’яті під час виконання програми. Він успадковує властивості та методи класу, що дозволяє йому виконувати визначені функції та взаємодіяти з іншими об’єктами.

Властивості – це змінна, яка дозволяє зберігати певний тип даних, визначає які дані можуть бути збережені в об’єкті і які операції можна виконати.

Методи – це функції, що визначають, як буде діяти об’єкт. Вони задають можливості об’єкта, дозволяючи йому виконувати певні завдання, такі як обробка інформації, виконання обчислень або показ результатів. Методи класу визначають, які операції доступні для його екземплярів.

Розглянемо детальніше головні принципи об’єктно-орієнтованого програмування:

a) Абстракція: в програмуванні дозволяє зосередитися на функціональності об’єкта, не заглиблюючись у деталі його реалізації. В ООП вона передбачає збирання максимальної кількості суттєвих даних про об’єкт, що сприяє створенню незалежних модулів, які можуть взаємодіяти між собою. У житті та програмуванні ми вибірково звертаємо увагу лише на те, що є важливим для нас або для певного модуля. Зміни в одному модулі не впливають на інші, а користувачеві достатньо знати лише його функціонал, не заглиблюючись у внутрішні механізми роботи. Це демонструє основний принцип абстракції – приховування деталей і спрощення взаємодії.

b) Інкапсуляція: принцип програмування, який допомагає приховати внутрішню реалізацію об’єкта та захистити його дані від некоректного використання. Вона дозволяє працювати з об’єктом через спеціальні методи, не заглиблюючись у те, як саме він влаштований. Це забезпечує цілісність і безпеку даних, запобігаючи їх ненавмисній зміні. Завдяки інкапсуляції програмісти можуть створювати зрозумілі, організовані та надійні системи.

c) Наслідування: спосіб повторного використання коду, що зменшує його дублювання та спрощує структуру програм. Воно дозволяє створювати класи, які отримують властивості та поведінку від загальних базових компонентів. Цей дає можливість визначати базову функціональність у загальних класах, а спеціалізовані класи можуть доповнювати або змінювати під конкретні потреби. Дочірній клас наслідує методи батьківського, при цьому батьківський клас не отримує характеристик дочірнього.

d) Поліморфізм: принцип, який дозволяє створювати нові об’єкти, що можуть взаємодіяти з уже існуючими. Наприклад, для написання тексту можна використовувати різні інструменти, такі як ручка, олівець чи маркер, і незалежно від вибору вони виконують одну й ту ж функцію – залишають слід на папері. У програмуванні аналогічний принцип застосовується до об’єктів, які виконують один метод, але можуть поводитись по-різному. Наприклад, літак і космічний корабель по-різному рухаються у просторі, але для спостерігача обидва просто

летять. Поліморфізм допомагає зробити системи гнучкішими та простішими для розширення [20 – 22].

UML (Unified Modeling Language) – це стандартизована мова моделювання, яка використовується для опису програмних систем. Modeling означає створення моделі, що відображає об'єкт, а Unified підкреслює її універсальність, оскільки UML підходить для різних типів програм, організацій та рівнів розробки. Головна особливість UML – єдиний синтаксис, що дозволяє створювати діаграми, які будуть зрозумілі всім, хто знайомий з цією графічною мовою. Застосування UML дозволяє візуалізувати основні компоненти системи, їх взаємодії та бізнес-процеси, що значно спрощує розуміння архітектури проекту як для розробників, так і для замовників. Використання UML-діаграм допоможе зрозуміти, яким чином будуть реалізовані функціональні модулі WEB-ресурсу.

Одне із завдань UML – служити засобом комунікації всередині команди та при спілкуванні з замовником. Розглянемо випадки, в яких можна застосувати діаграми:

- проєктування: UML-діаграми допомагають структурувати архітектуру великих систем, показуючи як загальні, так і детальні елементи, що стануть основою для коду;
- реверс-інжиніринг: дозволяє створити UML-модель на основі вже існуючого коду, що корисно при роботі з проєктами, де документація неповна або відсутня;
- документування: з моделей можна отримувати текстову інформацію, яка у поєднанні з графікою покращує розуміння структури програми.

Відомі такі основні види UML-діаграм:

- діаграма прецедентів (Use-case diagram);
- діаграма класів (Class diagram);
- діаграма активностей (Activity diagram);
- діаграма послідовності (Sequence diagram);
- діаграма розгортання (Deployment diagram);

- діаграма співробітництва (Collaboration diagram);
- діаграма об’єктів (Object diagram);
- діаграма станів (Statechart diagram).

Для представлення моделей було обрано діаграму класів та діаграму прецедентів, які далі розглянемо більш детально.

Діаграма класів – це статичне представлення моделі, яке описує зв’язки між елементами програмного коду, їхню функціональність та структуру. Вона надає детальне бачення взаємодії між класами, а також часто використовується як основа для генерації програмного забезпечення. Діаграма класів дозволяє візуально відобразити структуру системи, основні сутності (класи), їхні властивості, методи та взаємозв’язки між ними.

У серверній частині WEB-ресурсу «Ресторан» є такі ключові класи: Application, Model, Mailer, Translator.

Клас Application містить лише такі методи: static run() та static render(). Метод static run() визначає маршрут та викликає відповідний рендеринг сторінки. Метод static render($page, $data) рендерить сторінку, підключаючи заголовок, основний контент та футер.

Клас Model містить атрибути dish_file, review_file, booking_file, dishes_data, reviews_data. Містить такі методи: init() ініціалізує класи, getDishes() отримує та групує страви за категоріями, getReviews() отримує відгуки, getBookings() отримує бронювання, addReview($name, $text, $rating) додає новий відгук, addBooking($name, $telephone, $date, $time, $quantity, $email, $additional) додає нове бронювання.

Клас Mailer містить тільки метод static send(), який відправляє email з заданими параметрами.

Клас Translator містить атрибути text_file, text, default_lang, allowed_langs. Містить такі методи: init() ініціалізує переклади, getActiveLang() повертає активну мову, getCommonText() повертає переклади для активної мови, getDefaultLang() повертає мову за замовчуванням, getLangPostfix() повертає постфікс для URL, getCategoryNameByKey($key) повертає переклад назви

категорії, getLabelNameByKey($key) повертає переклад мітки, getDishField($field_key, $dish) повертає переклад поля страви.

На рисунку 2.2 зображена UML-діаграма класів серверної частини WEB-ресурсу «Ресторан».

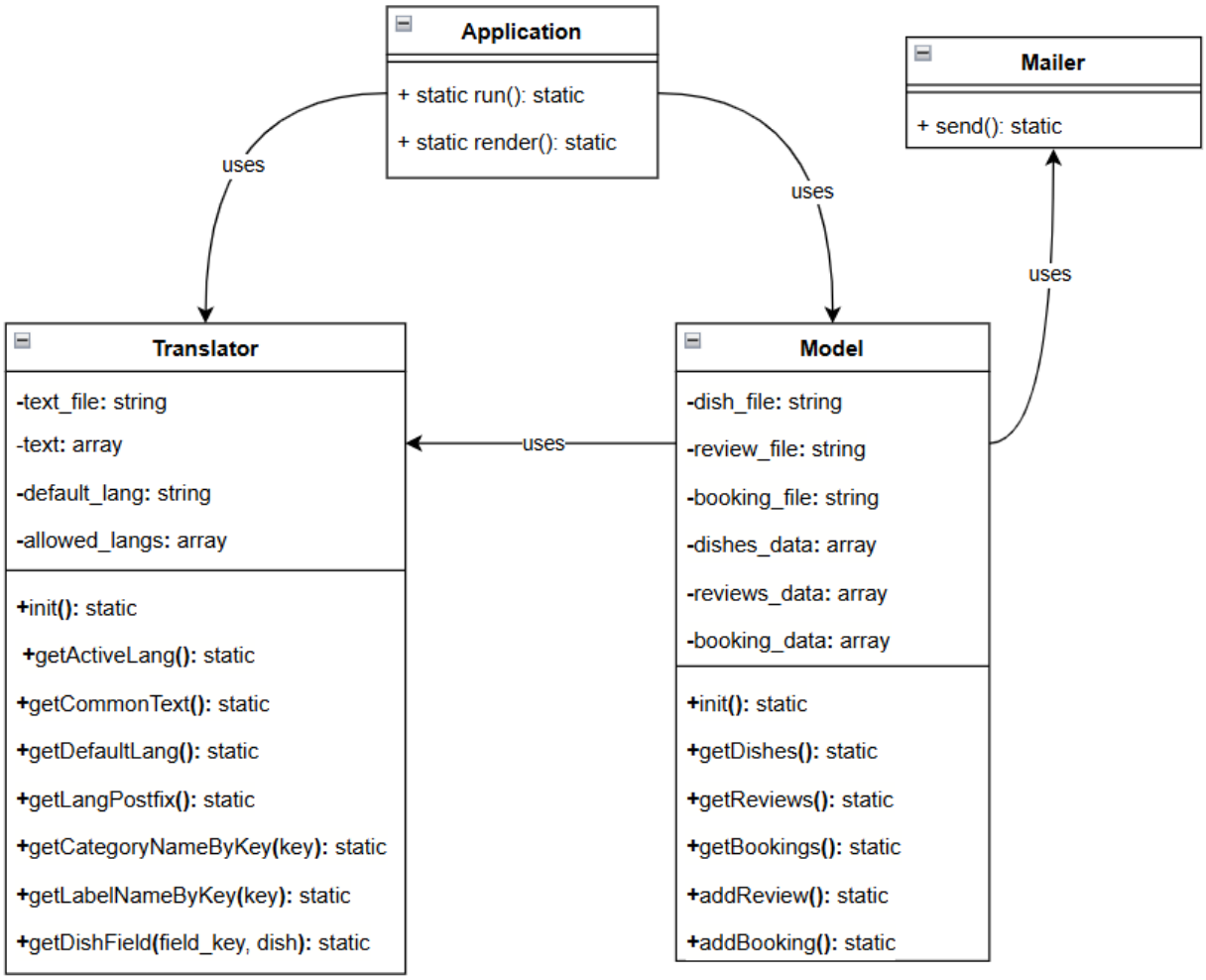

Рисунок 2.2 – UML-діаграма класів WEB-ресурсу «Ресторан»

Діаграма прецедентів дозволяє візуально відобразити функціональні можливості, які система повинна забезпечувати. Вона допомагає зрозуміти, які ролі взаємодіють із системою, які задачі користувачі можуть виконувати, а також як різні компоненти системи співпрацюють між собою.

Діаграма прецедентів складається з двох ключових елементів: учасник (Actor) і прецедент (Use case). Учасник – це набір ролей, які взаємодіють із системою, підсистемою або класом. Учасником може бути людина, її роль у

системі чи навіть інша система, що виконує певні функції. Прецедент – це опис конкретної поведінки системи з точки зору користувача. Він не пояснює, як саме досягається результат, а лише визначає, які дії виконує система [23]. На рис. 2.3 зображено діаграму прецедентів WEB-ресурсу «Ресторан».

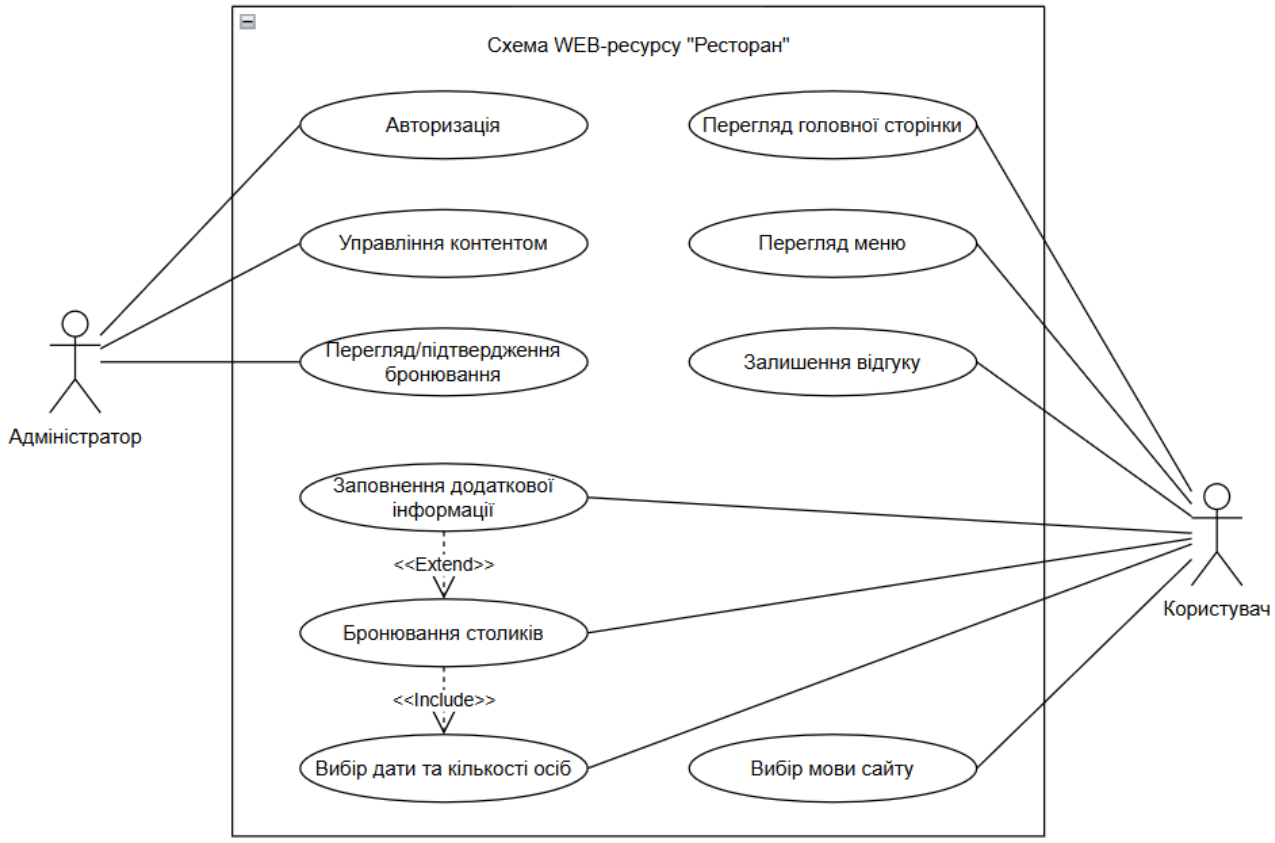

Рисунок 2.3 – Діаграма прецедентів WEB-ресурсу «Ресторан»

Зображені на діаграмі основні функціональні можливості, які виконує користувач, включають в себе перегляд сторінок, вибір мови інтерфейсу, залишення відгуку та оцінки і бронювання столика. Також адміністратор може авторизуватися для отримання доступу до сторінки адміністратора, управляти контентом WEB-ресурсу та переглядати чи підтверджувати бронювання столиків. Ці можливості забезпечують зручність та ефективність використання WEB-ресурсу «Ресторан», що в подальшому залишить позитивний досвід як в користувачів, так і в адміністраторів.

2.3 Розробка схеми алгоритму роботи WEB-ресурсу «Ресторан»

Алгоритм – це набір чітко визначених кроків і правил, що ведуть до розв'язання задачі або отримання бажаного результату. Він визначає послідовність операцій і дій, необхідних для виконання завдання. Алгоритми можуть містити умовні оператори, цикли та обробку даних, що допомагає програмі працювати ефективно й досягати поставленої мети [24]. Алгоритми складаються з набору конкретних інструкцій або дій, які комп'ютер виконує для розв'язання певної задачі. Вони є важливим елементом розробки програмного забезпечення, оскільки визначають порядок виконання операцій і забезпечують коректне функціонування програми [25].

Блок-схема – це візуальне представлення алгоритму або процесу у вигляді графічних елементів. Вона містить блоки, що позначають різні операції, умови, цикли та дані, а також лінії, які визначають послідовність виконання та напрямок руху інформації. Таке зображення допомагає зрозуміти структуру процесу та спрощує його аналіз і оптимізацію [26].

Для опису алгоритму роботи модуля бронювання було вибрано представлення алгоритму у вигляді блок-схеми. Схему алгоритму зображено на рисунку 2.4 і 2.5.

Розглянемо алгоритм роботи модуля бронювання (рис. 2.4 – 2.5). Алгоритм складається з таких кроків:

Крок 1. Система завантажує дані сторінки.

Крок 2. Відбувається перевірка на коректне відображення сторінки. Якщо все коректно, то перейти до кроку 3, інакше - до кроку 7.

Крок 3. Система перевіряє обрану мову інтерфейсу.

Крок 4. Якщо користувач обрав іншу мову інтерфейсу, перехід до кроку 5, інакше – крок 6.

Крок 5. Система оновлює мову інтерфейсу на обрану.

Крок 6. Залишається поточна мова інтерфейсу.

Крок 7. Виведення помилки, користувач повертається до кроку 2.

Крок 8. Користувач оформлює бронювання.

Крок 9. Користувач вводить дані: ім’я, телефон, дата, час, кількість гостей тощо.

Крок 10. Натискання користувачем кнопки «Забронювати».

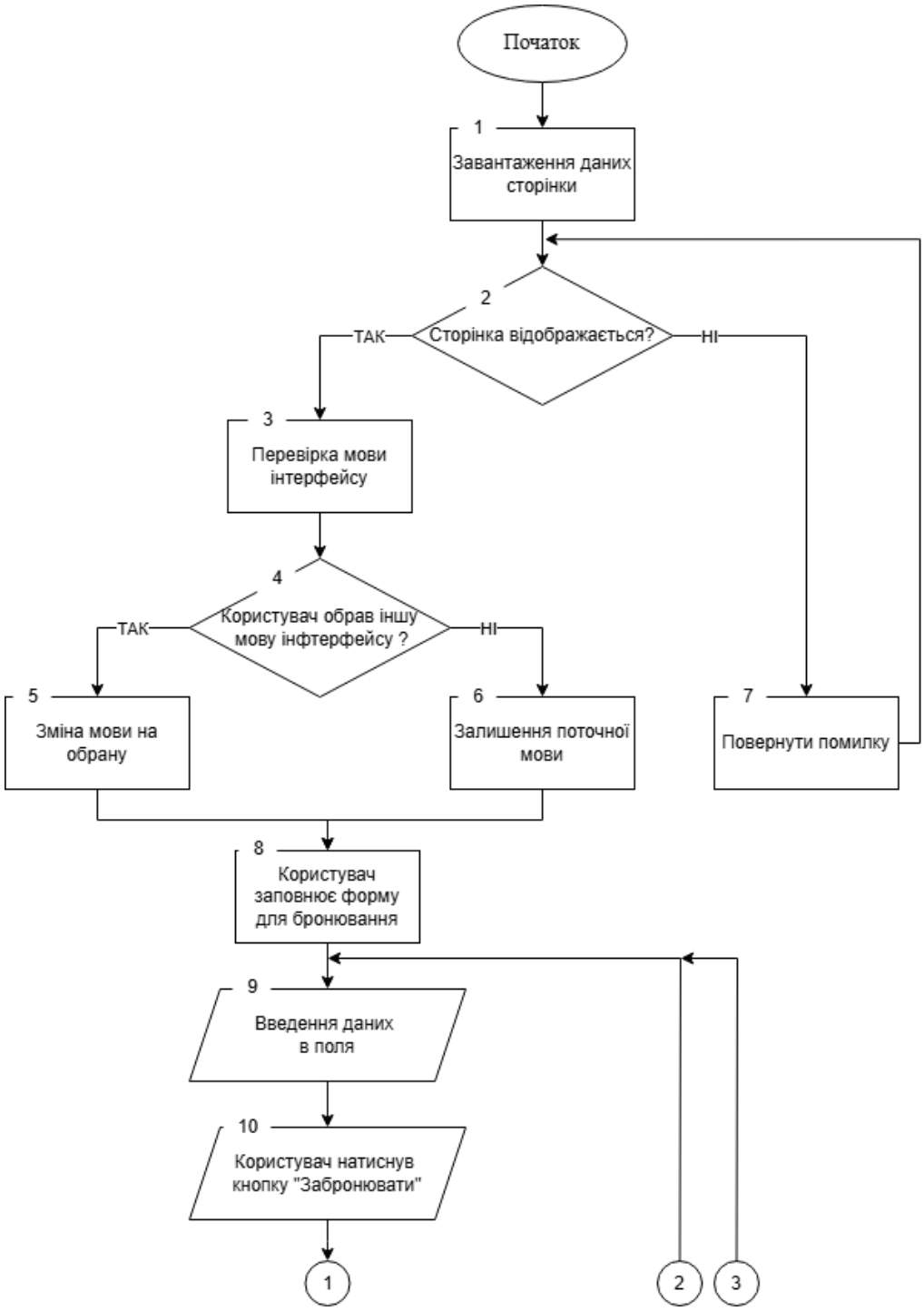

Рисунок 2.4 – Схема алгоритму роботи модуля бронювання

Крок 11. Система переходить до перевірки даних.

Крок 12. Відбувається перевірка обов’язкових полів. Якщо всі поля заповнені, тоді перехід до кроку 13, якщо ні – крок 14.

Крок 13. Система перевіряє чи є вільні столики на обрану дату та час.

Крок 14. Виведення помилки, користувач повертається до кроку 9.

Крок 15. Якщо є вільні столики на обрану дату та час, перехід до кроку 16, якщо ні – крок 17.

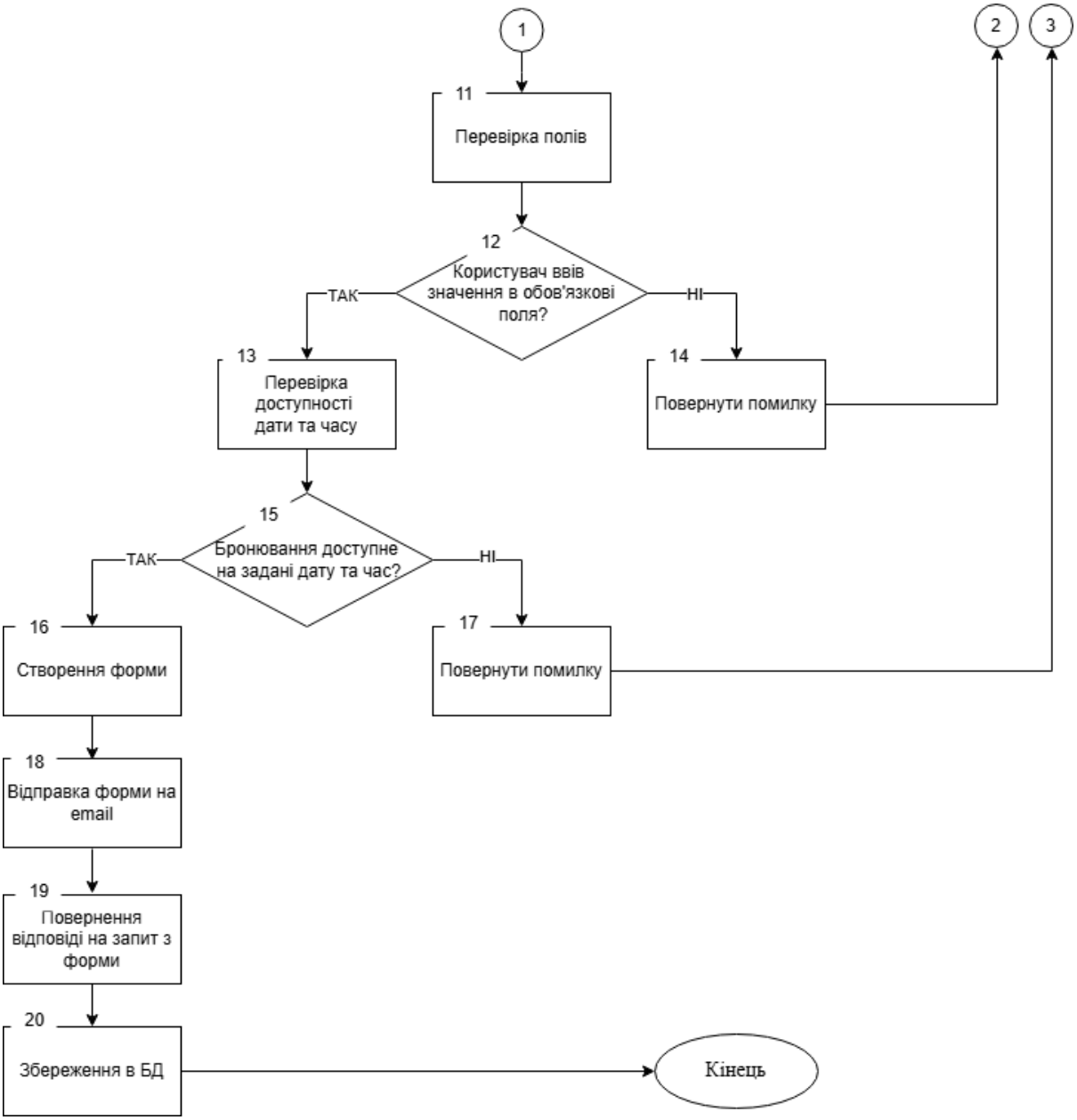

Рисунок 2.4, аркуш 2

Крок 16. Створення запиту на бронювання.

Крок 17. Виведення помилки, користувач повертається до кроку 9.

Крок 18. На адресу адміністратора надсилається лист з деталями бронювання.

Крок 19. Користувач отримує повідомлення про підтвердження бронювання.

Крок 20. Дані записуються в базу даних.

Також створено блок-схему алгоритму роботи самого WEB-ресурсу «Ресторан», зображену на рис. 2.5.

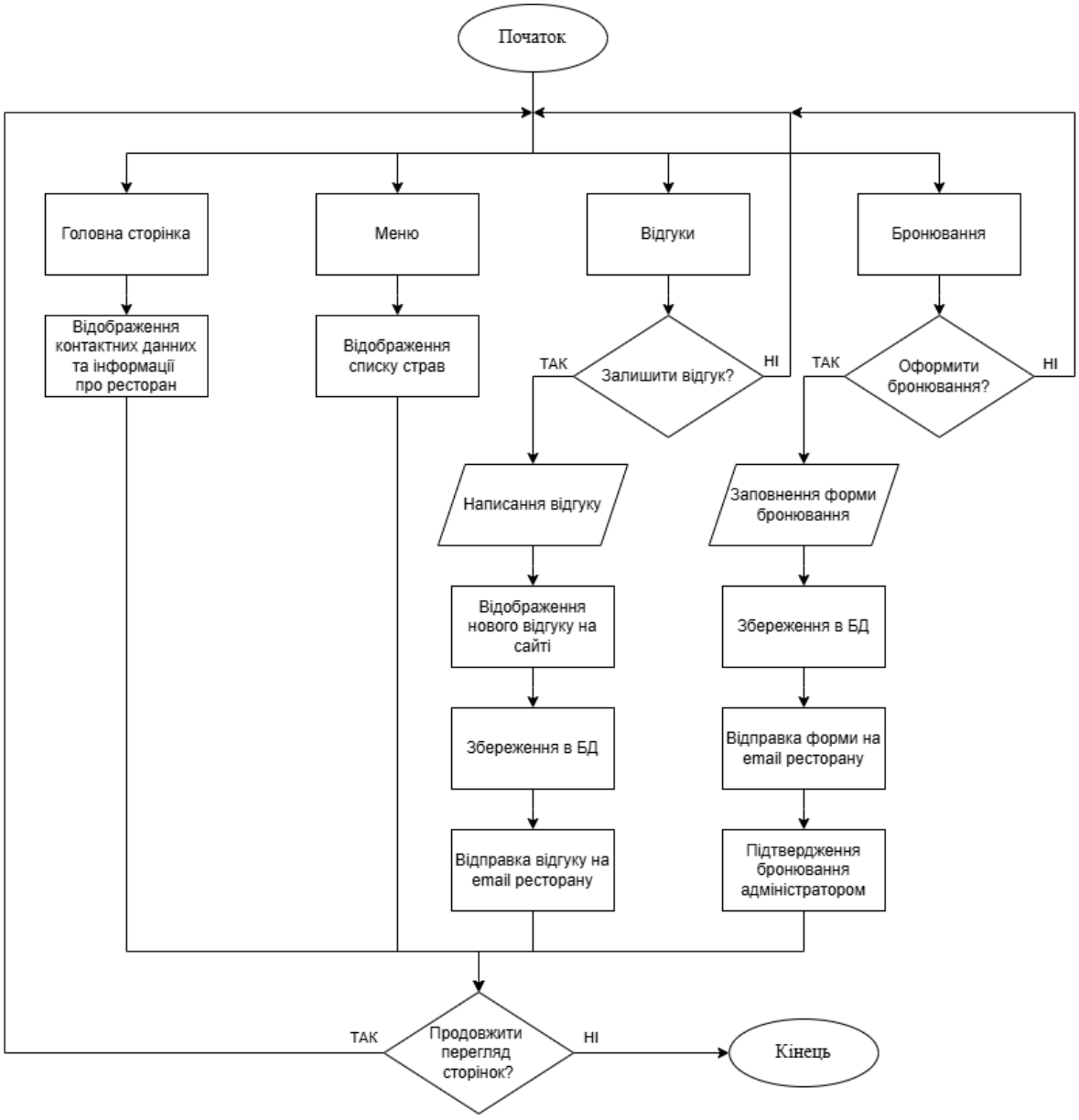

Рисунок 2.5 – Схема алгоритму роботи WEB-ресурсу «Ресторан»

2.4 Розробка ER-моделі бази даних

ER-модель дає змогу зобразити основні сутності системи та зв’язки між ними, що спрощує розуміння логіки взаємодії даних і допомагає попередньо спланувати структуру бази даних [27]. В універсальне відношення потрібно включити атрибути, що описують такі сутності: користувач, адміністратор, страва, меню, бронювання, відгук, категорія.

Універсальне відношення для реалізації бази даних WEB-ресурсу «Ресторан» буде мати такий вигляд: R (код користувача, ім’я користувача, електрона пошта користувача, номер телефону користувача, код бронювання, дата бронювання, час бронювання, кількість гостей, код страви, назва страви, ціна страви, вага страви, опис страви, фото страви, код категорії, назва категорії, код меню, назва меню, код адміністратора, ім’я адміністратора, пароль адміністратора, електронна пошта адміністратора, код відгуку, опис відгуку, дата відгуку, оцінка). Ступінь універсального відношення – 26.

Між двома сутностями у базі даних WEB-ресурсу «Ресторан» виділено типи зв’язку ОДИН-ДО-БАГАТЬОХ (1:Б) та БАГАТО-ДО-БАГАТЬОХ (1:Б).

Зв’язок ОДИН-ДО-БАГАТЬОХ є між сутностями «Користувач»-«Бронювання», «Адміністратор»-«Бронювання», «Користувач»-«Відгук», «Страва»-«Категорія», «Страва»-«Меню».

Зв’язок БАГАТО-ДО-БАГАТЬОХ є тільки між сутностями «Страва»-«Відгук». Характеристики відношень між сутностями представлено у таблиці 2.1.

Таблиця 2.1 – Характеристики відношень між сутностями WEB-ресурсу «Ресторан»

Ім’я сутності 1	Ім’я сутності 2	Тип зв’язку	Клас належності
Користувач	Бронювання	1:Б	Необов’язковий
Адміністратор	Бронювання	1:Б	Обов’язковий
Користувач	Відгук	1:Б	Необов’язковий
Страва	Відгук	Б:Б	Необов’язковий
Страва	Меню	Б:1	Обов’язковий
Страва	Категорія	Б:1	Обов’язковий

Представлення ER-моделі бази даних WEB-ресурсу «Ресторан» зображено на рисунку 2.6.

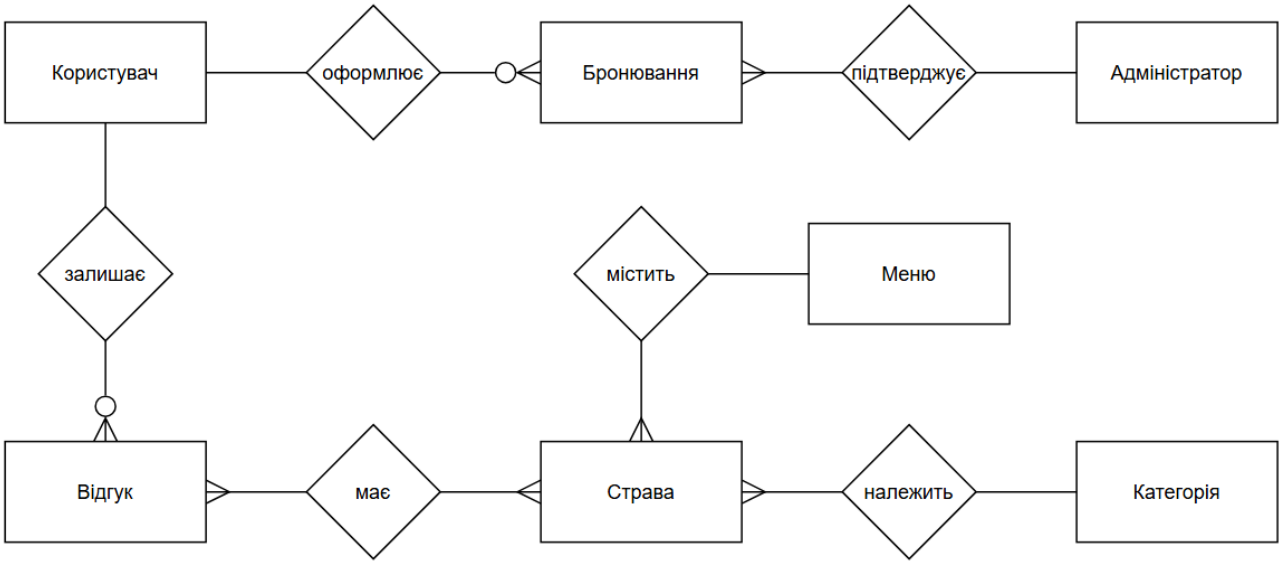

Рисунок 2.6 – ER-діаграма бази даних WEB-ресурсу «Ресторан»

2.5 Висновок

В другому розділі було обрано MVC архітектуру, яка забезпечила чітке розділення логіки. Використання цього підходу дозволило створити структурований код, який легко підтримувати та розширювати, забезпечивши ефективну взаємодію між різними компонентами системи ресторану. Разом з цим, було розглянуто принципи ООП, які дозволяють створювати гнучкі та масштабовані WEB-ресурси.

Було застосовано UML-діаграму класів і UML-діаграму прецедентів для проектування WEB-ресурсу «Ресторан». Це допомагає представити структуру програми у графічному вигляді, показати наявні класи, їхні зв’язки та методи, які вони містять. Графічно представленні структури програми дають можливість розробникам легко зрозуміти функціонування системи.

Також було розроблено схему алгоритму функціонування модуля бронювання і схему алгоритму роботи WEB-ресурсу «Ресторан». Схема

алгоритму дозволяє зрозуміти, як працює програма поетапно, що значно спрощує її розробку та мінімізує ризик помилок. Правильно спроєктований алгоритм підвищує ефективність програми, забезпечуючи швидке та оптимальне виконання завдань.

Крім того, було створено ER-модель, яка графічно зображає сутності та зв’язки між ними. Ця модель дозволяє чітко структурувати інформацію, забезпечуючи ефективне зберігання та обробку даних у базі, а також слугує фундаментом для подальшої реалізації бази даних і її інтеграції з програмною частиною. Правильне проектування ER-моделі сприяє уникненню надлишковості даних, забезпеченню цілісності та узгодженості інформації, що є критично важливим для коректного функціонування системи.

Ці етапи є важливими для успішної розробки продукту, оскільки вони забезпечують чітке уявлення про систему, допомагають структурувати робочий процес і дозволяють виявляти проблеми ще на ранніх стадіях. Такий підхід сприяє успішному запуску продукту, а також спрощує його підтримку та вдосконалення у майбутньому.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-РЕСУРСУ «РЕСТОРАН»

3.1 Обґрунтування вибору мови програмування та бібліотек

Перед початком розробки постає необхідність ретельного вибору мов програмування, фреймворків та бібліотек, які б найкраще відповідали технічним і функціональним вимогам проєкту. Вибір інструментів є одним із ключових етапів у створенні будь-якого програмного продукту, оскільки саме від нього залежать гнучкість архітектури, швидкість розробки, зручність супроводу, адаптивність до змін, а також загальна якість програмного забезпечення.

Сучасна WEB-розробка передбачає тісну взаємодію клієнтської та серверної частин застосунку, що обумовлює потребу у використанні кількох мов програмування та бібліотек одночасно. При цьому важливо враховувати не лише популярність чи технічні характеристики мови, а й її сумісність з іншими технологіями, рівень підтримки спільнотою, наявність документації та готових рішень.

PHP – скриптова мова програмування, що була створена для генерації HTML-сторінок на стороні WEB-сервера PHP, є однією з найпоширеніших мов, що використовуються у сфері WEB-розробок. PHP може обробляти дані, керувати сеансами, обробляти форми та інтегрувати бази даних. PHP широко застосовується у розробці WEB-додатків і програмних рішень у різних сферах. Ця мова підтримує розвиток таких технологій, як системи управління контентом, розробка WEB-додатків і сайтів, створення платформ електронної комерції, аналітика та візуалізація даних, обробка зображень, графічні інтерфейси додатків.

Переваги PHP:

- відкритий код і безкоштовність: PHP доступний кожному, його легко завантажити та використовувати для розробки WEB-додатків;
- кросплатформенність: працює на різних ОС, включаючи UNIX, Linux і Windows;

- швидке завантаження додатків: додатки на PHP працюють ефективніше навіть при низькій швидкості Інтернету;
- легкість у навчанні: простий синтаксис дозволяє швидко освоїти PHP, особливо для тих, хто знайомий із мовою C;
- стабільність: постійна підтримка та оновлення забезпечують надійну роботу протягом років;
- ефективне повторне використання коду: допомагає уникнути зайвого дублювання та спрощує структуру додатків;
- зручність керування кодом: чітка структура та наявність потужних бібліотек полегшують розробку;
- інтеграція з базами даних: просте підключення до різних СУБД дозволяє швидко створювати WEB-додатки на основі контенту;
- гнучкість: PHP легко поєднується з іншими мовами програмування, що дозволяє використовувати найкращі технології для кожної функції [28].

CSS (Cascading Style Sheets) – це мова таблиць стилів, яка використовується для стилізації HTML-елементів на WEB-сторінці. Це дозволяє розробникам контролювати та керувати макетом та дизайном сторінки. У WEB-розробці має такі переваги, як відокремлення контенту від дизайну, надання розширених можливостей стилізації, адаптивний дизайн та налаштовуваний стиль [29].

HTML (HyperText Markup Language) – це мова розмітки гіпертексту, стандартна мова розмітки документів, призначена для відображення та перегляду в Інтернеті, також допомагає створювати структуру WEB-сторінки. Оскільки це мова розмітки, вона складається з багатьох тегів. Є теги для відображення тексту, таблиць, упорядкованих списків і невпорядкованих списків тощо. На HTML-сторінці є два основних розділи: розділ head і body. HTML є незалежною від платформи мовою, тому її можна використовувати на будь-якій платформі, як-от Windows, Linux, Macintosh тощо.

Переваги HTML:

- широко використовувана мова розмітки;

- проста в освоєнні;
- багатоплатформенність;
- зберігання великих файлів дозволено завдяки функції кешу додатків;
- вільний синтаксис;
- легко редагувати;
- легко інтегрується з іншими мовами, такими як JavaScript, CSS тощо;
- дозволяє використовувати шаблони, що спрощує проектування WEB-сторінки [30].

JavaScrip – це скриптова мова програмування, яка активно використовується для створення інтерактивних WEB-застосунків. Завдяки JS WEB-сторінки стають динамічними: додаються анімації, спливаючі повідомлення, форма перевірки даних та можливість оновлення контенту без перезавантаження.

Переваги JavaScript:

- інтерпретованість: код виконується безпосередньо в браузері без попередньої компіляції, що дозволяє миттєво бачити зміни;
- інтеграція з HTML та CSS: JS чудово взаємодіє з цими технологіями, забезпечуючи повноцінну розробку WEB-інтерфейсів;
- динамічна типізація: змінні отримують тип під час виконання коду, що спрощує розробку;
- широкий набір бібліотек і фреймворків: готові рішення допомагають розробникам прискорити процес створення додатків;
- кросплатформеність: JS працює на різних пристроях і платформах;
- автоматичне управління пам’яттю: механізм збирання сміття автоматично видаляє непотрібні об’єкти;
- підтримка різних стилів програмування: мова поєднує імперативний, функціональний підхід та принципи ООП;
- асинхронність: JS ефективно керує задачами без блокування виконання інших процесів;

- робота з DOM: дає можливість змінювати елементи WEB-сторінки та обробляти події в режимі реального часу;

- відкритий вихідний код: мова доступна для використання без ліцензійних обмежень, що забезпечує її популярність [31].

TypeScript – це надмножина JavaScript, призначена для покращення розробки простого коду JavaScript шляхом додавання статичної типізації, класів та інтерфейсів. Ця необов’язкова статична типізація дозволяє краще організувати код і використовувати методи об’єктно-орієнтованого програмування. TypeScript вдосконалює інструменти розробки та розширює мову за межі стандартних декораторів, залишаючись при цьому конвертованим у простий JavaScript [32].

У таблиці 3.1 наведено основні відмінності між мовами програмування JavaScript та TypeScript.

Таблиця 3.1 – Порівняння мов програмування JavaScript і TypeScript

Критерій	JavaScript	TypeScript
Типізація	Динамічна, без статичної типізації	Статична типізація з можливістю анотацій типів
Сумісність з браузерами	Підтримується безпосередньо у всіх сучасних браузерах	Потребує компіляції в JavaScript для виконання у браузері
Складність	Простий синтаксис, легкий для початківців	Більш складний через типи і додаткові можливості
Інтелектуальні функції	Обмежені можливості автодоповнення	Розвинуті функції автодоповнення і перевірка типів на етапі розробки
Інструменти розробки	Підтримка у всіх популярних редакторах	Підтримка у редакторах з TypeScript
Масштабованість проектів	Підходить для невеликих і середніх проектів	Краще підходить для великих, складних проектів
Гнучкість та простота	Висока	Строгіша, з меншою «свободою» для розробника
Популярність використання	Дуже висока	Набирає популярність, але потребує додаткових інструментів

Як видно з результатів порівняння у таблиці 3.1, JavaScript має перевагу над мовою програмування TypeScript. Тому було обрано мову JavaScript через її простоту, гнучкість, прямолінійну інтеграцію у HTML-документи та широку підтримку у всіх сучасних браузерах без необхідності додаткових компіляторів чи конфігурацій.

Для розробки програмного забезпечення було використано бібліотеку PHPMailer – це популярна бібліотека для надсилання електронних листів у PHP. Вона дозволяє надсилати листи без використання локального поштового сервера, додає файли до листів і підтримує різні методи автентифікації [33].

Також було використано Bootstrap – фреймворк для прискорення розробки адаптивного дизайну та забезпечення кросбраузерної сумісності [34].

3.2 Обґрунтування вибору мови програмування для управління реляційною базою даних

У процесі розробки програмного забезпечення, зокрема WEB-ресурсів, вибір мови або засобу для організації та доступу до бази даних є критично важливим рішенням. База даних – це основа, на якій зберігається й обробляється ключова інформація, необхідна для функціонування застосунку. Її ефективність, швидкодія, безпека та масштабованість безпосередньо залежать від того, яка мова або технологія для роботи з базами даних була обрана.

Розглянемо мови програмування для управління організованою базою даних OQL (Object Query Language) та SQL (Structured Query Language).

SQL – це стандартна мова для управління реляційними базами даних. Вона дозволяє створювати, змінювати, видаляти та запитувати дані, що зберігаються у таблицях. SQL є декларативною мовою, тобто користувач описує, що потрібно отримати, а не як це зробити.

OQL – це мова запитів, розроблена для об’єктно-орієнтованих баз даних. Вона була створена Object Data Management Group (ODMG) як аналог SQL для об’єктних моделей даних. OQL дозволяє виконувати запити до об’єктів, їхніх

властивостей та зв’язків, підтримуючи вкладені об’єкти та складні структури даних. Проте під час використання методів, реалізованими мовою програмування OQL потрібно бути обережним, щоб не допустити мутацію даних та побічні ефекти. OQL є декларативним, тому порядок виконання виразів не завжди передбачуваний [35].

У таблиці 3.2 наведено порівняння основних характеристик SQL (Structured Query Language) та OQL (Object Query Language).

Таблиця 3.2 – Порівняння основних характеристик SQL та OQL.

Критерій	SQL	OQL
Тип бази даних	Реляційна	Об’єктно-орієнтована
Стандартизація	ANSI/ISO	ODMG (Object Data Management Group)
Синтаксис	Декларативний, з фіксованими командами (SELECT, INSERT тощо)	Подібний до SQL, але з підтримкою об’єктних концепцій та вкладених запитів
Підтримка вкладених структур	Обмежена	Повна підтримка вкладених об’єктів та їхніх властивостей
Використання у практиці	Широко використовується в різних сферах	Обмежене використання
Сумісність з мовами програмування	Широка підтримка (Java, Python, C# тощо)	В основному використовується в об’єктно-орієнтованих середовищах

Вибір між SQL і OQL визначається вимогами конкретного проєкту. Якщо головне – робота з реляційними базами даних, а також необхідність простоти й ефективності, то SQL стане оптимальним рішенням. Тому було прийнято рішення використовувати саме SQL.

3.3 Обґрунтування вибору середовища розробки

У процесі розробки програмного забезпечення однією з важливих складових є вибір середовища програмування, у якому буде реалізовуватися

логіка та інтерфейс майбутнього застосунку. Від цього вибору значною мірою залежать зручність розробки, швидкість виконання задач, можливості автоматизації, якість коду та ефективність командної роботи. Правильно підібране середовище розробки дозволяє значно зменшити час на розв'язання рутинних завдань, а також сприяє підтриманню високої якості коду завдяки автоматичній перевірці стилю, підказкам щодо помилок і рекомендаціям щодо оптимізації.

Інтегроване середовище розробки (IDE – Integrated Development Environment) – це програмне забезпечення, яке об'єднує в собі всі необхідні інструменти для створення, редагування, компіляції, налагодження та тестування програмного коду в єдиному інтерфейсі. Метою IDE є підвищення продуктивності розробника шляхом спрощення та автоматизації різних аспектів процесу розробки програмного забезпечення [36].

Під час вибору середовища розробки було розглянуто такі додатки: Visual Studio Code (VS Code), IntelliJ IDEA, Atom.

Visual Studio Code – це безкоштовний, кросплатформовий редактор коду, розроблений компанією Microsoft. Він поєднує простоту текстового редактора з потужними інструментами для розробників. Він підтримує багато мов програмування, таких як JavaScript, Python, C++, Java, Go та інші. VS Code також містить вбудований відлагоджувач для JavaScript, Node.js та інших мов, що спрощує процес тестування і пошуку помилок [37].

IntelliJ IDEA – це потужне інтегроване середовище розробки від JetBrains, яке надає розробникам широкий набір інструментів для ефективної роботи з різними мовами програмування та технологіями. Спочатку створене для Java, воно з часом отримало підтримку таких мов, як Kotlin, Scala, Groovy, Python, Ruby, PHP, JavaScript, TypeScript, SQL, Go, HTML, CSS та Markdown. Підтримує сучасні фреймворки, включаючи React, Angular, Vue.js і Node.js для WEB-розробки. Окрім цього, середовище доступне для Windows, macOS та Linux [38].

Atom – це безкоштовний і відкритий редактор коду, який зосереджений на гнучкості та налаштуванні, створений з інтеграцією HTML, JavaScript, CSS та

Node.js. Він підтримує Windows, macOS і Linux, має вбудовану підтримку Git для управління версіями безпосередньо з редактора, а також величезну кількість пакетів і тем для розширення функціоналу. Atom використовує фреймворк Electron, що дозволяє створювати кросплатформені додатки за допомогою WEB-технологій [39].

У таблиці 3.3 наведено порівняння основних характеристик Visual Studio Code (VS Code), IntelliJ IDEA та Atom.

Таблиця 3.3 – Порівняння середовищ розробки Visual Studio Code (VS Code), IntelliJ IDEA та Atom

	Visual Studio Code (VS Code)	IntelliJ IDEA	Atom
Вартість	Безкоштовний	Платний	Безкоштовний
Швидкодія	Легкий і швидкий	Потужний, але важкий	Часто повільний
Споживання ресурсів	Легке, оптимізоване споживання ресурсів	Помірне, але неоптимізоване	Високе споживання пам’яті та CPU
Розширюваність	Велика кількість розширень	Через плагіни JetBrains	Великий вибір, але підтримка обмежена
Можливості	IntelliSense для багатьох мов, підтримка AI-розширень, Git-підтримка	Потужне IntelliSense, рефакторинг, Git-підтримка	Примітивне автодоповнення, базова Git-інтеграція

Згідно табл. 3.3, середовище розробки Visual Studio Code має перевагу над іншими середовищами розробки завдяки своїй безкоштовності, легкості, потужним інструментам та відмінній підтримці плагінів. Тому було прийнято рішення обрати середовище Visual Studio для розробки програмного додатку.

3.4 Розробка клієнтської та серверної частин

Для розробки використовуються мови програмування PHP, HTML, CSS і JavaScript, а також бібліотека PHPMailer для відправки email-повідомлень і фреймворк Bootstrap для спрощення стилізації та адаптивності інтерфейсу.

Початковим етапом розробки стало створення модульної структури проєкту. Основний файл index.php виконує роль вхідної точки, з якої запускається додаток. У цьому файлі підключаються та ініціалізуються головні класи: Application, Model, Translator.

```
require_once 'Application.php';
require_once 'Model.php';
require_once 'Translator.php';

Model::init();
Translator::init();
Application::run();
```

Далі викликається метод Application::run(), що відповідає за маршрутизацію.

Клас Application обробляє маршрутизацію залежно від URL-адреси та викликає відповідні шаблони сторінок, передаючи їм потрібні дані. Це дозволяє централізовано керувати навігацією та зручно додавати нові маршрути (наприклад, /menu, /review, /booking).

```
class Application {
    public static function run() {
        $route = parse_url($_SERVER['REQUEST_URI'],
PHP_URL_PATH);
        $data = [];
        $data['text'] = Translator::getCommonText();
        $data['lang'] = Translator::getActiveLang();
        $data['lang_url_postfix'] =
Translator::getLangPostfix();
        $data['route'] = $route;

        switch ($route) {
            case '/':
                $data['title'] =
$data['text']['page_home_title'];
                self::render('home', $data);
                break;
            case '/menu':
```

```
                    $data['title'] =
$data['text']['page_dish_title'];
                    $data['grouped_dishes'] = Model::getDishes();
                    self::render('menu', $data);
                    break;
                case '/booking':
                    $data['title'] =
$data['text']['page_booking_title'];
                    self::render('booking', $data);
                    break;
                default:
                    http_response_code(404);
                    echo "Сторінку не знайдено.";
            }
        }

        public static function render($view, $data) {
            extract($data);
            require "layout/header.php";
            require "views/$view.php";
            require "layout/footer.php";
        }
    }
```

Клас Model виконує функції моделі даних: зчитування, перевірка та зберігання JSON-файлу з інформацією про страви. Це дає змогу легко змінювати або оновлювати дані без використання повноцінної СУБД. При ініціалізації виконується перевірка цілісності файлу та його коректного формату.

```
    class Model {

        private static $dish_file = 'data/dishes.json';
        private static $dishes_data = [];

        public static function init() {
            if (!is_file(self::$dish_file)) {
                throw new Error('Файл з базою страв не знайдено');
            }

            $data = file_get_contents(self::$dish_file);
            self::$dishes_data = json_decode($data, true);

            if (json_last_error() !== JSON_ERROR_NONE) {
                throw new Error('Файл JSON пошкоджений');
            }
        }

        public static function getDishes() {
            return self::$dishes_data;
        }
    }
```

Особливу увагу приділено багатомовності інтерфейсу. Вона реалізована у класі Translator, який надає функції вибору мови користувача, завантаження відповідних текстів та формування URL-постфіксів. Це забезпечує гнучкість і зручність локалізації.

```
class Translator {
    private static $text = [];
    private static $active_lang = 'uk';

    public static function init() {
        self::$active_lang = $_GET['lang'] ?? 'uk';
        $file = "lang/" . self::$active_lang . ".json";
        self::$text = json_decode(file_get_contents($file),
true);
    }

    public static function getCommonText() {
        return self::$text;
    }

    public static function getActiveLang() {
        return self::$active_lang;
    }

    public static function getLangPostfix() {
        return self::$active_lang === 'uk' ? '' : '?lang=' .
self::$active_lang;
    }
}
```

Для обробки бронювань впроваджено функціонал відправки електронних листів адміністратору. Для цього інтегровано бібліотеку PHPMailer, яка є надійним та гнучким засобом відправки email через SMTP. Клас Mailer інкапсулює логіку підключення до поштового сервера та відправки листів із потрібними параметрами. Це значно підвищує надійність у порівнянні зі стандартною функцією mail() у PHP.

```
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

require 'vendor/autoload.php'; // Підключення PHPMailer через
Composer

class Mailer {
    public static function send($to, $subject, $message, $from
= 'noreply@example.com') {
```

```
            $mail = new PHPMailer(true);
            try {
                $mail->isSMTP();
                $mail->Host = 'smtp.example.com';
                $mail->SMTPAuth = true;
                $mail->Username = 'your_username';
                $mail->Password = 'your_password';
                $mail->SMTPSecure                                    =
PHPMailer::ENCRYPTION_STARTTLS;
                $mail->Port = 587;

                $mail->setFrom($from, 'Restaurant Dolce Vita');
                $mail->addAddress($to);
                $mail->Subject = $subject;
                $mail->Body = $message;
                $mail->CharSet = 'UTF-8';

                return $mail->send();
            } catch (Exception $e) {
                error_log("Mailer Error: {$mail->ErrorInfo}");
                return false;
            }
        }
    }
```

Цей код у Mailer.php використовується в booking_form_handler.php для відправки повідомлення адміністратору:

```
    require_once 'Mailer.php';

    $name = $_POST['name'];
    $date = $_POST['date'];
    $admin_email = 'dianademiuk13@gmail.com';
    $from_email = 'myhost1@web797.default-host.net';

    $message = "$name зробив(-ла) нове бронювання на $date";
    $status   =   Mailer::send($admin_email,   'Нове   бронювання',
$message, $from_email);
```

Клієнтська частина реалізована з використанням HTML, CSS, JavaScript та фреймворку Bootstrap. HTML-структура сторінок побудована модульно: повторювані елементи (шапка, футер, меню) винесено в окремі шаблони (layout/header.php, layout/footer.php) для повторного використання.

CSS-оформлення базується на Bootstrap 5, який дозволяє швидко створювати адаптивні компоненти (навігаційне меню, форми, таблиці тощо) та забезпечує кросбраузерну сумісність. Підключення Bootstrap відбувається через

CDN, що скорочує час завантаження та спрощує підтримку актуальної версії бібліотеки. Додаткове стилізування реалізовано у файлі assets/style.css, а взаємодія з користувачем (наприклад, обробка форм, зміна мови інтерфейсу, динамічна підсвітка активних пунктів меню) – через файл assets/script.js.

Клієнтська частина також забезпечує адаптивність інтерфейсу – сайт коректно відображається на пристроях із різною шириною екрану (смартфони, планшети, десктопи), що є важливою вимогою до сучасних WEB-ресурсів. Для зручності користувача реалізовано вибір мови за допомогою динамічного елемента <select>. Після вибору мови змінюється URL і відповідна локалізована версія інтерфейсу.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1, shrink-to-fit=no">
    <title><?=$title ?></title>
    <!-- Підключення Bootstrap CSS -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootst
rap.min.css" rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
    <!-- Підключення власних стилів -->
    <link rel="stylesheet" href="/assets/style.css?v3">
    <script src="/assets/script.js?v3"></script>
</head>
<body>
<header class="header">
    <div class="container">
        <nav class="navbar navbar-expand-lg navbar-dark bg-
dark">
            <div class="container-fluid">
                <a href="/<?=$lang_url_postfix?>"
class="navbar-brand header__logo">
                    <span class="header__logo-
text"><?=$text['restaurant']?></span><span class="header__logo-
logo">Dolce Vita</span>
                </a>
                <button class="navbar-toggler" type="button"
data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-
controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
                    <span class="navbar-toggler-icon"></span>
```

```
                    </button>
                    <div class="collapse navbar-collapse"
id="navbarNav">
                        <ul class="navbar-nav header__menu">
                            <li class="nav-item
header__menu__item">
                                <a class="nav-link <?=$route ==
'/' ? 'active' : ''?>"
href="/<?=$lang_url_postfix?>"><?=$text['menu_main']?></a>
                            </li>
                            <li class="nav-item
header__menu__item">
                                <a class="nav-link <?=$route ==
'/menu' ? 'active' : ''?>"
href="/menu<?=$lang_url_postfix?>"><?=$text['menu_dish']?></a>
                            </li>
                            <li class="nav-item
header__menu__item">
                                <a class="nav-link <?=$route ==
'/review' ? 'active' : ''?>"
href="/review<?=$lang_url_postfix?>"><?=$text['menu_review']?></a>
                            </li>
                            <li class="nav-item
header__menu__item">
                                <a class="nav-link <?=$route ==
'/booking' ? 'active' : ''?>"
href="/booking<?=$lang_url_postfix?>"><?=$text['menu_booking']?></
a>
                            </li>
                        </ul>
                        <div class="header__lang ms-auto">
                            <select data-lang="<?=$lang?>"
class="form-select form-select-sm">
                                <option value="uk" <?=$lang ===
'uk' ? 'selected="selected"' : ''?>>UA</option>
                                <option value="en" <?=$lang ===
'en' ? 'selected="selected"' : ''?>>EN</option>
                            </select>
                        </div>
                    </div>
                </div>
            </nav>
        </div>
    </header>
```

Для коректної роботи Bootstrap у layout/footer.php додано підключення JavaScript-файлу Bootstrap:

```
    <footer>
        <div class="container">
            <p>© <?= date('Y') ?> <?=$text['restaurant']?> "Dolce
Vita"</p>
        </div>
    </footer>
```

```
    <!-- Підключення Bootstrap JS -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstra
p.bundle.min.js" integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
    </body>
    </html>
```

Важливою частиною є взаємодія між клієнтською та серверною частинами. Користувач, заповнюючи форму бронювання, надсилає дані на сервер, де вони обробляються у файлі booking_form_handler.php. У разі успішної обробки та відправки листа – формується повідомлення про підтвердження. Форма перевіряється як на клієнтському рівні (заповнення полів), так і на серверному (наявність даних, перевірка формату). Це дозволяє підвищити безпеку та надійність системи.

3.5 Тестування роботи програми та аналіз результатів

Тестування є критично етапом розробки WEB-ресурсу, оскільки дозволяє виявити та усунути помилки до введення сайту в експлуатацію. Аналіз результатів тестування допомагає чітко зафіксувати, які саме функції або компоненти працюють некоректно, за яких умов виникають збої та які частини системи потребують доопрацювання.

Проведемо комплексне тестування функціональності сайту для виявлення потенційних помилок та перевірки відповідності функціоналу вимогам. Тестування включає в себе багато різних напрямів. Для забезпечення якості та надійності розробленого WEB-ресурсу проведемо комплексне тестування, що включає наступні види:

1. Функціональне тестування – перевірка роботи основних функцій сайту, таких як навігація, відображення сторінок, кнопок, форм зворотного зв’язку та меню.

Критерії тестування:

- форми приймають та обробляють введені дані;

- кнопки виконують відповідні дії при натисканні;
- всі сторінки завантажуються без помилок.

2. UI/UX тестування – оцінка візуального оформлення сайту, правильності відображення контенту, адаптації елементів інтерфейсу.

Критерії тестування:

- елементи інтерфейсу відображаються коректно на різних розширеннях екрана;
- текст читається легко, шрифти та кольори узгоджені;
- зображення не спотворені та мають відповідну якість.

3. Крос-браузерне тестування – перевірка коректності роботи сайту в найпоширеніших браузерах: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari.

Критерії тестування:

- WEB-сайт коректно відображається в браузерах (Microsoft Edge, Safari);
- функціональність зберігається незалежно від браузера.

4. Тестування адаптивності – перевірка коректного відображення сайту на пристроях з різними розмірами екранів (мобільні телефони, планшети, десктопи).

Критерії тестування:

- WEB-сайт коректно відображається на мобільних пристроях;
- меню та інші елементи інтерфейсу адаптуються до розміру екрана;
- всі функції доступні на мобільних пристроях [40].

Для початку спробуємо запустити WEB-сайт на різних браузерах, а саме Microsoft Edge та Safari. На рис. 3.1 зображено коректне функціонування WEB-сайту в браузері Microsoft Edge.

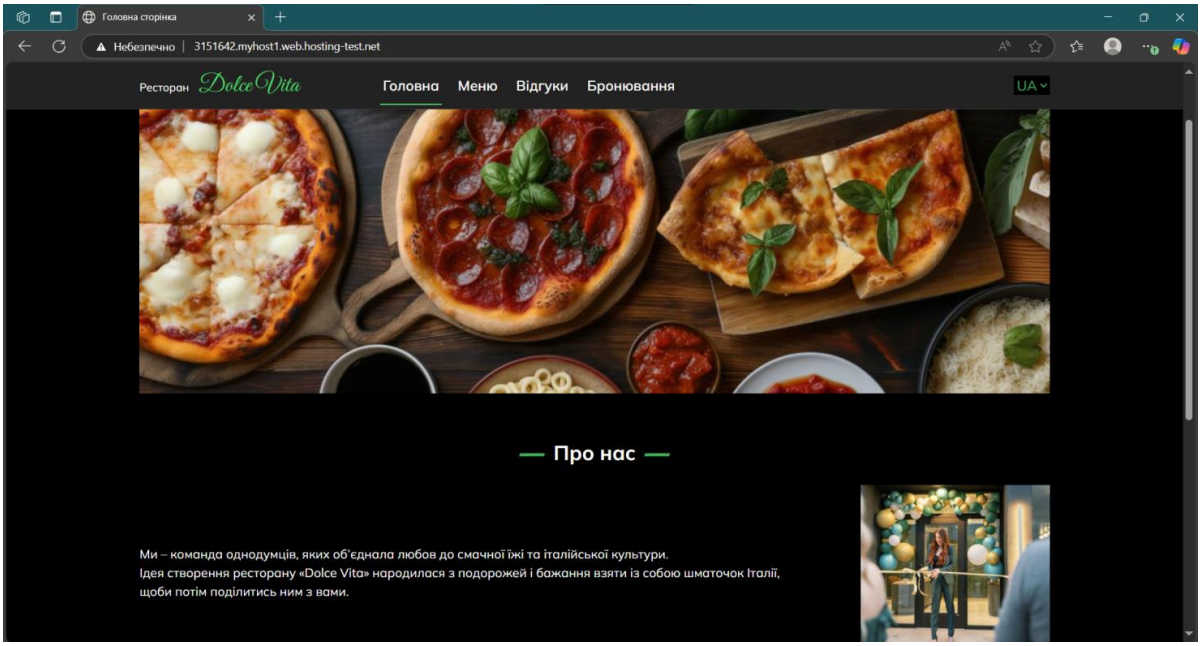

Рисунок 3.1 – Зображення роботи WEB-сайту в браузері Microsoft Edge

На рис. 3.2 зображено роботу WEB-сайту в браузері Safari. WEB-сайт коректно відображається в основних браузерах (Microsoft Edge, Safari). Функціональність зберігається незалежно від браузера, що відповідає вимогам.

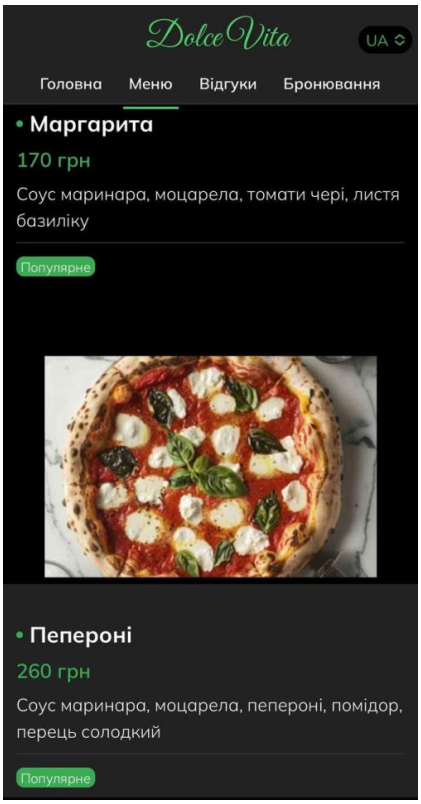

Рисунок 3.2 – Зображення роботи WEB-сайту в браузері Safari

Після запуску відкривається Головна сторінка (рис. 3.3). Звідси також можна перейти на сторінки Меню, Відгуки та Бронювання.

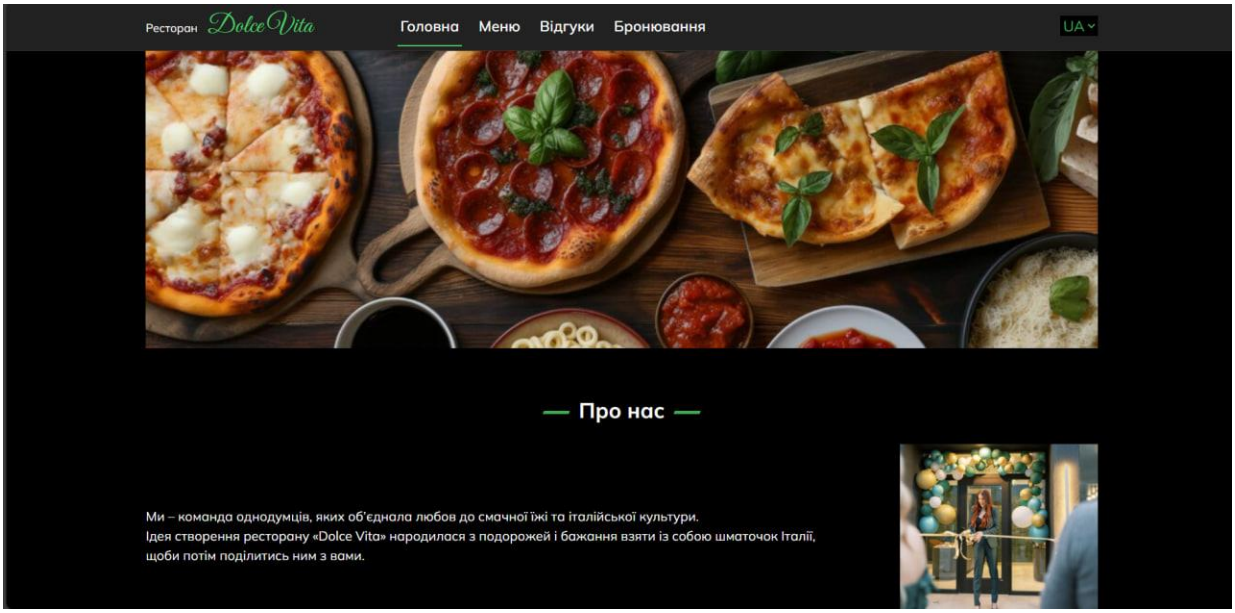

Рисунок 3.3 – Інтерфейс головної сторінки WEB-ресурсу «Ресторан»

Головна сторінка містить на панелі кнопки «Головна», «Меню», «Відгуки», «Бронювання», що відсилатимуть користувача на відповідні сторінки, і кнопку зміни мови інтерфейсу. Також містить в собі коротку інформацію про ресторан (рис. 3.4) та контактні дані (рис. 3.5).

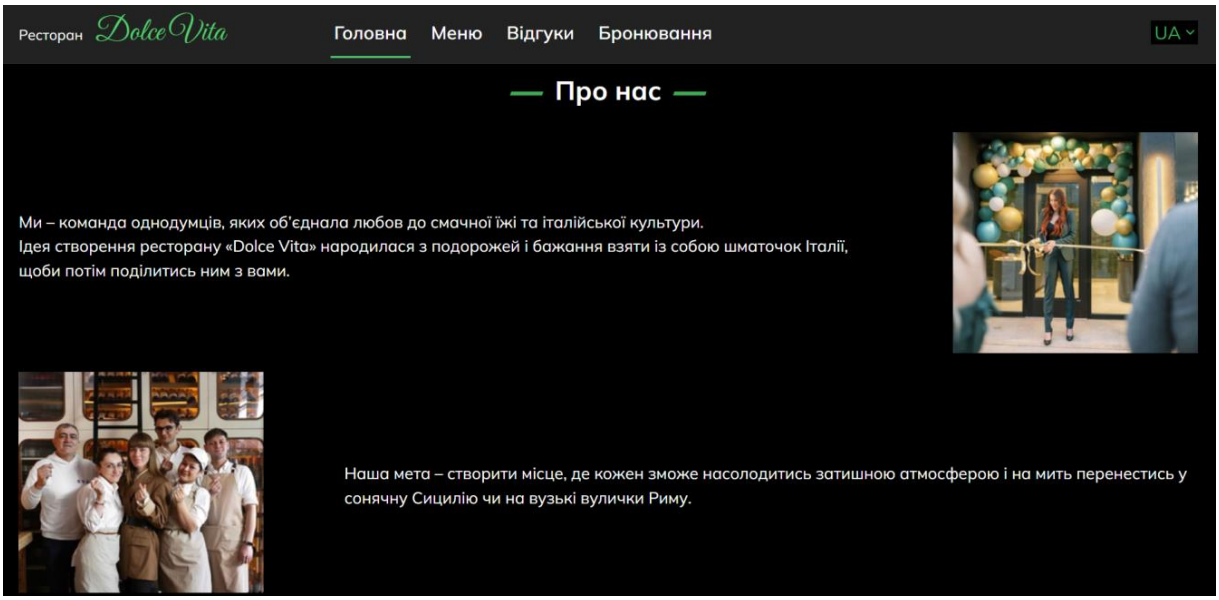

Рисунок 3.4 – Коротка інформація про ресторан

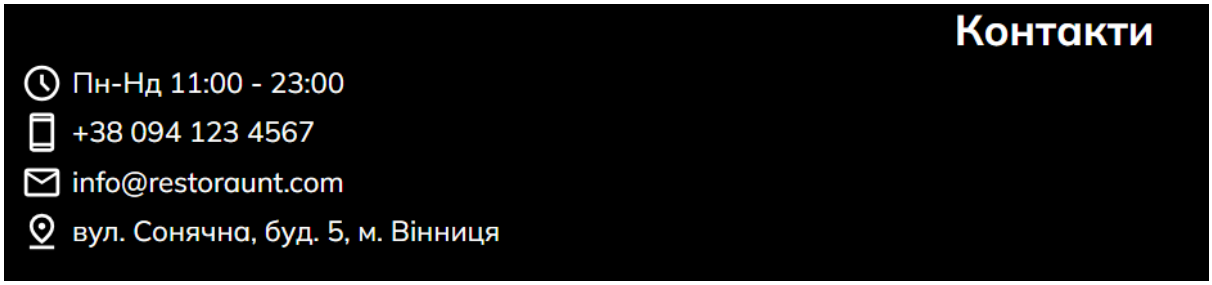

Рисунок 3.5 – Секція з контактною інформацією

Секція з контактною інформацією містить в собі наглядну інформацію про розклад роботи ресторану, адресу, номер телефону та email для зв’язку з адміністрацією.

На рис. 3.6 відображено коректне функціонування сайту на мобільному пристрої.

Рисунок 3.6 – Інтерфейс головної сторінки WEB-ресурсу на мобільному пристрої

Кнопка для зміни мови інтерфейсу також коректно працює як на мобільному пристрої (рис. 3.7), та і на ПК (рис 3.8).

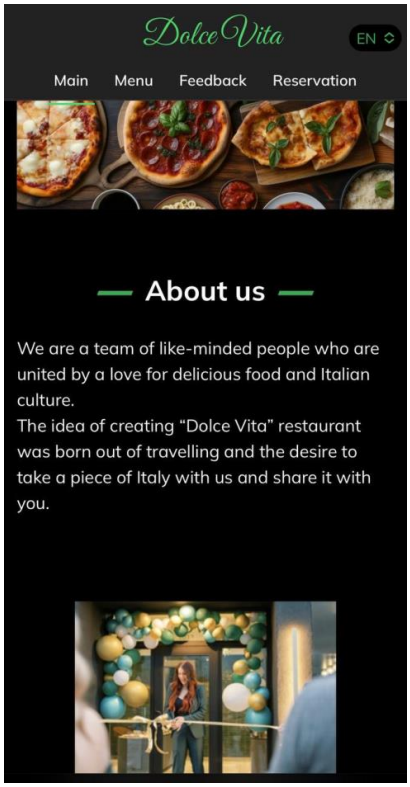

Рисунок 3.7 – Головна сторінка WEB-ресурсу на мобільному пристрої при зміні мови інтерфейсу на англійську

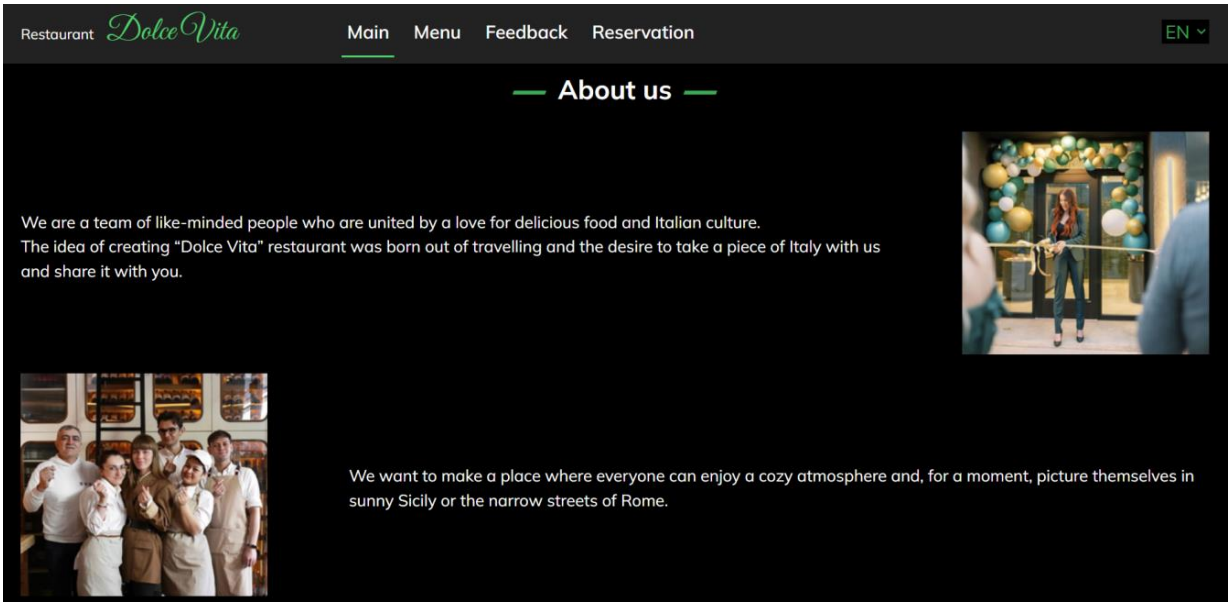

Рисунок 3.8 – Коротка інформація про ресторан при зміні мови інтерфейсу на англійську

Сторінка «Меню» (рис. 3.9) містить в собі різноманітні категорії страв з назвами страв, ціною, вагою, описом, та відповідними зображеннями.

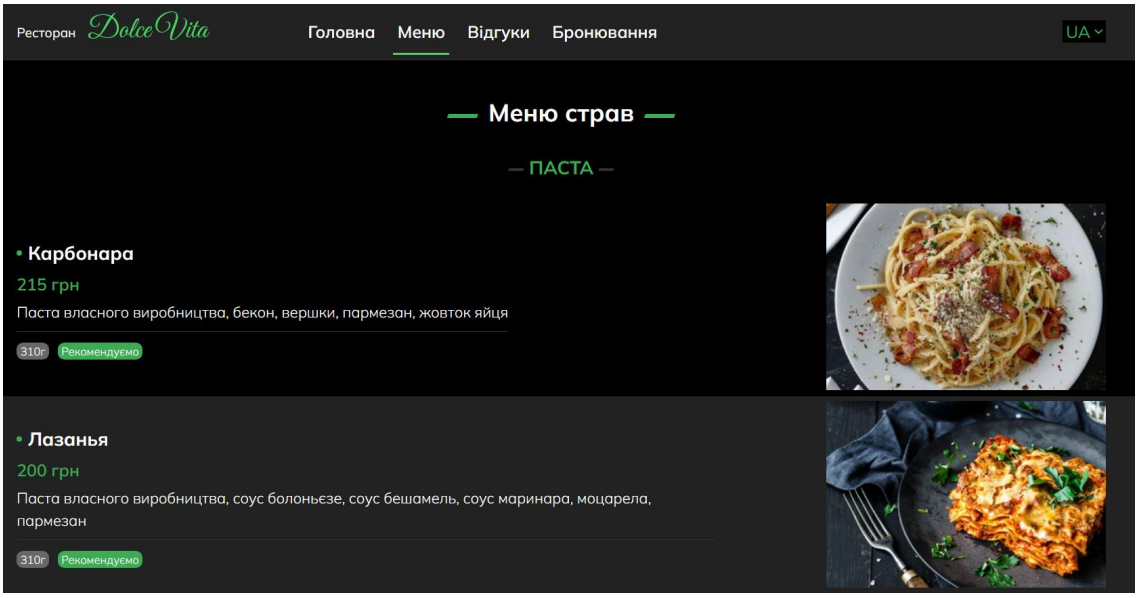

Рисунок 3.9 – Інтерфейс сторінки «Меню»

Весь графічний матеріал відображається чітко без втрат якості, в тому числі й на мобільному пристрої (рис. 3.10).

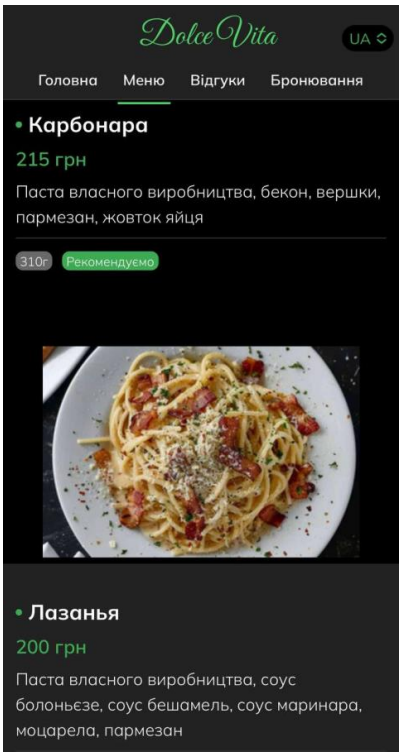

Рисунок 3.10 – Інтерфейс сторінки «Меню» на мобільному пристрої

Сторінка «Відгуки» (рис.3.11) містить форму для залишення відгуку на сайті. У форму можна вписати ім’я, сам відгук та залишити оцінку.

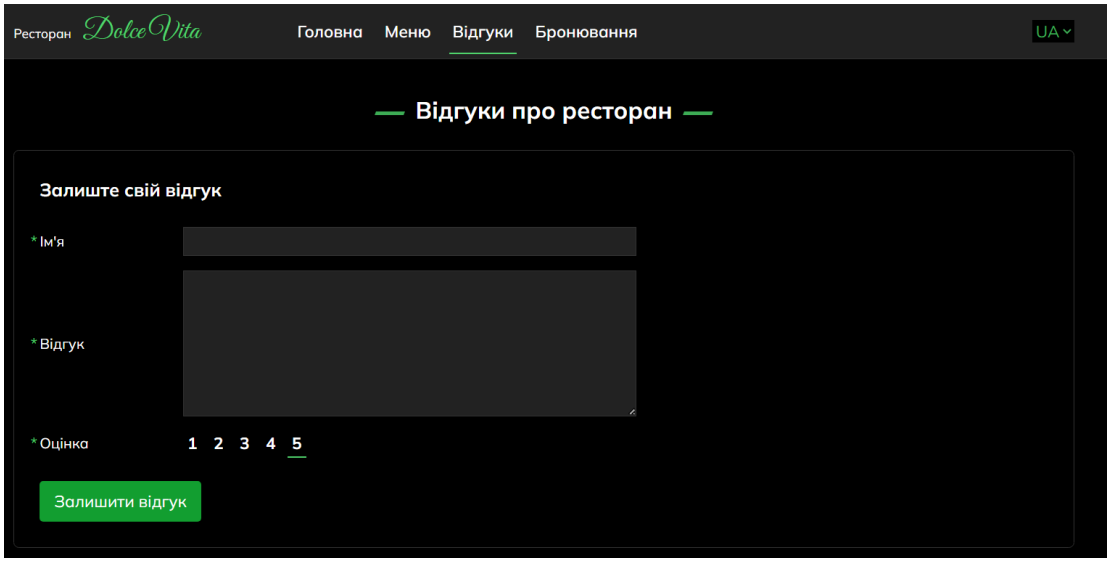

Рисунок 3.11 – Інтерфейс сторінки «Відгуки»

На рис. 3.12 зображено сторінку «Відгуки» на мобільному пристрої, на якій форма також коректно відображається.

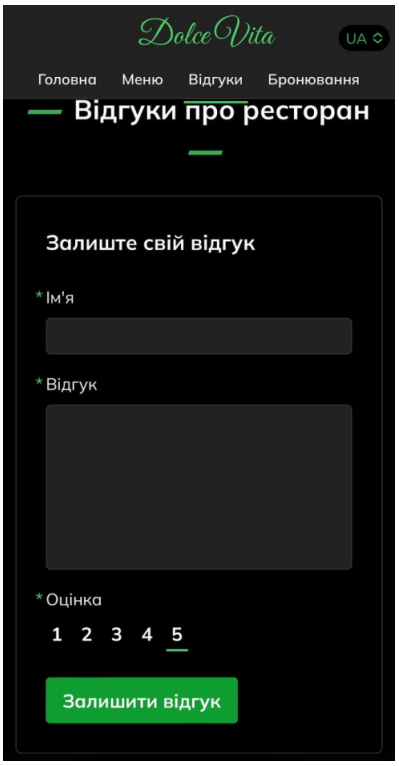

Рисунок 3.12 – Інтерфейс сторінки «Відгуки» на мобільному пристрої

Спробуємо залишити відгук, для цього заповнимо форму (рис. 3.13). Відгук відображається на сайті (рис. 3.14), отже форма коректно працює.

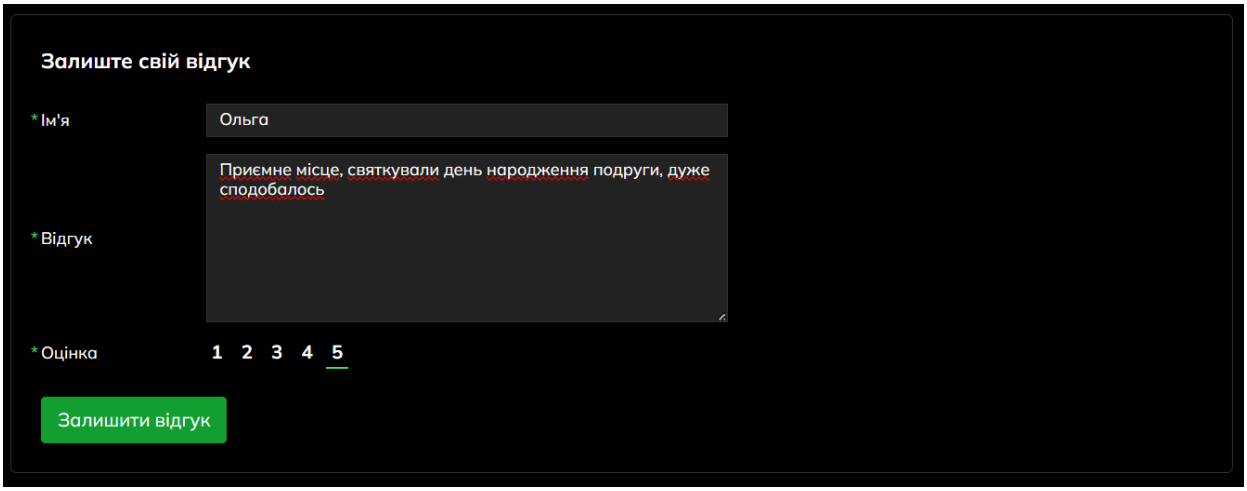

Рисунок 3.13 – Заповнена форма для відгуків

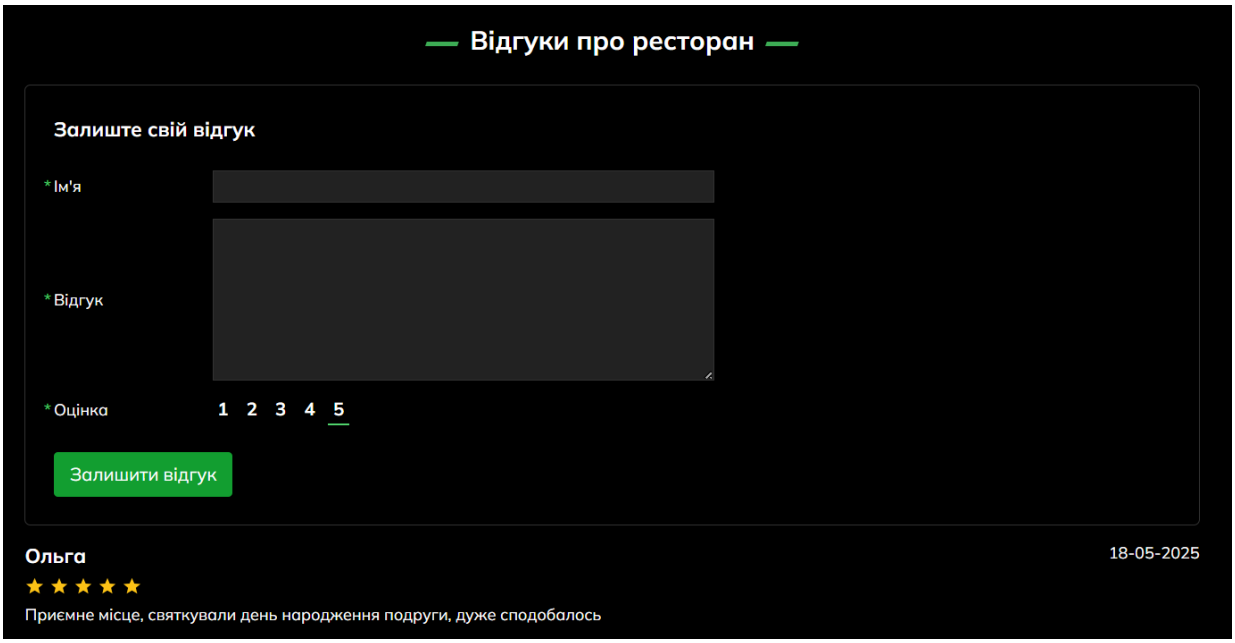

Рисунок 3.14 – Секція з відгуками

Перейдемо до сторінки «Бронювання» (рис. 3.15). На ній міститься форма для бронювання (рис. 3.16) і вказана інформація для користувачів щодо заповнення форми бронювання. Форма містить обов’язкові для заповнення поля «ім’я», «номер телефону», «дата», «час», «кількість людей», а також поля «електронна пошта» і «додаткова інформація», які не є обов’язковими.

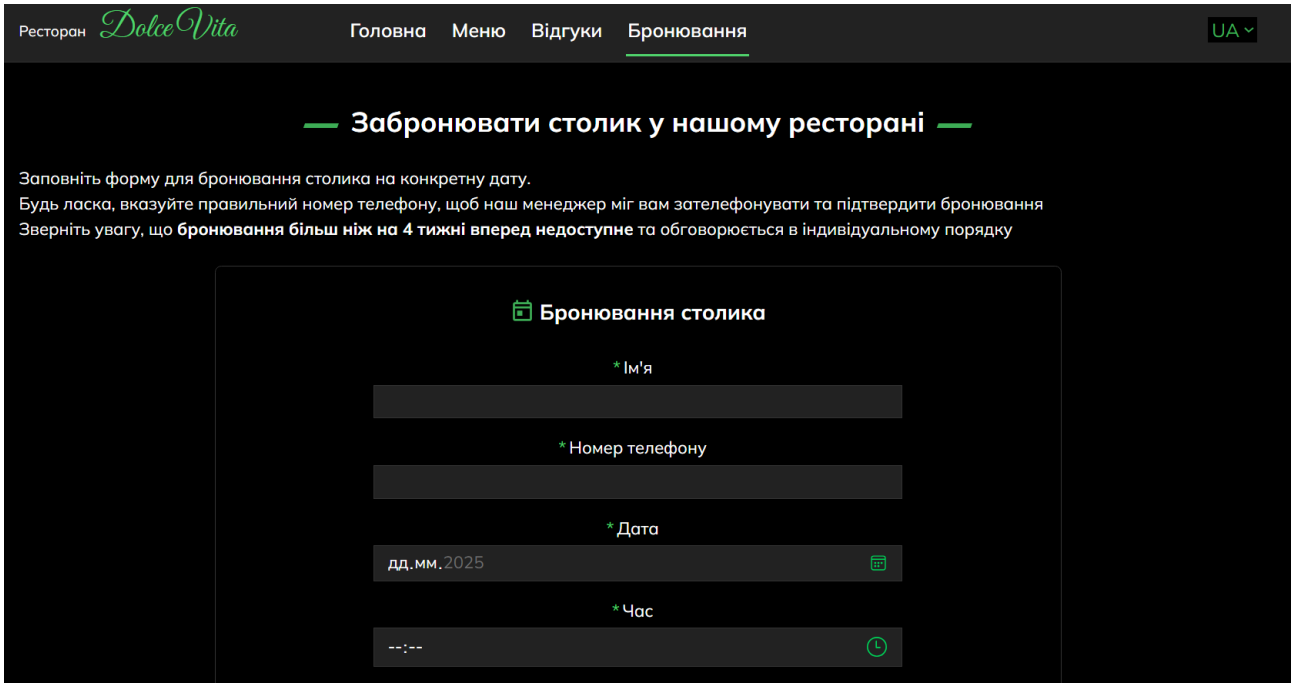

Рисунок 3.15 – Інтерфейс сторінки «Бронювання»

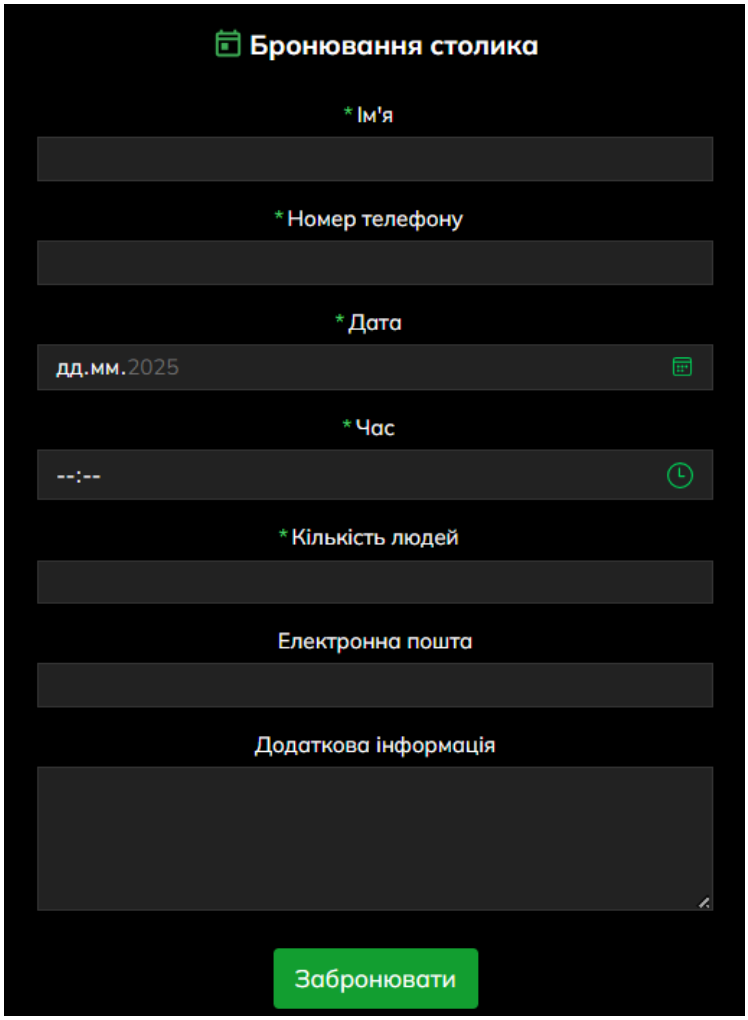

Рисунок 3.16 – Форма для бронювання

Також форма коректно відображається на мобільному пристрої, що зображено на рис. 3.17.

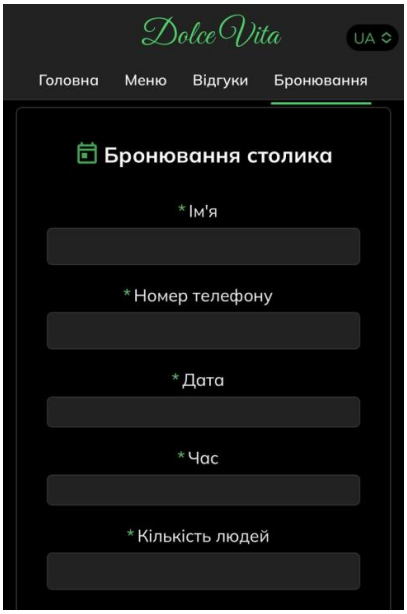

Рисунок 3.17 – Форма для бронювання на мобільному пристрої

Перевіримо коректність роботи системи бронювання. Для початку заповнимо форму (рис. 3.18) та натиснемо кнопку «Забронювати».

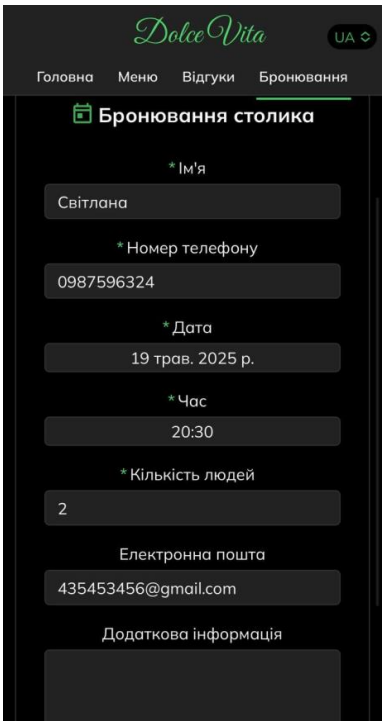

Рисунок 3.18 – Заповнена форма для бронювання на мобільному пристрої

В усі поля можна внести інформацію, кнопка працює. Після бронювання на пошту адміністрації надходить повідомлення про здійснення нового бронювання з інформацією з форми (рис. 3.19).

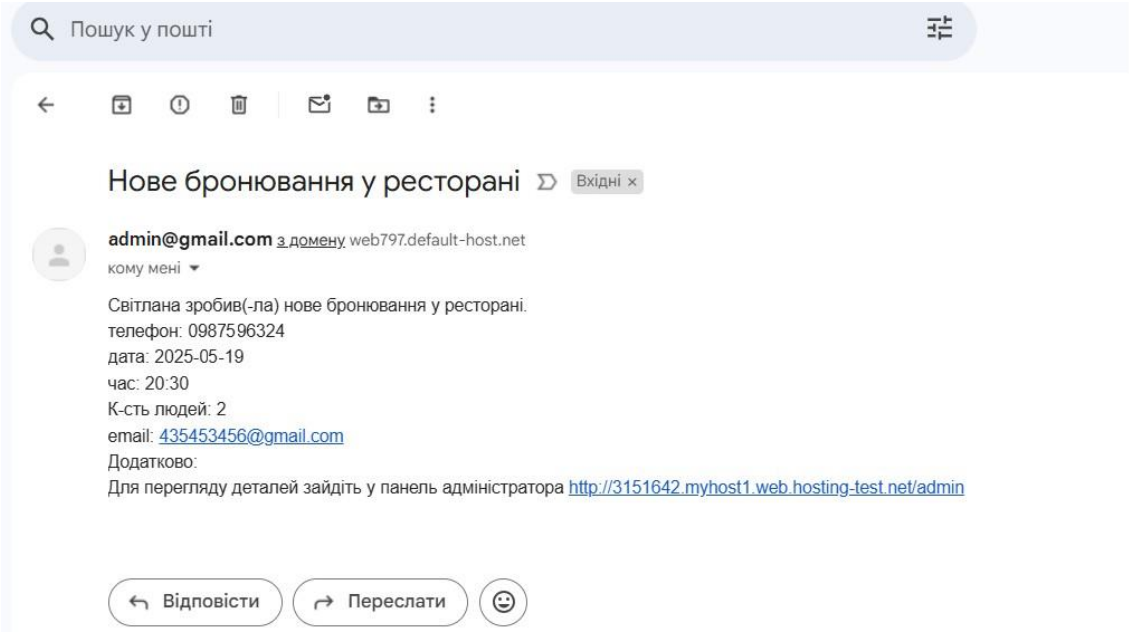

Рисунок 3.19 – Повідомлення про нове бронювання на пошті адміністрації

На рис. 3.20 зображено форму для авторизації адміністратора, з полями для вводу логіну та паролю.

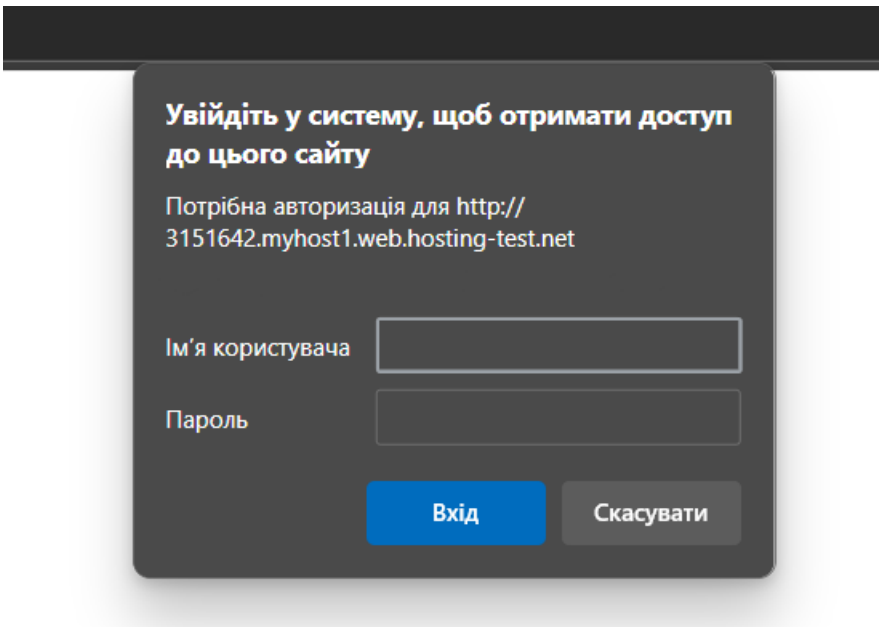

Рисунок 3.20 – Форма для входу на сторінку адміністратора

Результати тестування:

- сайт завантажується стабільно у всіх протестованих браузерах;
- основні функції сайту працюють коректно: меню, навігація між сторінками, форми і кнопки;
- зображення та текст відображаються чітко, усі елементи адаптуються до розміру екрана;
- адаптивна верстка працює коректно як на ПК, так і на мобільних пристроях;
- жодної критичної помилки або збоїв під час користування сайтом не виявлено.

Отже, розроблений WEB-ресурс успішно пройшов тестування та відповідає всім заданим вимогам.

3.6 Висновок

В третьому розділі було обрано мови програмування для розробки клієнтської частини – CSS, HTML, та для серверної – JavaScript і PHP.

Було проаналізовано мови програмування для управління організованою базою даних – SQL та OQL. В результаті аналізу обрано SQL, оскільки саме SQL призначений для роботи з реляційною базою даних.

Як середовище розробки обрано Visual Studio Code завдяки значним перевагам над іншими середовищами, а саме Visual Studio Code має відкритий вихідний код, простий, швидкий та зручний, має інтелектуальне автозавершення, відлагоджувач вбудований в термінал, підтримка Git та багато розширень.

Було проведено тестування WEB-ресурсу «Ресторан». Результати тестування свідчать про те, що розроблений WEB-ресурс є технічно стабільним, функціональним та зручним у використанні. Сайт повністю відповідає поставленим функціональним вимогам і готовий до подальшого використання.

ВИСНОВКИ

Всі задачі, поставлені для реалізації WEB-ресурсу «Ресторан» були виконані в повному обсязі, а саме: обґрунтована доцільність розробки WEB-ресурсу «Ресторан», враховуючи сучасні тенденції розвитку ресторанного бізнесу; обрано архітектуру для проектування WEB-ресурсу «Ресторан»; проаналізовано та обґрунтовано вибір інструментів для розробки; виконано програмну реалізацію клієнтської та серверної частин WEB-ресурсу «Ресторан»; виконано тестування WEB-ресурсу «Ресторан» та проведено аналіз отриманих результатів.

Під час виконання бакалаврської кваліфікаційної роботи було проведено комплексний аналіз галузі ресторанного бізнесу та можливостей використання WEB-ресурсів для автоматизації, просування та покращення обслуговування клієнтів. Встановлено, що більшість наявних сайтів ресторанів мають низку недоліків, зокрема обмежену функціональність, відсутність інтерактивності, слабку адаптацію до мобільних пристроїв, а також недостатню зручність користування.

Було спроектовано структуру WEB-ресурсу, для візуалізації якої було створено UML-діаграму класів, UML-діаграму прецедентів та ER-діаграму, також було розроблено схему алгоритму роботи сайту і модуля бронювання з використанням блок-схем.

У ході роботи було обрано архітектуру MVC, детально розглянуто її структуру, описано ключові переваги її використання для розробки WEB-ресурсу «Ресторан», що дозволило забезпечити логічне розділення компонентів системи, а також підвищити загальну гнучкість і масштабованість програмного рішення.

Крім того, було проведено аналіз вибору інструментів технічної реалізації, а саме розглянуто переваги використання PHP, CSS, HTML і на основі порівняльного аналізу було обрано JavaScript для фронтенду та SQL для

управління реляційною структурою даних. Також для розробки програмного забезпечення було обрано бібліотеки PHPMailer та Bootstrap.

Як середовище розробки було використано Visual Studio Code, що забезпечило зручність розробки завдяки широкому набору розширень, підтримці автозаповнення, налагодження, інтеграції з Git та високій швидкодії.

Здійснено тестування WEB-ресурсу «Ресторан», зокрема перевірено коректність роботи ключових функцій: здійснення бронювання та залишення відгуків на сайті. Результати тестування показали що розроблений WEB-ресурс відповідає поставленим функціональним вимогам.

Мета роботи, яка полягала в розширенні функціональних можливостей WEB-ресурсу «Ресторан», досягнута за рахунок створення функціоналу для перегляду меню із графічним поданням страв, онлайн-бронювання, залишення відгуків на сайті, адаптивності інтерфейсу до різних типів пристроїв, підтримки багатомовності та зручної навігація з орієнтацією на користувацький досвід.

Робота виконана у повному обсязі відповідно до поставлених цілей, вимог та методичних рекомендацій щодо підготовки бакалаврських кваліфікаційних робіт. Усі етапи були реалізовані належним чином, що підтверджує якість, завершеність і практичну цінність розробленого рішення [41].

Таким чином, розроблений WEB-ресурс повністю задовольняє актуальні вимоги користувачів, є зручним у використанні як для клієнтів, так і для працівників ресторану, а також має потенціал для подальшого масштабування та інтеграції з іншими сервісами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Демюк Д. Я., Белзецький Р. С. Аналіз передумов розробки WEB-ресурсу «Ресторан». *Молодь в науці: дослідження, проблеми, перспективи (МН-2025)* URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25645/21175 (дата звернення: 17.06.2025)

2. State of the Restaurant Industry 2025. URL: https://restaurant.org/research-and-media/research/research-reports/state-of-the-industry/ (дата звернення: 05.05.2025).

3. Serebryakova N.A., Avdeev I.V., “The content of structural transformations of the region's economy, adequate to the requirements of digitalization”, Proc. of the Voronezh State Univer. of Engineer. Technol. 2018, T. 80, №. 4, P. 408–412. URL: https://doi.org/10.20914/2310-1202-2018-4-408-412 (дата звернення: 06.05.2025).

4. Stolterman, E., Fors, A.C. Information Technology and the Good Life. In: Kaplan, B., Truex, D.P., Wastell, D., Wood-Harper, A.T., DeGross, J.I. (eds) Information Systems Research. IFIP International Federation for Information Processing. 2004. Vol 143. Springer, Boston, MA. https://doi.org/10.1007/1-4020-8095-6_45 (дата звернення: 06.05.2025).

5. Annual number of individuals using the internet worldwide between 2015 and 2025. URL: https://www.statista.com/statistics/1560762/global-internet-users-annual-number/ (дата звернення: 09.05.2025).

6. Ресторанна індустрія. URL: https://uk.dir.gg/list/restaurant-industry/ (дата звернення: 09.05.2025).

7. «За двома зайцями». URL: https://zaytsi.com.ua (дата звернення: 12.05.2025).

8. «EastWest Connection». URL: https://eastwest-connection.com (дата звернення: 12.05.2025).

9. «Plates». URL: https://plates-london.com/plant-based-restaurant-london/ (дата звернення: 12.05.2025).

10. Udo, G. J., Marquis, G. P. Factors Affecting E-Commerce Web Site Effectiveness. Journal of Computer Information Systems. 2002. Vol. 42, P. 10-16. URL: https://doi.org/10.1080/08874417.2002.11647481 (дата звернення: 13.05.2025).

11. Gehrke D., Turban E., "Determinants of successful Website design: relative importance and recommendations for effectiveness". Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences. 1999. Vol. 32, P. 8. URL: https://doi.org/10.1109/HICSS.1999.772943 (дата звернення: 14.05.2025).

12. Hentati A., Forsell E., Ljótsson B., Kaldo V., Lindefors N., Kraepelien, M. (2021). The effect of user interface on treatment engagement in a self-guided digital problem-solving intervention: A randomized controlled trial. Internet Interventions. 2021. Vol 26. https://doi.org/10.1016/j.invent.2021.100448. (дата звернення: 14.05.2025).

13. Архітектура програмного забезпечення: все що треба знати. URL: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya (дата звернення: 17.05.2025).

14. MVC Design Pattern. URL: https://www.geeksforgeeks.org/mvc-design-pattern/ (дата звернення: 17.05.2025).

15. Notes and Historical documents from Trygve Reenskaug, inventor of MVC. URL: https://folk.universitetetioslo.no/trygver/themes/mvc/mvc-index.html (дата звернення: 17.05.2025).

16. Концепція mvc для чайників. URL: https://yak.koshachek.com/articles/koncepcija-mvc-dlja-chajnikiv.html (дата звернення: 17.05.2025).

17. Six Benefits of Using MVC Model for Effective Web Application Development. URL: https://www.brainvire.com/six-benefits-of-using-mvc-model-for-effective-web-application-development/ (дата звернення: 17.05.2025).

18. Що таке ООП. URL: https://itpraktik.com/shho-take-oop-nazvit-jogo-principi-z-prikladami/ (дата звернення: 18.05.2025).

19. Класи ООП. Об’єктно-орієнтоване програмування. URL: https://what.com.ua/klasi-oop-obiektno-oriientovan/ (дата звернення: 18.05.2025).

20. ООП – Об’єктно-орієнтоване програмування – Що це таке. URL: https://javascript.org.ua/oop-ob%d1%94ktno-ori%d1%94ntovane-programuvannya-shho-cze-take/ (дата звернення: 18.05.2025).

21. Класи в програмуванні: занурення в об’єктно-орієнтоване програмування. URL: https://foxminded.ua/klasy-v-prohramuvanni (дата звернення: 18.05.2025).

22. В. А. Каплун, Ю. В. Баришев, А. В. Остапенко. Технологія програмування. Лабораторний практикум: навчальний посібник. Вінниця: ВНТУ, 2015. 125 с.

23. UML для бізнес-моделювання: для чого потрібні діаграми процесів. URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення: 18.05.2025).

24. Побудова та розуміння алгоритмів: крок за кроком для новачків. URL: https://cloud.itstep.org/blog/building-and-understanding-algorithms-a-step-by-step-guide-for-beginners (дата звернення: 19.05.2025).

25. Що таке алгоритми програмування: основи та застосування. URL: https://itproger.com/ua/news/chto-takoe-algoritmi-programmirovaniya-osnovi-i-primenenie (дата звернення: 19.05.2025).

26. Блок-схема: основні елементи та застосування в розробці ПО. URL: https://foxminded.ua/blok-skhema/ (дата звернення: 19.05.2025).

27. ER-model. URL: https://www.geeksforgeeks.org/introduction-of-er-model/ (дата звернення: 19.05.2025).

28. PHP. URL: https://www.geeksforgeeks.org/php/ (дата звернення: 20.05.2025).

29. Advantages and Disadvantages of CSS. URL: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-css/ (дата звернення: 20.05.2025).

30. HTML. URL: https://www.geeksforgeeks.org/html (дата звернення: 20.05.2025).

31. Що таке JavaScript і для чого він потрібен. URL: https://goit.global/ua/articles/shcho-take-javascript-i-dlia-choho-vin-potriben/ (дата звернення: 20.05.2025.)

32. Advantages and Disadvantages of TypeScript over JavaScript. URL: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-typescript-over-javascript/ (дата звернення: 20.05.2025).

33. PHPMailer Guide: Configuration, SMTP Setup, and Email Sending. URL: https://mailtrap.io/blog/phpmailer/ (дата звернення: 21.05.2025).

34. Bootstrap. URL: https://getbootstrap.com/ (дата звернення: 21.05.2025).

35. SQL vs MySQL. URL: https://www.geeksforgeeks.org/sql-vs-mysql/ (дата звернення: 21.05.2025).

36. Інтегроване середовище розробки (IDE). URL: https://w3schoolsua.github.io/hyperskill/ide (дата звернення: 22.05.2025).

37. Visual Studio Code. URL: https://code.visualstudio.com/ (дата звернення: 22.05.2025).

38. IntelliJ IDEA. URL: https://www.jetbrains.com/idea/ (дата звернення: 22.05.2025).

39. Atom. URL: https://atom-editor.cc/ (дата звернення: 22.05.2025).

40. Види тестування та відміннсоті між ними. URL:https://qagroup.com.ua/publications/vydy-testuvannya-ta-vidminnosti-mizh-nymy/ (дата звернення: 23.05.2025).

41. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп’ютерні науки» / уклад.: А. А. Яровий, О. В. Сілагін, І. В. Богач. – Вінниця : ВНТУ, 2023. 54 с.

ДОДАТКИ

ДОДАТОК А (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка WEB-ресурсу «Ресторан»

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 11,75 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____
(прізвище, ініціали, посада)

_____
(підпис)

Колесницький О.К., проф. каф КН _____
(прізвище, ініціали, посада)

_____
(підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Белзецький Р.С.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Демюк Д.Я.
(підпис) (прізвище, ініціали)

ДОДАТОК Б (обов'язковий)
Лістинг програми

```
==================== ./admin.php ====================
<?php
require_once 'Model.php';
Model::init();
$user = 'admin';
$pass = '12345';
if (!isset($_SERVER['PHP_AUTH_USER']) || $_SERVER['PHP_AUTH_USER'] !== $user ||
$_SERVER['PHP_AUTH_PW'] !== $pass) {
    header('WWW-Authenticate: Basic realm="Admin Panel"');
    header('HTTP/1.0 401 Unauthorized');
    exit;
}
$title = 'Бронювання у вашому ресторані:';
$bookings = Model::getBookings();
?>
<?php require_once 'layout/header.php' ?>
<main>
    <div class="container">
        <h2 class="page-title dish-page-title"><?=$title ?></h2>
        <table>
            <tr>
                <th>Ім'я</th>
                <th>Телефон</th>
                <th>Забронювано на дату</th>
                <th>Час</th>
                <th>К-сть людей</th>
                <th>Email</th>
                <th>Додатково</th>
            </tr>
        <?php foreach ($bookings as $booking){ ?>
            <tr>
                <td><?=$booking['name']?></td>
                <td><?=$booking['telephone']?></td>
                <td><?=$booking['date']?></td>
                <td><?=$booking['time']?></td>
                <td><?=$booking['quantity']?></td>
                <td><?=$booking['email']?></td>
                <td><?=$booking['additional']?></td>
            </tr>
        <?php } ?>
        </table>
    </div>
</main>
<?php  ?>
<?php require_once 'layout/footer.php' ?>
==================== ./Application.php ====================
<?php
class Application {
    static function run(){
        $route = parse_url($_SERVER['REQUEST_URI'],PHP_URL_PATH);
        $data = [];
        $data['text'] = $text = Translator::getCommonText();
        $data['lang'] = Translator::getActiveLang();
```

```
        $data['lang_url_postfix'] = Translator::getLangPostfix();
        $data['route'] = $route;
        switch ($route){
            case '/' :
                $data['title'] = $text['page_main_title'];
                self::render('home', $data);
            break;
            case '/menu' :
                $data['title'] = $text['page_dish_title'];
                $data['grouped_dishes'] = Model::getDishes();
                self::render('menu', $data);
            break;
            case '/review' :
                $data['title'] = $text['page_review_title'];
                $data['reviews'] = Model::getReviews();
                self::render('review', $data);
            break;
            case '/booking' :
                $data['title'] = $text['page_booking_title'];
                self::render('booking', $data);
            break;
            case '/admin' :
                header("Location: /admin.php");
            break;
            default: http_response_code(404);
        }
    }
    public static function render($page, $data){
        extract($data);
        require_once 'layout/header.php';
        require_once 'pages/'.$page.'.php';
        require_once 'layout/footer.php';
    }
}
?>
==================== ./booking_form_handler.php ====================
<?php
require_once 'Model.php';
Model::init();
require_once 'Mailer.php';
$data = json_decode(file_get_contents("php://input"), true);
$name = $data['name'] ?? '';
$telephone = $data['telephone'] ?? '';
$date = $data['date'] ?? '';
$time = $data['time'] ?? '';
$quantity = $data['quantity'] ?? '';
$email = $data['email'] ?? '';
$additional = $data['additional'] ?? '';
Model::addBooking($name, $telephone, $date, $time, $quantity, $email, $additional);
$protocol = (!empty($_SERVER['HTTPS']) && $_SERVER['HTTPS'] !== 'off') ? 'https' : 'http';
$host = $_SERVER['HTTP_HOST'];
$base_url = $protocol . '://' . $host;
$admin_link = $base_url.'/admin';
$message = $name.' зробив(-ла) нове бронювання у ресторані.'.PHP_EOL
    .'телефон: '.$telephone.PHP_EOL
    .'дата: '.$date.PHP_EOL
```

```
    .'час: '.$time.PHP_EOL
    .'К-сть людей: '.$quantity.PHP_EOL
    .'email: '.$email.PHP_EOL
    .'Додатково: '.$additional.PHP_EOL.
'Для перегляду деталей зайдіть у панель адміністратора '.$admin_link;
$admin_email = 'dianademiuk13@gmail.com';
$from_email = 'myhost1@web797.default-host.net';
$status = Mailer::send($admin_email, 'Нове бронювання у ресторані на '.$date, $message, $from_email);
$result = [
    'message' => $name.', дякуємо! Ми зв\'яжемося з Вами найближчим часом для підтвердження
бронювання!'
];
echo json_encode($result);
==================== ./index.php ====================
<?php
require_once 'Application.php';
require_once 'Model.php';
require_once 'Translator.php';
ini_set('display_errors',1);
ini_set('error_reporting',E_ALL);
ini_set('log_errors',1);
ini_set('error_log',__DIR__.'\error-log.txt');
Model::init();
Translator::init();
Application::run();
?>
==================== ./layout/footer.php ====================
<footer>
    <div class="container">
        <p>© <?= date('Y') ?> <?=$text['restaurant']?> "Dolce Vita"</p>
    </div>
</footer>
</body>
</html>
==================== ./layout/header.php ====================
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta charset="UTF-8">
    <title><?=$title ?></title>
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Mulish:ital,wght@0,200..1000;1,200..1000&display=swap"
rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Great+Vibes&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="/assets/style.css?v3">
    <script src="/assets/script.js?v3"></script>
</head>
<body>
<header class="header">
    <div class="container">
        <div class="header-inner">
            <a href="/<?=$lang_url_postfix?>" class="header__logo">
```

```
            <span class="header__logo-text"><?=$text['restaurant']?></span><span class="header__logo-
logo">Dolce Vita</span>
          </a>
          <ul class="header__menu">
            <li class="header__menu__item">
              <a class="header__menu__link <?=$route == '/' ? 'active' : ''?>"
href="/<?=$lang_url_postfix?>"><?=$text['menu_main']?></a>
            </li>
            <li class="header__menu__item">
              <a class="header__menu__link <?=$route == '/menu' ? 'active' : ''?>"
href="/menu<?=$lang_url_postfix?>"><?=$text['menu_dish']?></a>
            </li>
            <li class="header__menu__item">
              <a class="header__menu__link <?=$route == '/review' ? 'active' : ''?>"
href="/review<?=$lang_url_postfix?>"><?=$text['menu_review']?></a>
            </li>
            <li class="header__menu__item">
              <a class="header__menu__link <?=$route == '/booking' ? 'active' : ''?>"
href="/booking<?=$lang_url_postfix?>"><?=$text['menu_booking']?></a>
            </li>
          </ul>
          <div class="header__lang">
            <select data-lang="<?=$lang?>">
              <option value="uk" <?=$lang === 'uk' ? 'selected="selected"' : ''?>>UA</option>
              <option value="en" <?=$lang === 'en' ? 'selected="selected"' : ''?>>EN</option>
            </select>
          </div>
        </div>
    </div>
</header>
==================== ./Mailer.php ====================
<?php
class Mailer {
    public static function send($to, $subject, $message, $from = 'noreply@example.com') {
        $headers = "From: {$from}\r\n";
        $headers .= "Reply-To: {$from}\r\n";
        $headers .= "Content-Type: text/plain; charset=utf-8\r\n";
        return mail($to, $subject, $message, $headers);
    }
}
?>
==================== ./Model.php ====================
<?php
class Model
{
    private static $dish_file = 'db/dish.json';
    private static $review_file = 'db/review.json';
    private static $booking_file = 'db/booking.json';
    private static $dishes_data = [];
    private static $reviews_data = [];
    private static $booking_data = [];
    public static function init()
    {
        if(!is_file(self::$dish_file)){
            throw new Error('файлу '.self::$dish_file.' не знайдено');
        }
```

```
        if(!is_file(self::$review_file)){
            throw new Error('файлу '.self::$review_file.' не знайдено');
        }
        if(!is_file(self::$booking_file)){
            throw new Error('файлу '.self::$review_file.' не знайдено');
        }
        self::$dishes_data = json_decode(file_get_contents(self::$dish_file),1);
        if (json_last_error() != JSON_ERROR_NONE) {
            throw new Error('база даних '.self::$dish_file.' пошкоджена!');
        }
        self::$reviews_data = json_decode(file_get_contents(self::$review_file),1);
        if (json_last_error() != JSON_ERROR_NONE) {
            throw new Error('база даних '.self::$review_file.' пошкоджена!');
        }
        self::$booking_data = json_decode(file_get_contents(self::$booking_file),1);
        if (json_last_error() != JSON_ERROR_NONE) {
            throw new Error('база даних '.self::$booking_file.' пошкоджена!');
        }
    }
    public static function getDishes(){
        foreach (self::$dishes_data as $dish) {
            $category_name = Translator::getCategoryNameByKey($dish['category']);
            $dish['label'] = Translator::getLabelNameByKey($dish['label']);
            $dish['name'] = Translator::getDishField('name', $dish);
            $dish['description'] = Translator::getDishField('description', $dish);
            $grouped_dishes[$category_name][] = $dish;
        }
        return $grouped_dishes;
    }
    public static function getReviews(){
        //відсортуємо відгуки з старішого до новішого
        usort(self::$reviews_data, function($a, $b) {
            return $b['date'] <=> $a['date'];
        });
        return self::$reviews_data;
    }
    public static function getBookings(){
        //відсортуємо бронювання з старішого до новішого
        usort(self::$booking_data, function($a, $b) {
            return $b['date_added'] <=> $a['date_added'];
        });
        return self::$booking_data;
    }
    public static function addReview($name, $text, $rating){
        $new_review = [
            'name' => strip_tags($name) ?? 'Гість',
            'rating' => (int)$rating ?? 0,
            'text' => strip_tags($text) ?? '',
            'date' => time()
        ];
        $exist_reviews = self::getReviews();
        $exist_reviews[] = $new_review;
        $file_stream = fopen(self::$review_file,'w');
        fwrite($file_stream, json_encode($exist_reviews));
    }
    public static function addBooking($name, $telephone, $date, $time, $quantity, $email, $additional){
```

```
        $new_booking = [
            'name' => strip_tags($name),
            'telephone' => strip_tags($telephone),
            'date' => strip_tags($date),
            'time' => strip_tags($time),
            'quantity' => (int)$quantity,
            'email' => strip_tags($email),
            'additional' => strip_tags($additional),
            'date_added' => time(),
        ];
        $exist_bookings = self::getBookings();
        $exist_bookings[] = $new_booking;
        $file_stream = fopen(self::$booking_file,'w');
        fwrite($file_stream, json_encode($exist_bookings));
    }
}
==================== ./pages/booking.php ====================
<main>
    <div class="container">
        <h2 class="page-title booking-page-title"><?=$title ?></h2>
        <div class="booking-page__header">
            <?=$text['page_booking_description']?>
        </div>
        <form id="booking-form" class="booking-form">
            <label class="booking-form__title"><img src="/assets/img/reserve.svg">
<?=$text['page_booking_form_heading']?></label>
            <div class="booking-form__element">
                <label class="booking-form__label required"><?=$text['page_booking_form_name']?></label>
                <input type="text" required class="booking-form__input" name="name">
            </div>
            <div class="booking-form__element">
                <label class="booking-form__label
required"><?=$text['page_booking_form_telephone']?></label>
                <input type="text" required class="booking-form__input" name="telephone">
            </div>
            <div class="booking-form__element">
                <label class="booking-form__label required"><?=$text['page_booking_form_date']?></label>
                <input type="date" min="<?=date('Y-m-d')?>"  max="<?=date('Y-m-d',strtotime('+31 days'))?>"
required class="booking-form__input" name="date">
            </div>
            <div class="booking-form__element">
                <label class="booking-form__label required"><?=$text['page_booking_form_time']?></label>
                <input type="time" required class="booking-form__input" name="time">
            </div>
            <div class="booking-form__element">
                <label class="booking-form__label required"><?=$text['page_booking_form_qty']?></label>
                <input type="number" min="1" required class="booking-form__input" name="quantity">
            </div>
            <div class="booking-form__element">
                <label class="booking-form__label"><?=$text['page_booking_form_email']?></label>
                <input type="email" class="booking-form__input" name="email">
            </div>
            <div class="booking-form__element">
                <label class="booking-form__label"><?=$text['page_booking_form_comment']?></label>
                <textarea rows="5" class="booking-form__input" name="additional"></textarea>
            </div>
```

```
        <button class="booking-form__btn"
type="submit"><?=$text['page_booking_form_btn_submit']?></button>
      </form>
   </div>
</main>
==================== ./pages/home.php ====================
<main>
   <section class="main-banner">
      <div class="container">
         <img width="1170" height="450" src="/assets/img/content/main-banner.png">
      </div>
   </section>
   <section class="about-us">
      <div class="container">
         <h1 class="about-us__title page-title"><?=$text['page_main_about_heading']?></h1>
         <div class="about-us__grid">
            <div class="about-us__left">
               <?=$text['page_main_about_one']?>
            </div>
            <div class="about-us__right">
               <img src="/assets/img/content/about-1.png">
            </div>
         </div>
         <div class="about-us__grid">
            <div class="about-us__left">
               <img src="/assets/img/content/about-2.png">
            </div>
            <div class="about-us__right">
               <?=$text['page_main_about_two']?>
            </div>
         </div>
      </div>
   </section>
   <section class="contacts">
      <div class="container">
         <h2 class="contacts__title"><?=$text['page_main_contact_heading']?></h2>
         <div class="contacts__grid">
            <div class="contacts__left">
               <ul class="contacts__list">
                  <li>
                     <img src="/assets/img/schedule.svg"> <?=$text['page_main_contact_shedule']?> 11:00 -
23:00
                  </li>
                  <li>
                     <img src="/assets/img/phone.svg"> +38 094 123 4567
                  </li>
                  <li>
                     <img src="/assets/img/mail.svg"> info@restoraunt.com
                  </li>
                  <li>
                     <img src="/assets/img/location.svg"> <?=$text['page_main_contact_address']?>
                  </li>
               </ul>
            </div>
            <div class="contacts__right"></div>
         </div>
```

```
      </div>
   </section>
</main>
==================== ./pages/menu.php ====================
<main>
      <div class="container">
      <h2 class="page-title dish-page-title"><?=$title ?></h2>
   </div>
   <?php foreach ($grouped_dishes as $category => $dishes){ ?>
      <div class="container">
         <div class="dish-list__category">
            <?=$category?>
         </div>
      </div>
      <div class="dish-list">
      <?php foreach ($dishes as  $dish){ ?>
         <div class="dish-item">
            <div class="container">
               <div class="dish-item-inner">
                  <div class="dish-item__left">
                     <div class="dish-item__info">
                        <div class="dish-item__name"><?=$dish['name']?></div>
                        <div class="dish-item__price"><?=$dish['price']?> <?=$text['currency']?></div>
                        <div class="dish-item__description"><?=$dish['description']?></div>
                        <div class="dish-item__additional">
                           <?php if($dish['weight']){ ?>
                           <span class="dish-item__additional-
label"><?=$dish['weight']?><?=$text['grams']?></span>
                           <?php } ?>
                           <?php if (isset($dish['label']) && $dish['label']){ ?>
                           <span class="dish-item__additional-label additional-label-
offer"><?=$dish['label']?></span>
                           <?php } ?>
                        </div>
                     </div>
                  </div>
                  <div class="dish-item__right">
                     <img class="dish-item__image" src="<?=$dish['image']?>" alt="<?=$dish['name']?>">
                  </div>
               </div>
            </div>
         </div>
      <?php } ?>
      </div>
   <?php } ?>
</main>
```

ДОДАТОК В (обов'язковий)

ГРАФІЧНА ЧАСТИНА

«РОЗРОБКА WEB-РЕСУРСУ "РЕСТОРАН"»

Виконала: студентка 4 курсу, групи 2КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Діана ДЕМЮК
(ім'я ПРІЗВИЩЕ)

Керівник: к.т.н., доцент каф. КН
Руслан БЕЛЗЕЦЬКИЙ
(ім'я ПРІЗВИЩЕ)
«03» червня 2025 р.

Вінниця ВНТУ - 2025 рік

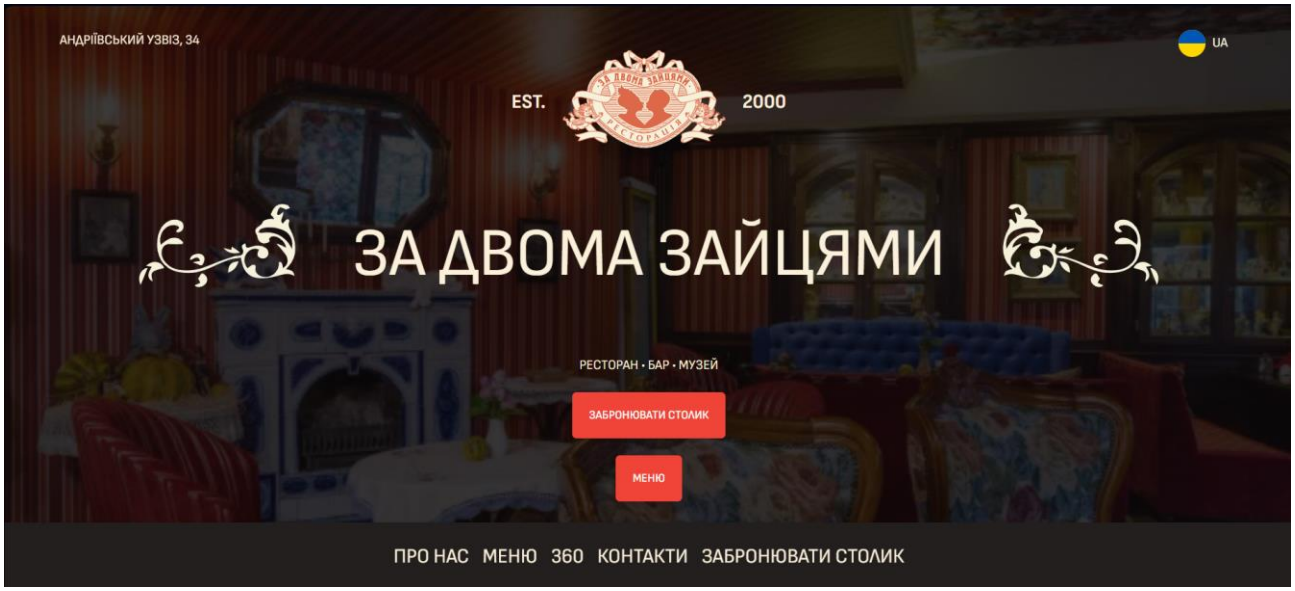

Рисунок В.1 – Загальний вигляд WEB-ресурсу «За двома зайцями»

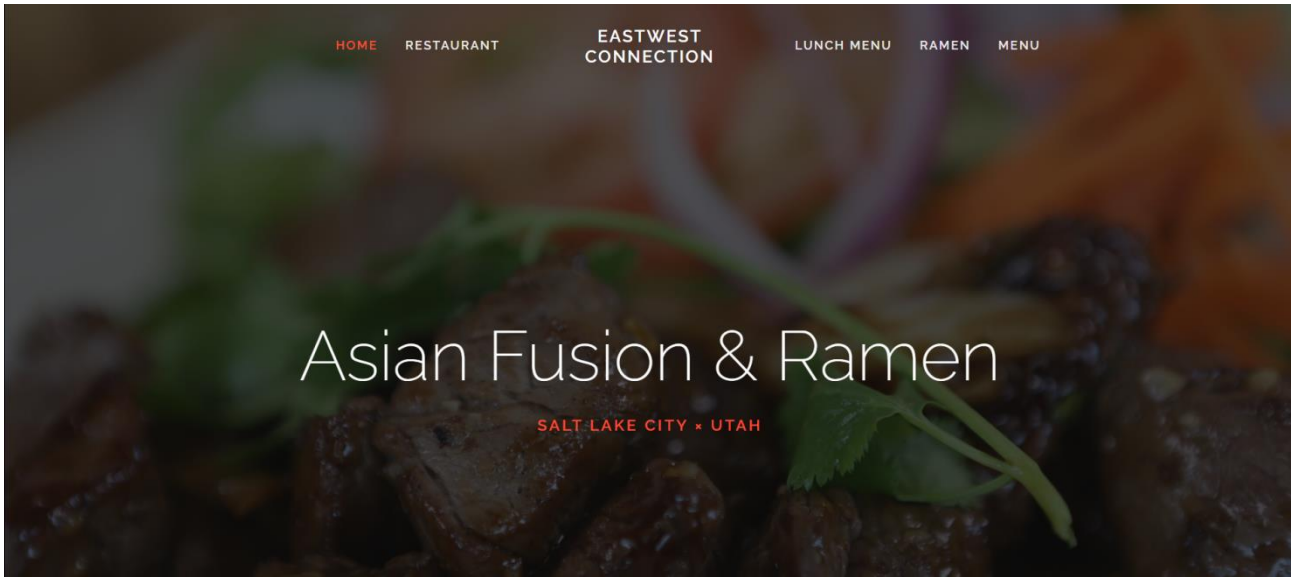

Рисунок В.2 – Загальний вигляд WEB-ресурсу «EastWest Connection»

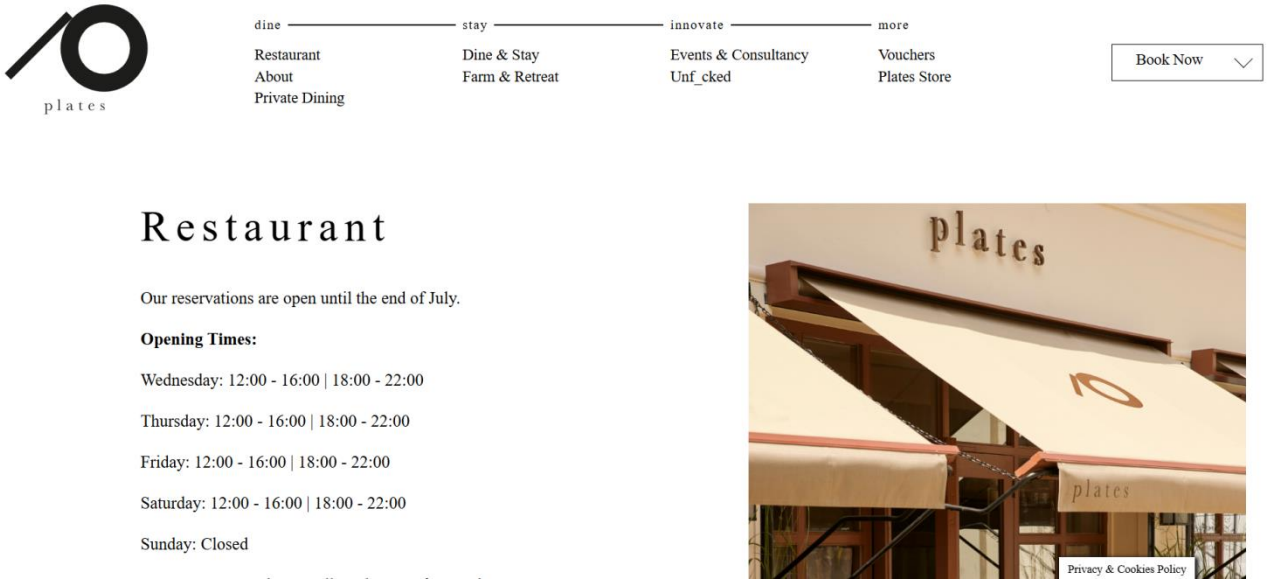

Рисунок В.3 − Загальний вигляд WEB-ресурсу «Plates»

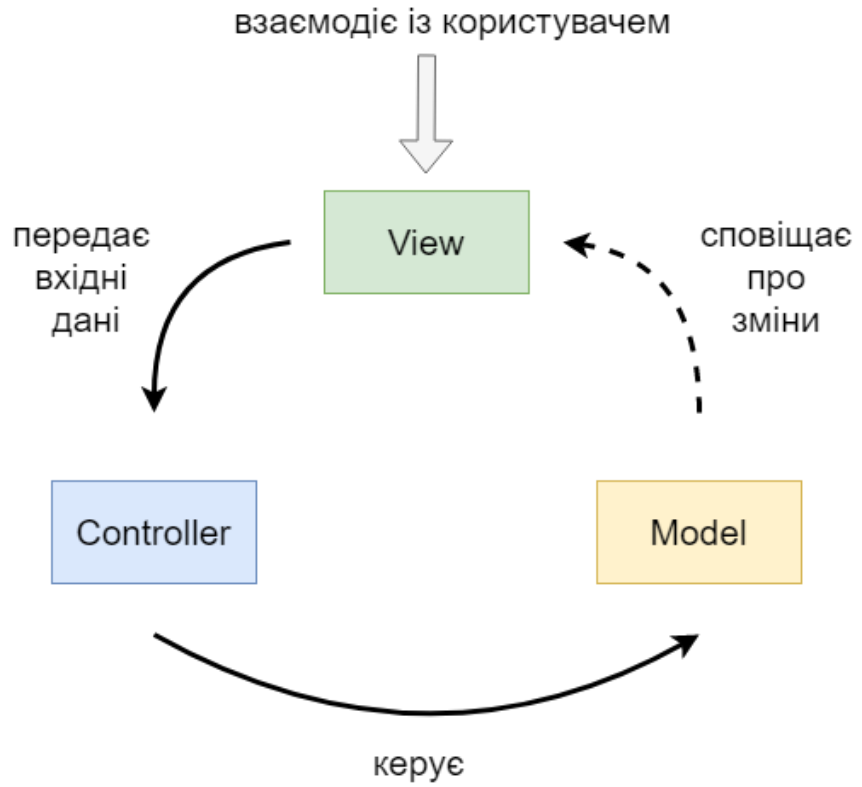

Рисунок В.4 – Діаграма взаємодії між компонентами шаблону MVC

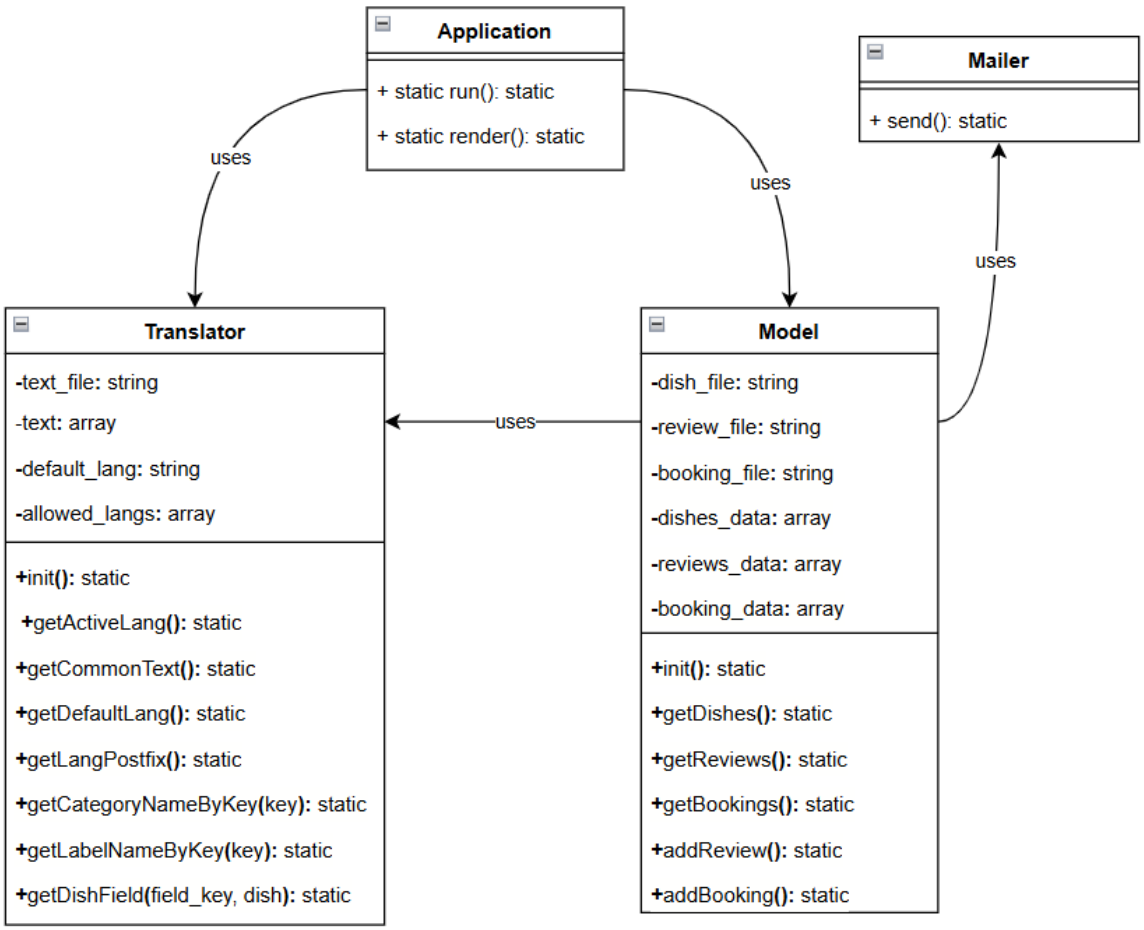

Рисунок В.5 – UML-діаграма класів WEB-ресурсу «Ресторан»

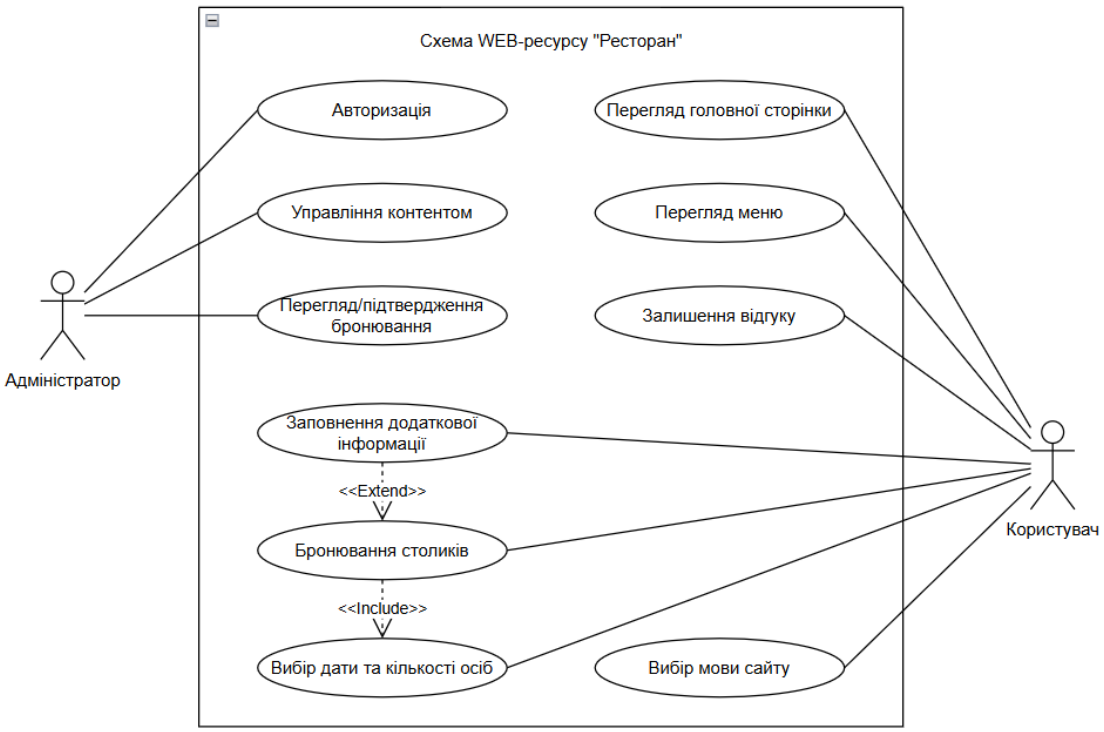

Рисунок В.6 – Діаграма прецедентів WEB-ресурсу «Ресторан»

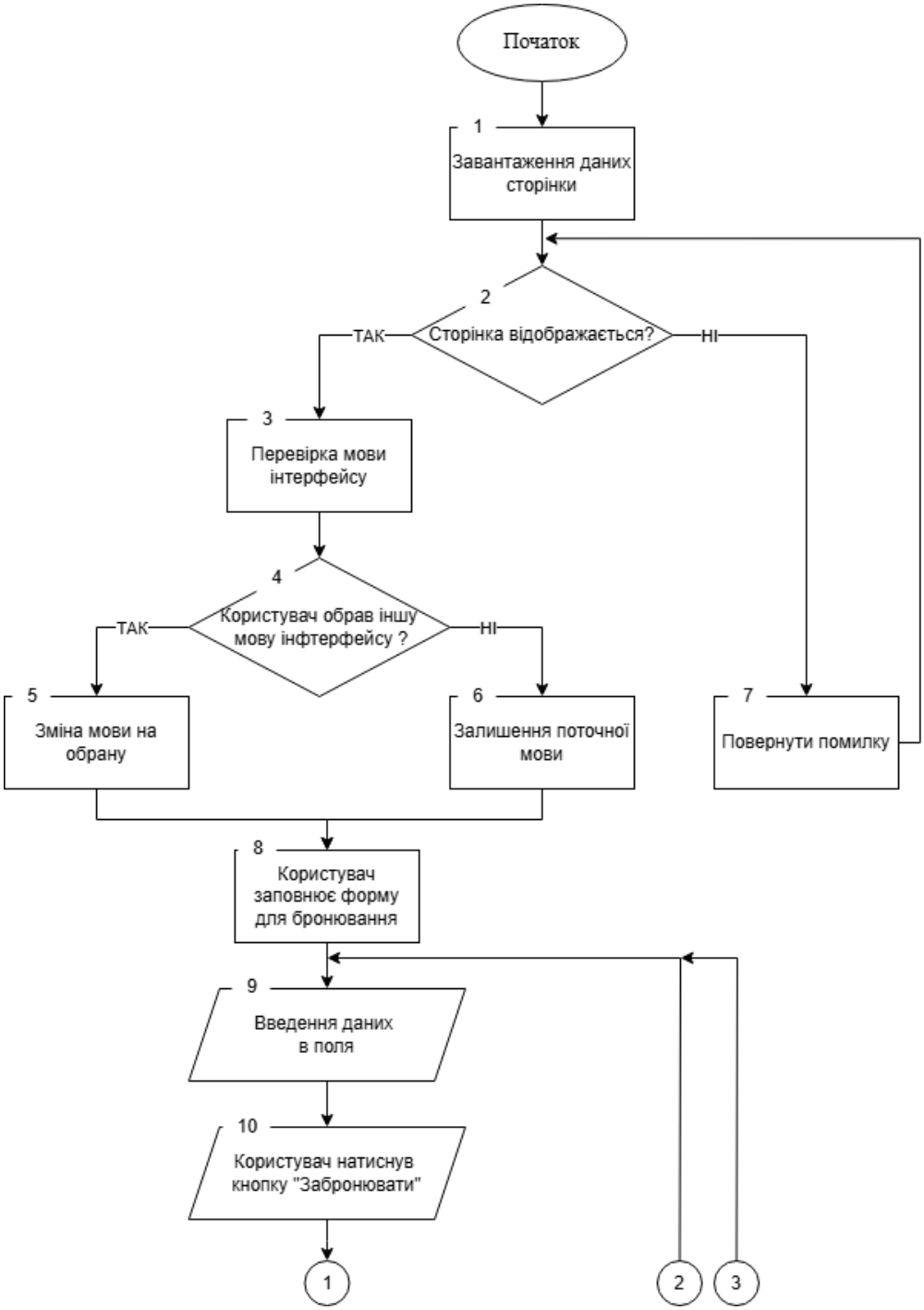

Рисунок В.7 – Схема алгоритму роботи модуля бронювання

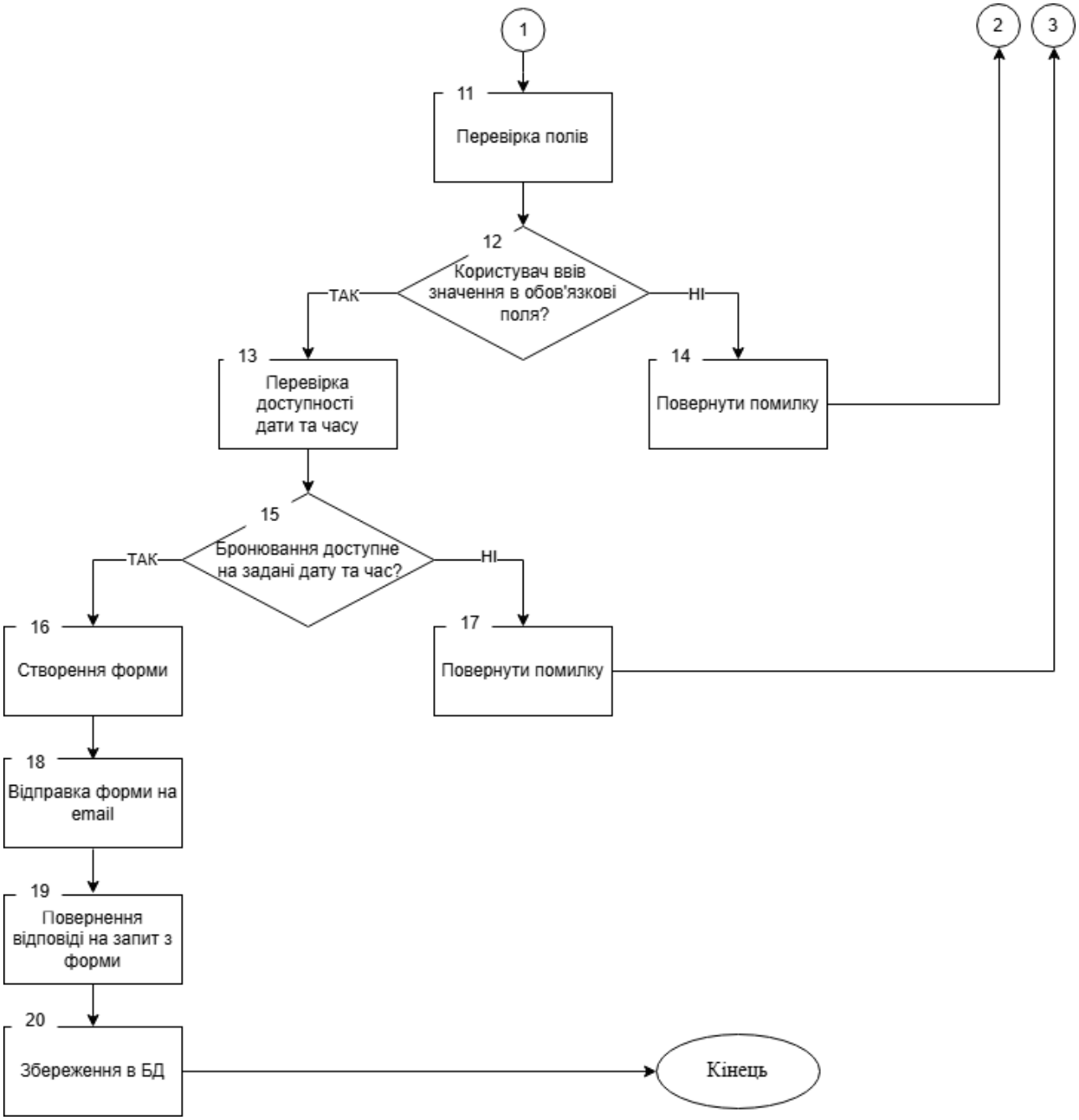

Рисунок В.7, аркуш 2

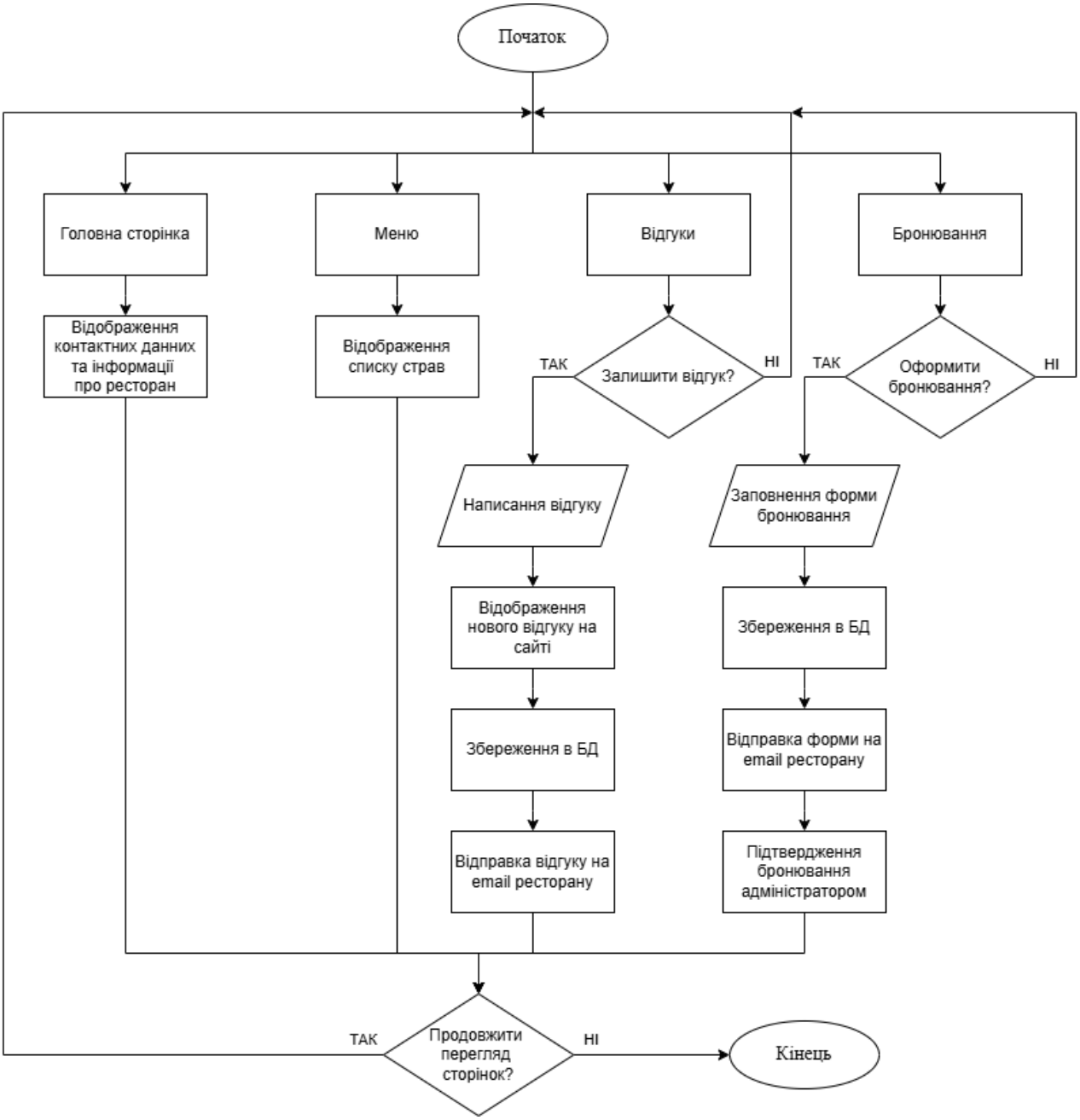

Рисунок В.8 – Схема алгоритму роботи WEB-ресурсу «Ресторан»

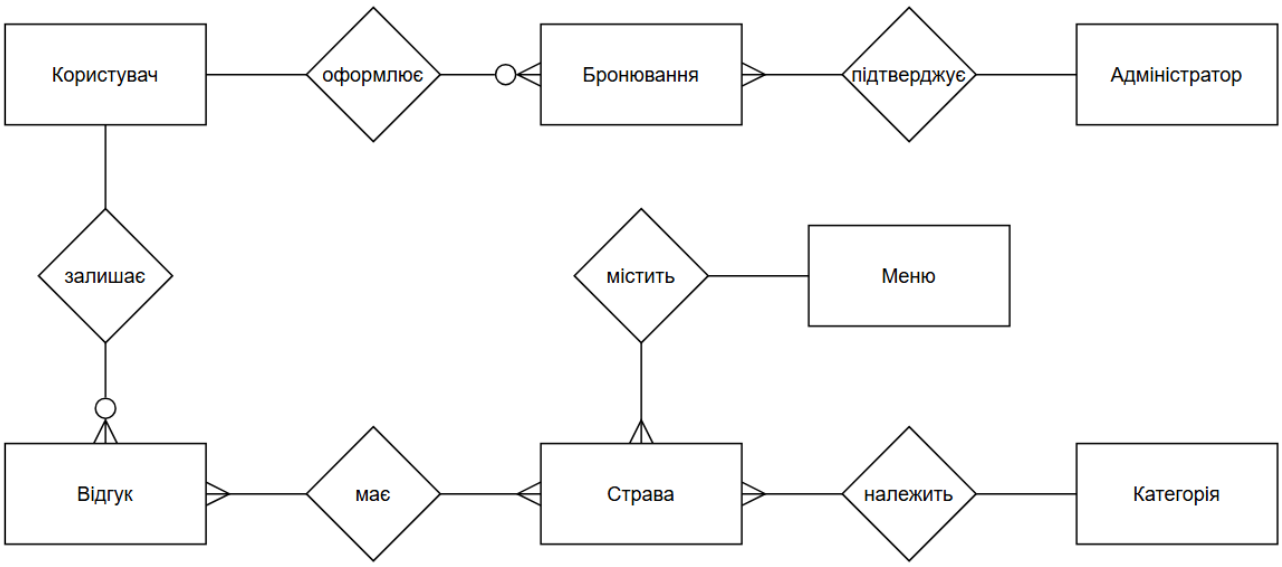

Рисунок В.9 – ER-діаграма бази даних WEB-ресурсу «Ресторан»

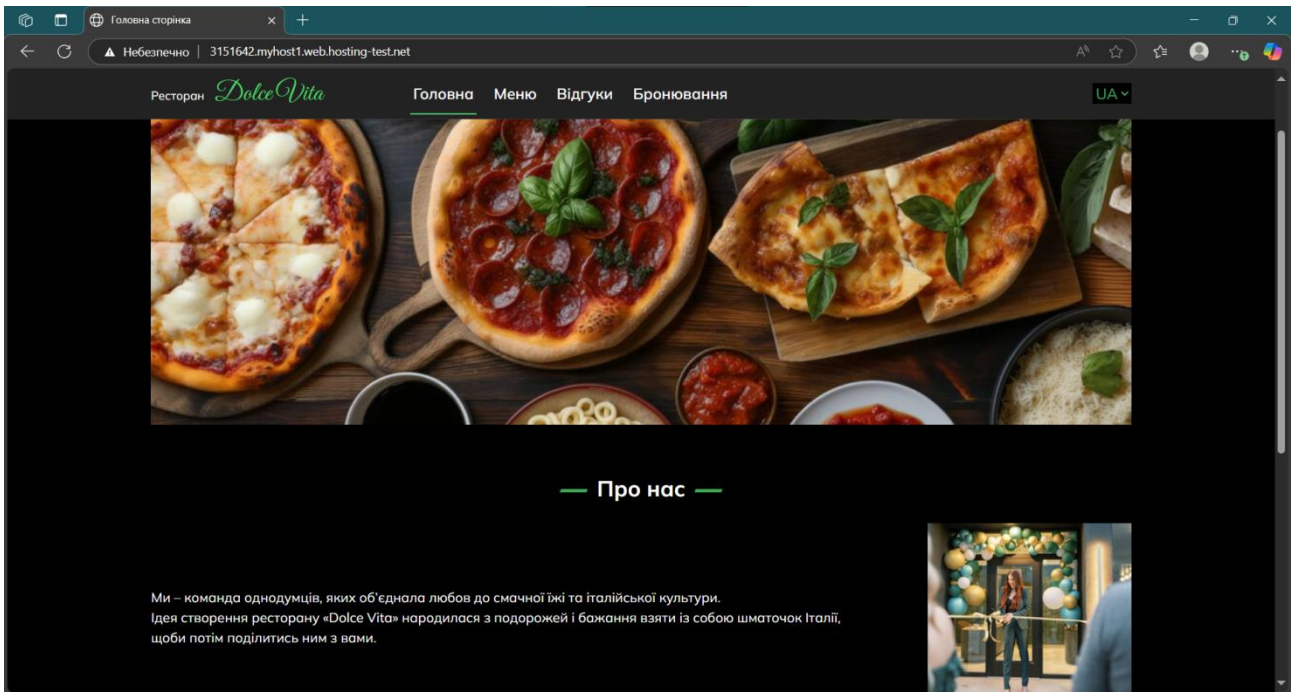

Рисунок В.10 – Зображення роботи WEB-сайту в браузері Microsoft Edge

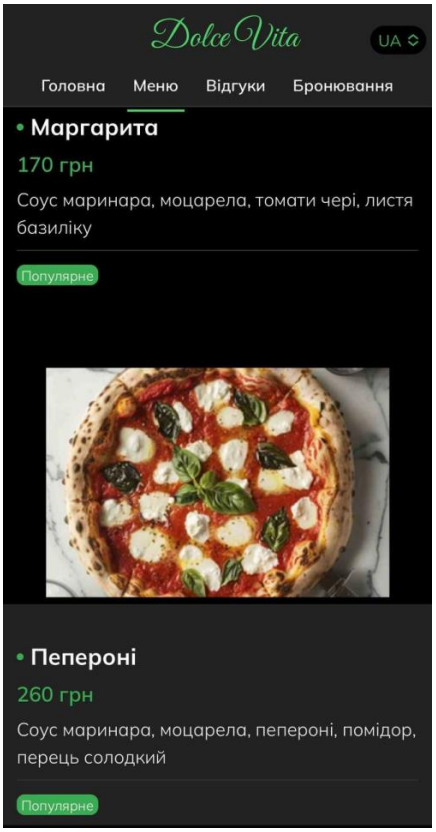

Рисунок В.11 – Зображення роботи WEB-сайту в браузері Safari

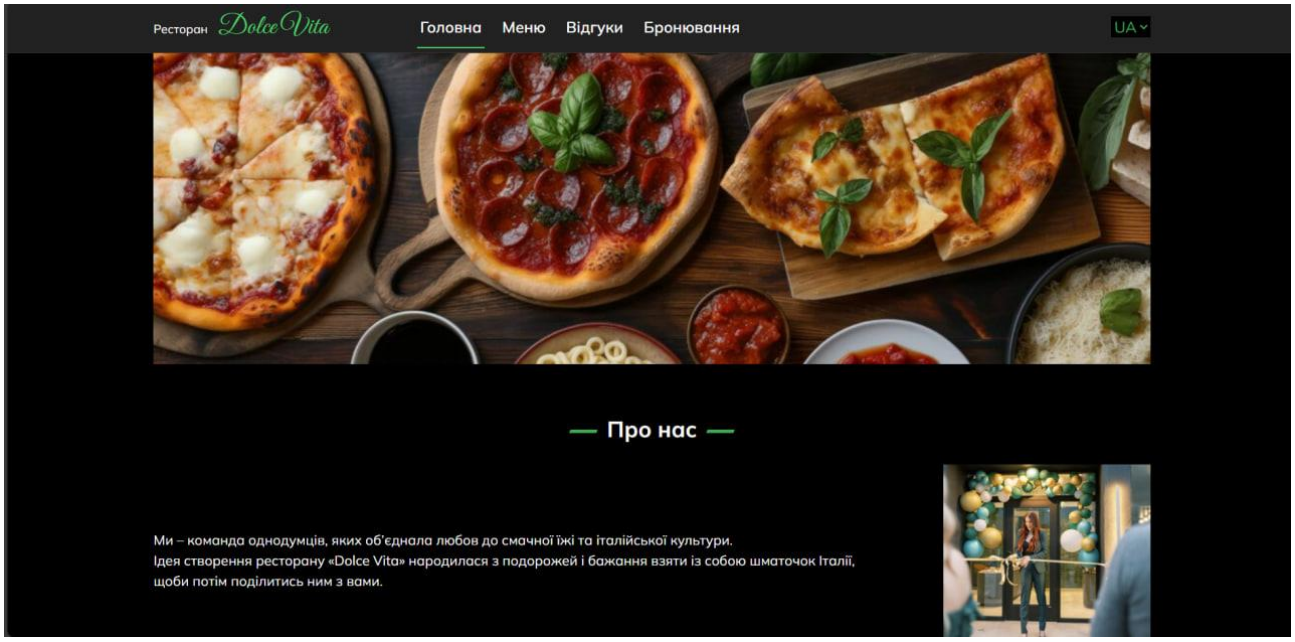

Рисунок В.12 – Інтерфейс головної сторінки WEB-ресурсу «Ресторан»

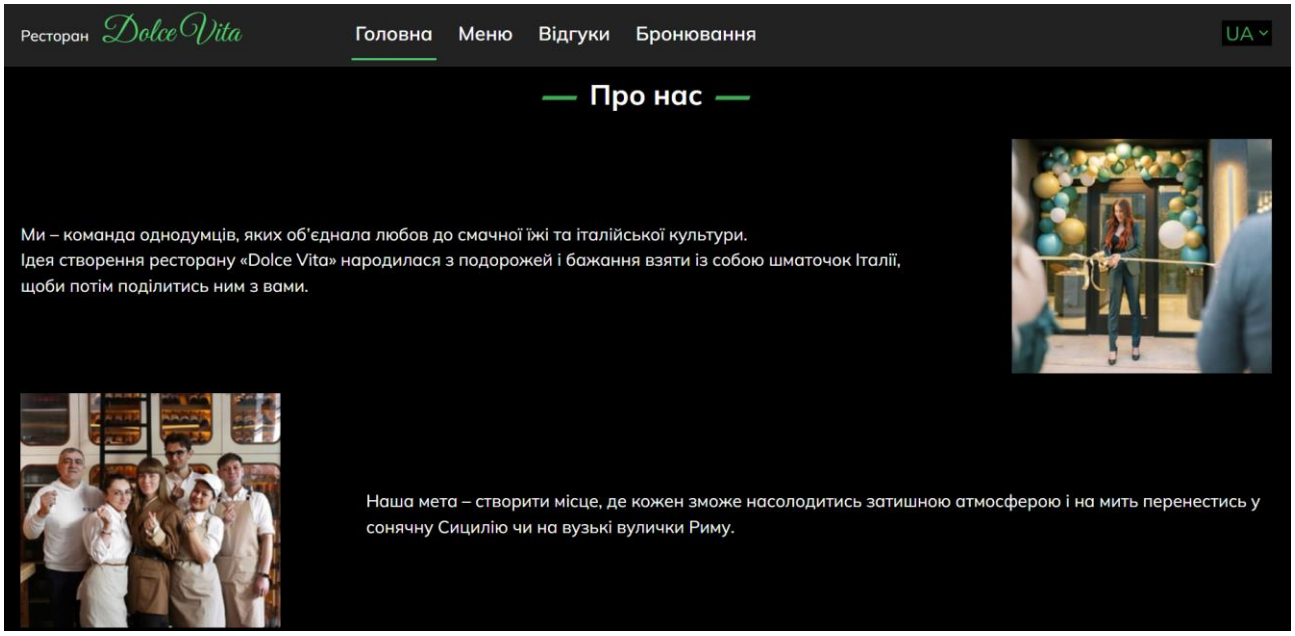

Рисунок В.13 – Коротка інформація про ресторан

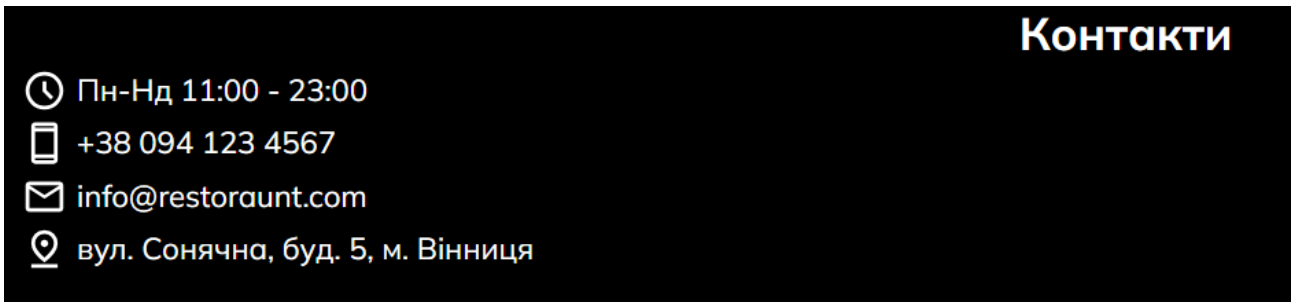

Рисунок В.14 – Секція з контактною інформацією

Рисунок В.15 – Інтерфейс головної сторінки WEB-ресурсу на мобільному пристрої

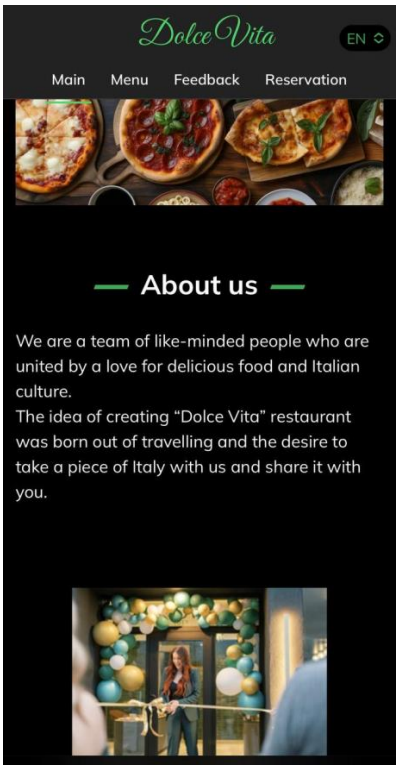

Рисунок В.16 – Головна сторінка WEB-ресурсу на мобільному пристрої при зміні мови інтерфейсу на англійську

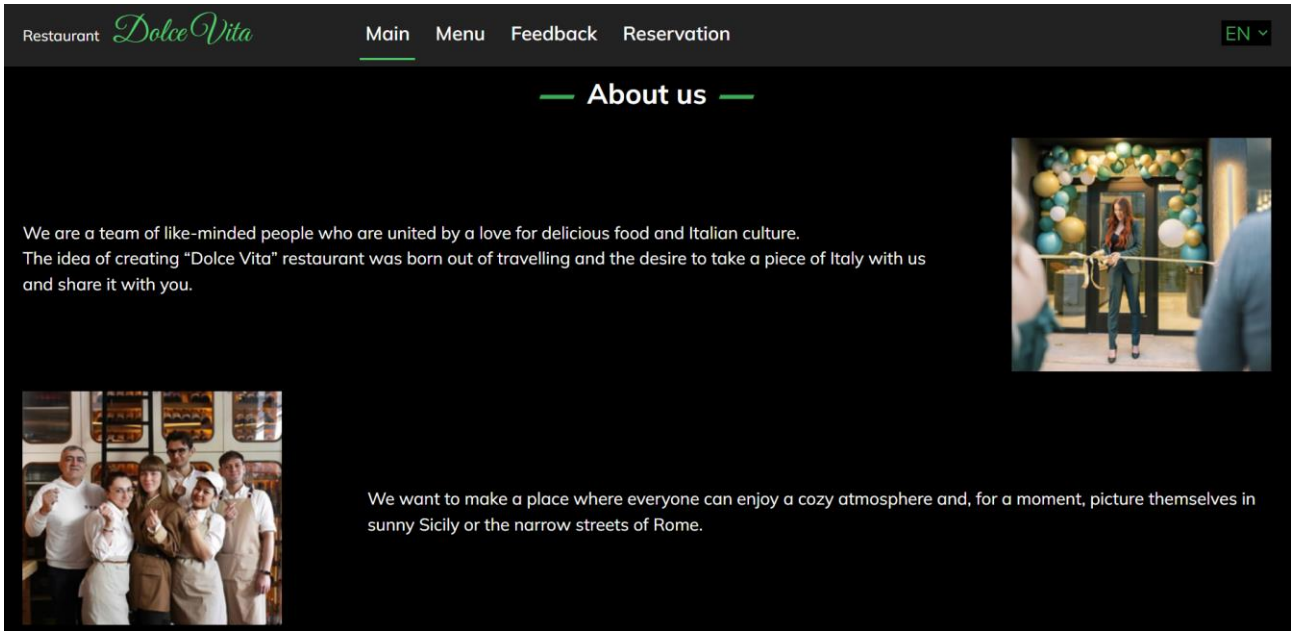

Рисунок В.17 – Коротка інформація про ресторан при зміні мови інтерфейсу на англійську

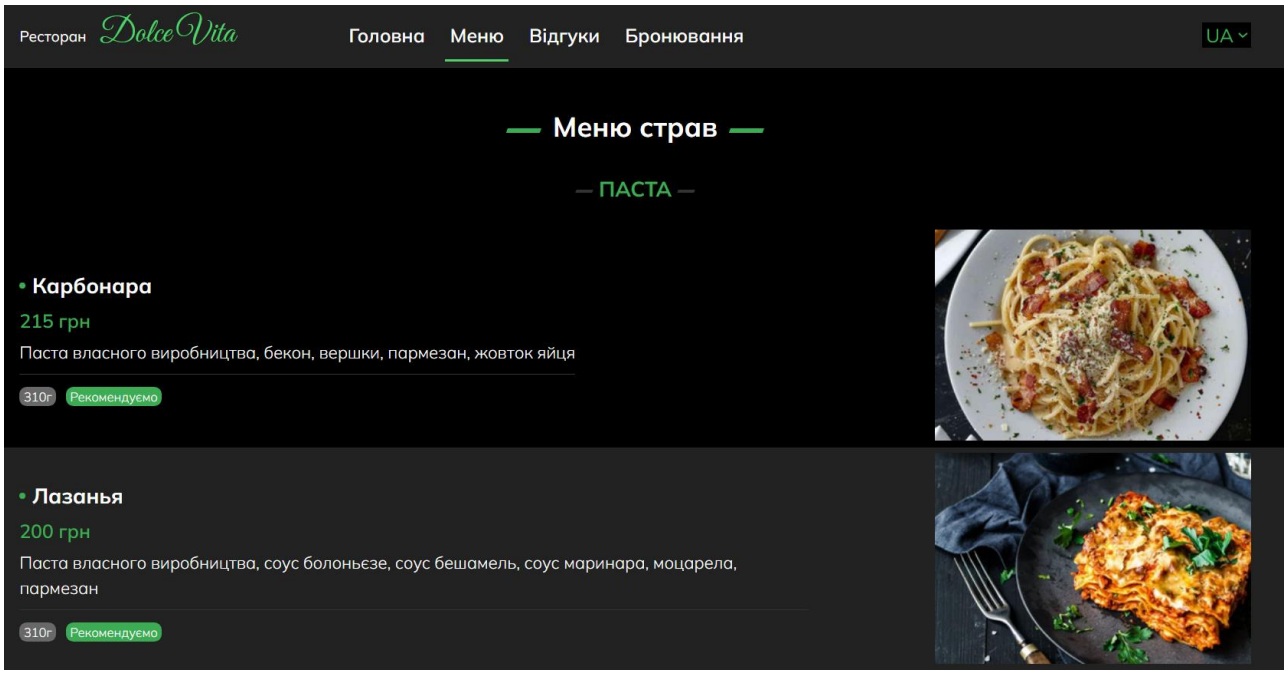

Рисунок В.18 – Інтерфейс сторінки «Меню»

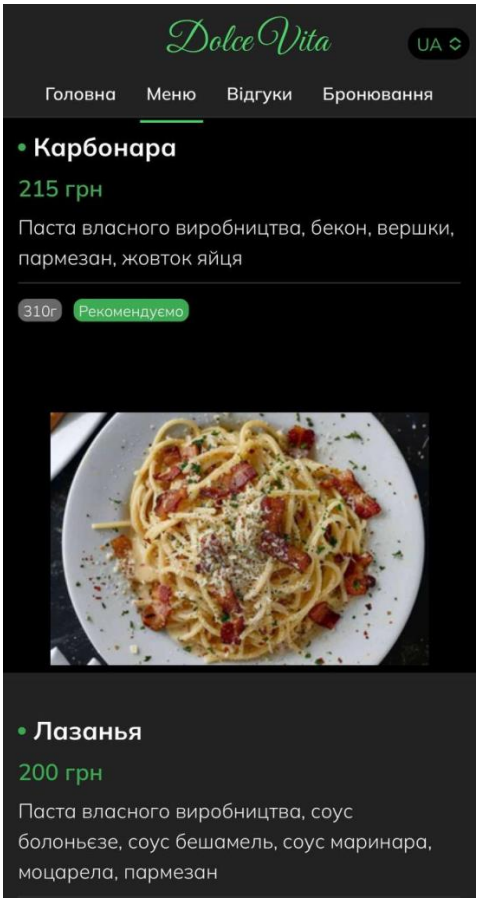

Рисунок В.19 – Інтерфейс сторінки «Меню» на мобільному пристрої

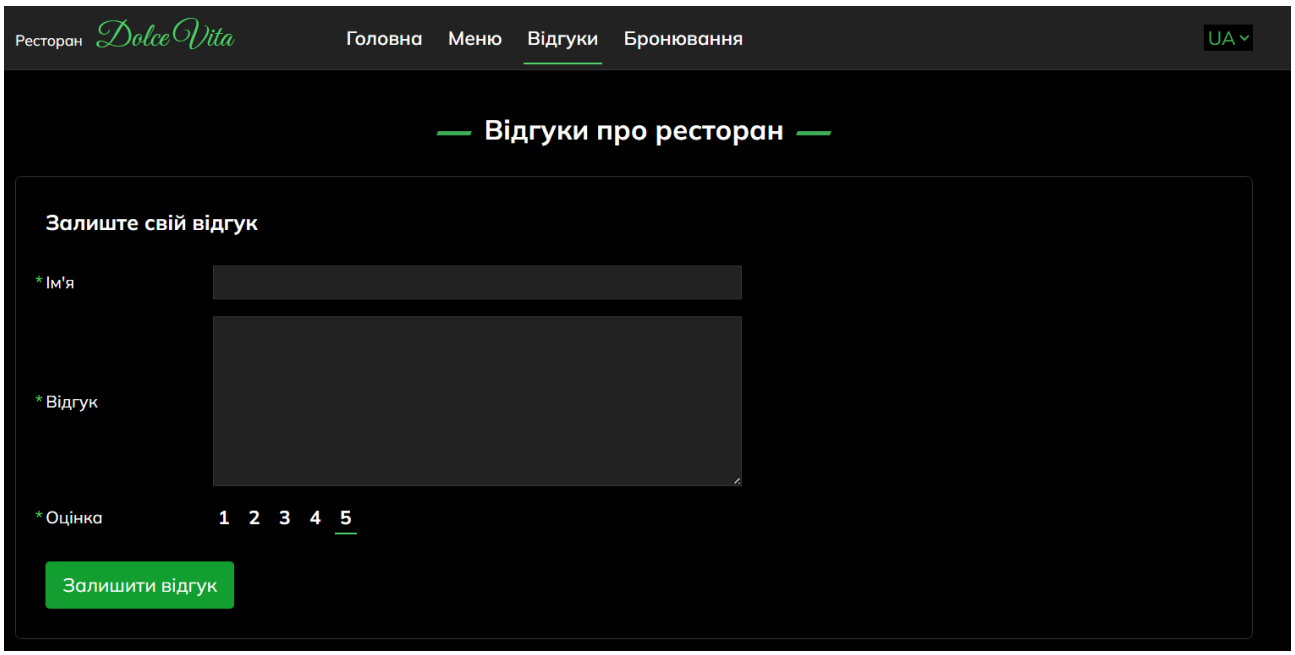

Рисунок В.20 – Інтерфейс сторінки «Відгуки»

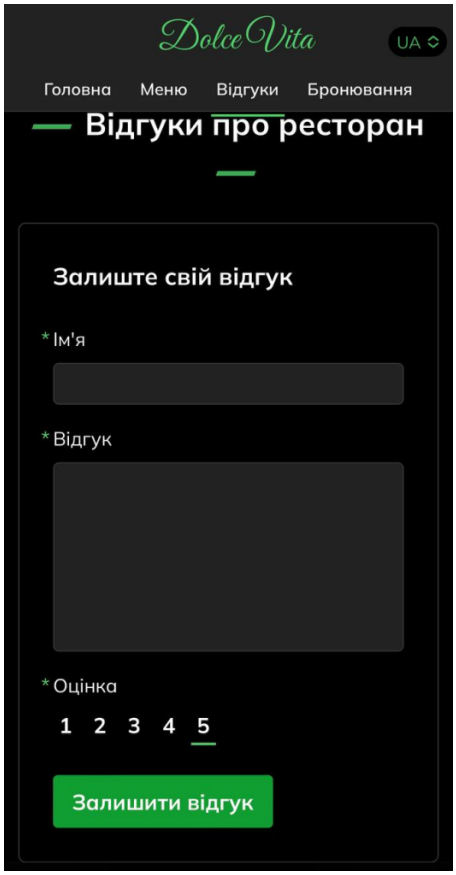

Рисунок В.21 – Інтерфейс сторінки «Відгуки» на мобільному пристрої

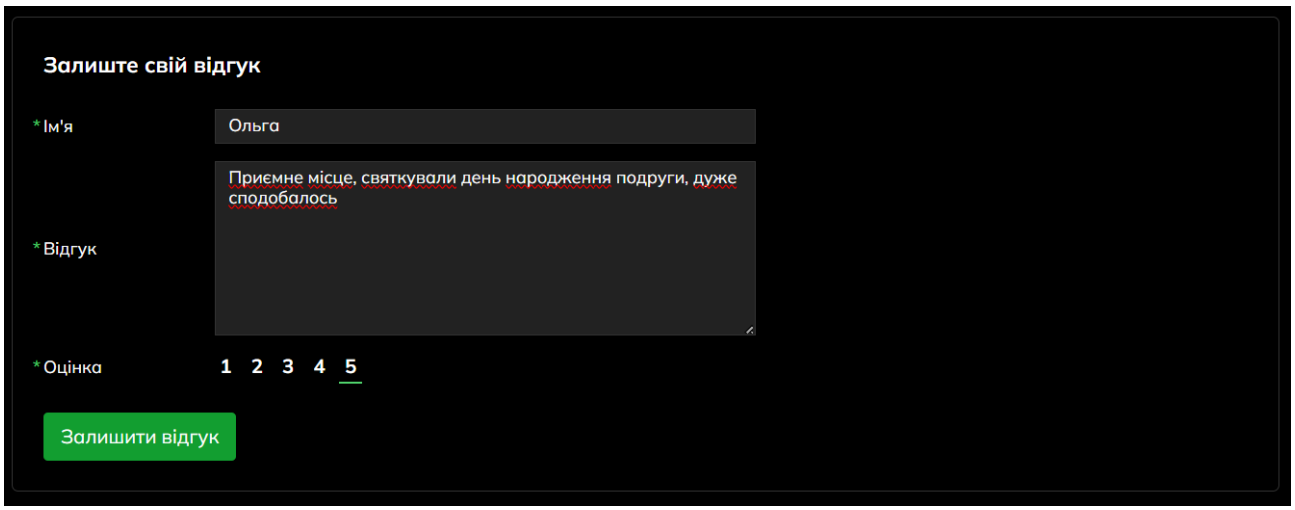

Рисунок В.22 – Заповнена форма для відгуків

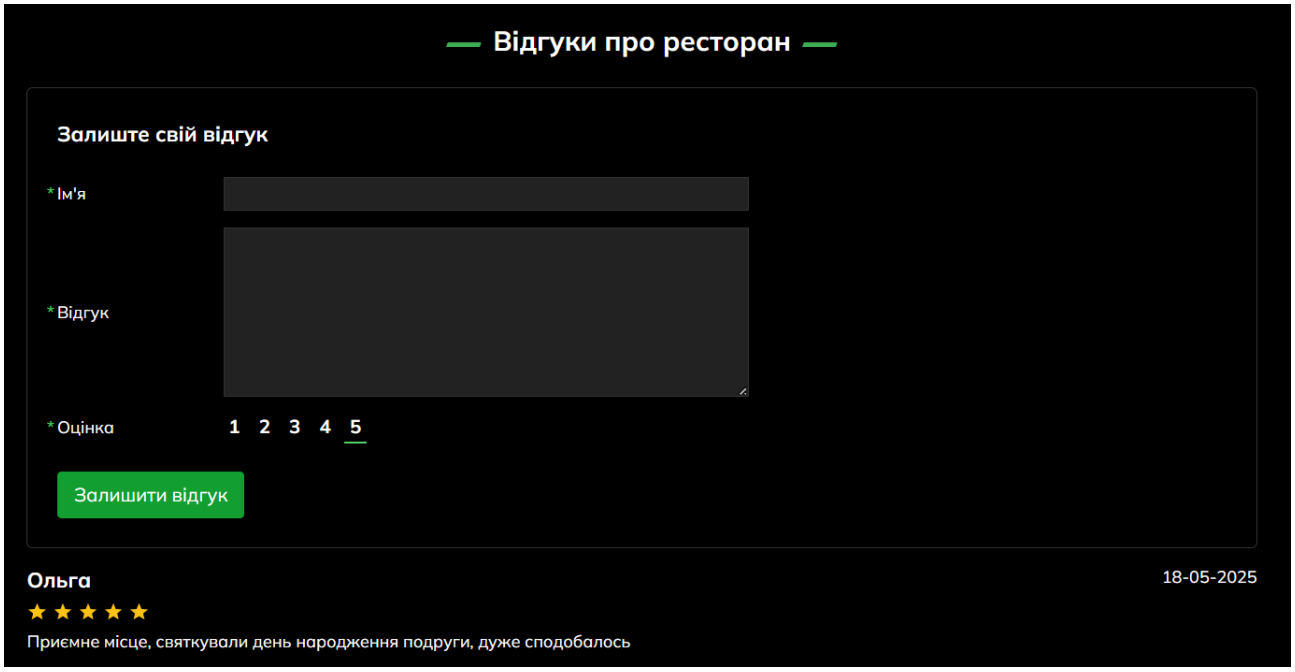

Рисунок В.23 – Секція з відгуками

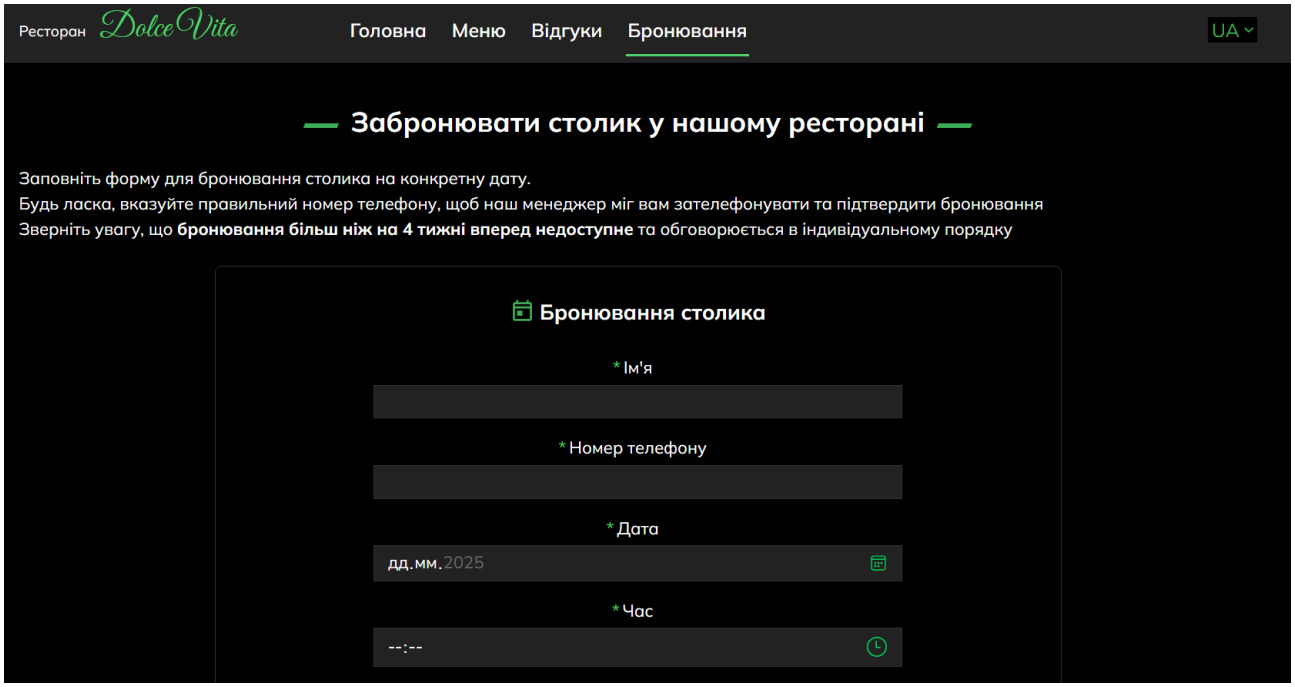

Рисунок В.24 – Інтерфейс сторінки «Бронювання»

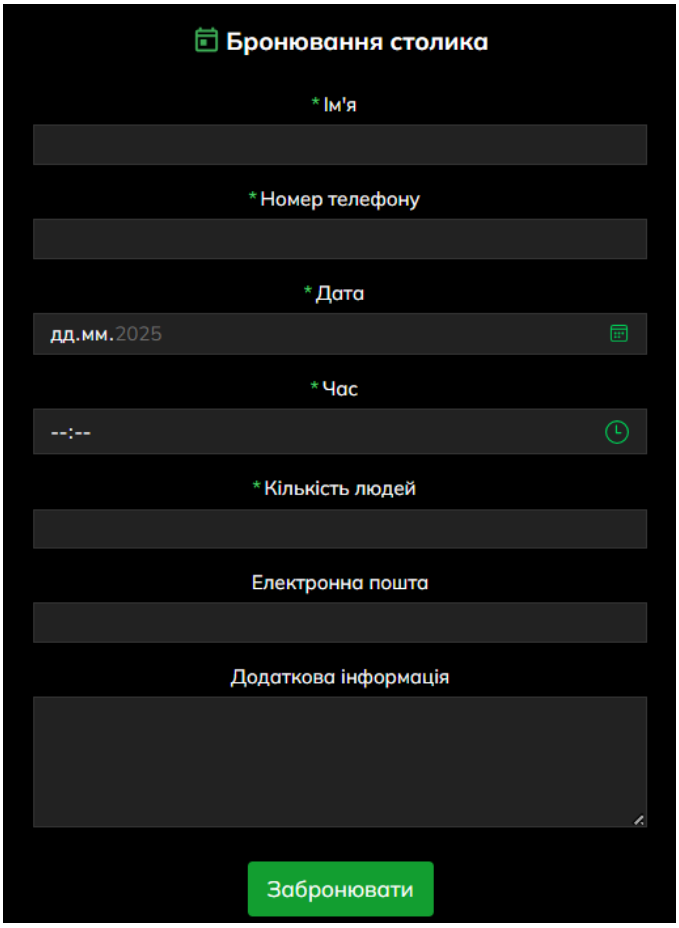

Рисунок В.25 – Форма для бронювання

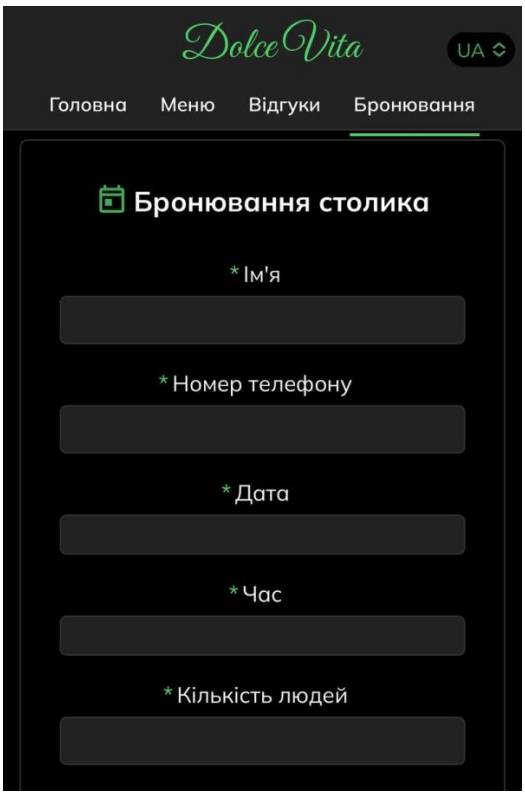

Рисунок В.26 – Форма для бронювання на мобільному пристрої

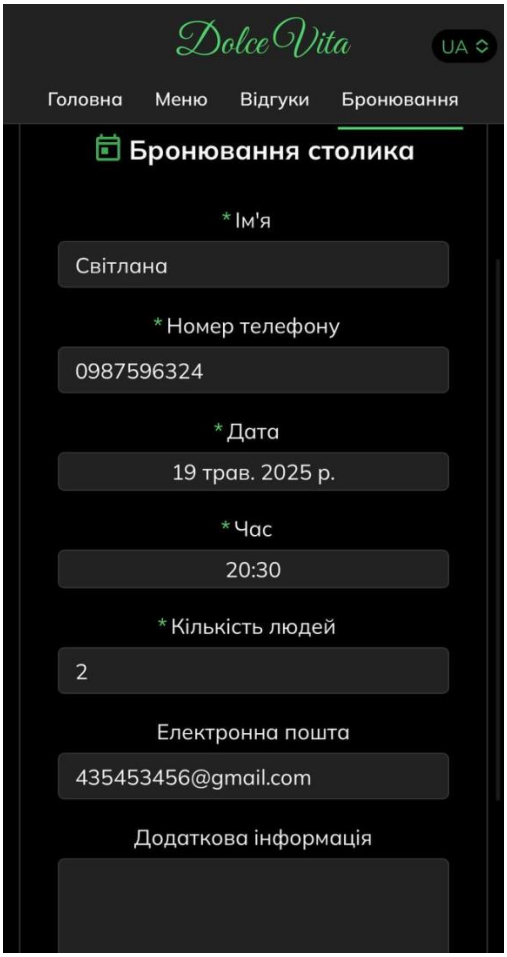

Рисунок В.27 – Заповнена форма для бронювання на мобільному пристрої

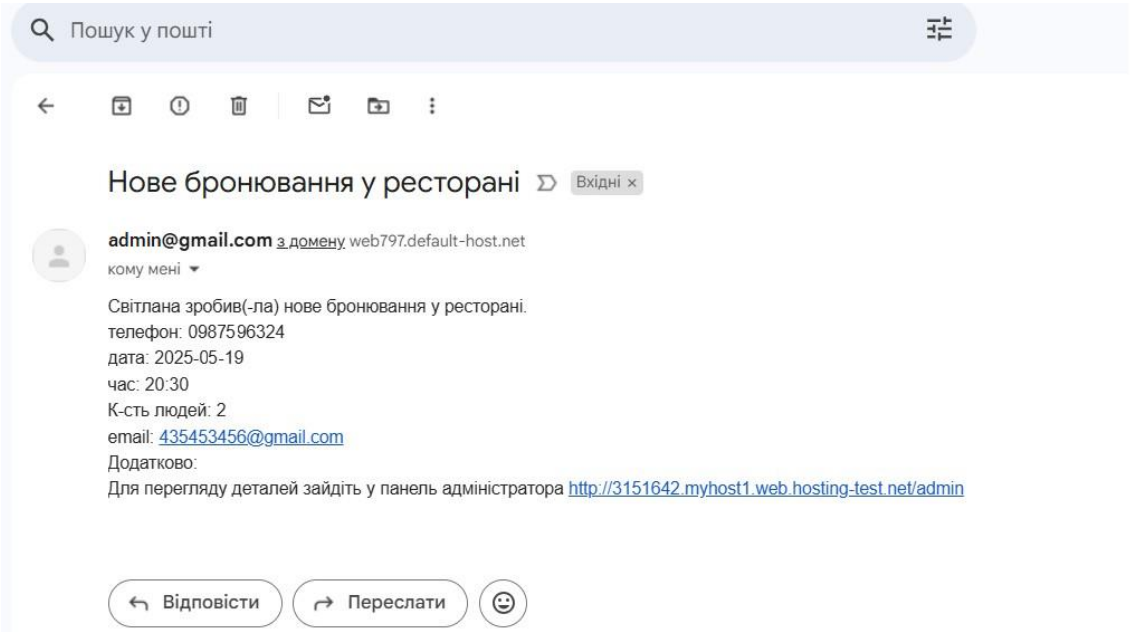

Рисунок В.28 – Повідомлення про нове бронювання на пошті адміністрації

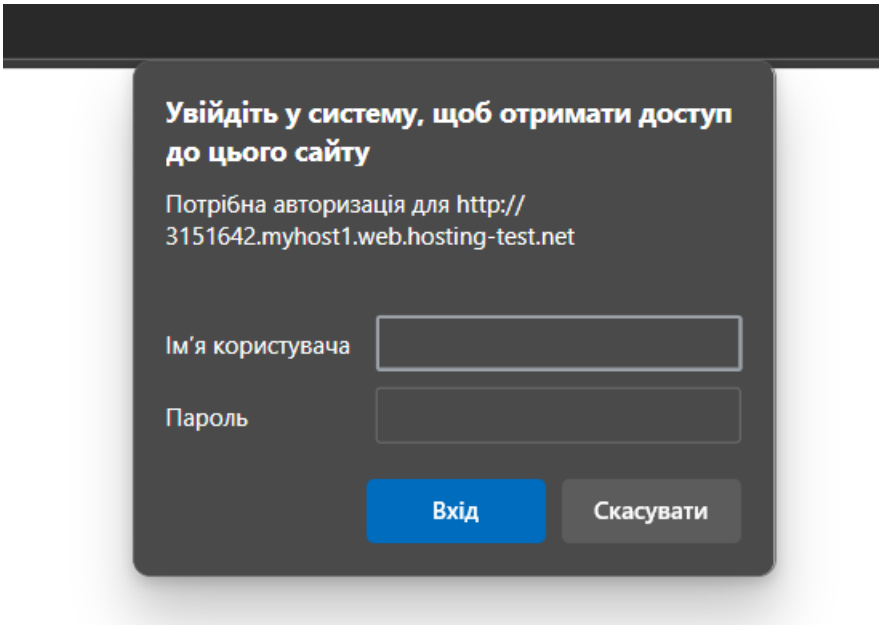

Рисунок В.29 – Форма для входу на сторінку адміністратора

ДОДАТОК Г (довідниковий)

Інструкція користувача

Для запуску WEB-ресурсу адміністратор повинен мати Visual Stusio Code зі встановленими бібліотеками PHPMailer і Bootstrap.

Після запуску при переході за адресою http://3151642.myhost1.web.hosting-test.net, відкривається головна сторінка сайту, що зображена на рисунку Г.1.

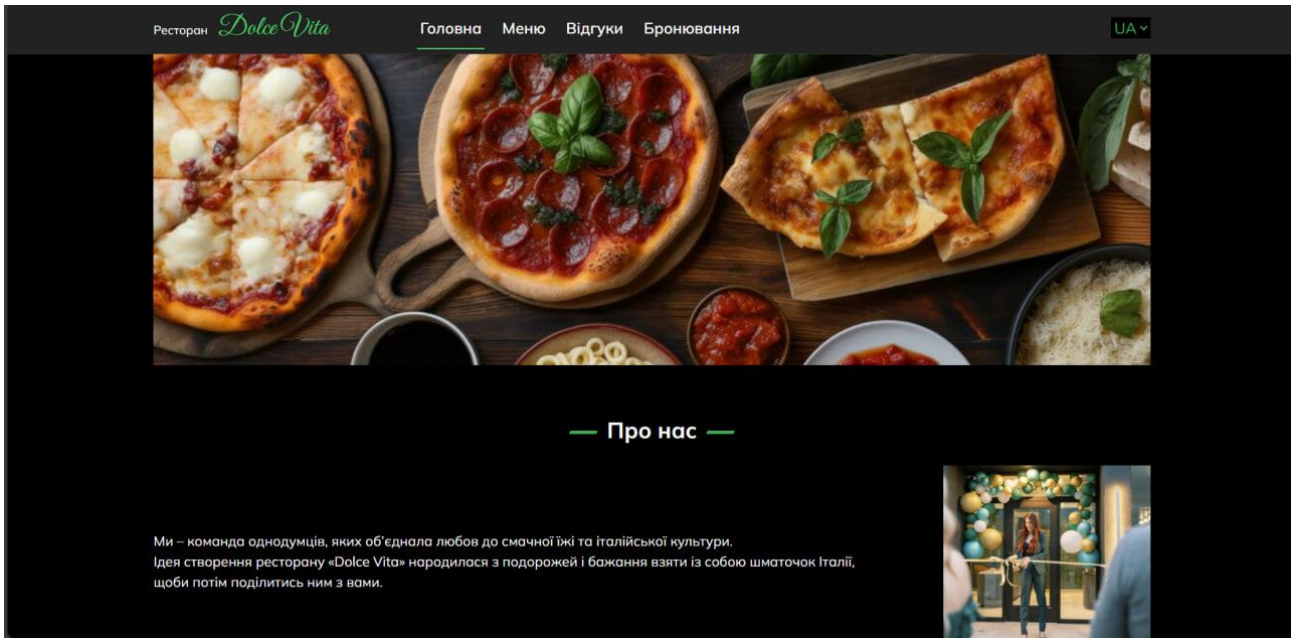

Рисунок Г.1 – Головна сторінка WEB-ресурсу «Ресторан»

Сайт призначений для ознайомлення з рестораном, перегляду меню, читання та залишення відгуків, а також здійснення онлайн-бронювання столиків.

Вгорі розміщена навігаційна панель, яка містить посилання на такі сторінки: «Головна сторінка», «Меню», «Відгуки» та «Бронювання». Також у правій частині панелі є перемикач мови.

Для перегляду страв потрібно натиснути на вкладку «Меню». На сторінці «Меню» (рис. Г.2) можна переглянути список страв, згрупованих за категоріями (закуски, основні страви, десерти тощо). Кожна страва має назву, опис, фото і ціну.

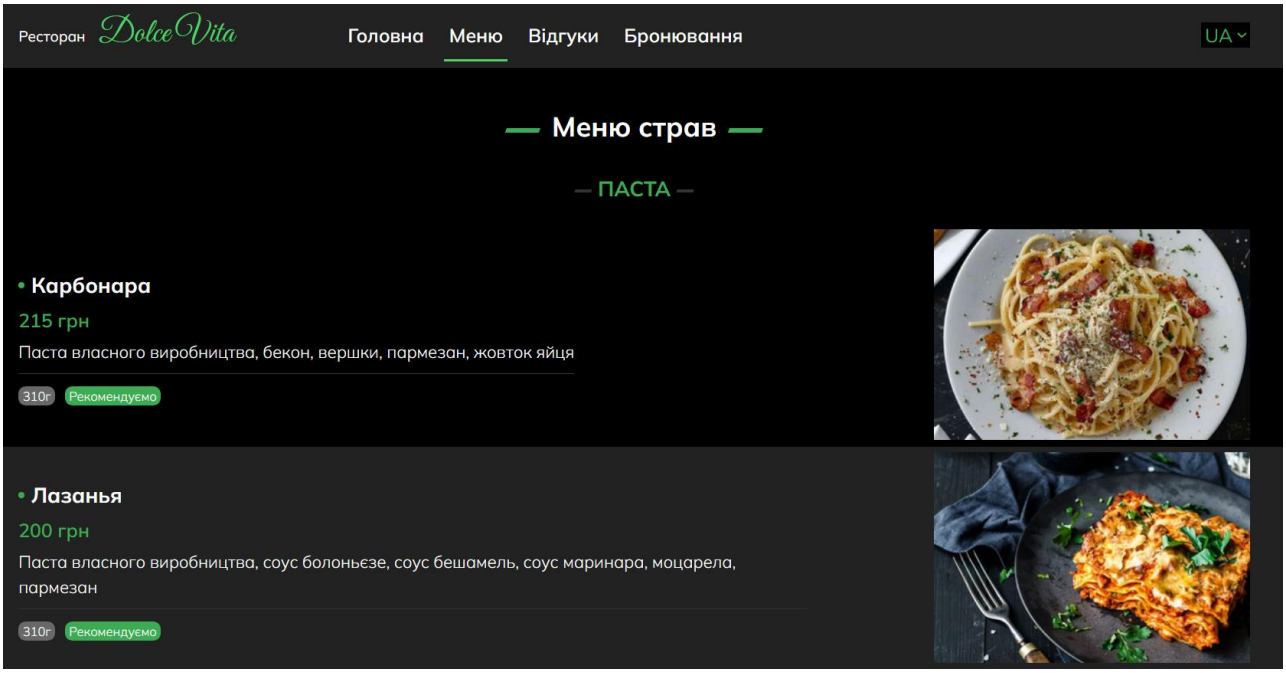

Рисунок Г.2 – Сторінка «Меню»

На сторінці «Відгуки» можна ознайомитися з відгуками попередніх відвідувачів. Це допомагає новим користувачам сформувати уявлення про сервіс і якість страв. Щоб залишити власний відгук, потрібно у формі для відгуків (рис. Г.3) заповнити поля «Ім’я», «Відгук», обрати оцінку та натиснути кнопку «Залишити відгук».

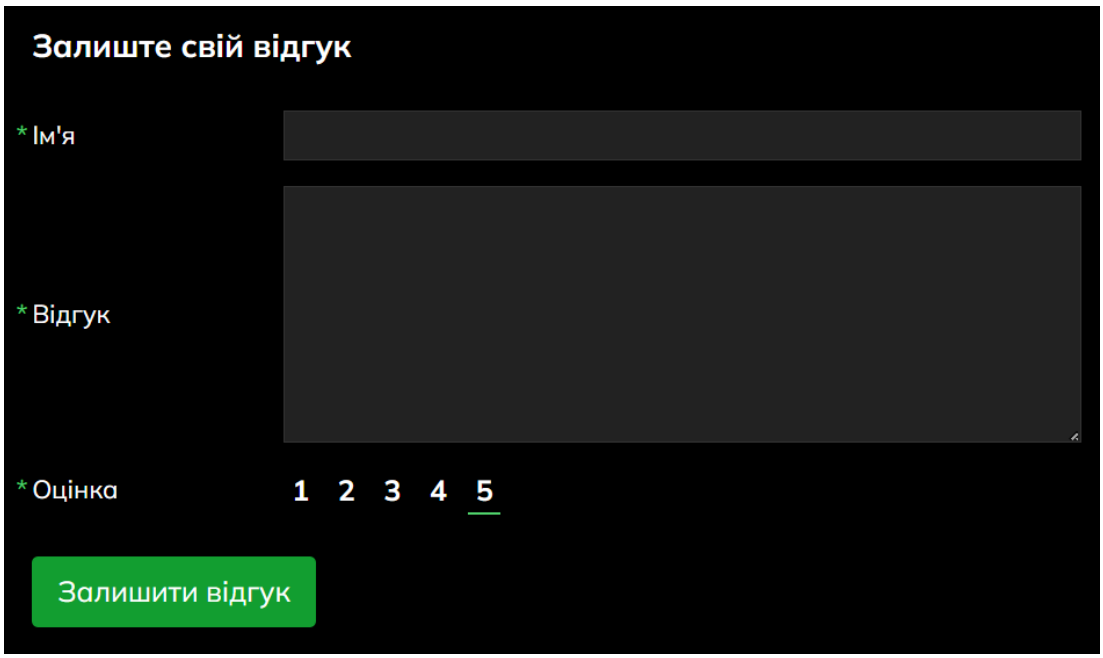

Рисунок Г.3 – Форма для відгуків

Після цього відгук відобразиться на сторінці сайту, що зображено на рисунку Г.4.

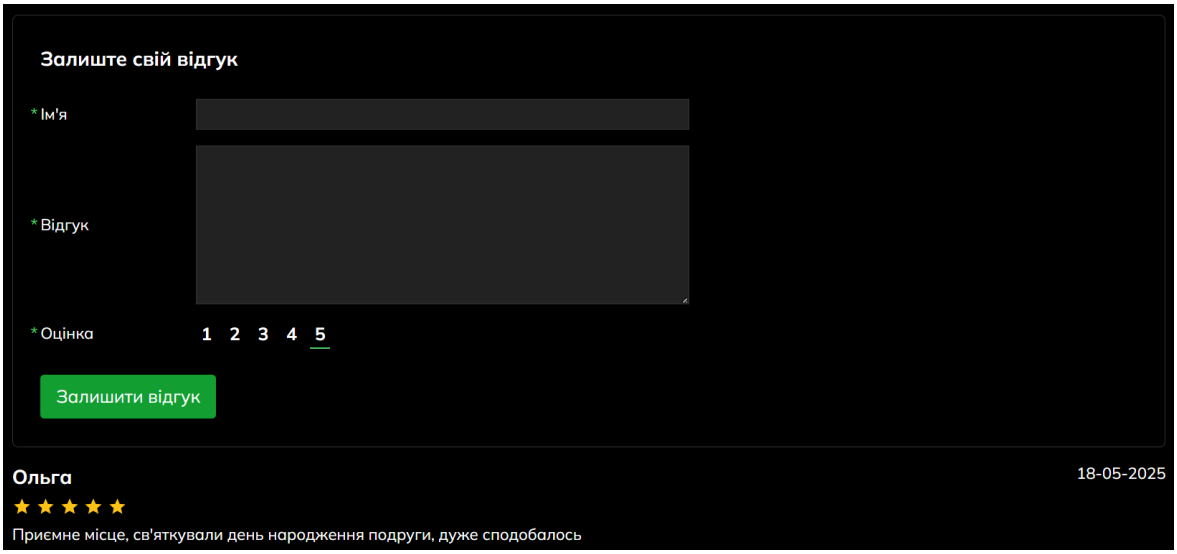

Рисунок Г.4 – Секція з відгуками

Щоб забронювати столик, потрібно перейти у розділ «Бронювання» та заповнити форму для бронювання (рис. Г.5).

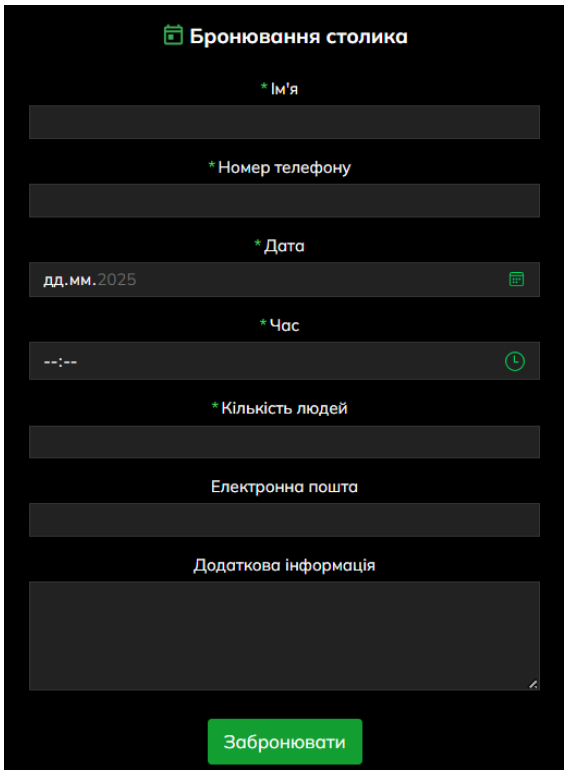

Рисунок Г.5 – Форма для бронювання

Потрібно вказати інформацію в полях «Ім’я», «Номер телефону», «Дата», «Час», «Кількість людей», «Електронна пошта» та «Додаткова інформація» та натиснути кнопку «Забронювати». Обов'язкові для заповнення поля відмічені символом «*». Якщо все правильно заповнено, після надсилання форми система автоматично сформує повідомлення і відправить його адміністратору електронною поштою (рис. Г.6). Користувач отримає підтвердження бронювання після обробки заявки.

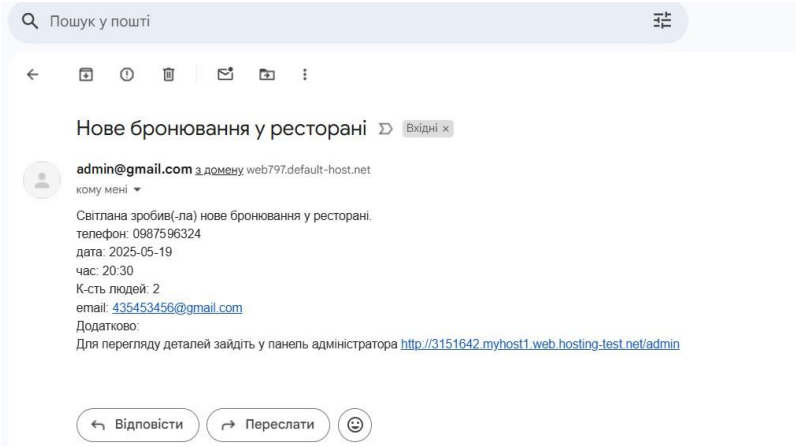

Рисунок Г.6 – Повідомлення про бронювання на пошті адміністрації

Щоб увійти на сторінку адміністратора, на якій можна редагувати зміст сторінок сайту, потрібно перейти за посиланням http://3151642.myhost1.web.hosting-test.net/admin та ввести логін та пароль адміністратора у відповідну форму, зображену на рисунку Г.7. На мобільному пристрої все працює так само.

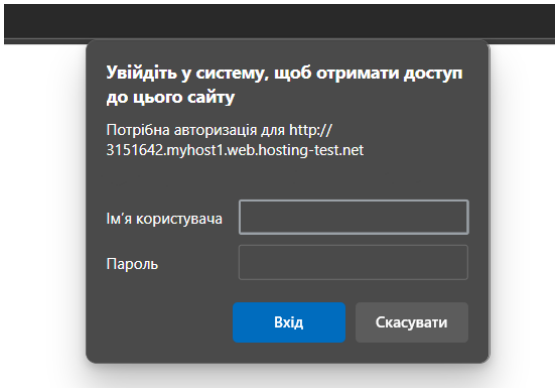

Рисунок Г.7 – Форма для входу на сторінку адміністратора