ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра інформаційних технологій

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Бакалавр»

«Розробка веб-ресурсу для ресторану з використанням технології ASP.NET Core»

(тема кваліфікаційної роботи українською мовою)

«Development of a web resource for a restaurant using ASP.NET Core technology»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання

спеціальності 122 Комп'ютерні науки

(код, назва спеціальності)

Освітня програма Комп'ютерні науки

(назва)

Присуха Микола Володимирович

(прізвище, ім'я, по-батькові здобувача)

Керівник асистент Гадяцький І.А.

(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент к.т.н., доцент Щербіна Ю.В.

(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій .
№ 1 від 09 червня 2024 р.

Завідувачка кафедри

(підпис) КАЗАКОВА Надія .
(прізвище,ім'я)

Захищено на засіданні ЕК № 13 .
протокол № 24 від 21 червня 2024 р.

Оцінка добре / B / 85 .
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК

(підпис) КОПИЧЕНКО Іван .
(прізвище,ім'я)

Одеса 2024

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

MVC – Model-View-Controller – архітектурний шаблон, який використовується під час проєктування та розробки програмного забезпечення.

СУБД – Система керування базами даних – набір взаємопов'язаних даних (база даних) і програм для доступу до цих даних

API – Application Programming Interface – опис способів взаємодії однієї комп'ютерної програми з іншими

ORM – Object-Relational Mapping – технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування

JWT – JSON Web Token – це стандарт токена доступу на основі JSON

JSON – JavaScript Object Notation – текстовий формат обміну даними між комп'ютерами

ІС – інформаційна система.

GPL – GNU General Public License – загальна публічна ліцензія GNU.

Хостинг – послуга з надання простору для розміщення сайтів в Інтернеті.

Apache – вільний WEB-сервер.

CSS – Cascading Style Sheets – каскадні таблиці стилів.

HTML – HyperText Markup Language – мова гіпертекстової розмітки.

MySQL – вільна система управління базами даних.

PHP – Hypertext Preprocessor – препроцесор гіпертекста.

ODBC – OpenDatabaseConnectivityStandard – стандарт підключення до відкритих баз даних.

SADT – Structured Analysis and Design Technique – методологія структурного аналізу і проектування.

JS – JavaScript – динамічна, об'єктно-орієнтована прототипна мова програмування.

ВСТУП

В сучасному інформаційному суспільстві ресторанна галузь постійно розвивається, шукаючи нові та інноваційні підходи до забезпечення якісного обслуговування клієнтів. У цьому контексті велике значення набуває використання веб-технологій для покращення ефективності та зручності роботи ресторанного бізнесу. Дипломний проект, присвячений розробці веб-ресурсу для ресторану з використанням технології ASP.NET, спрямований на створення модернізованого інструменту, який не лише полегшить процес управління ресторанним бізнесом, але й покращить взаємодію з клієнтами.

Технологія ASP.NET визнана однією з передових у сфері веб-розробки, що дозволяє створювати потужні та надійні веб-додатки з високою продуктивністю. Застосування цієї технології у проекті дозволить реалізувати широкий спектр функцій, від управління меню та замовленнями до ведення обліку і взаємодії з клієнтською базою. В результаті, веб-ресурс стане не лише важливим інструментом для оптимізації внутрішніх процесів ресторану, а й забезпечить зручний і привабливий інтерфейс для клієнтів, сприяючи підвищенню їхньої задоволеності та лояльності.

Проект передбачає використання сучасних методів дизайну та веб-розробки для створення інтуїтивно зрозумілого і естетичного інтерфейсу веб-ресурсу. Застосування дизайнерських рішень та графічних елементів допоможе підкреслити унікальний стиль ресторану та зробить користування веб-сайтом максимально комфортним для відвідувачів.

Крім того, важливо відзначити, що розробка веб-ресурсу з використанням технології ASP.NET також відкриває можливості для інтеграції з іншими системами управління, такими як системи обліку товарів, фінансові платформи та системи аналітики. Це створить цілісний інформаційний простір для ефективного управління всіма аспектами ресторанного бізнесу та надасть можливість швидко реагувати на зміни в індустрії та потреби клієнтів.

Не останню роль відіграє і питання безпеки, яке має важливе значення в галузі обробки конфіденційної інформації та електронних платежів. Використання технології ASP.NET забезпечить високий рівень захисту даних та персональної інформації, що стане гарантом надійності та конфіденційності для як ресторанного персоналу, так і для клієнтів веб-ресурсу.

Дипломна робота містить в собі 53 сторінки, 7 таблиць, 21 рисунок та 10 посилань.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Актуальність розробки веб-ресурсу для ресторанного бізнесу

У сучасному світі, де технології швидко розвиваються, ресторанний бізнес стає неабиякою лабораторією для впровадження новацій та високотехнологічних рішень. Актуальність розробки веб-ресурсу для ресторану надзвичайно велика, оскільки він відкриває безліч можливостей для оптимізації бізнес-процесів та поліпшення взаємодії з клієнтами.

З появою онлайн-платформ та змінами в споживчих практиках, споживачі все більше висловлюють попит на швидкі та зручні способи вибору ресторанів, ознайомлення з їхнім меню та замовлення страв чи столиків. Розробка веб-ресурсу стає стратегічно важливим етапом для ресторанного бізнесу в умовах конкуренції, дозволяючи привертати та утримувати клієнтів, а також оптимізувати внутрішні операції.

Проект спрямований на створення інноваційного веб-ресурсу, який надасть ресторану можливість ефективно конкурувати на ринку, надаючи клієнтам зручні та привабливі сервіси. Розробка буде акцентувати увагу на покращенні взаємодії з клієнтами, оптимізації управління замовленнями та внутрішніми процесами, що сприятиме підвищенню конкурентоспроможності та розвитку ресторанного бізнесу в цифрову еру [1].

В сучасному світі інтернет-технології перетворили не лише спосіб, яким ми спілкуємося та отримуємо інформацію, але й спосіб, яким ресторани взаємодіють із своєю аудиторією. З'явлення онлайн-платформ для ресторанів визначає нові стандарти в галузі обслуговування та управління бізнесом, надаючи значущий вплив на їх функціонування та взаємодію з клієнтами.

Перш за все, онлайн-платформи дозволяють ресторанам забезпечити своїй аудиторії простий та швидкий доступ до інформації про меню, акції та спеціальні пропозиції. Клієнти можуть зручно переглядати весь асортимент, детально досліджувати страви та знайомитися з відгуками інших гостей перед

тим, як прийняти рішення щодо вибору ресторану.

Додатково, онлайн-платформи дозволяють здійснювати онлайн-замовлення, що стає важливим фактором для забезпечення швидкості та зручності обслуговування. Клієнти можуть здійснити замовлення на доставку або резервування столиків, оптимізуючи свій час та максимізуючи зручність відвідування ресторану.

Загальне значення онлайн-платформ для ресторанів полягає в покращенні комунікації, збільшенні видимості та привабливості бізнесу в онлайн-середовищі. Це стає ключовим елементом для успішного ведення ресторанного бізнесу в умовах сучасного ринкового середовища.

Однією з основних переваг використання онлайн-платформ є можливість побудови ефективної системи лояльності та залучення нових клієнтів.

Ресторани, які інтегруються в онлайн-середовище, можуть запроваджувати програми винагородження, розсилати персоналізовані пропозиції та спеціальні знижки, що робить їх більш привабливими для постійних та потенційних гостей. Це не лише сприяє збільшенню обігу ресторану, але і підвищує рівень задоволеності клієнтів, що може призвести до позитивного впливу на репутацію закладу.

Застосування онлайн-платформ також відкриває ресторанам можливість збору та аналізу даних про свою аудиторію. Великий обсяг інформації, що виникає з онлайн-замовлень та резервацій, дозволяє вдосконалювати стратегії ціноутворення, асортимент страв та рекламних кампаній. Аналіз поведінки клієнтів стає основою для розробки персоналізованих підходів та оптимізації відповідно до попиту та тенденцій.

Неабияке значення має і взаємодія з рейтинговими та відгуковими платформами. Інтеграція з такими сервісами надає можливість ресторанам ефективно взаємодіяти з власним репутаційним образом. Високі рейтинги та позитивні відгуки можуть значно підвищити привабливість ресторану для нових клієнтів, сприяючи його популярності та розвитку в конкурентному середовищі.

Узагальнюючи, актуальність використання онлайн-платформ для ресторанного бізнесу полягає в їхній здатності забезпечити більш ефективну комунікацію, високий рівень зручності для клієнтів, а також в унікальних можливостях аналізу даних та взаємодії з рейтинговими платформами, що в сукупності сприяє успішному функціонуванню та розвитку ресторанного бізнесу.

Сучасні веб-ресурси ресторанів стали ключовим інструментом для взаємодії з клієнтами та покращення обслуговування. Їхні функціональні можливості та дизайн грають важливу роль у формуванні першого враження від закладу, а також впливають на рішення клієнтів щодо вибору ресторану. Аналізуючи такі веб-ресурси, можна визначити ряд ключових тенденцій та елементів успішного віртуальної присутності.

Починаючи з головної сторінки, багато веб-ресурсів ресторанів намагаються максимально зручно та привабливо представити своє меню та атмосферу. Інтерактивність та візуальна привабливість грають важливу роль, надаючи клієнтам можливість швидко ознайомитися з різноманіттям страв та їх описами. Важливим аспектом є також наявність фотографій страв, що відображають їхню привабливість та стимулюють апетит.

Онлайн-системи замовлення та резервації є необхідними компонентами сучасного веб-ресурсу ресторану. Клієнти очікують можливості легко та швидко здійснювати замовлення онлайн або бронювати столики, оптимізуючи свій час та забезпечуючи зручність обслуговування.

Спеціальні розділи для відгуків та рейтингів грають важливу роль у взаємодії зі споживачами. Активна участь ресторану у відповідях на відгуки та позитивні рейтинги сприяє підвищенню довіри та репутації.

Окрім цього, важливим є впровадження елементів персоналізації, таких як програми лояльності та індивідуальні рекомендації. Забезпечення персоналізованого підходу підвищує залученість та стимулює клієнтів до повторних відвідувань.

У підсумку, аналіз сучасних веб-ресурсів ресторанів вказує на важливість інновацій, ефективного дизайну та високофункціональних можливостей

для підтримки високого рівня обслуговування та конкурентоспроможності у галузі ресторанного бізнесу.

1.2 Аналіз предметної області

Ресторан пропонує клієнтам різноманітні страви та напої, орієнтуючись на високу якість обслуговування та унікальну атмосферу. Основні напрямки діяльності включають обслуговування гостей у залі, замовлення їжі

Основні бізнес-процеси:

- Включає прийом замовлень, подачу страв і розрахунок;
- Включає прийом і обробку замовлень через сайт, приготування страв;
- Оновлення інформації про страви, ціни;
- Збір відгуків, обробка запитів і вирішення питань клієнтів.

Прийом і обслуговування відвідувачів у залі охоплює весь спектр взаємодії з клієнтами, починаючи з моменту їхнього прибуття до ресторану і завершуючи оплатою рахунку. Він включає:

- співробітники ресторану зустрічають відвідувачів, допомагають їм знайти вільний стіл і розсаджують їх;
- офіціанти приймають замовлення на страви і напої, консультують клієнтів щодо меню та спеціальних пропозицій;
- після приготування страви подаються до столу. Офіціанти стежать за тим, щоб страви подавалися у відповідний час і мали належну температуру;
- після завершення трапези офіціанти приносять рахунок і приймають оплату. Важливо забезпечити клієнтам різноманітні способи оплати, такі як готівка, банківські картки та мобільні платежі.

Онлайн-замовлення і доставка їжі забезпечує клієнтів можливістю замовляти страви через веб-сайт ресторану. Він включає:

- клієнти обирають страви з меню на сайті, додають їх до кошика та оформлюють замовлення. Система автоматично передає замовлення

на кухню для приготування;

– кухарі отримують замовлення та починають приготування. Важливо дотримуватися всіх стандартів якості та санітарних норм;

– після приготування страви подається на стіл.

Оновлення інформації про страви і ціни важливий для підтримки актуальності меню на веб-сайті та в друкованих матеріалах ресторану. Він включає:

– при введенні нових позицій у меню адміністратори додають інформацію про страви, їхні опис, ціни на сайт;

– у разі змін у рецептурі, складі інгредієнтів або цін, інформація на сайті також оновлюється. Це допомагає уникнути непорозумінь і забезпечити клієнтам точну інформацію [2].

1.3 Постановка завдання

Завдання розробки веб-ресурсу для ресторану полягає у створенні зручного, функціонального та безпечного онлайн-простору, що слугуватиме не лише вітриною для представлення меню та послуг, а й повноцінним інструментом для взаємодії з клієнтами. Мета проєкту - надати користувачам можливість легко знаходити інформацію про ресторан, бронювати столики, замовляти їжу онлайн, а також отримувати актуальні новини та спеціальні пропозиції. Додатково, система має давати змогу адміністраторам і працівникам ресторану ефективно управляти контентом, замовленнями і бронюваннями, а також аналізувати статистику і взаємодіяти з клієнтами.

У результаті реалізації проєкту буде створено веб-ресурс, що включає такі компоненти:

– вітальне повідомлення, актуальні новини та спеціальні пропозиції;

– докладне меню з фотографіями страв, описами та цінами;

– адреса ресторану, контактні дані, інтерактивна карта та форма зворотного зв'язку;

– можливість створення облікового запису, входу через соціальні мережі та відновлення пароля;

– вибір страв, додавання в кошик, оформлення замовлення, вибір способу оплати;

– додавання та оновлення страв, зміна цін;

– перегляд і обробка замовлень і бронювань, що надходять;

– додавання і видалення співробітників, призначення ролей і прав доступу;

– захист даних користувачів, безпечні методи аутентифікації та авторизації, SSL-шифрування;

– система повідомлень про нові акції, зміни в меню і статус замовлень;

– адреса ресторану, контактні дані, інтерактивна карта та форма зворотного зв'язку.

2 ВИБІР АРХІТЕКТУРИ СИСТЕМИ ТА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

Для успішної розробки веб-ресурсу для ресторану, необхідно ретельно провести технічний аналіз та визначити оптимальний технологічний стек, який відповідатиме вимогам проекту. У цьому розділі розглянемо критичні аспекти вибору мови розробки, системи управління базами даних, середовища зберігання, а також бібліотек та фреймворків на основі .NET для оптимізації розробки та функціональності веб-ресурсу.

2.1 Обґрунтування вибору мови розробки

Вибір технологічного стеку є важливою стратегічною відправною точкою при розробці веб-ресурсу для ресторану. Технологія ASP.NET виявляється ключовою, надаючи ряд переваг, які визначають успішність та ефективність функціонування веб-проекту.

Однією з ключових переваг використання ASP.NET є його висока продуктивність та швидкодія. Завдяки вбудованим інструментам оптимізації та механізмам кешування, технологія забезпечує ефективну роботу великих та складних веб-систем, що є важливим аспектом для ресторанного бізнесу, де швидкість реакції та завантаження графіків меню можуть визначати задоволення клієнтів.

Додатково, ASP.NET пропонує високий рівень безпеки. Інтеграція засобів аутентифікації та авторизації дозволяє надійно захищати конфіденційну інформацію клієнтів та ресторанного бізнесу, а також забезпечує захист від потенційних кібератак [3].

Зручна інтеграція з базами даних також визначає вагомість ASP.NET у розробці веб-ресурсу для ресторану. Здатність ефективно управляти та оптимізувати великі обсяги даних про меню, замовлення та клієнтську інформацію стає ключовим фактором для успішного функціонування ресторанного

веб-ресурсу.

Наприклад, використання ASP.NET дозволить ефективно реалізувати функції обробки замовлень, систему резервацій та ведення бази клієнтів, надаючи адміністраторам ресторану інструменти для зручного управління та аналізу даних.

Узагальнюючи, технологія ASP.NET в розробці веб-ресурсу для ресторану визначається своєрідним синергетичним ефектом ефективності, безпеки та зручності і є стратегічним вибором для досягнення високої продуктивності та задоволення вимог сучасних веб-проектів.

Unit тестирование є важливою частиною розробки програмного забезпечення і грає ключову роль у впевненні в правильності та надійності коду. У контексті проекту з веб-ресурсу для ресторану, використання NUnit дозволить створювати тести для перевірки окремих компонентів програми, таких як функції обробки замовлень, взаємодії з базою даних та інші аспекти бізнес-логіки. Це сприяє забезпеченню високої якості коду та швидкого виявлення можливих помилок під час розробки.

.NET 7 є новою версією фреймворку .NET, яка принесла ряд нововведень та поліпшень. У контексті розробки веб-ресурсу для ресторану, використання .NET 7 може дозволити користуватися останніми технологічними можливостями, поліпшити продуктивність, забезпечити більші можливості для оптимізації та використовувати нові функції для покращення взаємодії з клієнтами та адміністраторами.

Ідея використання QR кодів для замовлення блюд є вельми зручною та актуальною. Кожне блюдо може мати свій унікальний QR код, який клієнт може зісканувати за допомогою мобільного телефону. Це відкриє спеціальну сторінку на веб-ресурсі, де клієнт може переглядати інформацію про блюдо, додавати його до кошика та робити замовлення.

OAuth 2.0 – це протокол авторизації, який може бути використаний для забезпечення безпечного та зручного входу в систему. Застосування OAuth 2.0 у веб-ресурсі для ресторану дозволить користувачам входити,

використовуючи свої облікові записи з інших платформ (наприклад, Google, Facebook), що полегшить процес авторизації та реєстрації.

SignalR – це технологія для розробки реального часу веб-застосунків. Використання SignalR у веб-ресурсі для ресторану дозволить реалізувати функціонал миттєвого оновлення інформації про замовлення. Наприклад, коли нове замовлення надходить, воно може автоматично з'являтися на екрані кухні або в інтерфейсі адміністратора без необхідності оновлення сторінки.

Паттерн Unit of Work дозволяє зберігати стан об'єктів та забезпечує виконання всіх змін в одній транзакції. У контексті веб-ресурсу для ресторану, використання Unit of Work полегшить управління транзакціями бази даних під час обробки замовлень та інших операцій.

Паттерн Repository дозволяє відокремити логіку доступу до даних від решти програми. Це полегшить роботу з базою даних та забезпечить структурованість коду при взаємодії з даними ресторанного веб-ресурсу.

Паттерн MVC використовується для відокремлення логіки бізнесу, представлення та управління подіями в програмному забезпеченні. Його використання в розробці веб-ресурсу для ресторану дозволить створити добре структуровану та легко розширювану систему, в якій можливі зміни в одному компоненті не впливатимуть на інші [4].

2.2 Вибір системи управління базами даних

Microsoft SQL Server обрано як систему управління базами даних (СУБД) з урахуванням його надійності та продуктивності. Його інтегровані засоби забезпечують ефективне зберігання, управління та аналіз даних, що є критичним для ресторанного веб-ресурсу. Рішення засноване на SQL Server буде забезпечувати надійність операцій з базою даних та гнучкість у роботі з великим обсягом даних.

Вибір системи управління базами даних (СУБД) є критичним етапом при розробці веб-ресурсу для ресторану. В контексті проекту обрано Microsoft

SQL Server, одну з найпопулярніших та найрозповсюдженіших реляційних СУБД, розроблену корпорацією Microsoft. Давайте розглянемо ключові переваги та властивості Microsoft SQL Server, які роблять її оптимальним вибором для даного проекту:

1. Microsoft SQL Server відомий своєю високою надійністю та стабільністю. Вона дозволяє ефективно керувати об'ємами даних, забезпечуючи стабільну роботу в умовах високого навантаження, що є важливим для веб-ресурсу ресторану з активним обігом замовлень та іншої інформації;
2. SQL Server володіє потужною оптимізацією запитів та механізмами кешування, що дозволяє виконувати операції з базою даних швидко та ефективно. Це особливо важливо для веб-ресурсу, де швидкість обробки замовлень та відображення інформації є ключовими;
3. SQL Server надає широкий спектр можливостей, таких як тригери, процедури, визначені користувачем функції та інші засоби, що полегшують роботу з базою даних. Зручний інтерфейс управління, такий як SQL Server Management Studio (SSMS), надає розробникам та адміністраторам зручні інструменти для налагодження та оптимізації роботи з базою;
4. Засоби безпеки Microsoft SQL Server включають аутентифікацію, авторизацію та шифрування даних. Це робить її надійним вибором для проектів, де важлива конфіденційність та захист даних, так як у ресторанному бізнесі може бути важливо зберігати особисті дані клієнтів та іншу конфіденційну інформацію;
5. SQL Server ідеально інтегрується з іншими продуктами та технологіями в екосистемі Microsoft, такими як .NET Framework та Azure. Це полегшує розробку та підтримку веб-ресурсу, підвищуючи ефективність розробки.

Узагальнюючи, вибір Microsoft SQL Server для управління базою даних у веб-ресурсі для ресторану обґрунтовується його надійністю, високою

продуктивністю, широким функціоналом та зручним інтерфейсом, а також легкою інтеграцією з іншими технологіями Microsoft.

Вибір ефективного середовища для зберігання та керування версіями коду є ключовим етапом у розробці програмного забезпечення. Для проекту веб-ресурсу для ресторану обрано GitHub, і ось деякі аргументи, які обґрунтовують це рішення:

1. GitHub надає потужні інструменти для керування версіями коду. Система контролю версій Git, яка використовується GitHub, дозволяє ефективно відстежувати зміни в коді, створювати та об'єднувати гілки, що полегшує паралельну розробку та тестування різних функціональностей;
2. GitHub створений для співпраці. Команди розробників можуть легко спільно працювати над проектом, роблячи покращення, рецензуючи код та взаємодіючи через питання та обговорення. Розподілена розробка в GitHub дозволяє командам працювати над проектом, не залежачи від географічного розташування;
3. GitHub інтегрується з великою кількістю інших інструментів для автоматизації різних аспектів розробки, таких як CI/CD системи, сервіси тестування, засоби розгортання та інші. Це забезпечує неперервну інтеграцію та доставку, спрощуючи процес розробки та випуску нових версій програмного забезпечення;
4. GitHub має зручний веб-інтерфейс, який полегшує перегляд та взаємодію з кодом, відстеження змін та створення питань. Також, можливість додавати документацію до проекту та вести обговорення через питання дозволяє забезпечити чітку документацію та вирішення можливих питань;
5. GitHub надає ефективні засоби для забезпечення безпеки коду та даних. Регульярні аудити, можливість обмежити доступ до приватних репозиторіїв, а також можливість використовувати двофакторну аутентифікацію забезпечують високий рівень захисту.

GitHub – це онлайн-сервіс зберігання та синхронізації коду для програмістів та розробників додатків. Головною метою цього сервісу є підтримка спільної розробки проектів та контролю версій (рис. 2.1).

Рисунок 2.1 – Інтерфейс GitHub

Git Bash – це програма для середовищ Microsoft Windows, що емулює роботу командного рядка Git. Bash – абревіатура від Bourne Again Shell. Оболонка (Shell) є додатком терміналу для взаємодії з операційною системою за допомогою письмових команд (рис. 2.2).

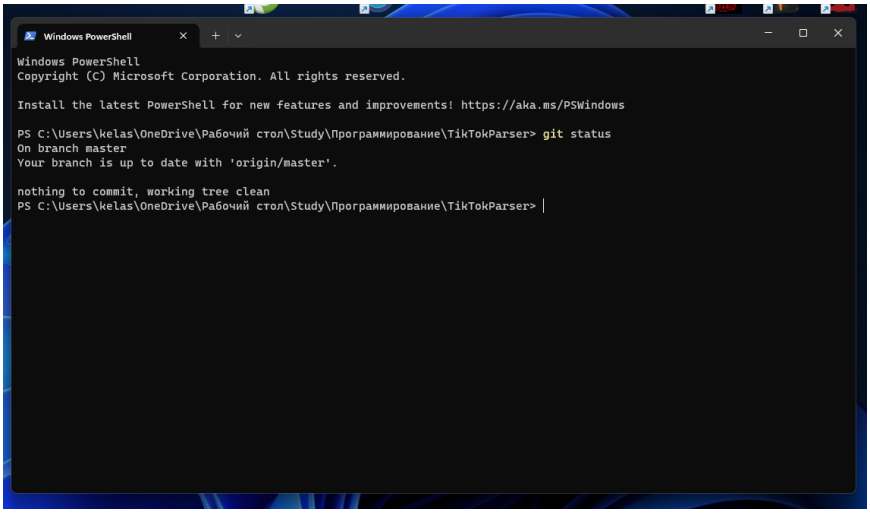

Рисунок 2.2 – Інтерфейс GitBash

2.3 Вибір бібліотек та фреймворку .NET

Розробка веб-ресурсу для ресторану потребує ретельного вибору бібліотек та фреймворків для побудови надійної та функціональної системи [5]. Для оптимального функціоналу та швидкого розгортання вибрано ряд ключових бібліотек та фреймворків з екосистеми .NET:

1. ASP.NET Core став основним фреймворком для розробки веб-ресурсу. Цей фреймворк від Microsoft надає високу продуктивність, переносимість та ефективність. ASP.NET Core дозволяє розробникам писати високоякісний код та швидко впроваджувати нові функції завдяки модульній архітектурі;
2. для взаємодії з базою даних використовується Entity Framework Core - ORM (Object-Relational Mapping), що полегшує роботу з базою даних та дозволяє працювати з об'єктами даних безпосередньо в коді, спрощуючи роботу з реляційними таблицями;
3. для забезпечення реального часу та миттєвої комунікації використовується SignalR. Ця бібліотека дозволяє надсилати повідомлення та оновлення клієнтам в реальному часі, що особливо важливо для відстеження статусу замовлень та сповіщення користувачів про нові події;
4. для тестування розроблених модулів та функціоналу використовується бібліотека NUnit. Вона дозволяє створювати та виконувати тести для перевірки коректності роботи різних частин системи;
5. автентифікація та авторизація реалізована за допомогою протоколу OAuth 2.0. Це забезпечує безпеку та взаємодію з іншими сервісами, що можуть бути використані для реєстрації та входу;
6. для зручного замовлення страв із меню використовуються QR-коди. Дана функціональність реалізована з використанням бібліотек.

В проекті застосовуються паттерни проєктування Unit of Work та Repository для ефективної роботи з базою даних. Архітектурний паттерн MVC

(Model-View-Controller) використовується для відокремлення логіки відображення, бізнес-логіки та управління станом додатку.

2.4 Пояснення праці ресурсу

Проект розроблено мовою програмування C# з використанням потужного фреймворку ASP.NET Core. ASP.NET Core входить до складу обширної платформи .NET, що надає високий рівень гнучкості та ефективності для розробки веб-застосунків.

Центральним елементом архітектури проекту є паттерн Model-View-Controller (MVC) фреймворку ASP.NET Core. Цей паттерн розподілення відповідальностей дозволяє зробити модель (Model) незалежною від контролера (Controller) та представлення (View), що спрощує розробку та підтримку коду.

ASP.NET Core принесе до вашого проекту численні переваги, які варто відзначити:

- відкриті вихідні коди: Можливість доступу до вихідного коду відкриває широкі можливості для внесення змін та вдосконалення фреймворку згідно з власними потребами проекту;
- потужні пакетні менеджери: ASP.NET Core використовує потужні пакетні менеджери, такі як NuGet або Paket, що спрощує управління залежностями та пакетами в проекті;
- тег-хелпери: Ці зручні інструменти дозволяють спрощувати та полегшувати роботу з HTML-кодом в середині розмітки сторінок;
- вбудований bootstrap та AJAX: Наявність вбудованих засобів для роботи з Bootstrap і AJAX полегшує розробку динамічних та привабливих веб-сторінок;
- покращена обробка помилок: ASP.NET Core забезпечує ефективну обробку помилок під час виконання, дозволяючи вам ефективно виявляти та вирішувати проблеми у запущених програмах за допомогою блоків try-catch.

Ці переваги створюють робочий середовище, яке прискорює розробку та підтримку вашого веб-проекту, забезпечуючи високий рівень якості та зручності в програмуванні.

Розробка архітектури веб-проекту є критичним етапом, оскільки вона визначає структуру та взаємодію компонентів системи. В данному випадку використовується архітектура з розділенням на рівні бізнесу, що дозволяє ефективно управляти функціональністю та розділяти обов'язки між різними компонентами:

- HTML, CSS, та JavaScript визначають вигляд та інтерактивність вебсторінок;
- ASP.NET MVC використовується для розділення логіки бізнесу, представлення та обробки взаємодії з користувачем;
- ASP.NET Web API надає інтерфейс для взаємодії з клієнтським рівнем та обробляє HTTP-запити;
- unit тестування використовується для автоматизованого тестування функціоналу бізнес-логіки;
- Microsoft SQL Server: Система управління базами даних для зберігання та взаємодії з даними;
- Entity Framework: ORM для спрощення роботи з базою даних та взаємодії з об'єктами в програмному коді;
- використовується для зберігання та відстеження змін у кодові проекту, спрощуючи роботу в команді;
- ASP.NET Identity використовується для реалізації системи аутентифікації та авторизації користувачів;
- SignalR впроваджений для забезпечення функціоналу реального часу, зокрема відстеження статусу замовлень та сповіщення.

3 ПРОЕКТУВАННЯ МЕРЕЖЕВОГО WEB-РЕСУРСУ ЗА ДОПОМОГОЮ МЕТОДОЛОГІЙ ФУНКЦІОНАЛЬНОГО МОДЕЛЮВАННЯ

3.1 Проектування мережевого WEB-ресурсу за допомогою методології функціонального моделювання SADT

При проектуванні інтернем-магазину була обрана методологія функціонального моделювання SADT, стандарт IDEF0. Методологія SADT, яка є методологією аналізу та проектування систем, вона дозволяє відобразити важливі характеристики: управління, зворотній зв'язок і ресурси [6]. Особливість SADT полягає в тому, що ця методологія здійснює загальний опис функціонування системи. За допомогою графічної мови IDEF0, мережевий WEB-ресурс компанії, яка надання послуги перегляду та замовлення страв, можливо представити у вигляді набору взаємопов'язаних функціональних блоків, що дозволить вивчити загальне функціонування системи. Контекстна діаграма мережевого WEB-ресурсу компанії, яка надає послуги перегляду та замовлення страв, наведена на рис. 3.1.

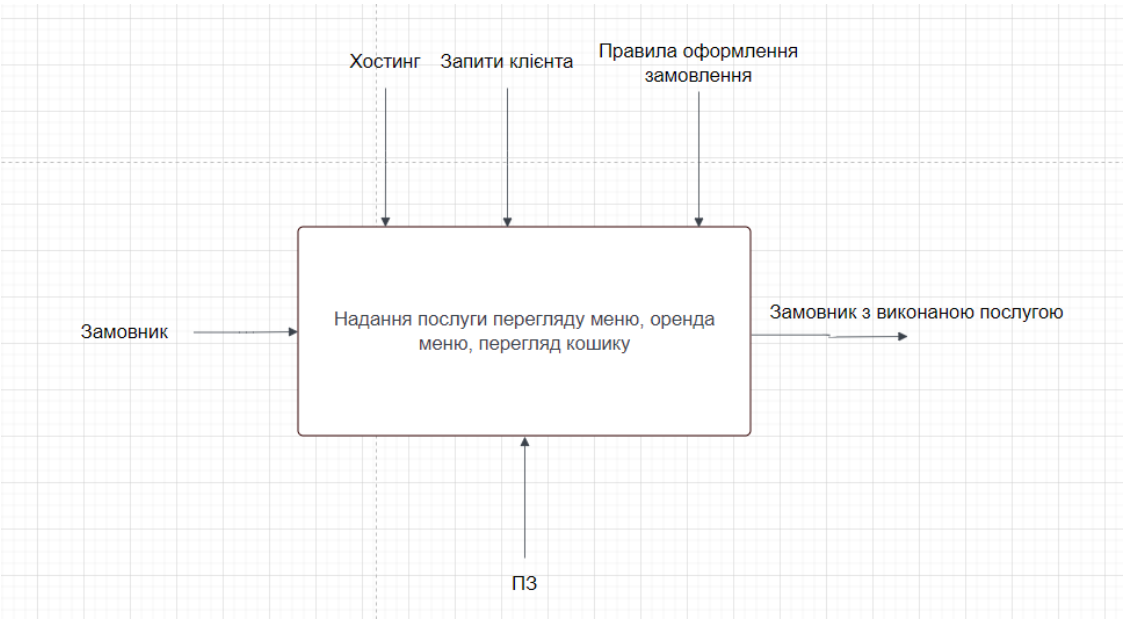

Рисунок 3.1 – Контекстна діаграма мережевого WEB-ресурсу

Методологія SADT є важливим інструментом для розробки інформаційних систем, оскільки вона надає розробникам набір методів, правил і процедур для аналізу та побудови функціональної моделі обраної предметної області. Одним з ключових компонентів SADT є графічна мова IDEF0, яка дозволяє описати бізнес-процеси у вигляді ієрархічної системи взаємопов'язаних функцій.

Контекстна діаграма є важливим етапом у розробці інформаційних систем, оскільки вона надає загальний опис інформаційної системи та її взаємодії з зовнішнім середовищем. Після створення головної роботи онлайн ресторану, проводиться декомпозиція, яка полягає у розбитті системи на менші частини. Це дозволяє докладно описати кожну з них та їх функції, а також з'ясувати взаємозв'язок між ними.

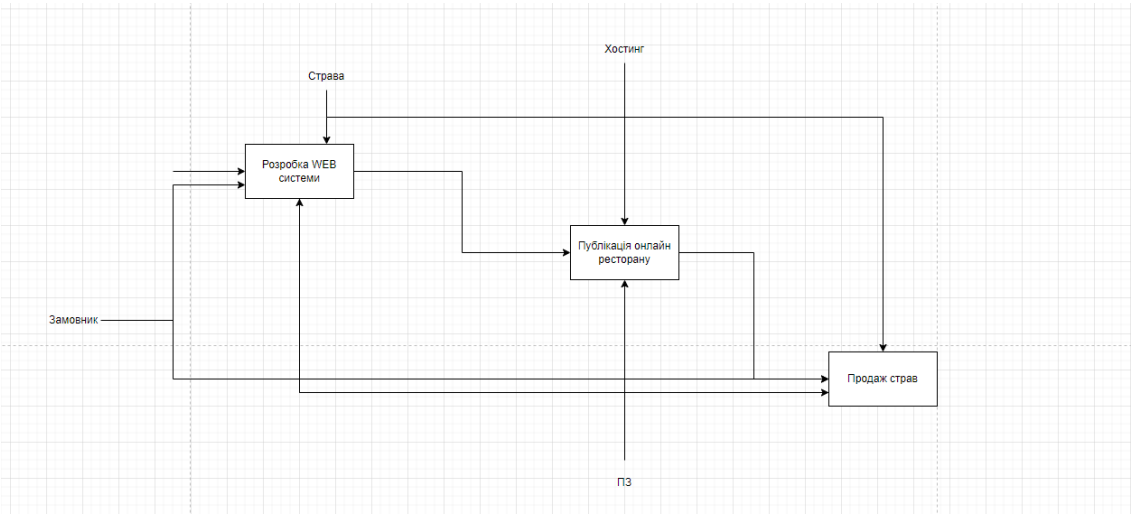

Рисунок 3.2 – Діаграма декомпозиції онлайн ресторану

Робота «Розробка WEB-системи» забезпечує виконання інформаційної системи на локальному комп'ютері: здійснюється розробка інтерфейсів, скриптів і баз даних, що забезпечують функціонування системи. Входом у неї внесення даних користувача, які потрібно занести у БД системи. Управляється за

допомогою візуального інтерфейсу, запитом клієнта та правилами проектування системи. Механізмом є програмне забезпечення, яке потрібне для здійснення розробки, а результатом роботи є готова інформаційна система.

Наступна робота «публікування онлайн ресторану» забезпечує отримання доменного ім'я, а потім дозволяє здійснити публікацію WEB-системи на хостингу. Входом для роботи є готова інформаційна система для розміщення. Управляється робота правилами оформлення оренди хостингу, а механізмом є – програмне забезпечення.

Робота «Замовлення страв» призначена для того, щоб згідно з правилами додавання товару в кошик, клієнт мав можливість додавання необмеженої кількості страв а потів редагування замовлення та покупки. У цієї роботи є два входи, це: дані про клієнта і доступна WEB-система, розміщена на хостингу. Управляється правилами оформлення замовлення і за умовами надання послуг Інтернет-провайдером. Механізмом є – програмне забезпечення, а виходом даної роботи є клієнт зі стравою.

Робота «Розробка інтерфейсу» має вхідні данні про характеристику страви, управління здійснюється за допомогою панелі адміністратора, запитом клієнта та правилами будівлі WEB-системи; механізм – програмне забезпечення, вихід розроблений прототип інтерфейсу онлайн ресторану. Робота «Програмування системи» має входом прототип інтерфейсу макет – вихід попередньої роботи також замовник; управлінням є, як і в попередньому блоці, адміністративна панель; механізмом, як і у всіх роботах, є – програмне забезпечення; вихід – готова інформаційна система з необхідним функціоналом. Робота «Створення БД» передбачає, що входом є функціонально готова система зберігати дані користувача та характеристики товару. Даний блок управляється правила будівлі WEB-системи та правила оформлення замовлення; робота виконується за допомогою програмного забезпечення; виходом є – функціонуюча, наповнена система [7].

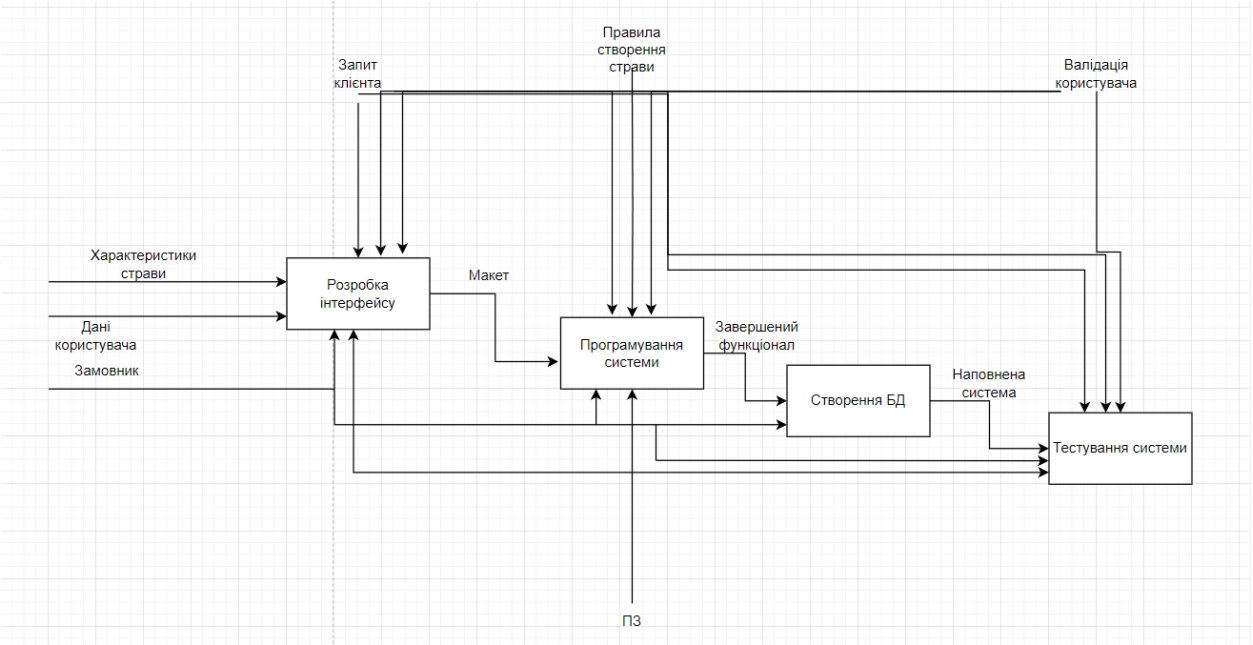

Рисунок 3.3 – Діаграми декомпозиції блоку «Розробка WEB-системи»

При декомпозиції другого А2 блоку – «публікування на хостингу» виділені наступні чотири роботи: «реєстрація домена» – входом є інформаційна система, робота керується хостингом; виходом є зареєстрований домен; «реєстрація хостингу» – входом є зареєстрований домен та інформаційна система, яка управляється хостингом; виходом роботи є зареєстрований хостинг. «Прив'язка домену до сайту» – входом є зареєстрований хостинг та інформаційна система, управляється хостингом; виходом роботи є закріплений за системою домен.

Робота «розміщення файлів на хостингу» передбачає на вході закріплений домен (домене ім'я,) та інформаційна система, управляється хостингом; виходом роботи є доступний у мережі Інтернет WEB-ресурс. Діаграма декомпозиції другого А2 блоку – «публікування на хостингу» наведена на рис. 3.4.

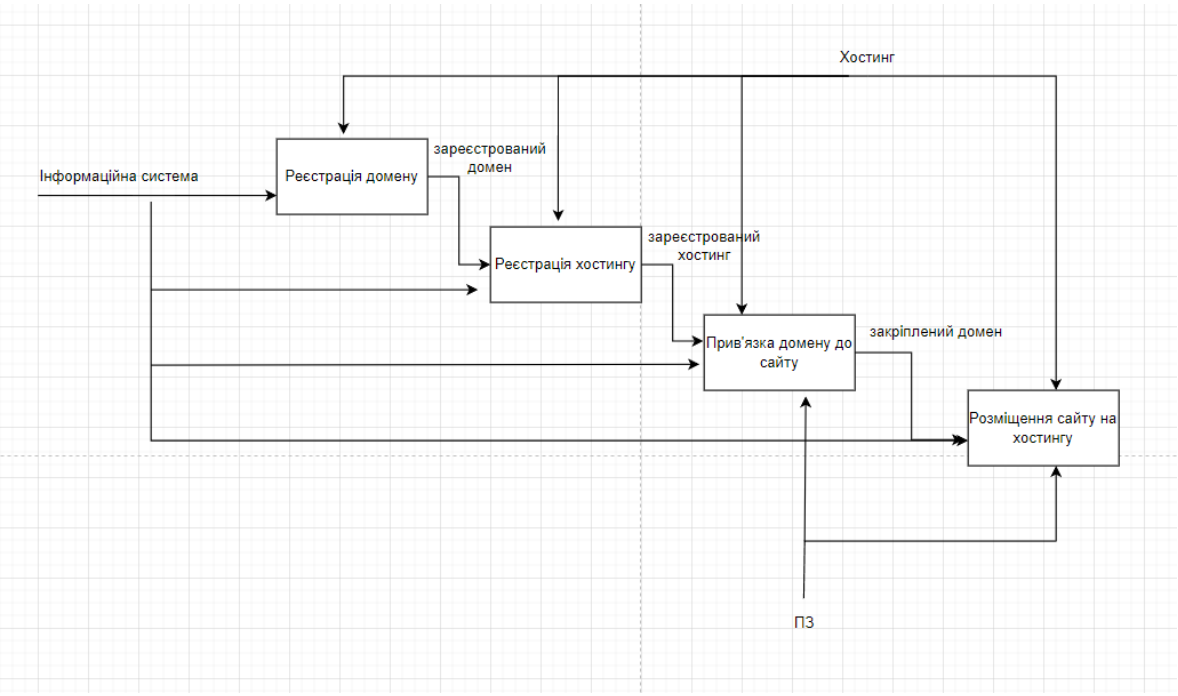

Рисунок 3.4 – Діаграми декомпозиції блоку «Розміщення на хостингу»

В подальшому було проведено наступний етап здійснення деком позиції інформаційної системи. При декомпозиції третього А3 блоку – «Замовлення страв» виділені наступні три роботи.

Робота «Реєстрація користувачів», щоб здійснити реєстрацію необхідні данні користувачів та доступний WEB-ресурс у мережі Інтернет; робота буде керуватися валідацією користувача; результатом роботи є зареєстрований у системі користувач. Робота «Вибір страви» передбачає, що організовано меню та в системі є зареєстровані користувачі для вибору страви, управлінням є вибір страви, а результатом є додані до кошика страви. Робота «Оформлення замовлення» передбачає, щоб організувати оформлення замовлення потрібно обрати страви; механізм даної роботи є програмне забезпечення, а результатом є замовник з оформленим замовленням. Діаграма декомпозиції третього А3 блоку – «Замовлення страв» наведена на рис. 3.5.

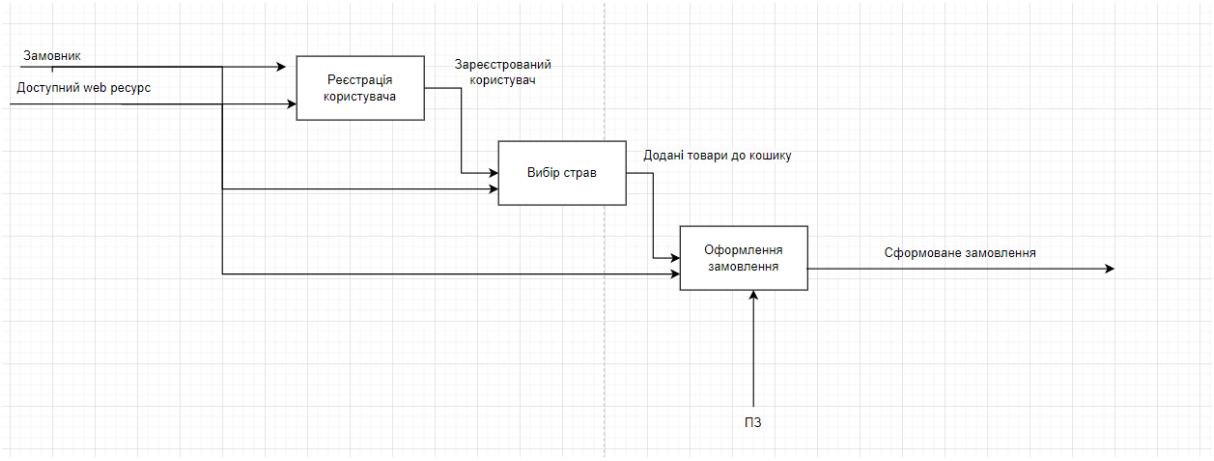

Рисунок 3.5 – Діаграми декомпозиції блоку «продаж доступних товарів»

3.2 Проектування мережевого WEB-ресурсу за допомогою методології функціонального моделювання Workflow Diagramming

Подальше проектування онлайн ресторану було здійснено за допомогою методології послідовного виконання процесів Workflow Diagramming (стандарт IDEF3). Цей стандарт описує логіку виконання дій. IDEF3 може використовуватися самостійно і спільно з методологією IDEF0: будь-який функціональний блок IDEF0 може бути представлений у вигляді послідовності процесів або операцій засобами IDEF3. Якщо IDEF0 описує, що здійснюється в системі, то стандарт IDEF3 описує, як це робиться. Згідно з методологією IDEF3 онлайн ресторану має єдину роботу – «замовлення доступних страв», яку можливо навести у контекстній діаграмі системи [8].

Провівши декомпозицію контекстної діаграми, спостерігається наступна послідовність виконання робіт (рис. 3.6).

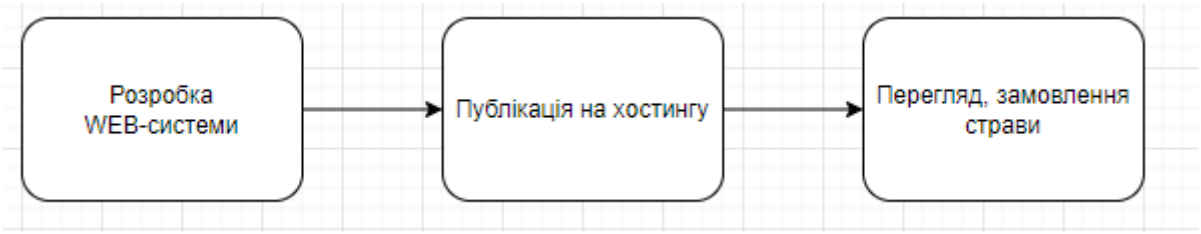

Рисунок 3.6 – Діаграма декомпозиції інтернет-магазину

Першою роботою системи є «розробка WEB-системи», далі – робота «публікування на хостингу». Зв'язок між цими роботами означає, що робота-приймач може завершитись ще до закінчення роботи-джерела. Наступною є робота «замовлення страв», яка пов'язана з блоком «публікування на хостингу» старшим зв'язком, що передбачає завершення всіх попередніх робот. При декомпозиції наступного рівня роботи «Розробка WEB-системи» отримано чотири блоки – це роботи з двома перехрестями (рис. 3.7).

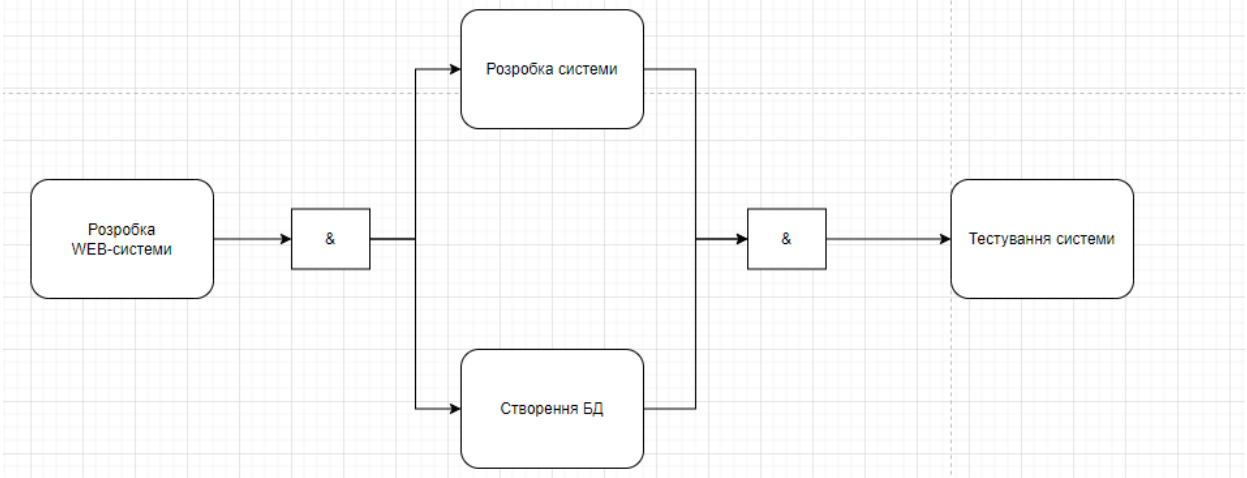

Рисунок 3.7 – Діаграма декомпозиції блоку «Розробка WEB-системи»

Після завершення першої роботи «Розробка інтерфейсу» йде перше перехрестя «Асинхронне І», що передбачає, що подальші роботи можуть початися не одночасно, але обов'язково повинні бути запущені. Це такі роботи як: «Розробка системи» та «Створення бази даних». При злитті стрілок-виходів з

цих робіт.Наступною є робота «Тестування системи», яка і буде завершальною.

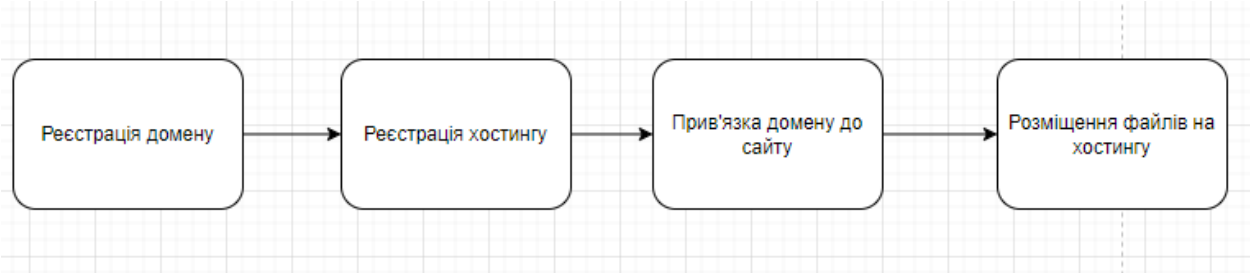

Рисунок 3.8 – Діаграма декомпозиції роботи «публікування на хостингу»

При декомпозиції роботи «публікування на хостингу» всі роботи пов'язані старшим зв'язком і розміщені послідовно: «реєстрація домену», «реєстрація хостингу», «прив’язка домену до сайту», «розміщення файлів на хостингу» (рис. 3.8).

При декомпозиції блоку «надання послуг» отримано три блоки робіт, які розміщені послідовно і пов'язані між собою старшим зв'язком (рис. 3.9).

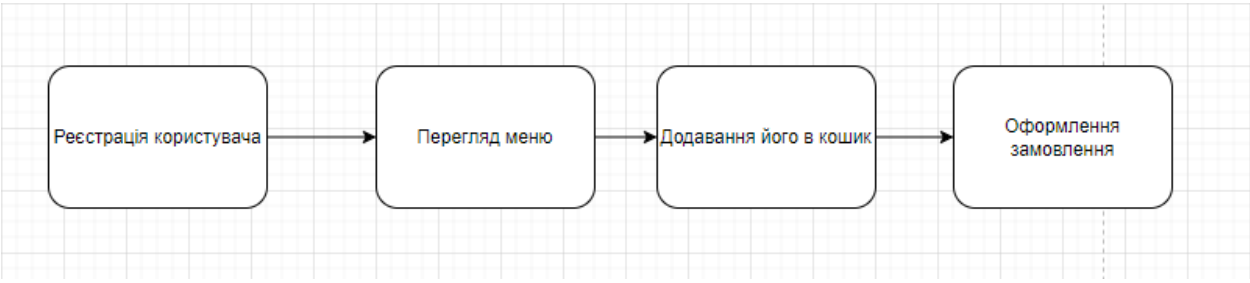

Рисунок 3.9 – Діаграма декомпозиції роботи «Замовлення страв»

3.3 Проектування бази даних системи

Для здійснення мережевого веб-ресурсу ресторану з використанням технології ASP .NET обрана трирівнева архітектура «клієнт-сервер». Це передбачає, що база даних розміщена на сервері локальної комп'ютерної мережі. На робочих станціях розміщують клієнтську програму-браузер, за

допомогою якої здійснюється формування та відсилка запитів до сервера бази даних. Сервер бази даних, в свою чергу, забезпечує виконання запиту і видачу клієнтові результату, який відображається на формах HTML-документу.

Перший етап процесу проектування бази даних називається концептуальним проектуванням бази даних. Він полягає у створенні концептуальної моделі даних для аналізованої частини системи. Ця модель даних створюється на основі функціональних вимог користувачів. Концептуальне проектування бази даних абсолютно не залежить від таких подробиць її реалізації, як тип обраної СУБД, набір створюваних прикладних програм, що використовуються мови програмування, тип обраної обчислювальної платформи, а також від будь-яких інших особливостей фізичної реалізації.

Концептуальне проектування включає створення концептуального уявлення бази даних, що включає визначення типів найважливіших сутностей, існуючих між ними зв'язків і атрибутів. Визначимо послідовність етапів проектування концептуальної моделі даних:

– визначення сутностей;
– визначення взаємозв'язків між сутностями;
– визначення атрибутів сутностей;
– завдання первинних та альтернативних ключів.

Побудуємо концептуальну модель бази даних для веб-ресурсу ресторану. Перейдемо до виконання послідовності проектування. Оберемо спроектовані таблиці бази даних та запишемо їх головні сутності. Для розробки бази даних мережевого веб-ресурсу ресторану, була обрана та використана СУБД MS SQL. При здійснені проектування були визначені основні сутності бази даних системи.

Основні сутності бази даних:

Таблиця 1 – Сутність «Користувачі»

№	Атрибут	Тип
1	Id	Guid
2	FirstName	varchar(255)
3	LastName	varchar(255)
4	DateOfBirth	DateTime
5	PhoneNumber	bool
6	TwoFactor	bool
7	SecurityStamp	varchar(255)
8	PasswordHash	varchar(255)

Таблиця 2 – Сутність «Ролі»

№	Атрибут	Тип
1	Id	Guid
2	Name	varchar(255)

Таблиця 3 – Сутність «РоліКористувачів»

№	Атрибут	Тип
1	Id	Guid
2	RoleId	Guid
3	UserId	Guid

Таблиця 4 – Сутність «Страви»

№	Атрибут	Тип
1	Id	Guid
2	Name	varchar(255)
3	Price	int(16)

Таблиця 5 – Сутність «Замовлення»

№	Атрибут	Тип
1	Id	Guid
2	Message	varchar(255)
3	TableName	varchar(255)
4	UserId	Guid

Таблиця 6 – Сутність «Столи»

№	Атрибут	Тип
1	Id	Guid
2	Link	varchar(255)
3	UserId	Guid
4	Name	varchar(255)

Таблиця 7 – Сутність «АвторизаціяКористувачів»

№	Атрибут	Тип
1	LoginProvider	varchar(255)
2	ProviderKey	varchar(255)
3	ProviderDisplayName	varchar(255)
4	UserId	Guid

Після визначення таблиць, полів, індексів і зв'язків між таблицями слід нормалізувати проектовану базу даних. Важливість нормалізації полягає в тому, що вона дозволяє розбити великі відносини, які зазвичай містять велику надмірність інформації, на дрібніші логічні одиниці. Процес нормалізації включає: усунення повторюваних груп (приведення до 1НФ); видалення частково залежних атрибутів (приведення до 2НФ); видалення транзитивно залежних атрибутів (приведення до 3НФ).

Приведення до першої нормальної форми включає усунення повторюваних груп даних. Це означає, що кожен стовпець таблиці повинен містити лише атомарні значення, а кожен рядок - бути унікальним. У нашому випадку всі таблиці бази даних вже відповідають вимогам 1НФ, оскільки всі атрибути мають атомарні значення і кожна таблиця має первинний ключ.

Для приведення до другої нормальної форми необхідно усунути часткові залежності, тобто кожен неключовий атрибут повинен бути повністю функціонально залежним від первинного ключа. Це досягається шляхом розділення таблиць, де це необхідно.

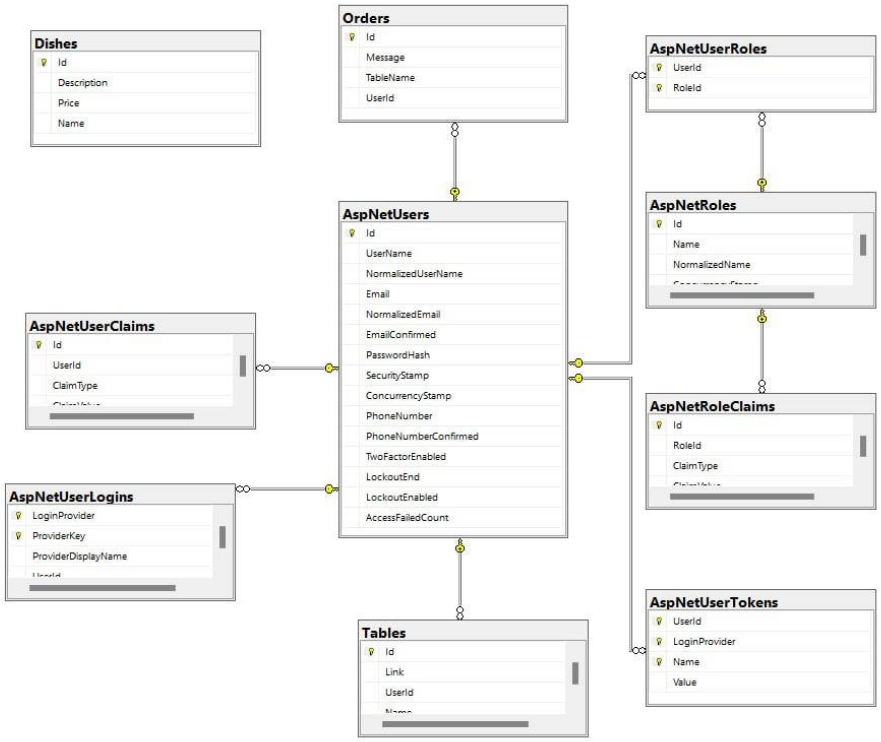

Рисунок 3.10 – Зв’язки між таблицями в БД

Для приведення до третьої нормальної форми необхідно усунути транзитивні залежності. Це означає, що неключові атрибути не повинні залежати один від одного. У нашій базі даних сутності вже приведені до 3НФ.

4 РОЗРОБКА ТЕХНІЧНОГО ПРОЕКТУ

4.1 Керівництво додатком користувача-клієнта веб-ресурсу

Загальний огляд ресторану представлений у вигляді великого банера з привабливим зображенням, яке створює атмосферу затишку та елегантності. Центральне місце на сторінці займає назва ресторану "THE RESTO" та слоган "An industry that starts with your stomach" (Індустрія, що починається з вашого шлунка). Цей слоган підкреслює важливість їжі у нашому житті та викликає апетит [9].

На головній сторінці також розміщена кнопка "MENU" (Меню), яка дозволяє користувачам швидко перейти до перегляду асортименту страв, доступних у ресторані. Це зручно, оскільки дозволяє одразу ознайомитися з пропозиціями кухні без потреби у додаткових пошуках.

У верхньому правому куті головної сторінки розташовані посилання "Home" (Головна) та "Login" (Вхід), що дозволяє користувачам легко переміщатися по сайту та виконувати авторизацію, якщо це необхідно. Також є можливість вибору мови інтерфейсу, що забезпечує зручність для відвідувачів з різних країн.

Додатково, на головній сторінці може бути представлена інформація про нові страви та інші важливі повідомлення. Це дозволяє ресторану оперативно інформувати відвідувачів про спеціальні пропозиції та новинки в меню, залучаючи їх до повторних візитів та підвищуючи лояльність клієнтів.

Таким чином, головна сторінка веб-ресурсу є ключовим елементом взаємодії з користувачами, надаючи їм необхідну інформацію та створюючи приємне враження про ресторан з перших секунд. Вона допомагає ефективно презентувати ресторан, його концепцію та особливості, а також забезпечує легкий доступ до найважливіших розділів сайту.

Рисунок 4.1 – Головна сторінка

Доступ до меню ресторану можливий через відповідний розділ головної сторінки. Користувач може переглянути страви, ознайомлюватися з описом та цінами (рис. 4.2). Меню представлено у зручному форматі, що дозволяє легко знаходити необхідні страви та отримувати інформацію про них. Кожна страва супроводжується коротким описом та ціною, що допомагає користувачам робити свій вибір.

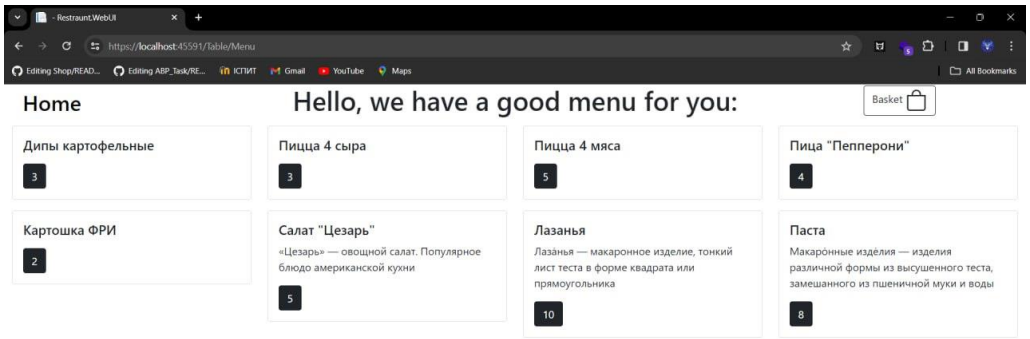

Рисунок 4.2 – Сторінка «Меню»

Після входу в акаунт користувач має доступ до персонального кабінету (рис. 4.3). У персональному кабінеті зберігаються індивідуальні налаштування користувача, історія замовлень, інформація про поточні замовлення та інші особисті дані. Це дозволяє користувачу легко відстежувати свої замовлення та керувати ними. Доступ до персонального кабінету забезпечує зручний та персоналізований користувацький досвід, що сприяє підвищенню задоволеності клієнтів.

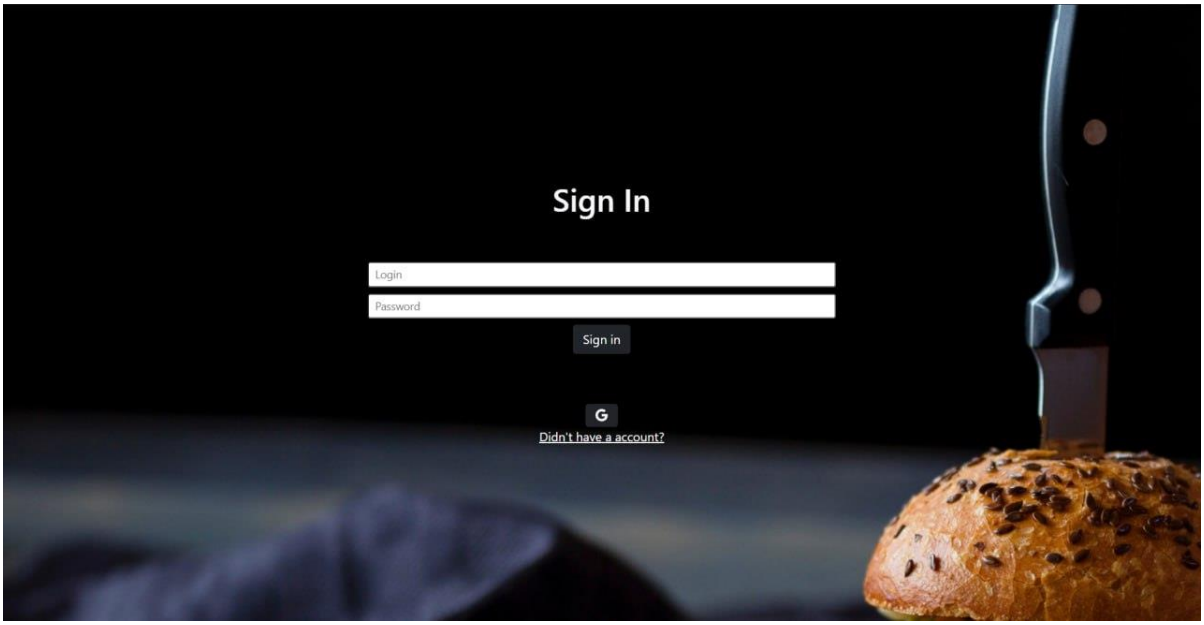

Рисунок 4.3 – Сторінка авторизації

Користувач може завершити роботу з веб-ресурсом, натискаючи кнопку виходу або закриваючи вікно браузера. Це безпечно завершить сеанс і захистить особисті дані користувача. Безпечне завершення сеансу є важливим аспектом, оскільки воно запобігає несанкціонованому доступу до облікового запису та особистої інформації користувача. Кнопка виходу зазвичай розташована у верхньому правому куті інтерфейсу, що робить її легко доступною для користувача.

Після вибору страв користувач може додавати їх до свого кошика, де відображається актуальний зміст його замовлення. Кошик забезпечує

можливість перегляду та відредагування замовлення перед підтвердженням (рис. 4.4).

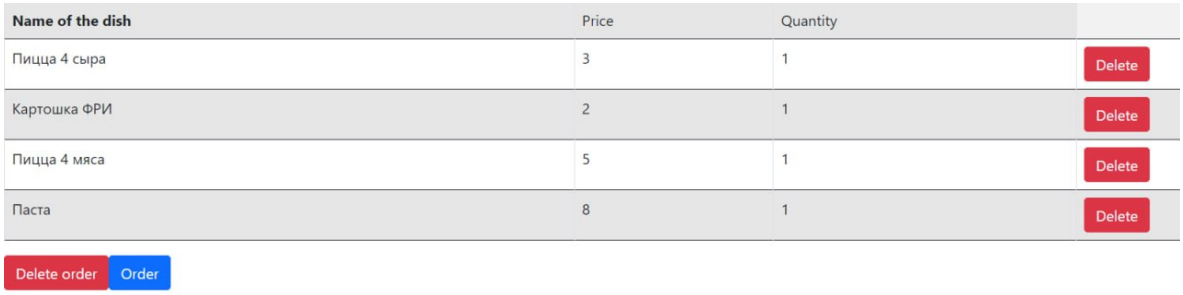

Рисунок 4.4 – Кошик

Після натискання кнопки "Order", система починає обробку замовлення, а користувачу відображається повідомлення про успішне підтвердження замовлення. Як видно на рис. 4.5, на екрані з'являється повідомлення "Thank you for order!" (Дякуємо за замовлення!) і підтвердження, що офіціанта вже повідомлено ("waiter already notified"). Це означає, що замовлення було успішно зареєстровано в системі і передано на виконання обслуговуючому персоналу. Важливим аспектом цього етапу є те, що користувачу не потрібно виконувати додаткові дії для інформування офіціанта, що значно спрощує процес обслуговування і зменшує час очікування [10].

Кнопка "Go to home page" (Перейти на головну сторінку) дозволяє користувачу повернутися на головну сторінку застосунку після завершення процесу замовлення. Це зручно, оскільки користувач може продовжити перегляд меню або здійснити нове замовлення, не потребуючи додаткових зусиль для навігації по сайту.

Система автоматизації замовлень, як показано на рисунку 4.6, значно підвищує ефективність роботи ресторану та покращує користувацький досвід. Користувачі отримують швидке підтвердження своїх дій, а обслуговуючий персонал оперативно дізнається про нові замовлення, що дозволяє швидше розпочати їх виконання.

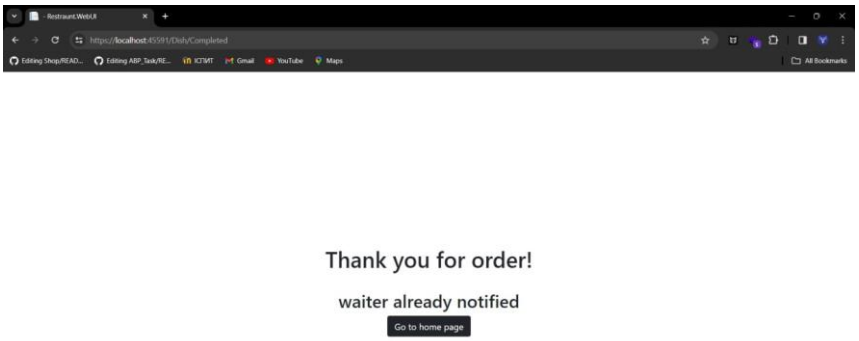

Рисунок 4.6 – Сторінка подяки за замовлення

4.2 Реалізація додатку користувача-адміністратора веб-ресурсу

Адміністраторський функціонал розроблено для забезпечення повноцінного та ефективного управління рестораном через веб-ресурс.

Адміністратор має можливість інтегрувати нові страви до ресторанного меню безпосередньо через адмін-панель. Вибираючи відповідний розділ, він вводить назву страви, опис, встановлює ціну та додає зображення для візуальної привабливості. Цей функціонал забезпечує меню до змін та нововведень у ресторані (рис. 4.7).

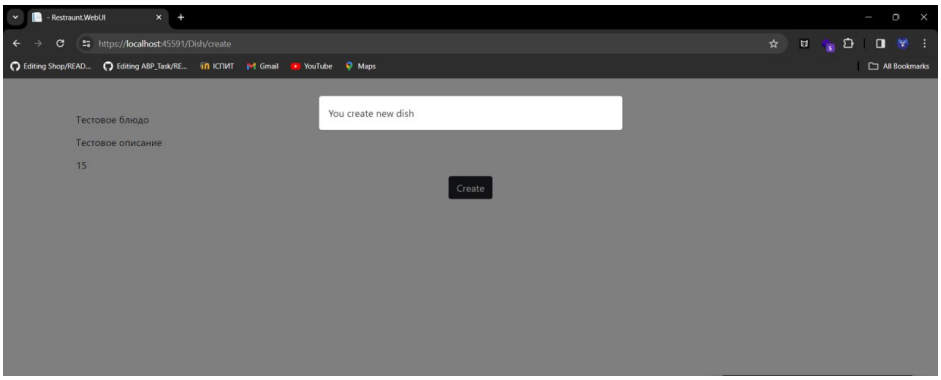

Рисунок 4.7 – Додавання страви

Адміністратор отримує можливість вносити зміни у вже існуючі страви. Це означає зміну параметрів, таких як ціна, склад чи опис, або видалення страви, яка більше не є актуальною. Редагування відбувається в реальному часі, щоб забезпечити актуальність інформації на веб-ресурсі (рис. 4.8).

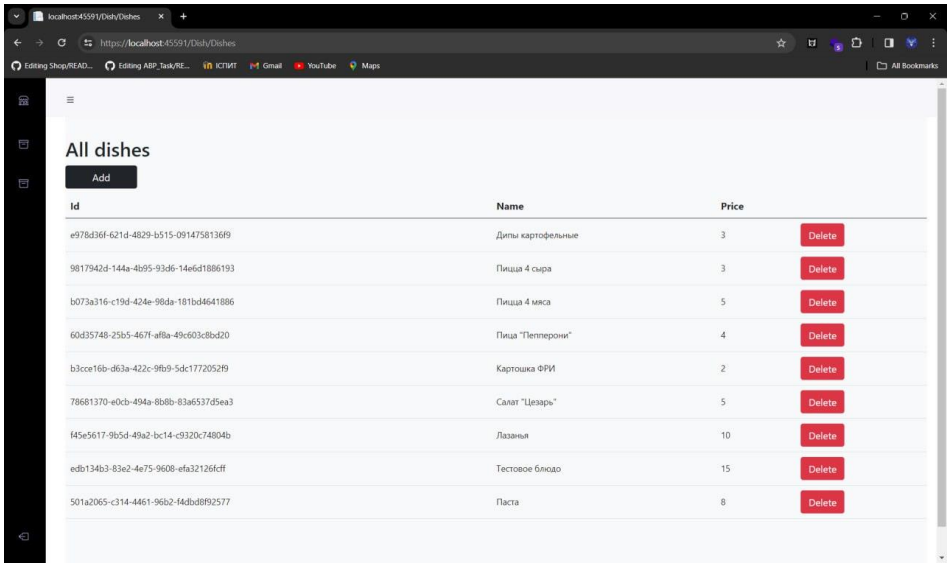

Рисунок 4.8 – Сторінка страв

Для підвищення ефективності роботи ресторану адміністраторам надається зручний та інтуїтивно зрозумілий набір інструментів для управління столиками. Ці інструменти значно полегшують процес додавання нових столиків у систему, забезпечуючи гнучкість і простоту в налаштуванні та управлінні розсадженням гостей. Наприклад, адміністратор може легко додати нові столики, що дає змогу оперативно реагувати на зміни та потреби закладу (рис. 4.9). Таким чином, використання цих інструментів сприяє поліпшенню якості обслуговування клієнтів та оптимізації процесів усередині ресторану.

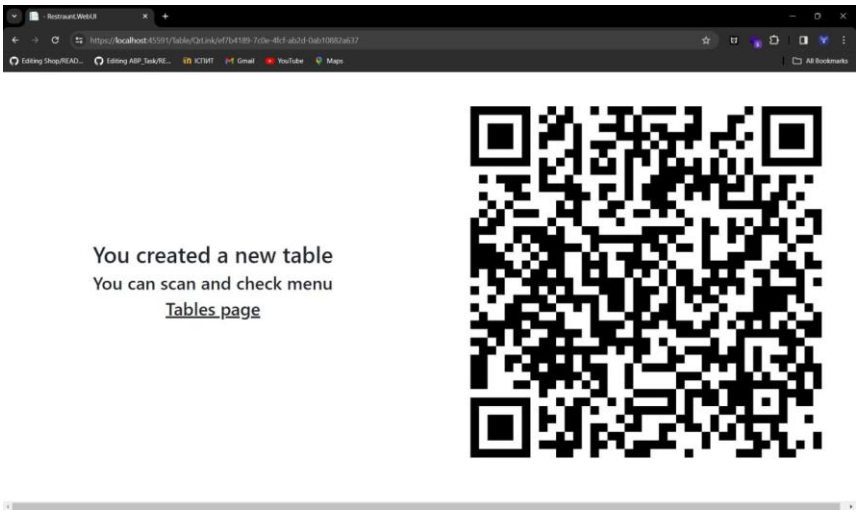

Рисунок 4.9 – Сторінка створення стола

4.3 Витримки коду та опис до них

Архітектурна конструкція веб-ресурсу для ресторану включає в себе неабияку складність, яка проявляється в аспектах безпеки та ідентифікації користувачів. Однією з ключових складових цієї архітектури є авторизація, яка виокремлена у вигляді окремого проекту:

– виділення авторизації у окремий проект є стратегічним рішенням, що дозволяє забезпечити максимальну гнучкість та масштабованість системи. Для реалізації цього підходу використовується стандарт OAuth 2.0, а також IdentityServer4 як реалізація сервера авторизації та видачі токенів доступу;
– OAuth 2.0 визначає стандарти для надання третім сторонам обмеженого доступу до ресурсів користувача без необхідності передачі йому облікових даних. Цей протокол використовується у поєднанні з IdentityServer4, який служить як потужний та належно налаштований сервер авторизації;
– bearer Tokens використовуються для безпечної передачі інформації про авторизацію між сторонами. Ці токени, видача яких

контролюється IdentityServer4, дозволяють зберігати важливу інформацію про користувача та його права в зашифрованому вигляді;

– виділення авторизації у вигляді окремого проекту не лише полегшує підтримку та розширення системи, але й забезпечує високий рівень безпеки. Винесення цієї функціональності у відокремлений шар дозволяє уникнути непотрібних залежностей та підвищує стійкість системи до потенційних загроз безпеки.

Авторизація у вигляді окремого проекту виступає як ключовий стовпчик, що підтримує стабільність та безпеку веб-ресурсу для ресторану, гарантуючи захист конфіденційної інформації користувачів та надійність системи в цілому.

Цей фрагмент встановлює параметри аутентифікації та авторизації для користувачів за допомогою сервісу Identity в ASP.NET Core. Зокрема, відключається вимога наявності великих літер, спеціальних символів та чисел у паролі. Крім того, вказується використання Entity Framework для зберігання даних користувачів та ролей в базі даних:

```
builder.Services.AddIdentity<User, Role>(config =>
{
    config.Password.RequireUppercase = false;
    config.Password.RequireNonAlphanumeric = false;
    config.Password.RequiredLength = 4;
    config.Password.RequireDigit = false;
})
.AddEntityFrameworkStores<ApplicationDbContext>();
builder.Services.ConfigureApplicationCookie(config =>
{
    config.Cookie.Name = "IdentityServer.Cookie";
    config.LoginPath = "/Account/Login";
    config.LogoutPath= "/Account/LogOut";
});
```

Тут визначаються параметри кукі для зберігання інформації про аутентифікацію. Вказується ім'я кукі, шляхи для входу та виходу:

```
builder.Services.AddAuthentication()
    .AddGoogle(options =>
    {
```

```
    options.ClientId="937172952204tm8qh7anmd-
bifhsses1mi7mrlnqpni.apps.googleusercontent.com";
            options.ClientSecret = "GOCSPX-
JT0Nmf5F4dOvLQoyGcj2ONpp";
            options.ReturnUrlParameter = "/";
        });
```

Цей блок коду відповідає за додавання та налаштування IdentityServer4. Використовуються інтегровані сервіси Identity з Entity Framework, а також конфігурація ресурсів, ідентифікаційних ресурсів та клієнтів. При цьому використовується вбудований підписчик для розробки. Після цього IdentityServer додається та конфігурується в додатку:

```
    builder.Services.AddIdentityServer()
        .AddAspNetIdentity<User>()
        .AddInMemoryApiResources(Configuration.GetApis())
        .AddInMemoryIdentityResources(Configuration.GetIden-
tityRecourses())
        .AddInMemoryClients(Configuration.GetClients())
        .AddDeveloperSigningCredential();
```

В даному фрагменті коду встановлюються Middleware для використання IdentityServer, а також Middleware для обробки аутентифікації та авторизації в додатку:

```
    app.UseIdentityServer();
    app.UseAuthentication();
    app.UseAuthorization();
```

Конструктор приймає об'єкт типу ApplicationDbContext в якості параметра та ініціалізує його для подальшого використання в методах репозиторію:

```
    public TableRepository(ApplicationDbContext db)
    {
        _db = db;
    }
```

Цей метод створює новий об'єкт Table на основі даних з об'єкта TableDto, додає його до контексту бази даних та повертає true:

```
    public async Task<bool> Create(TableDto? entity)
    {
        var result = new Table { Id = entity.Id, Name = en-
tity.Name, Link = entity.Link };
```

```
        await _db.Tables.AddAsync(result);
        return true;
    }
```

Цей метод видаляє запис з таблиці Tables за ідентифікатором. Якщо запис знайдено і успішно видалено, повертає true, інакше – false:

```
    public async Task<bool> Delete(Guid Id)
    {
        var entity = await _db.Tables.Where(t => t.Id == Id).FirstOrDefaultAsync();
        if (entity != null)
        {
            _db.Tables.Remove(entity);
            return true;
        }
        else
        {
            return false;
        }
    }
```

Цей метод повертає об'єкт типу Table за ідентифікатором, включаючи також зв'язаного користувача:

```
    public async Task<Table> Get(Guid id)
    {
        var table = await _db.Tables.Where(t=>t.Id==id).Include(u=> u.User).FirstOrDefaultAsync();
        return table;
    }
```

Метод повертає всі записи з таблиці Tables, включаючи зв'язаного користувача для кожного столика:

```
    public async Task<IEnumerable<Table>> Select()
    {
        return _db.Tables.Include(u => u.User).ToList();
    }
```

Цей метод редагує існуючий запис у таблиці Tables за допомогою даних з об'єкта EditTableDto та повертає оновлений об'єкт:

```
    public async Task<Table> Edit(EditTableDto entity)
```

```
    {
        var table = await _db.Tables.Where(t => t.Id == en-
tity.Id).FirstOrDefaultAsync();

        if (table != null)
        {
            table.Name = entity.Name;
            _db.Update(table);
        }
        return table;
    }
```

Метод прив'язує користувача до столика, оновлюючи відповідний запис у таблиці:

```
    public async Task<Table> BindUserToTable(BindUserToTableDto
model)
    {
        Table? table = await _db.Tables.Where(t => t.Id ==
model.Id).FirstOrDefaultAsync();
        User? user = await _db.Users.FirstOrDefaultAsync(u =>
u.Id == model.UserId);

        if (table != null)
        {
            table.User = user;
            _db.Update(table);
        }
        return table;
    }
```

Метод від'єднує користувача від столика, оновлюючи відповідний запис у таблиці та знульовуючи зв'язаного користувача:

```
    public async Task<Table> UnBindUserToTable(UnBindUserToTa-
bleDto model)
    {
        Table? table = await _db.Tables.Where(t => t.Id ==
model.Id).Include(u => u.User).FirstOrDefaultAsync();
        if (table != null)
        {
            table.User = null;
            _db.Update(table);
        }
        return table;
    }
```

Конструктор отримує об'єкт UnitOfWork в якості параметра та ініціалізує його для подальшого використання в методах контролера:

```
public TableController(UnitOfWork unitOfWork)
{
    _unitOfWork = unitOfWork;
}
```

Цей метод обробляє HTTP GET-запит на отримання всіх записів таблиці Tables. Повертає список столиків у форматі JSON:

```
[HttpGet]
public async Task<ActionResult<List<Table>>> Get()
{
    return Ok(await _unitOfWork.Tables.Select());
}
```

Цей метод обробляє HTTP GET-запит на отримання конкретного столика за його ідентифікатором. Повертає об'єкт типу Table у форматі JSON або повідомлення про помилку, якщо столик не знайдено:

```
[HttpGet("{Id}")]
public async Task<ActionResult<Table>> Get(Guid Id)
{
    var entity = await _unitOfWork.Tables.Get(Id);
    if (entity == null)
        return BadRequest("Entity not found");

    return Ok(await _unitOfWork.Tables.Get(Id));
}
```

Цей метод обробляє HTTP POST-запит на створення нового столика. Повертає посилання на створений столик або повідомлення про помилку:

```
[HttpPost]
[Authorize]
public async Task<ActionResult<List<Table>>> AddTable([From-
Body] TableDto? table)
{
    if (ModelState.IsValid)
    {
        table.Link  =  $"{HttpContext.Request.Scheme}://lo-
calhost:45591/Table/Menu/{table.Id.ToString()}";
        await _unitOfWork.Tables.Create(table);
        await _unitOfWork.Save();
        return Ok(table.Link);
```

```
        }
        else
        {
            return BadRequest("Problem..");
        }
    }
```

Цей метод обробляє HTTP PUT-запит на редагування існуючого столика. Повертає оновлені дані столика або повідомлення про помилку:

```
    [HttpPut]
    [Authorize]
    public async Task<ActionResult<List<Table>>> EditTable([FromBody] EditTableDto table)
    {
        await _unitOfWork.Tables.Edit(table);
        await _unitOfWork.Save();
        return Ok(table);
    }
```

Цей метод обробляє HTTP DELETE-запит на видалення столика. Повертає підтвердження видалення або повідомлення про помилку:

```
    [HttpDelete]
    [Authorize]
    public async Task<ActionResult<List<Table>>> DeleteTable([FromBody] Guid Id)
    {
        var result = await _unitOfWork.Tables.Delete(Id);
        if (result == true)
        {
            await _unitOfWork.Save();
            return Ok("Table delete");
        }
        else
        {
            return BadRequest("Table not found");
        }
    }
```

Цей метод обробляє HTTP PATCH-запит на прив'язку користувача до столика. Повертає підтвердження прив'язки або повідомлення про помилку:

```
    [HttpPatch]
    [Authorize]
    public async Task<ActionResult<List<Table>>> BindUserToTable(BindUserToTableDto model)
    {
```

```
        await _unitOfWork.Tables.BindUserToTable(model);
        await _unitOfWork.Save();
        return Ok("User bind to table");
    }
```

Цей метод обробляє HTTP PUT-запит на від'єднання користувача від столика. Повертає підтвердження від'єднання або повідомлення про помилку:

```
    [HttpPut("{Id}")]
    [Authorize]
    public async Task<ActionResult<List<Table>>> UnBindUserToTa-
ble(UnBindUserToTableDto model)
    {
        await _unitOfWork.Tables.UnBindUserToTable(model);
        await _unitOfWork.Save();
        return Ok("User unbind from table");
    }
```

Конструктор отримує об'єкт ApplicationDbContext в якості параметра та ініціалізує його для подальшого використання в репозиторіях:

```
    public UnitOfWork(ApplicationDbContext db)
    {
        _db = db;
    }
```

Ця властивість повертає екземпляр DishRepository, якщо він існує, або створює новий, якщо ще не був створений. Це забезпечує взаємодію з репозиторієм для сутностей Dish:

```
    public IDishRepository Dishes
    {
        get
        {
            return _dishRepository = _dishRepository ?? new
DishRepository(_db);
        }
    }
```

Аналогічно до попередньої властивості, ця повертає екземпляр TableRepository для взаємодії з репозиторієм для сутностей Table:

```
    public ITableRepository Tables
```

```
    {
        get
        {
            return _tableRepository = _tableRepository ?? new
TableRepository(_db);
        }
    }
```

Також аналогічно до попередніх, ця властивість повертає екземпляр OrderRepository для взаємодії з репозиторієм для сутностей Order:

```
    public IOrderRepository Orders
    {
        get
        {
            return _orderRepository = _orderRepository ?? new Or-
derRepository(_db);
        }
    }
```

Цей метод зберігає всі зміни, внесені до бази даних, викликаючи асинхронний метод SaveChangesAsync контексту бази даних:

```
    public async Task Save()
    {
        await _db.SaveChangesAsync();
    }
```

Підключення сервісів до проєкту:

```
    builder.Services.AddScoped<IDishRepository,      DishReposi-
tory>();
    builder.Services.AddScoped<ITableRepository,    TableReposi-
tory>();
    builder.Services.AddScoped<IOrderRepository,    OrderReposi-
tory>();
    builder.Services.AddTransient<UnitOfWork>();
```

4.4 Тести працездатності

Тестування – це процес оцінювання програмного забезпечення для виявлення його відповідності заданим вимогам та очікуванням. Це важлива частина розробки програмного забезпечення, яка допомагає виявити помилки, дефекти та недоліки в програмі перед її випуском. Тестування включає в себе запуск програми з метою виявлення помилок і перевірки її працездатності, а також порівняння її поведінки з очікуваним результатом. У процесі тестування використовують різні методи, техніки та інструменти, щоб забезпечити якість і надійність програмного продукту.

Процес тестування працездатності включає в себе різноманітні аспекти, які покликані забезпечити надійність та ефективність веб-сайту. Нижче наведено основні кроки та аспекти, які були взяті до уваги в процесі тестування працездатності веб-ресурсу:

- виконано тестування для перевірки правильності відображення головної сторінки та інших важливих сторінок веб-ресурсу на різних пристроях та розширеннях екрану;
- виконано тести для перевірки коректності функціонування навігації по веб-ресурсу, переконуючись, що користувач може легко переходити між різними сторінками та розділами;
- здійснено перевірку коректності роботи форм для введення даних, таких як замовлення, резервація столиків, а також перевірено обробку помилок при некоректному введенні;
- виконано тести для перевірки правильності процесу додавання блюд до корзини, перевірки замовлення та підтвердження замовлення, забезпечуючи коректну роботу веб-ресурсу у відношенні обробки замовлень користувачів;
- проведено тести на виявлення потенційних вразливостей та використання веб-ресурсу, такі як SQL-ін'єкції, перехоплення сесій та інші загрози безпеки;

– проведено тести для визначення відповідності веб-ресурсу під час великої кількості користувачів, а також перевірено оптимізацію завантаження сторінок та ресурсів;
– проведено тести для перевірки правильності роботи цього API;
– проведено тести для перевірки роботи функціоналу в реальному часі, такого як надходження нових замовлень у реальному часі за допомогою SignalR.

Додаток було покрито на 100% юніт-тестами, а також були використані інтеграційні тести і ручне тестування, яке показало хороший результат (рис. 4.10).

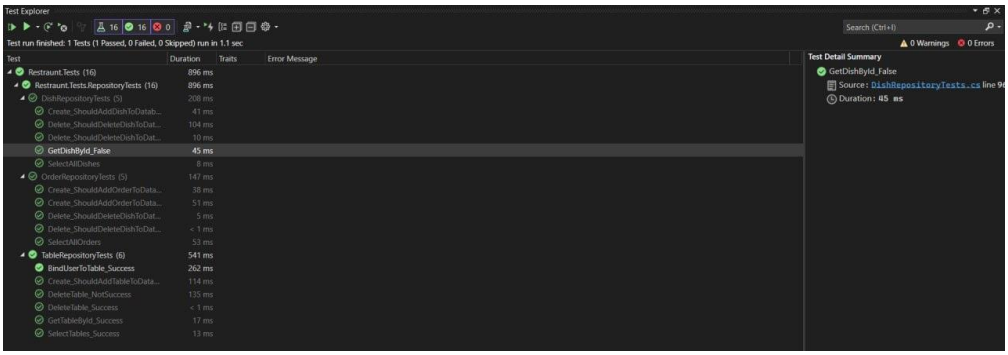

Рисунок 4.10 – Unit тестування

ВИСНОВКИ

В рамках даного дипломного проекту був успішно реалізований веб-ресурс для ресторану, що базується на технології ASP.NET. Проект був ініційований з метою надання ресторанному бізнесу високоефективного та зручного інструменту для управління замовленнями, меню та іншими аспектами роботи.

Проект використовує передові технології розробки, зокрема, фреймворк ASP.NET Core, мову програмування C#, та базу даних Microsoft SQL Server. Використання технологій також включає в себе використання єдиного патерну проектування MVC (Model-View-Controller), що забезпечує високу роздільну здатність компонентів проекту.

Веб-ресурс успішно реалізує ряд функціональностей, включаючи перегляд меню, здійснення замовлень, резервацію столиків та інші функції, спрямовані на полегшення взаємодії з користувачами та оптимізацію роботи ресторану.

Проект пройшов комплексне тестування, охоплюючи юніт-тестування, інтеграційне тестування, ручне тестування та тести безпеки. Це дозволило виявити та виправити потенційні проблеми та забезпечити високу якість роботи веб-ресурсу.

В рамках дипломного проекту були визначені та досягнуті основні завдання, включаючи розробку функціоналу, створення бази даних, використання безпеки та автентифікації через OAuth 2.0, інтеграцію з Google API, а також забезпечення високої швидкодії та ефективності веб-ресурсу.

Враховуючи досягнуті результати, проект може бути подальшим чином розвинений та вдосконалений. Можливості для майбутнього включають розширення функціоналу, оптимізацію взаємодії з користувачами та можливість впровадження нових технологій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Карпова Т.С. – Базы данных: модели, разработка, реализация. СПб.: Питер, 2001. 304 с.
2. Ричард Мартин – Agile Principles, Patterns, and Practices in C#: пер. с англ. 2-е изд. СПб.: ООО ДиаСофтЮП, 2011. 672 с.
3. Лок Ендрю – ASP.Net Core в действии 2021: пер. с англ. 2-е изд. СПб.: Питер, 2021. 278 с.
4. Гамма, Хелм, Джонсон, Влисседес – Паттерны обьектно-ориентированного проектирования: пер. с англ. 2-е изд. СПб.: Минск, 2020. 254 с.
5. Фреймворк – важливий інструмент програміста. [Електронний ресурс]. Режим доступу: https://fructcode.com/ru/blog/features-of-popular-frameworks-html-css-php-and-python-frameworks. Дата звернення: 02.06.2021
6. .NET Framework. [Електронний ресурс]. Режим доступу: https://ru.wikipedia.org/wiki/.NET_Framework. Дата звернення: 02.01.2024
7. C# Introduction. [Електронний ресурс]. Режим доступу: https://www.w3schools.com/cs/cs_intro.php. Дата звернення: 08.02.2024
8. What is SQL. [Електронний ресурс]. Режим доступу: https://aws.amazon.com/what-is/sql. Дата звернення: 18.03.2024
9. Unit testing. [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Unit_testing. Дата звернення: 10.01.2024
10. What Is Unit Testing. [Електронний ресурс]. Режим доступу: https://smartbear.com/learn/automated-testing/what-is-unit-testing. Дата звернення: 10.01.2024