МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри, канд. тех. наук, доцент Є.О. Давиденко

«___» _____2024р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Автоматизоване робоче місце адміністратора ресторану з підтримкою динамічного контенту

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 408.22010813

Здобувач освіти _____Н. С. Мещеряков
підпис
«__» _____2024 р.

Керівник д-р техн. наук, професор _____А. В. Швед
підпис
«__» _____2024 р.

Консультант канд. техн. наук, доцент _____А. О. Алєксєєва
підпис
«__» _____2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри Інженерії

Програмного Забезпечення

Є. О. Давиденко

«06» грудня 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано здобувачу освіти групи 408 факультету комп'ютерних наук

Мещерякову Нікіті Сергійовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи
Автоматизоване робоче місце адміністратора ресторану з підтримкою динамічного контенту

Затверджена наказом по ЧНУ від «22» грудня 2023 р. № 269

2. Строк представлення кваліфікаційної роботи « » _____ 20___ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Очікуваним результатом роботи є вебзастосунок автоматизованого робочого місця адміністратору ресторану з підтримкою динамічного контенту

4. Перелік питань, що підлягають розробці Аналіз специфіки діяльності ресторанів; аналіз та специфікація вимог до ПЗ, проєктування та моделювання ПЗ вебзастосунку, проєктування та реалізація бази даних вебзастосунку, розробка дизайну ресторанного вебзастосунку; кодування та тестування

програмного коду вебзастосунку.

5. Перелік графічних матеріалів

Презентація

6. Завдання до спеціальної частини

Дослідження питань охорони праці, які безпосередньо пов'язані з діяльністю розробника програмного забезпечення

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алексєєва А. О.	Кафедра екології Медичного інституту ЧНУ ім. Петра Могили	Спеціальна частина з охорони праці

Керівник роботи д-р техн. наук, професор Швед Альона Володимирівна

(посада, прізвище, ім'я, по батькові)

(підпис)

Завдання прийнято до виконання

Мещеряков Нікіта Сергійович

(прізвище, ім'я, по батькові здобувача)

(підпис)

Дата видачі завдання «06» грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: Автоматизоване робоче місце адміністратора ресторану з підтримкою динамічного контенту

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КРБ	05.12.2023	06.12.2023	виконано
2	Огляд літератури за темою роботи	20.01.2024	22.01.2024	виконано
3	Складання календарного плану КРБ	21.03.2024	22.03.2024	виконано
4	Аналіз предметної області	01.03.2024	22.03.2024	виконано
5	Розробка проєктних рішень	27.02.2024	12.03.2024	виконано
6	Моделювання та конструювання ПЗ	27.02.2024	12.03.2024	виконано
7	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	13.04.2024	04.06.2024	виконано
8	Розробка спеціальної частини з охорони праці	15.05.2024	23.05.2024	виконано
9	Відгук керівника КРБ	29.05.2024	30.05.2024	виконано
10	Оформлення КРБ та презентації	01.06.2024	02.06.2024	виконано
11	Попередній захист	03.06.2024	05.06.2024	виконано
12	Рецензування	13.06.2024	14.06.2024	виконано
13	Захист кваліфікаційної роботи	25.06.2024	26.06.2024	виконано

Розробив здобувач освіти Мещеряков Н. С. _____

(*прізвище, ім'я, по батькові)* *(підпис)*

«22» Березня 2024 р

Керівник роботи д-р, техн.,наук, професор Швед А. В. _____

(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«22» Березня 2024 р

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Автоматизоване робоче місце адміністратора ресторану з підтримкою динамічного контенту»

Здобувач 408 гр.:Мещеряков Нікіта Сергійович

Керівник: д-р., техн., наук, професор Швед Альона Володимирівна

Автоматизація процесів управління в ресторанному бізнесі відіграє ключову роль у підвищенні ефективності обслуговування та оптимізації робочого процесу. Розвиток цифрових технологій та їх інтеграція в управління ресторанами дозволяє адміністраторам керувати замовленнями, персоналом, запасами та фінансами в реальному часі. Використання систем з підтримкою динамічного контенту сприяє швидкому оновленню інформації про меню, ціни та акції, що забезпечує актуальність пропозицій для клієнтів та підвищує загальну задоволеність сервісом.

Об’єкт роботи – процес автоматизації бізнес-процесів робочого місця адміністратора ресторану.

Предмет роботи – програмні засоби та методи реалізації функціоналу та інтерфейсу автоматизованого робочого місця адміністратора ресторану.

Метою кваліфікаційної роботи є розробка та впровадження комплексного рішення автоматизованого робочого місця адміністратора ресторану, яке оптимізує управління рестораном, підвищує продуктивність персоналу та покращує якість сервісу для клієнтів.

Кваліфікаційна робота складається з вступу, 4 розділів, висновків та переліку джерел посилань.

У вступі визначається актуальність теми, мета, предмет та об’єкт дослідження.

У першому розділі проводиться аналіз предметної області та існуючих вебзастосунків-аналогів, формується постановка задачі та специфікація вимог до програмного забезпечення.

Другий розділ присвячений розробці сценаріїв використання, моделюванню та проєктуванню програмного забезпечення.

У третьому розділі подано огляд використаних технологій розробки та реалізацію вебзастосунку.

У четвертому розділі продемонстровано результат виконання роботи та тестування.

У висновках проводиться аналіз виконаних робіт та отриманих результатів.

Кваліфікаційна робота бакалавра викладена на 60 сторінках, вона містить 4 розділи, 33 ілюстрацій, 7 таблиці, 37 джерел в переліку посилань.

Ключові слова: розробка вебзастосунку, автоматизована ресторанна система, динамічний контент, ефективність обслуговування, технології розробки, ASP.NET.

ABSTRACT

of the Bachelor's Thesis

«Automated restaurant administrator workplace with support for dynamic content»

Student of group 408: Meshcheriakov Nikita Serhiyovych

Supervisor: Dr.Sc., Professor Shved Alona Volodymyrivna

The automation of management processes in the restaurant business plays a key role in enhancing service efficiency and optimizing the workflow. The development of digital technologies and their integration into restaurant management allow administrators to manage orders, personnel, inventory, and finances in real time. The use of systems with dynamic content support facilitates the rapid update of information about the menu, prices, and promotions, ensuring the relevance of offers for customers and increasing overall satisfaction with the service.

The object of work is the automation of business processes of the restaurant administrator's workplace.

The subject of work is the software tools and methods for implementing the functionality and interface of the automated restaurant administrator's workplace.

The purpose of the Bachelor's thesis is to develop and implement a comprehensive solution for the automated workplace of a restaurant administrator, which optimizes restaurant management, increases staff productivity, and improves the quality of service for customers.

The qualification work consists of an introduction, 4 sections, conclusions and a list of reference sources.

The introduction defines the relevance of the topic, the purpose, subject and object of the research.

In the first section, an analysis of the subject area and existing analogue web applications is carried out, a problem statement and a specification of software requirements are formed.

The second section is devoted to the development of usage scenarios, modeling and software design.

The third section provides an overview of the used technologies for the development and implementation of the web application.

The fourth section shows the results of the work and testing.

The conclusions analyze the work performed and the results obtained.

The bachelor's qualification work is laid out on 60 pages, it contains 4 sections, 33 illustrations, 7 tables, 37 sources in the list of references.

Keywords: web application development, automated restaurant system, dynamic content, service efficiency, development technologies, ASP.NET.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ... 3
ВСТУП ... 4
1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ ... 6
1.1 Аналіз предметної області автоматизації бізнес-процесів закладів харчування ... 6
1.2 Огляд застосунків-аналогів ресторанних систем ... 7
1.3 Специфікація вимог до програмного забезпечення АРМ адміністратора ресторану ... 9
Висновки до розділу 1 ... 16
2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ЗАСТОСУНКУ ... 17
2.1 Сценарії використання ... 17
2.2 Діаграма варіантів використання ... 21
2.3 Розробка мокапу системи ... 22
Висновки до розділу 2 ... 29
3 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ ... 30
3.1 Огляд технологій та їх порівняння ... 30
3.2 Обґрунтування вибору технологій для реалізації системи. Вибір програмних засобів для розробки ... 34
3.3 Опис архітектури системи ... 37
Висновки до розділу 3 ... 39
4 ПРОГРАМНА РЕАЛІЗАЦІЯ ... 40
4.1 Діаграма класів та діаграми сутність-зв’язок ... 40
4.2 Діаграма сутність-зв’язок ... 43
4.3 Діаграми послідовностей ... 45
4.4 Інструкція користувача ... 48
4.5 Тестування програмного забезпечення застосунку ... 53
Висновки до розділу 4 ... 55
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ... 58
ВИСНОВКИ ... 56

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ	–	програмне забезпечення
ОС	–	операційна система
ТЗ	–	технічне завдання
АРМ	–	Автоматизоване робоче місце
API	–	application programming interface
UML	–	unified modeling language
UI	–	user interface
UX	–	user experience

ВСТУП

Сучасний ресторанний бізнес стикається з високою конкуренцією та підвищеними вимогами з боку клієнтів до якості обслуговування, швидкості та зручності сервісу. Ринок громадського харчування характеризується не лише високою конкуренцією, але й стрімкими змінами у споживацьких вподобаннях та очікуваннях, що змушує бізнес адаптуватися, впроваджуючи інноваційні рішення для забезпечення високого рівня сервісу. Автоматизація робочих процесів стає необхідною умовою для підвищення ефективності управління рестораном та оптимізації всіх аспектів його діяльності. Впровадження автоматизованого робочого місця адміністратора з підтримкою динамічного контенту дозволить ресторанам підвищити рівень обслуговування, забезпечити оперативне управління замовленнями, меню, персоналом і запасами, а також підвищити загальну продуктивність роботи.

Актуальність теми обумовлена стрімким розвитком технологій і поширенням цифровізації у всіх сферах життя суспільства, включаючи сферу громадського харчування. Застосування інформаційних технологій дозволяє оптимізувати роботу адміністратора ресторану, швидко реагувати на зміни у попиті та запропонувати клієнтам актуальне та привабливе меню. Розробка автоматизованого робочого місця допоможе забезпечити динамічне оновлення контенту, управління замовленнями, запасами та персоналом, а також підвищити рівень звітності та аналітики бізнес-процесів Усі перераховані аспекти вказують на актуальність теми кваліфікаційної роботи, оскільки вона відповідає сучасним тенденціям у розвитку ринку та враховує потреби споживачів.

Об’єкт роботи – процес розробки вебзастосунку для створення автоматизованої ресторанної системи.

Предмет роботи – програмні засоби та методи реалізації функціоналу та інтерфейсу ресторанної системи.

Метою кваліфікаційної роботи є розробка та впровадження комплексного рішення автоматизованого робочого місця адміністратора

ресторану, яке оптимізує управління рестораном, підвищує продуктивність персоналу та покращує якість сервісу для клієнтів.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- проаналізувати специфіку діяльності ресторанів;
- проаналізувати вимоги користувачів до ресторанів;
- розробити вебдизайн ресторану;
- спроєктувати базу даних;
- розробити інтерфейс та функціонал вебзастосунку.

КРБ викладена на 60 сторінках, містить 33 ілюстрацій, 7 таблиць та 37 використаних джерел.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Аналіз предметної області автоматизації бізнес-процесів закладів харчування

Автоматизація бізнес-процесів у закладах харчування, таких як ресторани, кафе та інші, є ключовим аспектом для досягнення успіху та підвищення конкурентоспроможності. Цей підхід передбачає впровадження різноманітних технологій та систем з метою автоматизації рутинних операцій та оптимізації бізнес-процесів.

В першу чергу, автоматизація дозволяє раціоналізувати процеси приготування їжі. Вона передбачає використання спеціалізованого програмного забезпечення для управління кухнею, що дозволяє оптимізувати час приготування страв, контролювати запаси та забезпечувати їх вчасну поповненість.

По-друге, автоматизоване керування сприяє поліпшенню обслуговування клієнтів. Це включає в себе використання POS-систем для прийому та обробки замовлень, онлайн-систем бронювання столиків та замовлення їжі, а також систем лояльності для залучення та утримання клієнтів.

По-третє, автоматизація допомагає ефективно керувати фінансовими процесами. Це включає в себе використання програмного забезпечення для обліку фінансів, управління витратами та прибутками, а також автоматизовані системи оплати для зручності клієнтів та підвищення ефективності обслуговування.

Крім того, автоматизація бізнес-процесів сприяє підвищенню продуктивності працівників, зменшенню помилок та покращенню якості обслуговування. Вона дозволяє власникам закладів харчування зосередитися на стратегічних аспектах свого бізнесу, таких як розробка нових страв, взаємодія з клієнтами та розвиток бренду.

Аналіз предметної області автоматизації бізнес-процесів закладів харчування вказує на необхідність інтеграції сучасних технологій у роботу ресторанів та кафе з метою підвищення ефективності та конкурентоспроможності в сучасному ринковому середовищі.

1.2 Огляд застосунків-аналогів ресторанних систем

Першочерговим завданням перед конструюванням будь-якого програмного забезпечення є аналіз предметної галузі та пошук застосунків-аналогів. Такий аналіз є невід'ємною складовою для успішного проєктування будь-якого застосунку, так як це допомагає визначити потрібний функціонал, основні недоліки та переваги конкурентів. Для аналізу аналогів було обрано вебзастосунки: Daily Sport[1] (табл. 1.1), Ресторан «Вареники»[2] (табл. 1.2) та Ресторан «Дача»[3] (табл. 1.3).

Таблиця 1.1 – Опис вебзастосунку Daily Sport

Назва	Daily Sport
Розробник	Не доступно
Архітектура	Web application
Мова реалізації	Не доступно
Функції	– перегляд меню закладу; – інформація про час роботи та контакти; – можливість залишити відгук; – перегляд різноманітних категорій страв (сніданки, закуски, основні страви тощо).
Переваги	– зручний навігаційний інтерфейс; – широкий асортимент меню.

Кінець таблиці 1.1:

Недоліки	– відсутність функціоналу онлайн-замовлення; – обмежена інформація про заклад – без додаткових сервісів типу віртуального туру або резервації столиків онлайн.
Джерело інформації	https://dailysport.choiceqr.com/menu

Таблиця 1.2 – Опис вебзастосунку Ресторан "Вареники"

Назва	Вареники
Розробник	Не доступно
Архітектура	Web application
Мова реалізації	Не доступно
Функції	– перегляд меню доставки; – інформація про ресторан та його послуги; – контактна інформація та робочі години; – слідкування за рестораном в соціальних мережах.
Переваги	– чіткий та зрозумілий інтерфейс; – багате меню української кухні.
Недоліки	– відсутність онлайн замовлення через вебсайт; – обмежена інтерактивність та функціональність вебсайту.
Джерело інформації	https://vareniki.mk.ua/

Таблиця 1.3 – Опис вебзастосунку Ресторан «Дача»

Назва	Ресторан «Дача»
Розробник	Не доступно
Архітектура	Web application
Мова реалізації	CSS, JavaScript,HTML,
Функції	– перегляд інформації про ресторан і його кухню; – вивчення меню ресторану; – можливість ознайомлення з улюбленими стравами; – інформація про доставку та самовивіз; – відгуки гостей; – блог з корисною інформацією.
Переваги	– інформативний та легкий у навігації сайт; – багатий вибір страв одеської кухні; – велика кількість візуального контенту, що демонструє атмосферу ресторану
Недоліки	– недосконала фільтрація товарів.
Джерело інформації	https://savva-libkin.com/restaurant/odessa/dacha

1.3 Специфікація вимог до програмного забезпечення АРМ адміністратора ресторану

Автоматизоване робоче місце адміністратора ресторану з підтримкою динамічного контенту - це комплексне рішення, яке забезпечує ефективне управління всіма аспектами роботи ресторану.

Основні функції системи:

а) управління меню:

1) відображення усіх товарів з описом;

2) динамічне оновлення пунктів меню, ціни, описи страв та їх фотографії

3) швидке додавання спеціальних пропозицій, акційних страв та меню на основі сезонності продуктів

4) інтеграція з вебсайтом ресторану для актуалізації інформації про меню в реальному часі

б) управління замовленнями:

1) автоматизація процесу приймання замовлень з вебсайту;

2) управління статусами замовлень в реальному часі, включаючи обробку, готування та доставку;

3) автоматичний розрахунок вартості замовлень з урахуванням спеціальних пропозицій та знижок

в) управління запасами:

1) моніторинг запасів інгредієнтів в реальному часі,особистий кабінет для користувачів з можливістю редагувати особисту інформацію, переглядати історію замовлень;

2) планування закупівель на основі аналізу використання продуктів і тенденцій продажів;

г) управління персоналом

1) автоматизація планування графіків роботи співробітників на основі потреб ресторану;

2) відстеження відвідуваності, робочих годин та продуктивності персоналу.

д) звітність та аналітика

1) генерація детальних звітів про продажі, популярні страви, ефективність персоналу.

Користувачі системи – клієнт, адміністратор та персонал ресторану.

Функції клієнта:

- резервація столів;
- перегляд меню;
- пошук і перегляд товарів;
- перегляд інформації про товар;

- оформлення замовлення;
- перегляд замовлень;
- написання відгуків;

Функції персоналу ресторану:

- керування товарними залишками;
- управління товарами (створення, редагування, видалення);
- зміна статусу замовлень;
- прийом замовлень;
- редагування профілю;
- облікова система.

Функції адміністратора:

- управління товарами;
- управління замовленнями;
- управління персоналом
- управління звітами.
- управління акціями.

Не менш важливими завданнями виступають складення специфікації вимог[4], функцій та опис основних варіантів використання системи, що проєктується, у вигляді use-case діаграми. Створення діаграм класів, діаграм використання та розгортання також є ключовими завданнями для якісної розробки програмного забезпечення. Нижче наведено специфікацію вимог:

Призначення застосунку, для якого розробляється програмне забезпечення

Призначенням застосунку є допомога адміністратору ресторану приймати, обробляти, відстежувати замовлення в реальному часі, забезпечити ресторан зручною системою бронювання столів та облегшити інвентрацізацію продуктів.

Призначення застосунку, для якого розробляється програмне забезпечення

Було погоджено, що для створення ПЗ та його роботи буде використано такі технології: мова програмування C# у якості базової мови програмування,

платформу розробки вебзастосунків ASP.NET для створення Backend частини, платформу для розробки вебзастосунків Angular для створення Frontend частини застосунку.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Вебзастосунок призначений для використання у сфері споживання, а саме допомоги персоналу ресторану в зручному та гнучкому управлінні.

Характеристика користувачів

а) адміністратори ресторану. Основні користувачі системи, відповідальні за управління всіма аспектами ресторанного бізнесу, включаючи меню, кадри, столи, замовлення та фінанси;

б) офіціанти: Персонал, який приймає замовлення від клієнтів та забеспечує їх обслуговування за столами;

Загальна структура і склад системи

а) **клієнтська частина (Angular):**

1) **інтерфейс користувача (UI):** Розроблений з використанням Angular, включаючи сторінки для перегляду меню, резервацій столиків, обробки замовлень, оплати та інші функції;

2) **модулі та компоненти:** Розділення на модулі та компоненти для кращого управління кодом та підтримки масштабування;

3) **аутентифікація та авторизація:** Механізми аутентифікації користувачів та забезпечення прав доступу до функціоналу.

б) **серверна частина (ASP.NET):**

1) **API:**

– контролери та маршрутизація: Реалізація контролерів для обробки запитів користувачів та маршрутизація їх до відповідних методів;

– автентифікація та авторизація: Логіка для перевірки аутентифікації та авторизації користувачів перед виконанням дій на сервері.

2) **служби:**

– обробка бізнес-логіки: Логіка застосунку, така як обробка замовлень, управління меню, керування столиками тощо;

– взаємодія з базою даних: Спілкування з базою даних для отримання та збереження інформації про страви, клієнтів, замовлення та інше.

3) **база даних:**

– схема бази даних: Розроблена структура бази даних для збереження даних про меню, клієнтів, замовлення, столи та інше.

Загальні обмеження

Обмеження для роботи: наявність електронної обчислювальної системи (наприклад компьютер).

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела інформації

а) **внутрішні джерела:**

1) бази даних організації (кадрові, фінансові, операційні тощо);

2) відділи та співробітники, які надають звіти, запити, та інші форми даних;

3) внутрішні документи та архіви.

б) **зовнішні джерела:**

4) державні реєстри та бази даних;

5) інформація від партнерів та контрагентів;

6) відкриті джерела (наприклад, новини, аналітичні звіти, наукові публікації).

Нормативно-довідкова інформація

Вимоги відсутні.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Вимоги до технічного забезпечення мають певні обмеження у вигляді наявності електронної обчислювальної системи з операційною системою

Windows/MacOS/Linux дистрибутиву такої як комп’ютер з наявністю інтернет з’єднання щонайменш 5 Мбіт/с.

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Системне програмне забезпечення

Вебзастосунок має підтримувати різні платформи, такі як Windows, macOS або Linux, мати серверне программне забезпечення: вебсервер, систиму управління базами даних, проміжне програмне забезпечення для підтримки API для інтеграції з зовнішніми сервісами(наприклад система бронювання). Застосунок має бути побудований з використанням мови програмування C#, Frontend частина розроблена на фреймфоркі angular, Backend частина за допомогою ASP.Net. Процес отримання даних має відбуватись за допомогою запитів до СУБД SQL Server.

Мова і технологія розробки ПЗ

ПЗ має розроблюватись за допомогою мови C#, фреймворків Angular та ASP.Net.

ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

Інтерфейс має задовольняти вимоги UI/UX-дизайну для легкого розуміння користувачем. Інтерфейс має містити sidebar панель для загальної навігації та dashboard сторінку, яка надає швидкий доступ до основних компонентів системи.

Апаратний інтерфейс

Обчислювальна система користувача на базі ОС Windows/Linux/MacOS.

Програмний інтерфейс

У якості програмного інтерфейсу використовується 2 фреймворки: Angular та ASP.Net. Основні компоненти інтерфейсу: сторінка входу та реєстрації, дашборд, меню та управління замовленнями, управління запасами, фінанси, управління персоналом. Основні компоненти серверної частини: API для обробки запитів, база даних, автентифікація та авторизація, логування та аудит, інтеграція з зовнішніми сервісами.

Комунікаційний протокол

HTTP/HTTPS для зв’язку між клієнтом та сервером, Web API для

модульності та легкості інтеграції, формат обміну даних JSON, JWT та OAuth 2.0 для аутентифікації та авторизації, websockets для встановлення двостороннього зв’язку між клієнтом і сервером без постійних запитів, валідація даних на клієнтській та серверній стороні.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

Система повинна забезпечувати стабільну роботу з мінімальним простоєм.

Супроводженість

Система повинна підтримувати легке оновлення компонентів без значного переривання робочих процесів.

Переносимість

Система повинна бути сумісна з різними операційними системами та пристроями, що дозволяє її використання як на персональних комп’ютерах, так і на мобільних проистроях.

Продуктивність

Ефективне використання серверних та клієнтських ресурсів для забезпечення швидкої відповіді системи на запити користувачів. Здатність системи легко масштабуватися зі збільшенням кількості користувачів та обробки даних.

Надійність

Здатність системи продовжувати функціонування навіть при часткових збоях окремих компонентів. Гарантія актуальності та точності даних, навіть при високих навантаженнях.

Безпека

Строгі механізми для перевірки ідентичності користувачів та контролю доступу до ресурсів системи.

Висновки до розділу 1

Проведено аналітичний огляд застосунків-аналогів ресторанних систем, покращено навички аналізу функціоналу аналогічних систем у сфері обслуговування. Під час огляду були виокремлені як переваги, так і недоліки кожної з аналізованих систем. Такий підхід дозволив підготувати детальний аналіз застосунку, який розробляється.

Основні кроки аналізу включали визначення функцій, сценаріїв використання та складання специфікації вимог програмного забезпечення до майбутнього застосунку. Кожна з цих складових була ретельно вивчена та розглянута з точки зору їхнього впливу на роботу ресторанного бізнесу та користувачів.

Результатом цієї роботи стало чітке уявлення про те, які саме функції повинен виконувати розроблюваний застосунок, які сценарії використання він має підтримувати, а також які конкретні вимоги до програмного забезпечення мають бути враховані під час його розробки.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ЗАСТОСУНКУ

2.1 Сценарії використання

Use Case або сценарій використання[5] – перелік дій або ж сценарій, згідно з яким користувач взаємодіє із проєктованим застосунком, виконуючи будь- яку дію, що дозволена функціоналом застосунку.

Коротка форма: Резервація столика

Актор: Клієнт

Ціль: Успішно забронювати столик у ресторані через вебсайт.

Сценарій:

1. клієнт заходить на вебсайт ресторану;
2. вибирає опцію "Резервація столика";
3. вводить необхідну інформацію (дату, час, кількість осіб);
4. підтверджує резервацію;
5. система надсилає підтвердження на електронну пошту клієнта.

Поверхнева форма: Управління меню

Актор: Адміністратор

Ціль: Оновлення пунктів меню на вебсайті ресторану.

Сценарій:

1. адміністратор входить в систему через адміністративний інтерфейс;
2. вибирає розділ "Управління меню";
3. вносить зміни до меню (додає нові страви, оновлює ціни, та ін.);
4. підтверджує оновлення;
5. система автоматично оновлює інформацію на вебсайті.

Альтернативні сценарії:

– адміністратор виявляє помилку в описі страви та відразу ж редагує її;

– необхідність термінового видалення страви з меню через закінчення інгредієнтів;

– додавання сезонного меню;

– зміна цін на страви відповідно до зміни вартості інгредієнтів;
– оновлення фотографій страв у меню.

Замовлення товару.

Таблиця 2.1 – Сценарій обробки замовлення

Usecase section	**Comment**
Use Case Name	Обробка замовлення
Scope	Система управління рестораном
Level	Обробка замовлення в системі управління рестораном
Primary Actor	Адміністратор ресторану
Stakeholders and interests	1. адміністратор ресторану: Зацікавлений у ефективному управлінні замовленнями, задоволенні клієнтів, та оптимізації робочого часу; 2. клієнт: Зацікавлений в швидкому та зручному процесі замовлення, високій якості обслуговування; 3. персонал кухні: Зацікавлений в чіткому отриманні замовлень та специфікацій до них для ефективної роботи; 4. власник ресторану: Зацікавлений у збільшенні прибутку ресторану та ефективності управління.
Preconditions	1. адміністратор має бути авторизований в системі; 2. в системі мають бути актуальні дані про меню та наявність страв.
Success guarantee	1. замовлення успішно прийняте, оброблене та передане на кухню; 2. клієнт отримав підтвердження замовлення з орієнтовним часом доставки або готовності.

Продовження Таблиці 2.1

Main Success Scenario	1. клієнт робить замовлення через вебінтерфейс ресторану; 2. система автоматично надсилає замовлення в систему управління рестораном; 3. адміністратор переглядає замовлення та перевіряє наявність страв; 4. замовлення передається на кухню; 5. кухар підтверджує прийняття замовлення; 6. після приготування страви маркується як готова; 7. адміністратор відправляє клієнту повідомлення про готовність замовлення з часом доставки або самовивозу; 8. клієнт отримує замовлення; 9. замовлення маркується в системі як виконане; 10. система генерує звіт для адміністратора про статус замовлення.
Extensions	– якщо будь-яка страва з замовлення недоступна, адміністратор пропонує клієнту альтернативу або видаляє страву з замовлення; – якщо замовлення не може бути оброблене з технічних причин, система надсилає повідомлення адміністратору для ручного втручання; – у разі затримки приготування страви, система автоматично інформує клієнта про новий орієнтовний час готовності; – якщо клієнт хоче змінити замовлення після його підтвердження, адміністратор має можливість внести зміни в систему, за умови, що приготування ще не розпочате; – у випадку відміни замовлення клієнтом, замовлення маркується як скасоване, і клієнту надається повне або часткове повернення коштів згідно з політикою ресторану.

Кінець Таблиці 2.1:

Special Requirements	- система повинна мати високий рівень захисту даних клієнтів, включаючи платіжну інформацію; - інтерфейс користувача для адміністратора та кухні має бути інтуїтивно зрозумілим і забезпечувати швидкий доступ до всіх необхідних функцій; - система має забезпечувати можливість масштабування для обробки великої кількості замовлень одночасно;
Technology and Data Variations List	- використання мобільного застосунку для сповіщення кухарів про нові замовлення; - автоматична інтеграція з системами доставки для відслідковування статусу доставки замовлень клієнтам; - використання штучного інтелекту для прогнозування найпопулярніших страв і оптимізації запасів продуктів.
Frequency of Occurrence	Обробка замовлень є постійною діяльністю під час робочого часу ресторану.
Miscellaneous	- розробка детальних інструкцій для персоналу щодо використання системи; - налагодження комунікацій з клієнтами через систему для збору відгуків про якість обслуговування та страв; - планування регулярних оновлень системи для вдосконалення функціональності та виправлення помилок.

2.2 Діаграма варіантів використання

Діаграма варіантів використання (Use Case Diagram)[5] в UML — це високорівнева діаграма, яка показує взаємодію між зовнішніми акторами та системними функціями, які представлені у вигляді варіантів використання. Така діаграма є ключовим інструментом для визначення функціональних вимог до системи та для розуміння того, як користувачі (або інші системи) взаємодіють з системою.

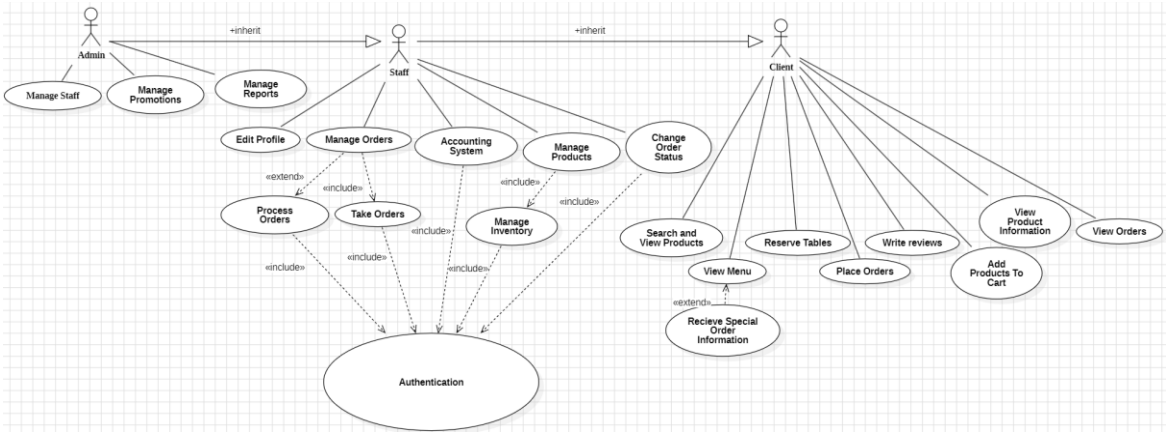

Рисунок 2.1 – Діаграма використання ресторанної системи

На діаграмі використання(рис 2.1) зображено трьох акторів, та їх функції:

1) адміністратор (Admin):
   - управління персоналом (Manage Staff);
   - управління акціями (Manage Promotions);
   - управління звітами (Manage Reports).
2) співробітник (Staff):
   - редагувати профіль (Edit Profile);
   - управління замовленнями (Manage Orders);
   - облікова система (Accounting System);
   - управління продуктами (Manage Products);
   - процес замовлень (Process Orders);
   - прийом замовлень (Take Orders);

- управління інвентарем (Manage Inventory);
- зміна статусу замовлення (Change Order Status).

3) клієнт (Client):

- пошук та перегляд продуктів (Search and View Products);
- перегляд меню (View Menu);
- отримання спеціальної інформації про замовлення (Receive Special Order Information);
- бронювання столиків (Reserve Tables);
- оформлення замовлень (Place Orders);
- написання відгуків (Write Reviews);
- перегляд інформації про продукт (View Product Information);
- перегляд замовлень (View Orders);
- додавання продуктів в кошик (Add Products To Cart).

Усі актори повинні проходити аутентифікацію перед виконанням будь-яких дій.

2.3 Розробка мокапу системи

Мокап[6] системи представляє собою деталізовану імітацію зовнішнього вигляду та функціональності розроблюваного програмного забезпечення. Цей інструмент використовується для наочної візуалізації інтерфейсу користувача та демонстрації основних функцій та процесів роботи системи, дозволяючи розробникам краще уявити кінцевий продукт та оцінити його потенціал до запуску в розробку.

На рисунку 2.9 зображено мокап головної сторінки сайту. У лівому кутку хедера розташована назва ресторана. Праворуч від назви знаходиться кнопка меню навігації по різним сторінкам, при натисканні на яку можна сховати назви та залишити тільки іконки. Під хедером, вся ліва частина зайнята sidebar-menu, яке і є основною навігацією по сторінці, яка відповідає за управління рестораном.

Праворуч від sidebar-menu є основний контент головної сторінки, яка відображає основну інформацію про те, що відбувається в ресторані і також допомогає швидко перейти до необхідної категорії.

Нижче, на рисунку 2.9 зображено мокап dashboard сторінки:

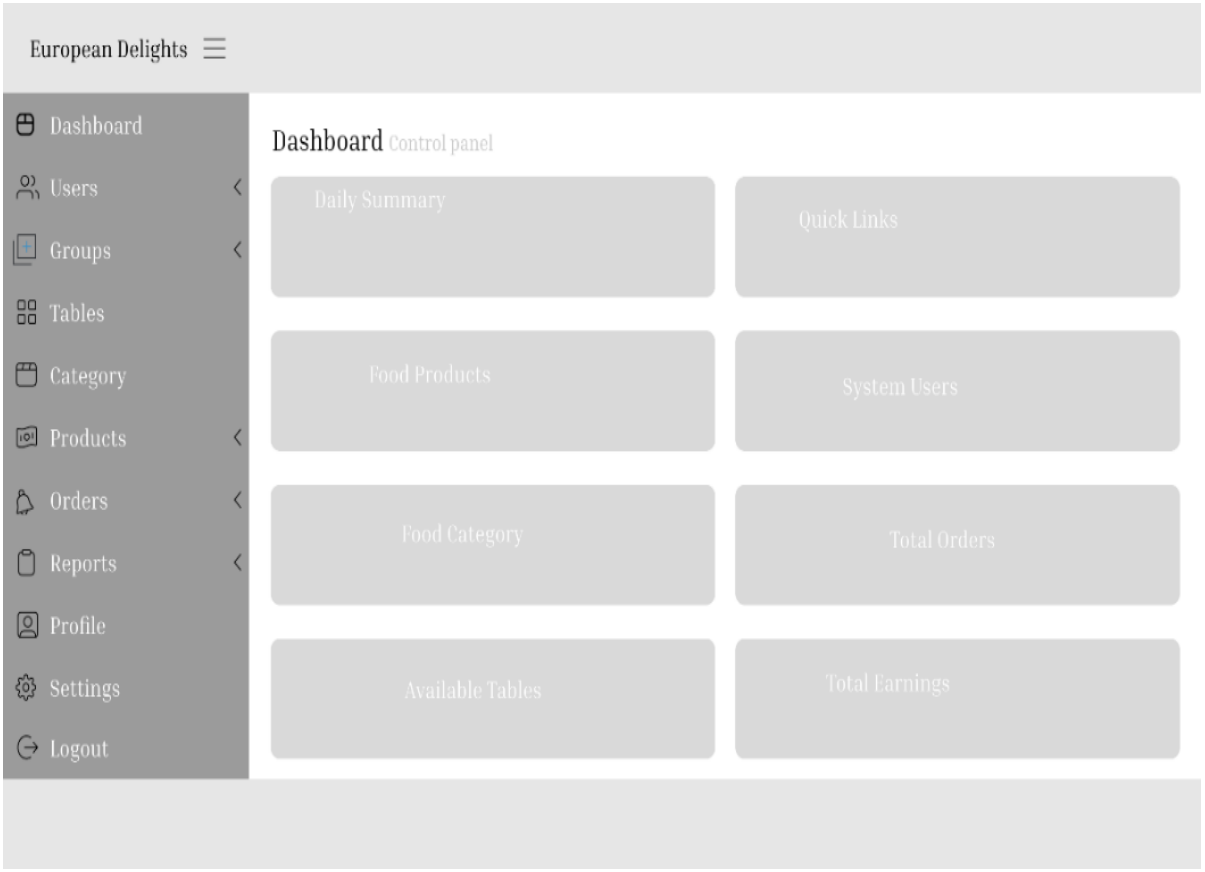

Рисунок 2.9 – Мокап dashboard сторінки

На рисунку 2.10 зображено мокап сторінки користувача, опція Add. При натисканні на категорію User в sidebar, випадає меню, в якому при натисканні на Add User можна створити користувача. Ця фунція буде доступна лише адміністратору, оскільки права звичайного персоналу не мають розповсюджуватись на всю функціональність автоматизованної ресторанної системи. В цьому мокапі продемонстровано основні поля, за якими створюється користувач.

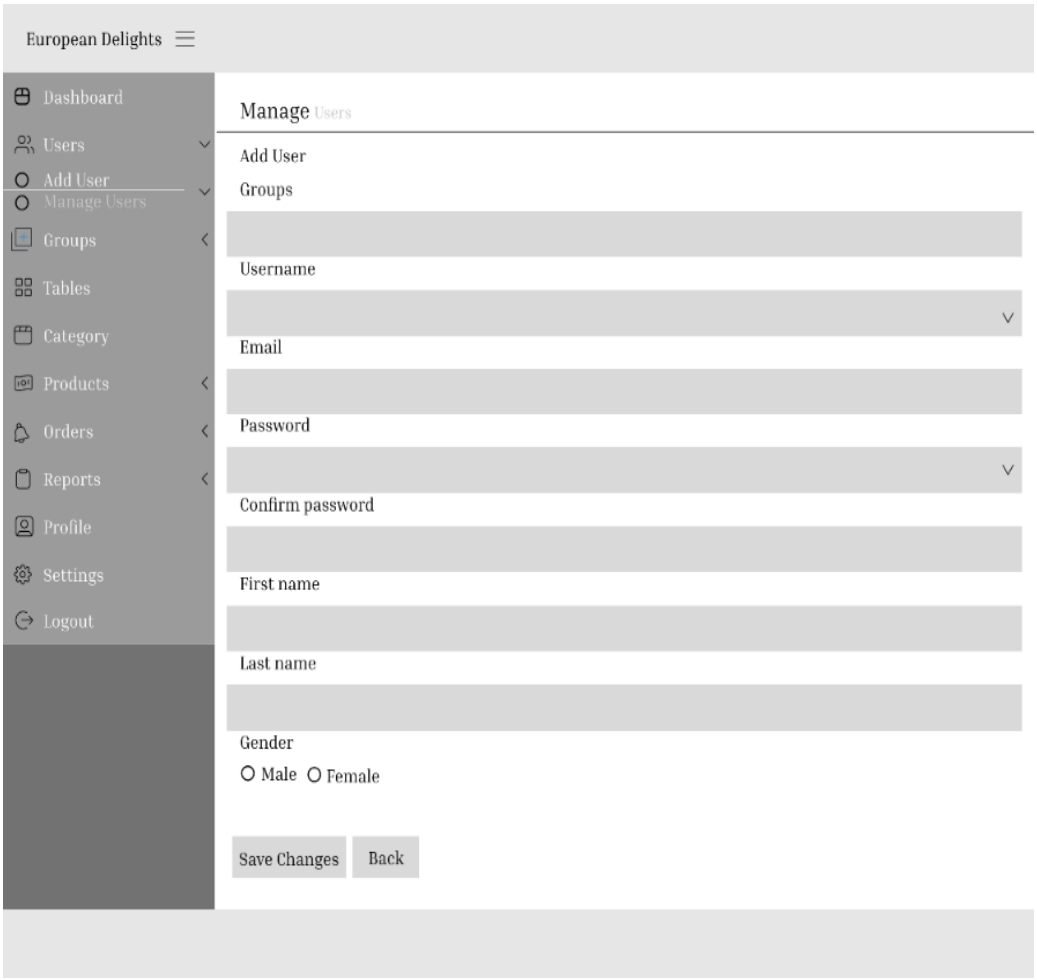

Рисунок 2.10 – Мокап сторінки створення користувача

На рисунку 2.11 зображено мокап сторінки користувача, опція Manage. При натисканні на категорію User в sidebar, випадає меню, в якому при натисканні на Manage User можна змінити властивості користувача. На мокапі з рисунка 3 продемонстровано можливість перегляду списку користувачів. В контенті, в правій частині таблиці є 2 кнопки, червона відповідає за видалення користувача з системи, голубого кольору за редагування користувача. Також над таблицею є 2 опції, перша відповідає за відображення кількості відображаємих користувачів в таблиці, і розташована вона з лівої частини. Друга опція відповідає за пошук користувача, її розташовано праворуч над таблицею в контенті.

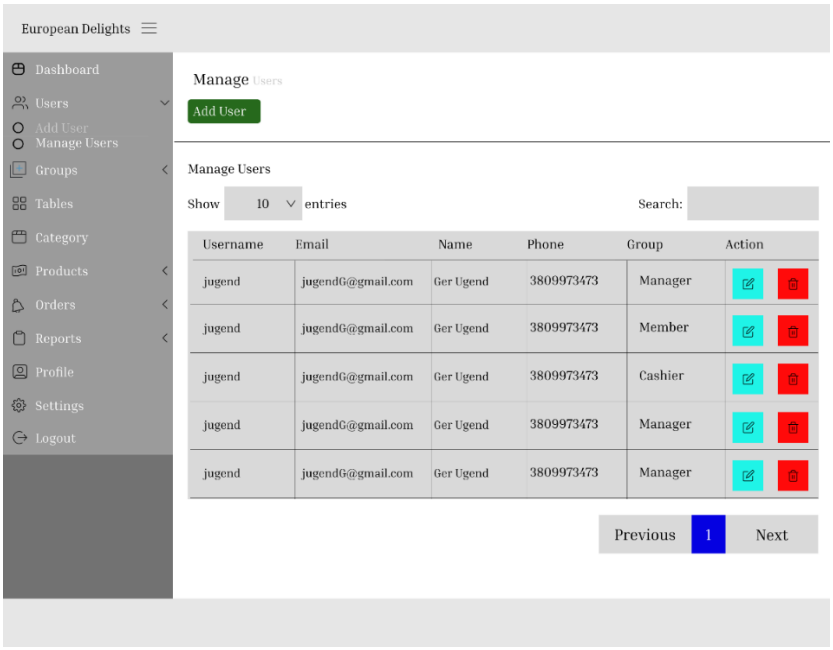

Рисунок 2.11 – Мокап сторінки створення користувача

На рисунку 2.12 зображено мокап групи, а саме функцію додавання груп. В sidebar-menu обрано Groups, в dropdown menu – Add Group. В контенті розташовано check-box контейнери, які відповідають за права, які надаються створеній групі.

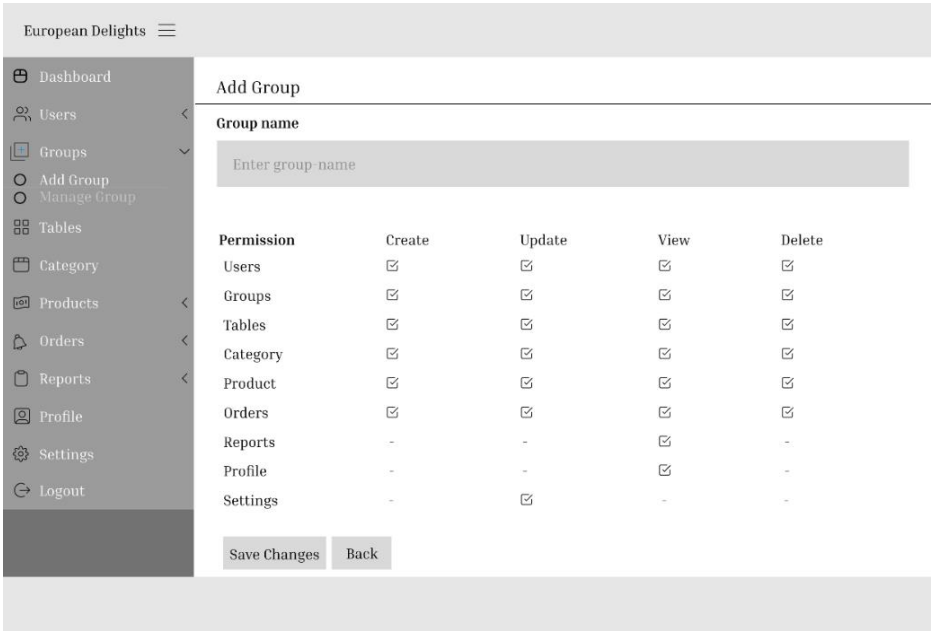

Рисунок 2.12 – Мокап сторінки створення груп

На рисунку 2.13 зображено мокап сторінки перегляду груп. В контенті створена таблиця, в якій можна побачити назву групи, редагувати групу, видалити групу, шукати групу та налаштувати відображення.

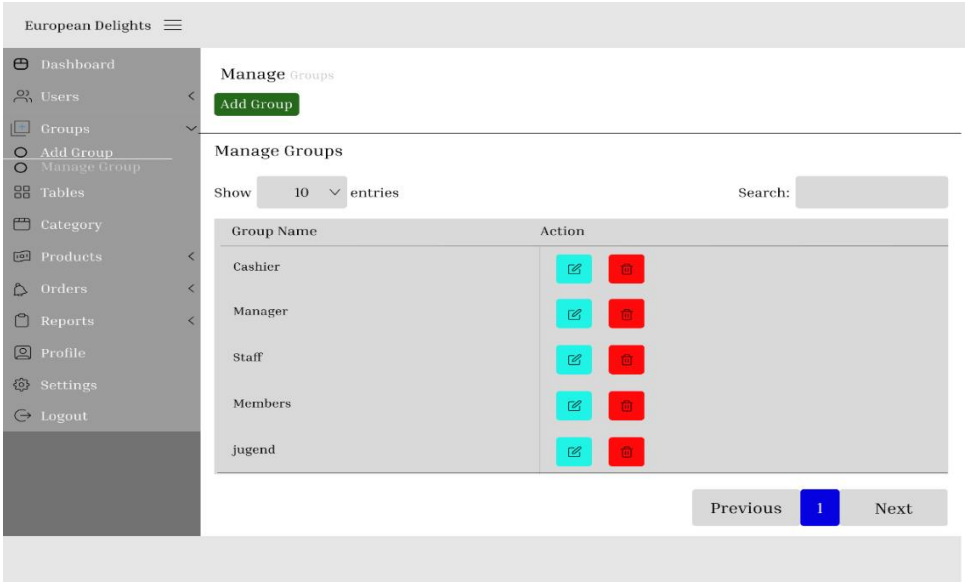

Рисунок 2.13 – Мокап сторінки перегляду груп

На рисунку 2.14 зображено мокап сторінки управління столами. Під назвою Manage Tables є кнопка, за допомогою якої можна додати стіл, нижче наведено таблицю, в якій демонструється назва столу, його вміст, доступність, статус та дії для редагування та видалення столу.

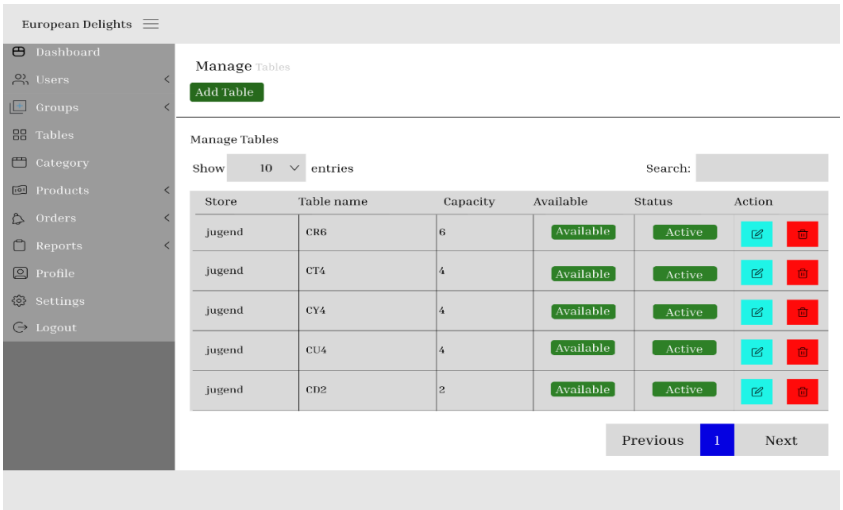

Рисунок 2.14 – Мокап сторінки управління столами

На рисунку 2.15 зображено мокап управління категоріями. В ньому мається можливість переглянути категорії, додати категорію, редагувати та видалити категорію. Також є поле для налаштування відображення та пошуку категорії.

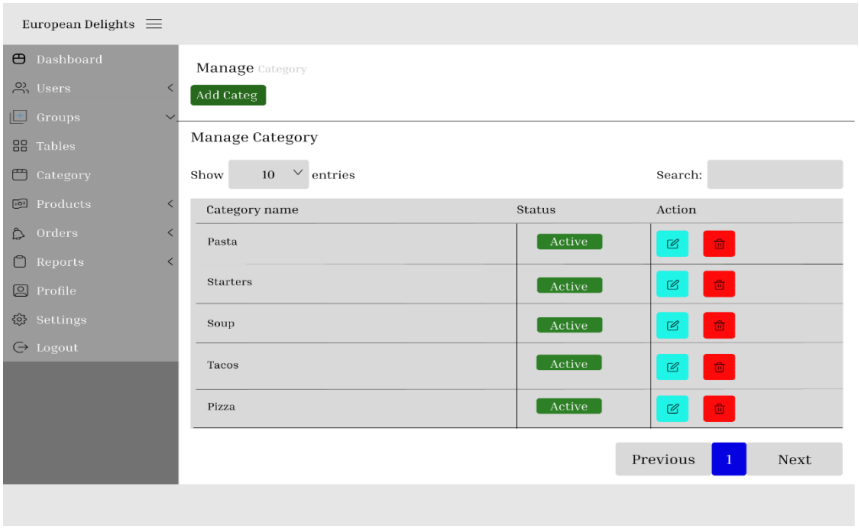

Рисунок 2.15 – Мокап управління категоріями

На рисунку 2.16 відображено мокап сторінки перегляду профіля користувача. В контенті є можливість переглянути основну інформацію про користувача, та його групу.

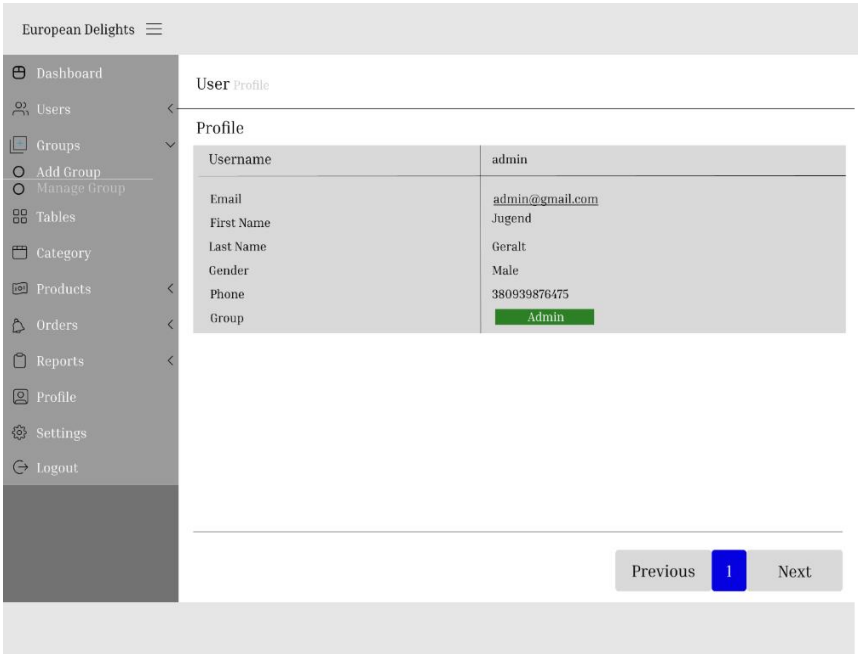

Рисунок 2.16 – Мокап сторінки перегляду користувача

На рисунку 2.17 зображено мокап сторінки звітів. На ньому можна побачити звіти за вказаний рік, в таблиці буде продемонстровано звіт за кожний місяць

Зображення мокапу сторінки звітів:

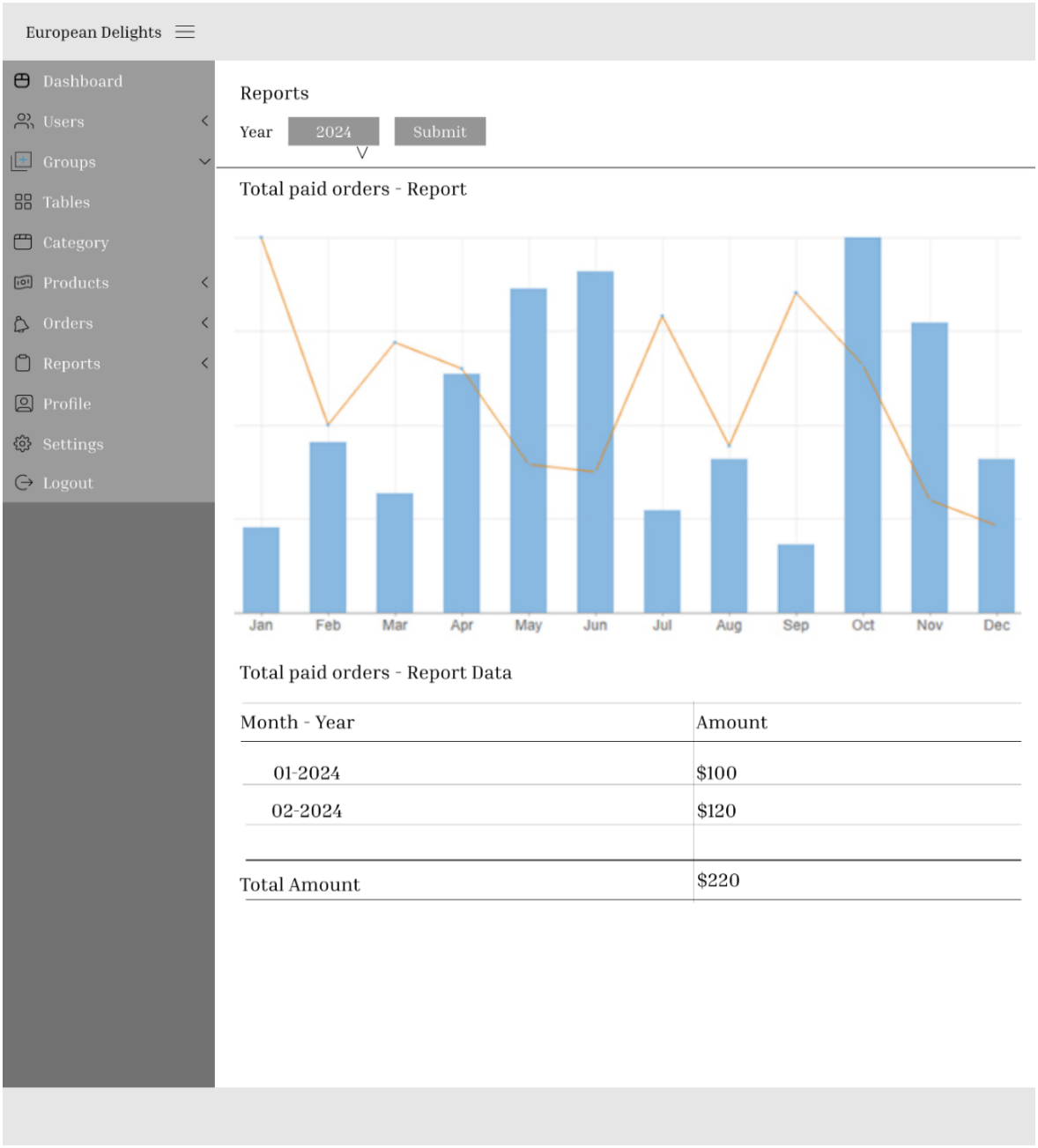

Рисунок 2.17 – Мокап сторінки звітів

Наведені мокапи зображають майбутній інтерфейс користувача. Мокапи допомагають уникнути помилок у дизайні, забезпечуючи більш гладке впровадження функціональності та оптимізацію користувацького досвіду

Висновки до розділу 2

Спроєктовано вебзастосунок автоматизованої системи управління рестораном, основними етапами якого стали розробка діаграм сценаріїв використання та створення мокапів системи. Діаграми сценаріїв використання допомогли ідентифікувати ключові функціональні можливості системи та визначити дії, які повинні бути доступні користувачам. Це сприяло структуризації процесів у ресторані та забезпечило розуміння, як система може автоматизувати ці процеси.

Завдяки розробці діаграм варіантів використання, були деталізовані основні сценарії, які система повинна підтримувати, надавши більш чітке уявлення про взаємодію користувачів із системою в конкретних ситуаціях. Діаграма класів визначила структуру даних і об'єктів системи, що допомогло встановити класи та взаємодії між ними.

Мокапи системи продемонстрували інтерфейс користувача, дозволяючи краще уявити, як система буде виглядати та як користувачі взаємодіятимуть з нею. Вони допомогли уникнути помилок у дизайні, забезпечуючи плавне впровадження функціональності та оптимізацію користувацького досвіду.

3 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

3.1 Огляд технологій та їх порівняння

Розробка програмного забезпечення вимагає ретельного вибору відповідних технологій для забезпечення оптимальної продуктивності, гнучкості та підтримуваності системи. Існує безліч технологій для реалізації різних аспектів проєкту. У цьому розділі буде проведено огляд основних технологій для фронтенду, бекенду та баз даних, а також їх порівняння.

Фронтенд технології:

Angular:

Angular[7] — це фреймворк для розробки вебзастосунків, створений компанією Google. Він забезпечує повний набір інструментів для побудови динамічних вебзастосунків, включаючи двостороннє зв'язування даних, модульну структуру та підтримку компонентного підходу. Angular відомий своєю строгістю та чіткою структурою, що полегшує масштабування проєктів.

React.js

React.js[8] — це бібліотека JavaScript для створення користувацьких інтерфейсів, розроблена Facebook. React використовує концепцію компонентів та віртуальний DOM для досягнення високої продуктивності. React часто використовують разом з іншими бібліотеками, такими як Redux або MobX для управління станом, і React Router для маршрутизації.

Vue.js[9]

Vue.js — це прогресивний фреймворк для створення користувацьких інтерфейсів, який поєднує в собі найкращі риси інших фронтенд технологій. Він забезпечує високу продуктивність та простоту використання, має чітку документацію та простий синтаксис. Vue.js підтримує реактивні двосторонні зв'язки даних, компоненти та інтеграцію з іншими бібліотеками. Нижче наведено таблицю 3.1, в якій порівнюються основні frontend технології.

Таблиця 3.1 – Порівнняня frontend технологій

Технологія	Продуктивність	Гнучкість	Спільнота	Простота використання	Підтримка корпоративних рішень
Angular	Висока	Висока	Висока	Середня	Висока
React.js	Висока	Висока	Висока	Висока	Висока
Vue.js	Висока	Висока	Середня	Висока	Середня

Angular, React.js та Vue.js є потужними інструментами для створення сучасних фронтенд застосунків. Angular надає комплексний підхід з великою кількістю вбудованих функцій, що полегшує роботу над великими проєктами. React.js забезпечує високу продуктивність і гнучкість завдяки віртуальному DOM і компонентній архітектурі. Vue.js виділяється своєю простотою і легкістю у навчанні, що робить його ідеальним вибором для швидкого старту проєктів.

Бекенд технології:

ASP.NET Core

ASP.NET Core[10] — це кросплатформовий, високопродуктивний фреймворк для створення сучасних хмарних та вебзастосунків, розроблений Microsoft. ASP.NET Core забезпечує високу продуктивність, масштабованість та надійність. Він підтримує модульну структуру, вбудовані засоби для роботи з безпекою, а також інтеграцію з різними базами даних через Entity Framework Core.

Node.js з Express.js

Node.js[11] — це серверна платформа для виконання JavaScript, яка використовує неблокуючу модель вводу-виводу, що робить її легкою та ефективною. Express.js[12] - це мінімалістичний фреймворк для Node.js, який забезпечує основні інструменти для побудови вебзастосунків та API. Node.js разом з Express.js дозволяє використовувати JavaScript як на клієнтській, так і на серверній частині.

Django

Django[13] — це високорівневий фреймворк для Python, який сприяє швидкій розробці та зручності використання. Він забезпечує велику кількість вбудованих компонентів для роботи з базами даних, безпеки, аутентифікації та адміністрування. Django відомий своєю чіткою архітектурою та високою продуктивністю для більшості вебзастосунків.

Таблиця 3.2 – Порівняння Backend технологій

Технологія	Продуктивність	Гнучкість	Спільнота	Простота використання	Підтримка корпоративних рішень
ASP.NET Core	Висока	Висока	Висока	Середня	Висока
Node.js + Express.js	Висока	Висока	Висока	Висока	Висока
Django	Висока	Середня	Висока	Висока	Висока

ASP.NET Core, Node.js з Express.js та Django є відмінними виборами для бекенд розробки. ASP.NET Core пропонує потужні можливості для розробки корпоративних застосунків з високою продуктивністю та підтримкою від Microsoft. Node.js з Express.js забезпечує швидкість і гнучкість завдяки неблокуючій моделі вводу-виводу і використанню JavaScript на всіх рівнях застосунка. Django, з іншого боку, надає багато вбудованих функцій для швидкої розробки та зручності використання, особливо підходить для проєктів на Python.

Технології для баз даних

SQL Server

SQL Server[14] — це реляційна база даних, розроблена Microsoft. Вона забезпечує високу продуктивність і масштабованість, підтримує складні транзакції та запити, має високий рівень безпеки. SQL Server широко використовується в корпоративних рішеннях завдяки своїй надійності та широким можливостям.

MySQL [15]

MySQL — це одна з найпопулярніших реляційних баз даних з відкритим кодом. Вона забезпечує високу продуктивність і масштабованість, підтримує широкий спектр операційних систем та має велику спільноту користувачів. MySQL часто використовується в веброзробці завдяки своїй доступності та простоті використання.

PostgreSQL [16]

PostgreSQL — це потужна реляційна база даних з відкритим кодом, яка підтримує складні запити, транзакції та зберігання даних у форматі JSON. PostgreSQL відома своєю високою продуктивністю, надійністю та можливістю розширення. Вона забезпечує підтримку різних мов програмування та має велику спільноту розробників.

Таблиця 3.3 – Порівняння баз даних

Технологія	Продуктивність	Гнучкість	Спільнота	Простота використання	Підтримка корпоративних рішень
SQL Server	Висока	Середня	Висока	Середня	Висока
MySQL	Висока	Середня	Висока	Висока	Висока
PostgreSQL	Висока	Висока	Середня	Середня	Висока

SQL Server, MySQL та PostgreSQL є відмінними базами даних для різних сценаріїв використання. SQL Server надає потужні можливості для корпоративних застосунків з високим рівнем безпеки та підтримкою складних транзакцій. MySQL є популярним вибором для веброзробки завдяки своїй простоті та доступності. PostgreSQL забезпечує високу продуктивність і гнучкість, особливо підходить для застосунків, що потребують складних запитів та транзакцій.

3.2 Обґрунтування вибору технологій для реалізації системи. Вибір програмних засобів для розробки

Розробка вебзастосунків вимагає ретельного вибору технологій та підходів. Важливо зрозуміти, як вони були відібрані для проєкту. При виборі технології важливо враховувати наступне:

- складність програмного забезпечення;
- наявність ресурсів;
- наявність готових компонентів;
- наявність технічної документації;
- характеристики якості ПЗ;
- витрати на технологію;
- ліцензійну політику;
- вимоги безпеки.

У рамках даного проєкту для реалізації автоматизованого робочого місця адміністратора ресторану з підтримкою динамічного контенту було обрано наступні технології:

Фронтенд

Angular:

Angular - це фреймворк для розробки вебзастосунків від Google, що забезпечує розробникам інструменти для створення динамічних та інтерактивних вебзастосунків. Основні причини вибору Angular:

- rxJS: бібліотека для роботи з асинхронним програмуванням за допомогою Observable;
- standalone компоненти: дозволяють створювати незалежні модулі, що полегшує масштабування та тестування;
- angular 17: остання версія, яка забезпечує покращення продуктивності та нові функції;

– HttpClientModule: модуль для роботи з HTTP-запитами, що спрощує взаємодію з сервером;

– AuthService: сервіс для управління аутентифікацією та авторизацією користувачів.

Бекенд

ASP.NET

– ASP.NET Core: кросплатформовий, високопродуктивний фреймворк для створення сучасних, хмарних та інтернет-застосунків;

– ASP.NET MVC: архітектурний шаблон Model-View-Controller для розділення застосунку на три компоненти, що сприяє полегшенню розробки, тестуванню та підтримці;

– ASP.NET Core WebAPI: інструмент для створення HTTP-служб, який дозволяє різним клієнтам (веб, мобільні тощо) взаємодіяти з сервером;

– FluentValidation: бібліотека для впровадження валідації моделей за допомогою зручного і чіткого синтаксису;

– Azure: хмарна платформа від Microsoft, що забезпечує надійність, масштабованість та безпеку;

– Azure Blob Storage: сервіс для зберігання великих об'ємів неструктурованих даних, що підходить для зберігання зображень, документів та інших файлів;

– Azure Telemetry: інструмент для моніторингу та аналізу роботи застосунку в реальному часі;

– .NET Aspire: платформа для швидкої розробки корпоративних застосунків;

– EF Core: ORM (Object-Relational Mapper) для роботи з базами даних, що спрощує взаємодію з реляційними БД;

– MediatR: бібліотека для впровадження патерну CQRS (Command Query Responsibility Segregation), що сприяє покращенню організації коду;

– FluentAssertions: бібліотека для написання зрозумілих та легких у підтримці тестів.

База даних

SQL Server:

надійна, масштабована та високопродуктивна реляційна база даних, яка забезпечує високий рівень безпеки та підтримку складних запитів і транзакцій.

Архітектура

Багатошарова архітектура типу "цибулина"(рис3.1) (Multilayered Onion-like 3 tier architecture)[17]

– Презентаційний шар (Presentation Layer): відповідає за взаємодію з користувачем, використовуючи Angular для створення динамічних та інтерактивних інтерфейсів.

– Логічний шар (Business Logic Layer): реалізує бізнес-логіку застосунку з використанням ASP.NET Core та відповідних бібліотек і сервісів.

– Шар даних (Data Access Layer): забезпечує взаємодію з базою даних через EF Core, забезпечуючи ефективний доступ та маніпуляцію даними.

Обрані технології забезпечують створення надійного, масштабованого та підтримуваного рішення для автоматизації роботи адміністратора ресторану з підтримкою динамічного контенту, що дозволяє ефективно управляти ресурсами та забезпечувати високу якість обслуговування клієнтів.

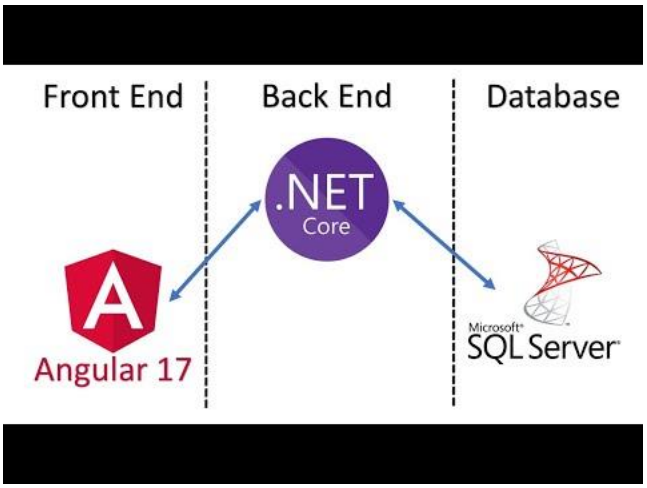

Рисунок 3.1 – Angular, ASP.NET та SQLServer

3.3 Опис архітектури системи

У цьому розділі розглядається архітектура автоматизованого робочого місця адміністратора ресторану. Архітектура системи побудована на багатошаровій структурі типу "цибулина", що забезпечує високу гнучкість, масштабованість та підтримуваність. Основні шари системи включають презентаційний шар, логічний шар, шар даних та інфраструктурний шар.

Архітектура системи включає наступні основні шари та компоненти (рис. 3.1):

1) презентаційний шар (Presentation Layer)
   - angular: Фреймворк для побудови динамічних та інтерфейсів користувача;
   - компоненти: Реалізують різні частини інтерфейсу, включаючи форми для введення даних, таблиці для відображення інформації та навігаційні елементи;
   - сервіси: Обробляють логіку на стороні клієнта, зокрема аутентифікацію, авторизацію та роботу з HTTP-запитами.
2) логічний шар (Business Logic Layer)
   - ASP.NET Core: Фреймворк для створення бізнес-логіки та API;
   - контролери: Обробляють запити від клієнта, викликаючи відповідні сервіси;
   - сервіси бізнес-логіки: Реалізують основні функціональні можливості системи, включаючи управління замовленнями, резервуванням столиків, обробкою платежів та управління меню;
   - FluentValidation: Забезпечує валідацію даних, що надходять від користувача.
3) шар даних (Data Access Layer)
   - EF Core (Entity Framework Core)**:** ORM для взаємодії з базою даних;

– репозиторії: Реалізують доступ до даних, забезпечуючи базові операції CRUD (Create, Read, Update, Delete);

– моделі даних: Визначають структуру даних у базі, включаючи таблиці для зберігання інформації про користувачів, замовлення, меню тощо.

4) інфраструктурний шар (Infrastructure Layer)

– Azure Blob Storage: Сервіс для зберігання зображень, документів та інших файлів;

– Azure Telemetry: Інструмент для моніторингу та аналізу роботи системи;

– MediatR: Паттерн CQRS для управління запитами та командами в системі;

– FluentAssertions**:** Бібліотека для створення зрозумілих та підтримуваних тестів.

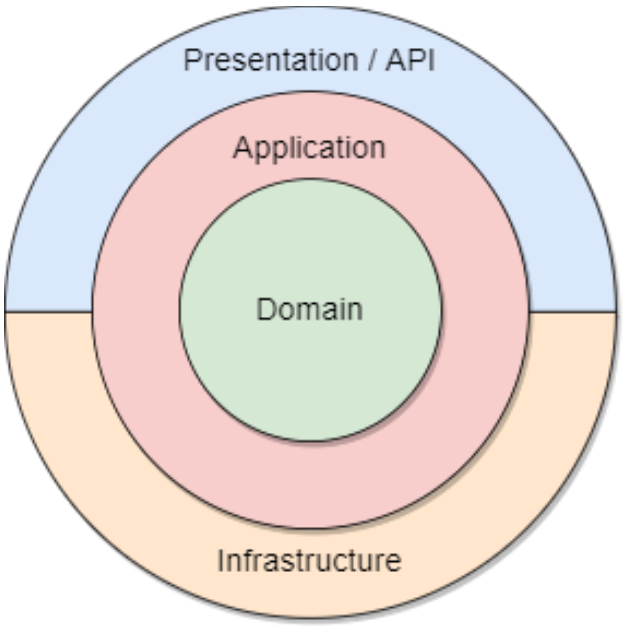

Рисунок 3.2 – Спрощена архітектурна схема системи

Висновки до розділу 3

Проведено глибокий аналіз технологій, що можуть бути використані для реалізації автоматизованого робочого місця адміністратора ресторану. Відповідно до проведеного огляду, були розглянуті технології для фронтенду, бекенду та баз даних. Кожна з цих технологій має свої переваги та недоліки, що детально розглядаються в порівняльних таблицях.

Фронтенд технології надають розробникам потужні інструменти для створення динамічних та інтерактивних інтерфейсів користувача. Angular відзначається своєю комплексністю та великою кількістю вбудованих функцій, що робить його особливо підходящим для великих проєктів. React.js забезпечує високу продуктивність завдяки використанню віртуального DOM, а Vue.js вирізняється своєю простотою та легкістю у навчанні.

Серед бекенд технологій, ASP.NET Core, Node.js з Express.js та Django пропонують високий рівень продуктивності та гнучкості. ASP.NET Core є ідеальним вибором для розробки корпоративних застосунків з підтримкою від Microsoft. Node.js з Express.js, завдяки своїй неблокуючій моделі вводу-виводу, забезпечує швидкість і гнучкість розробки, а Django надає багато вбудованих функцій для швидкої та зручної розробки вебзастосунків.

Серед баз даних, SQL Server, MySQL та PostgreSQL пропонують відмінні рішення для різних сценаріїв використання. SQL Server є надійним вибором для застосунків завдяки високій продуктивності та безпеці. MySQL є популярним завдяки своїй доступності та простоті використання. PostgreSQL забезпечує високу продуктивність та гнучкість.

На основі проведеного аналізу, для реалізації проєкту було обрано Angular для фронтенду, ASP.NET Core для бекенду та SQL Server для бази даних. Ці технології забезпечують необхідний рівень продуктивності, гнучкості та підтримуваності, що є критично важливими для успішної реалізації проєкту . Обрані технології також мають велику спільноту розробників та хорошу документацію, що додатково підтверджує їх доцільність використання в даному проєкті.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Діаграма класів та діаграми сутність-зв’язок

Діаграма класів[5] є одним з ключових інструментів в UML (Unified Modeling Language) для візуалізації архітектури програмного забезпечення. Вона детально відображає структуру програмного коду, представляючи класи, їхні атрибути, методи та типи взаємозв'язків між класами, такі як асоціації, спадкування та реалізації інтерфейсів.

Діаграму класів наведено на рисунку 4.1

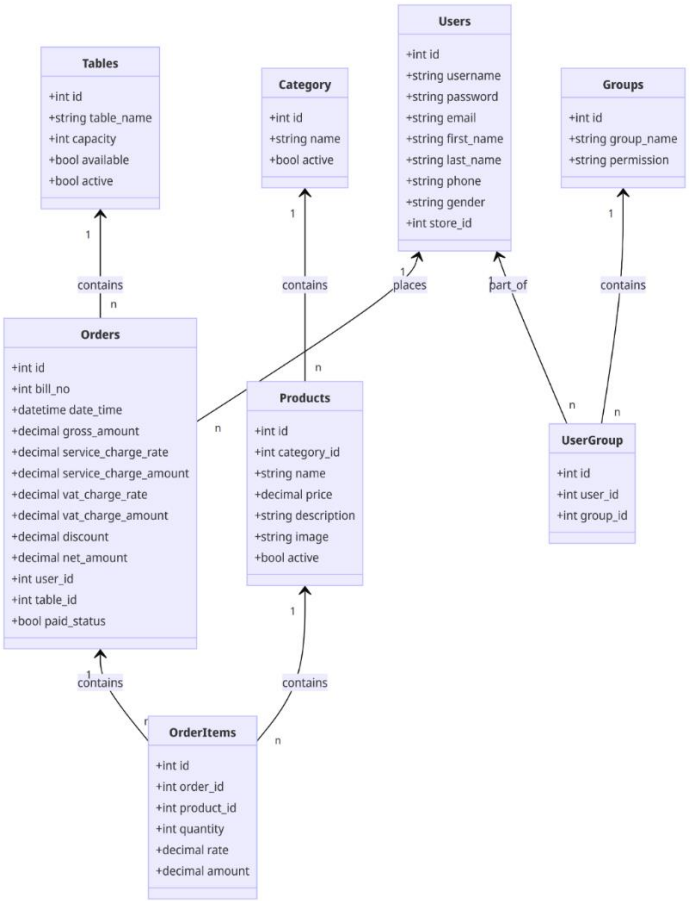

Рисунок 4.1 – Діаграма класів

Опис класів:

1) клас «Tables» (Столи):
   - атрибути:
     - +int id;

- +string table_name;
- +int capacity;
- +bool available;
- +bool active.

– зв'язки:

- один-до-багатьох з класом «Orders».

2) клас «Users» (Користувачі):

– атрибути:

- +int id;
- +string username;
- +string password;
- +string email;
- +string first_name;
- +string last_name;
- +string phone;
- +string gender;
- +int user_id;

– зв'язки:

- один-до-багатьох з класом «Orders»;
- багато-до-одного з класом «UserGroup».

3) клас «Groups» (Групи):

– атрибути:

- +int id;
- +string group_name;
- +string permission.

– зв'язки:

- один-до-багатьох з класом «UserGroup».

4) клас «Category» (Категорії):

- атрибути:
  - +int id;
  - +string name;
  - +string active.
- зв'язки:
  - один-до-багатьох з класом «Products».

5) клас «Products» (Продукти):

- атрибути:
  - +int id;
  - +int category_id;
  - +string name;
  - +decimal price;
  - +string description;
  - +string image;
  - +bool active.
- зв'язки:
  - один-до-багатьох з класом «OrderItems».

6) клас «Orders» (Замовлення):

- атрибути:
  - +int id;
  - +int bill_no;
  - +datetime date_time;
  - +decimal gross_amount;
  - +decimal service_charge_rate;
  - +decimal service_charge_amount;
  - +decimal vat_charge_rate;
  - +decimal vat_charge_amount;
  - +decimal discount;

- +decimal net_amount;
- +int user_id;
- +int table_id;
- +bool paid_status.

– зв'язки:

- один-до-багатьох з класом «OrderItems».

7) клас «OrderItems» (Продукти замовлення):

– атрибути:

- +int id;
- +int order_id;
- +int product_id;
- +int quantity;
- +decimal rate;
- +decimal amount.

– зв'язки:

- один-до-багатьох з класом «Orders».

8) клас «UserGroup» (Група користувача):

– атрибути:

- +int id;
- +int user_id;
- +int group_id.

– зв'язки:

- багато-до-одного з класом «Users»;
- багато-до-одного з класом «Groups».

4.2 Діаграма сутність-зв’язок

Діаграма "сутність-зв'язок" (Entity-Relationship Diagram, ERD) є одним з основних інструментів для моделювання та візуалізації структури бази даних.

ERD діаграма дозволяє детально відобразити сутності (таблиці), їх атрибути (поля) та взаємозв'язки між ними, що є критично важливим для розробки ефективної та логічної структури бази даних.

ERD діаграма допомагає зрозуміти, як дані взаємодіють між собою, і забезпечує чітке уявлення про архітектуру бази даних. Вона є основою для проєктування реляційних баз даних і використовується на початкових етапах розробки, щоб визначити структуру даних та їх взаємозв'язки.

Опис діаграми сутність-зв'язок

В контексті автоматизованого робочого місця адміністратора ресторану, ERD діаграма[5] включає такі сутності: "Користувачі", "Групи", "Столи", "Продукти", "Замовлення", "Позиції замовлення", "Звіти", "Профіль" та "Налаштування". Кожна сутність має свій набір атрибутів, а також взаємозв'язки між собою.

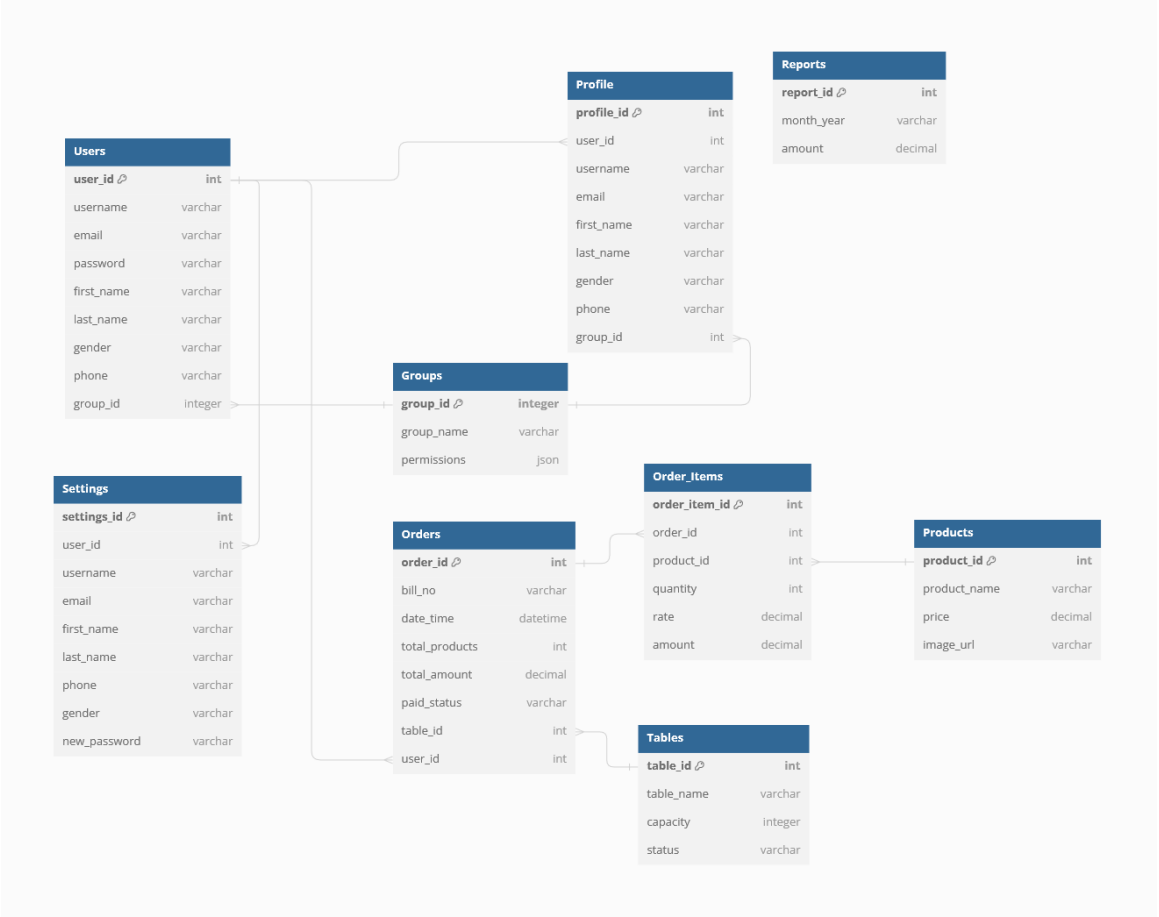

Рисунок 4.2 – Діаграма сутність-зв’язок

4.3 Діаграми послідовностей

Діаграма послідовності[5] є одним із типів діаграм в UML (Unified Modeling Language), призначених для зображення взаємодій між об'єктами в контексті конкретного процесу або сценарію. Ця діаграма демонструє послідовність повідомлень, що обмінюються між об'єктами протягом часу, ілюструючи як вони співпрацюють для виконання певної задачі. Діаграма послідовності в UML зображує взаємодію об'єктів у визначеній послідовності для виконання конкретного сценарію, уточнюючи хід подій та взаємодії між об'єктами в процесі.

Нижче на рисунках 4.3-4.8 наведено діаграми послідовності автоматизованної ресторанної системи.

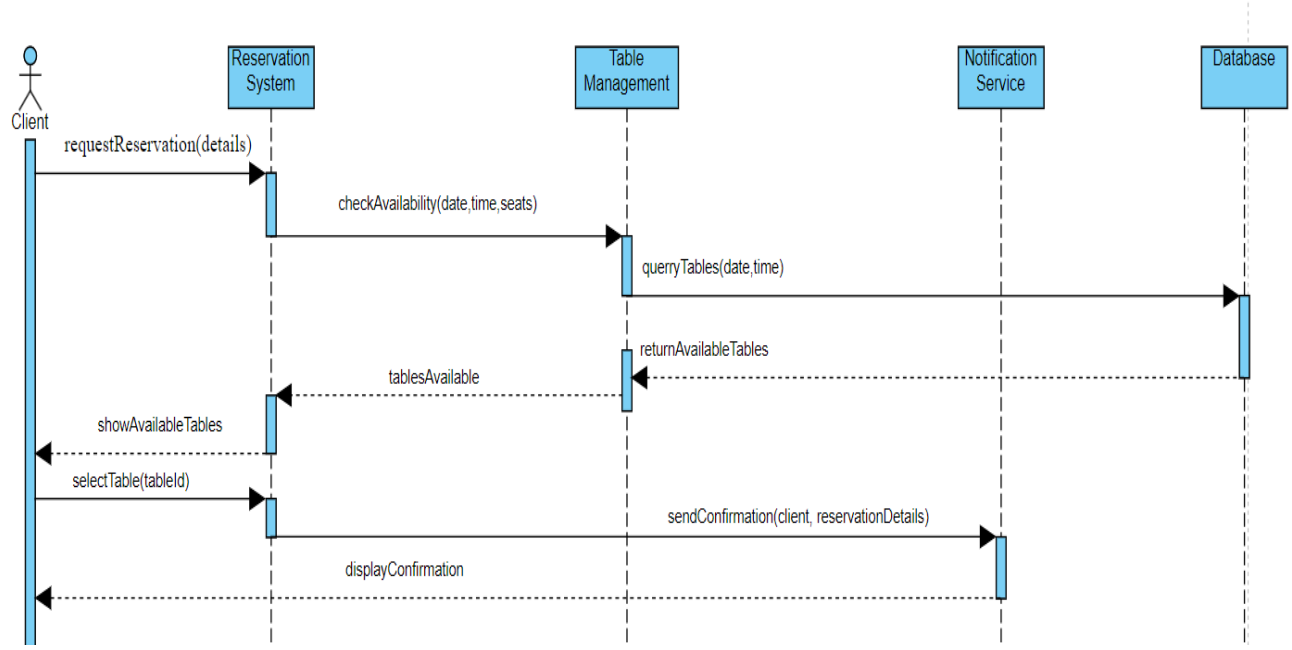

Рисунок 4.3 – Sequence diagram для варіанту використання «Резервація столу»

Нижче на рисунку 4.4 наведено діаграму послідовності для варіанту використання для перегляду меню:

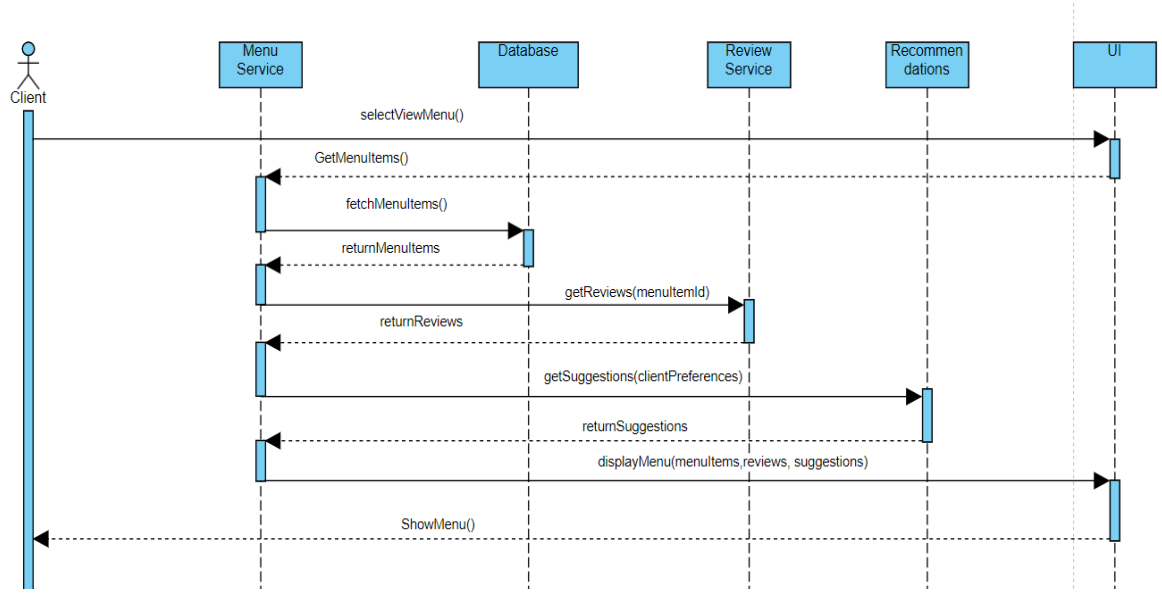

Рисунок 4.4 – Sequence diagram для варіанту використання «Перегляд меню»

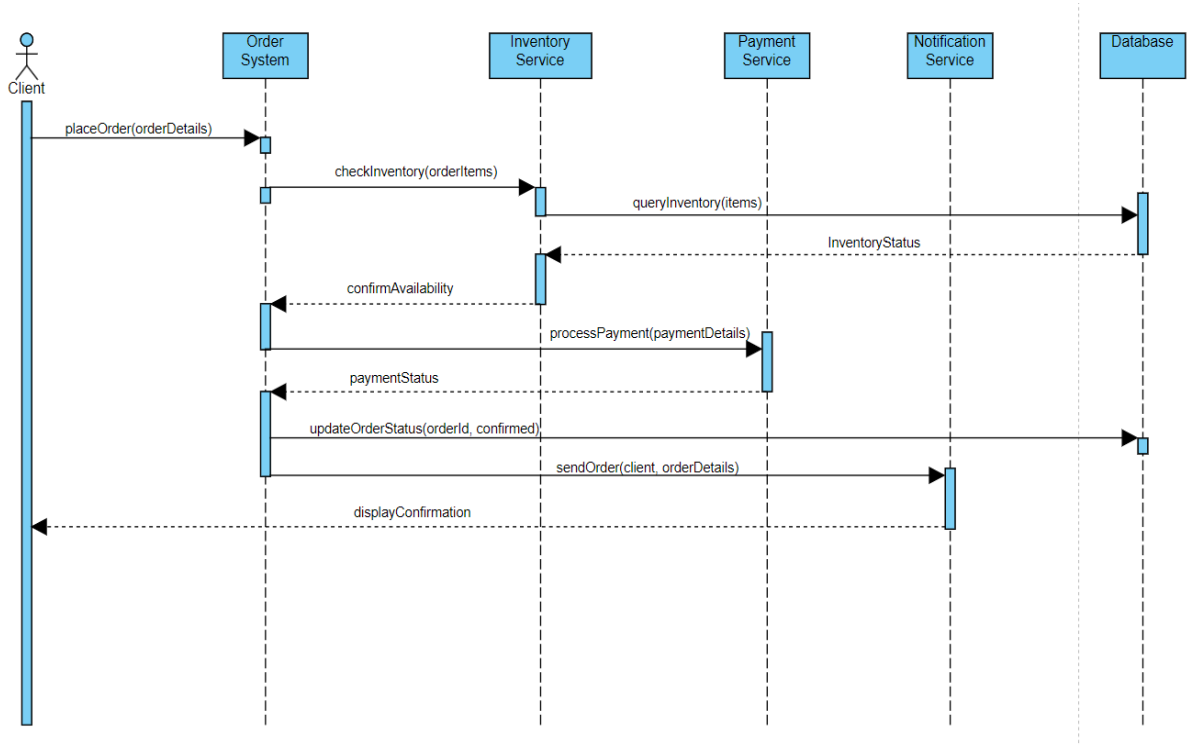

Рисунок 4.5 – Sequence diagram для варіанту використання «Розміщення замовлення»

На рисунку 4.6 зображено діаграму послідовності для варіанту використання «Управління інвентарем»:

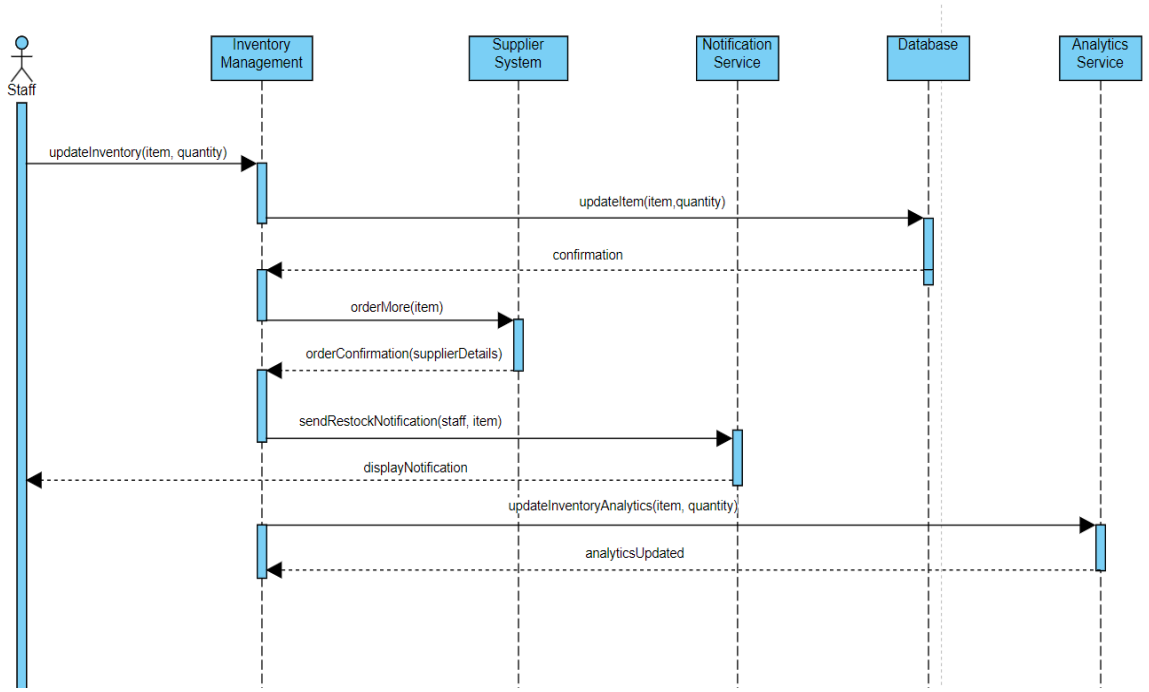

Рисунок 4.6 – Sequence diagram для варіанту використання «Управління інвентарем»

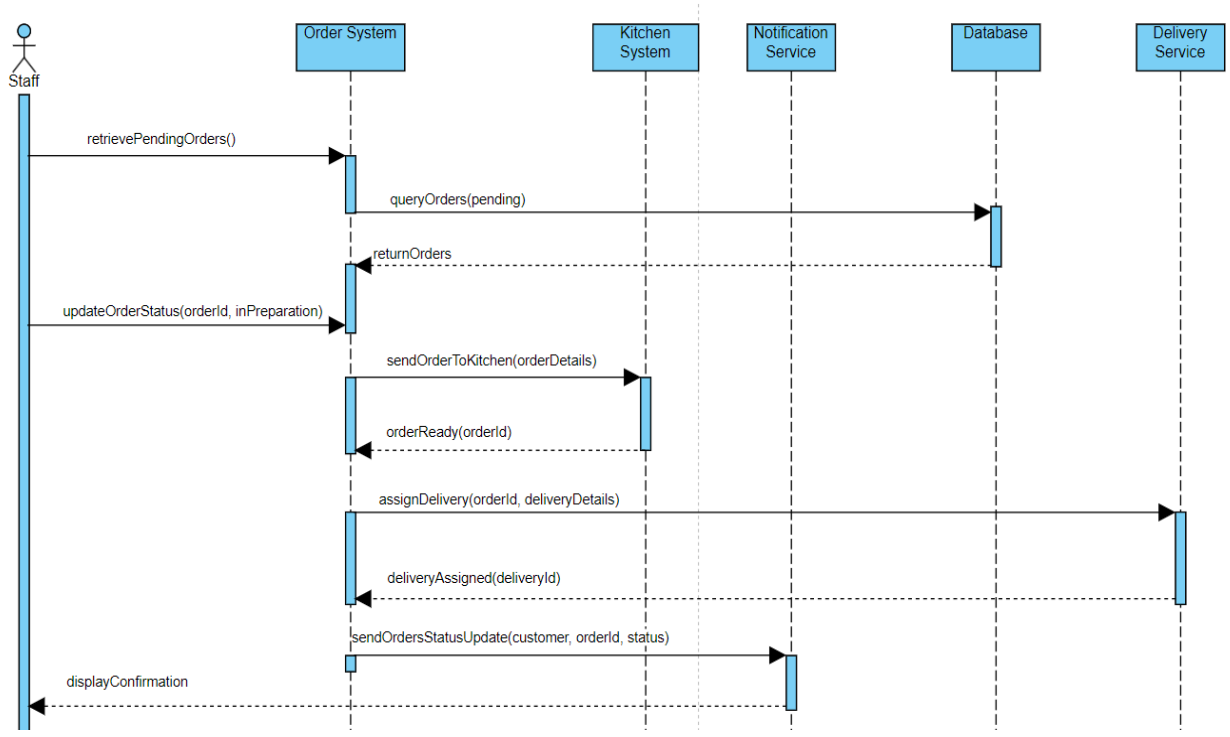

Рисунок 4.7 – Sequence diagram для варіанту використання «Управління замовленнями»

На крайній діаграмі(рис.4.8) продемонстровано діаграму послідовності для варіанту використання авторизації.

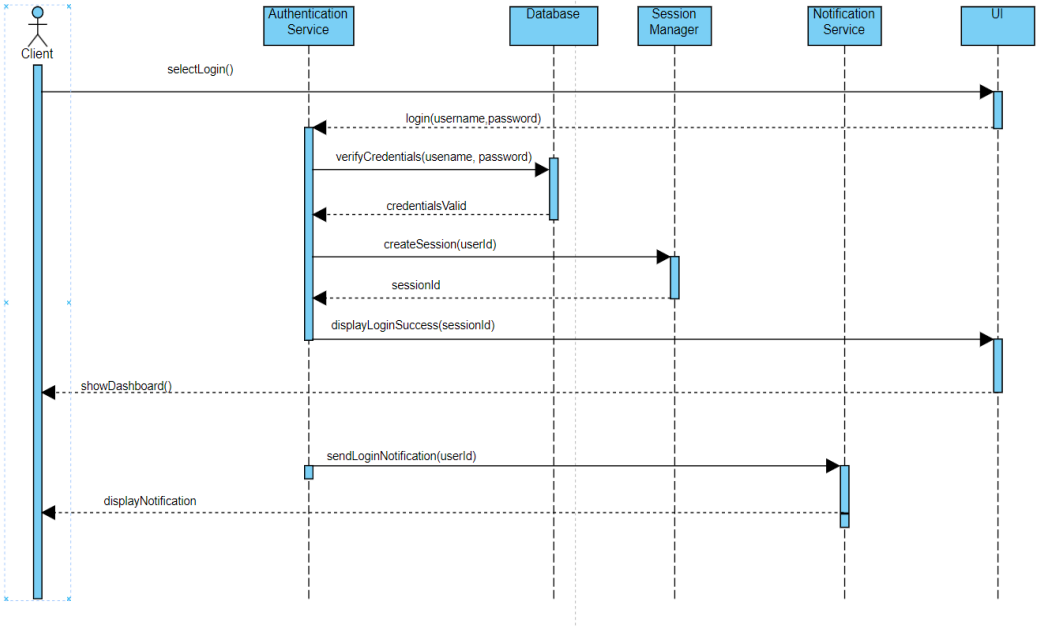

Рисунок 4.8 – Sequence diagram для варіанту використання «авторизація»

4.4 Інструкція користувача

Використання Visual Studio Code для роботи з Angular

Visual Studio Code (VS Code) є безкоштовним редактором коду, розробленим компанією Microsoft, який забезпечує широкі можливості для розробки на Angular завдяки підтримці численних розширень та інтеграцій. Перш за все, необхідно встановити Visual Studio Code, завантаживши його з офіційного сайту[18]. Після завантаження інсталяційного файлу виконайте інсталяцію, слідуючи інструкціям на екрані.

Для роботи з Angular у Visual Studio Code необхідно встановити відповідні розширення. Запустіть VS Code і відкрийте розділ розширень (Extensions) з лівої панелі або натисніть `Ctrl+Shift+X`. У пошуковому рядку введіть "Angular" і встановіть рекомендовані розширення, такі як "Angular Language Service" та "Debugger for Chrome".

Після налаштування середовища можна створити новий проєкт Angular. Відкрийте термінал у VS Code (`Ctrl+``) і виконайте команду `ng new my-angular-app`, щоб створити новий проєкт. Перейдіть до каталогу проєкту командою cd my-angular-appі запустіть проєкт за допомогою команди `ng serve`. Відкрийте браузер і перейдіть за адресою http://localhost:4200`, щоб побачити робочу версію вашого Angular застосунку.

Редагування коду в Angular здійснюється через відповідні файли компонентів. Наприклад, відкрийте файл app.component.ts для редагування головного компонента застосунку. Зробіть необхідні зміни та збережіть файл. Зміни автоматично відобразяться в браузері завдяки режиму гарячого перезавантаження (hot-reload).

Використання Visual Studio 2022 для роботи з ASP.NET Core

Visual Studio 2022 (VS 2022) є потужним інтегрованим середовищем розробки (IDE) від Microsoft, яке підтримує розробку на платформі ASP.NET Core. Щоб розпочати роботу, завантажте Visual Studio 2022 з офіційного сайту [19] та виконайте інсталяцію, обравши компоненти для розробки вебзастосунків на ASP.NET Core.

Для створення нового проєкту ASP.NET Core запустіть Visual Studio 2022 і оберіть "Create a new project". Виберіть шаблон "ASP.NET Core Web Application" і натисніть "Next". Введіть назву проєкту, оберіть місце для збереження та натисніть "Create". Виберіть тип проєкту (наприклад, "Web Application (Model-View-Controller)") та натисніть "Create".

Для запуску проєкту натисніть зелену кнопку "Start" у верхній частині вікна Visual Studio або натисніть `F5`. Вебзастосунок буде запущено у вбудованому вебсервері, і браузер відкриється автоматично для відображення застосунку

Використання Azure Data Studio для роботи з SQL Server

Azure Data Studio - це кросплатформенне інструментальне середовище для роботи з базами даних, в тому числі і SQL Server. Для початку завантажте Azure

Data Studio з офіційного сайту [20] та виконайте інсталяцію, слідуючи інструкціям на екрані.

Після запуску Azure Data Studio натисніть "New Connection" на початковій сторінці для підключення до SQL Server. Введіть необхідні дані для підключення до сервера (ім'я сервера, тип аутентифікації, логін та пароль) і натисніть "Connect".

Для створення нової бази даних натисніть правою кнопкою миші на вузлі "Databases" у панелі з'єднань і оберіть "New Database". Введіть назву нової бази даних і натисніть "OK".

Виконання SQL-запитів здійснюється через створення нових запитів. Натисніть "New Query" у верхньому меню, введіть ваш SQL-запит у новому вікні редактора і натисніть "Run" для виконання запиту. Результати запиту відобразяться у нижній частині вікна.

Управління даними здійснюється через панель з'єднань, де можна переглядати та керувати таблицями, представленнями та іншими об'єктами бази даних. Для додавання, зміни або видалення даних використовуйте відповідні SQL-запити.

Використання розробленого програмного забезпечення

Після налаштування всіх необхідних інструментів можна приступити до використання розробленого програмного забезпечення. Запустіть фронтенд-застосунок Angular за допомогою Visual Studio Code, виконавши команду `ng serve`, а бекенд застосунок ASP.NET Core у Visual Studio 2022, натиснувши кнопку "Start" або `F5`.

Перейдіть на головну сторінку вебзастосунку і використовуйте форму авторизації(рис 4.8) для входу в систему, ввівши свої облікові дані. Після авторизації можна приступити до управління замовленнями(рис 4.9), відкривши

відповідний розділ у навігаційному меню. Інтерфейс дозволяє переглядати, додавати, редагувати та видаляти замовлення.

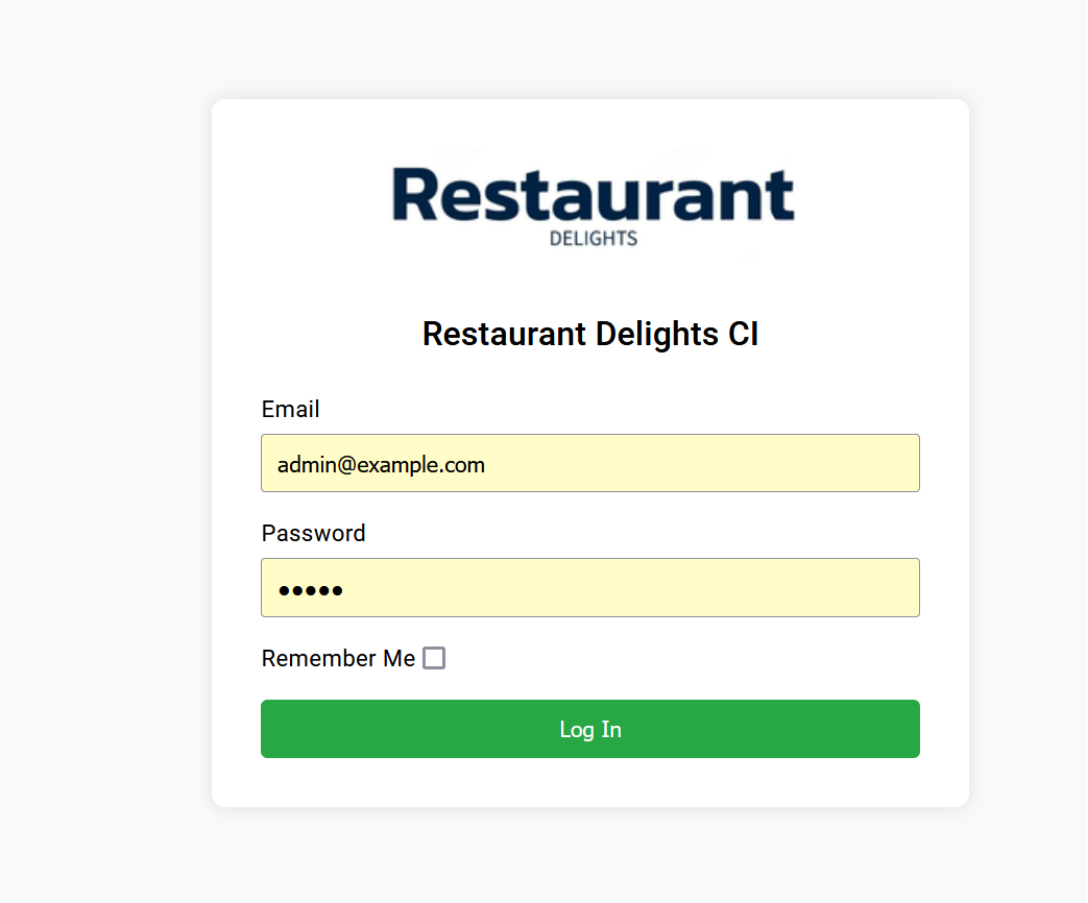

Рисунок 4.8 – Форма авторизації

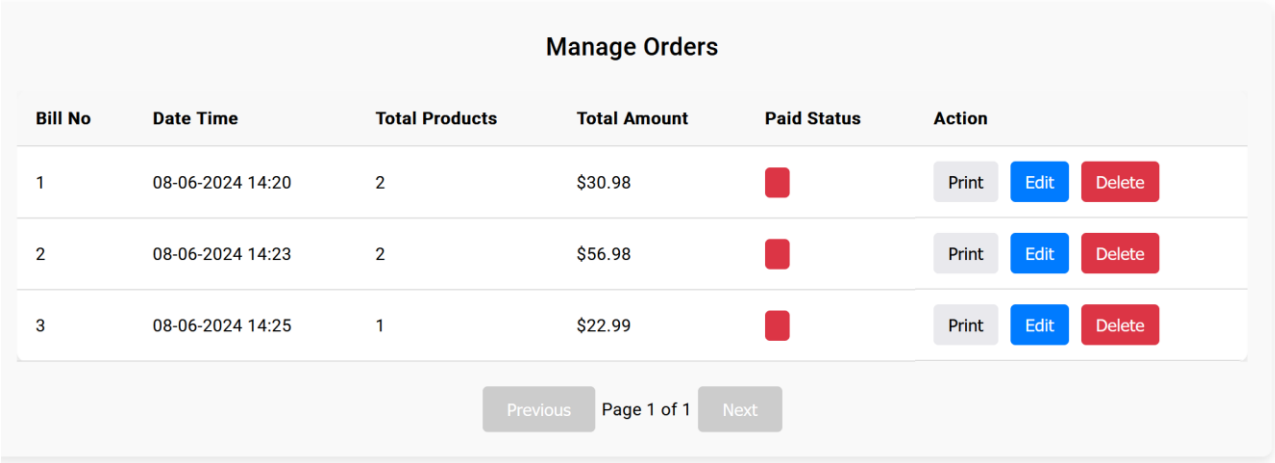

Рисунок 4.9 – Управління замовленнями

Для управління меню(рис 4.10) відкрийте відповідний розділ і використовуйте інтерфейс для додавання нових страв, редагування існуючих та

видалення непотрібних позицій. Резервування столиків(рис 4.11) здійснюється через розділ резервування, де можна створювати нові резервування, переглядати існуючі та керувати ними.

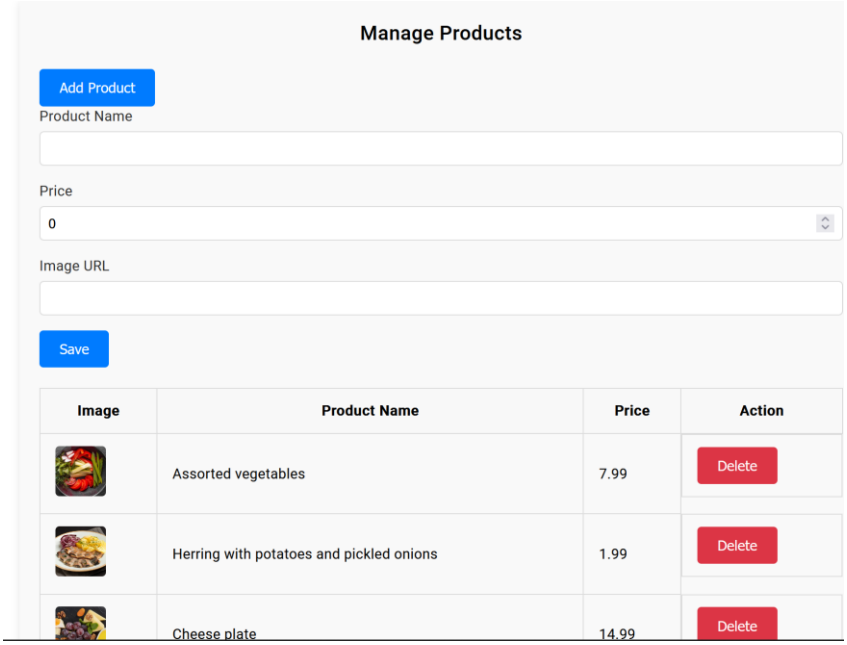

Рисунок 4.10 – Управління меню

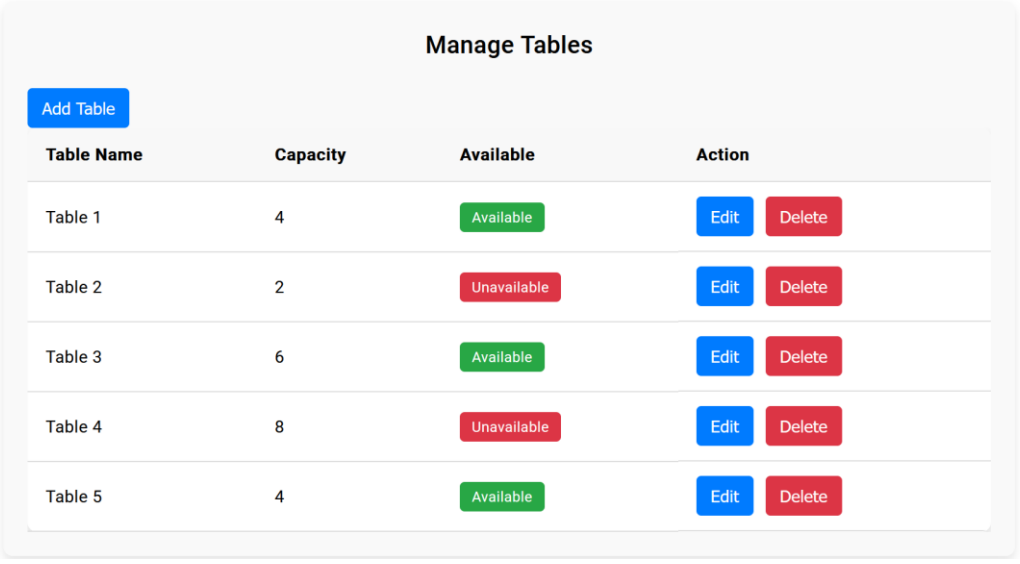

Рисунок 4.11 – Резервування столів

Для аналізу даних, збережених у SQL Server, використовуйте Azure Data Studio. Виконуйте складні SQL-запити для отримання звітів та аналітики, використовуючи відповідні інструменти середовища.

4.5 Тестування програмного забезпечення застосунку

Тестування програмного забезпечення є ключовим етапом у розробці застосунків, що забезпечує їх якість та надійність. Для тестування fronted частини обрано інструмент тестування cypress, для backend частини xunit.

Cypress

Cypress — це сучасний інструмент для енд-то-енд тестування, який забезпечує швидке та зручне написання тестів для вебзастосунків. Cypress особливо добре інтегрується з Angular завдяки можливості тестувати реальні браузери та автоматизувати тестування інтерактивності застосунків.

Встановити Cypress можна за допомогою команди `npm install cypress --save-dev`. Відкрити інтерфейс інструментарія можна за допомогою команди `npx cypress open`. Cypress інтерфейс продемонстровано на рисунку 4.12.

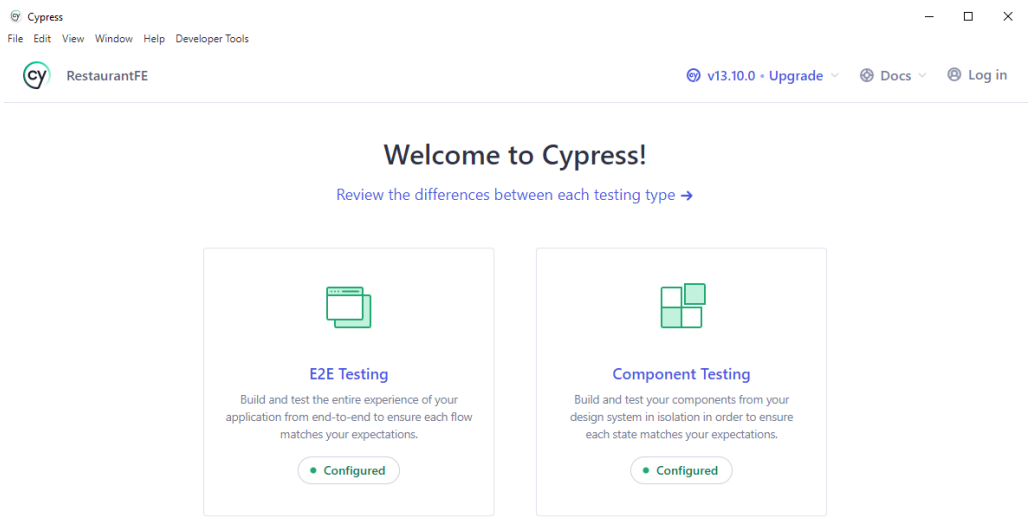

Рисунок 4.12 – Cypress інтерфейс

Cypress забезпечує швидкий зворотній зв’язок завдяки миттєвому виконанню тестів. Також можна отримати детальний звіт про помилки, за необхідністю є можливість тестування окремих компонентів. Тестування авторизації продемонстровано на рисунку 4.13.

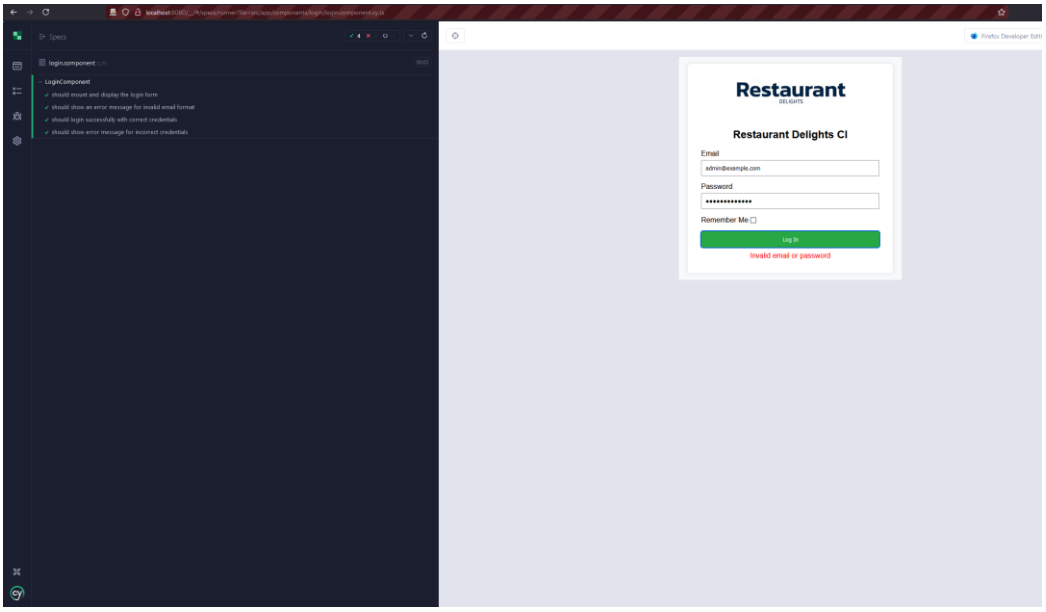

Рисунок 4.13 – Тестування авторизації за допомогою Cypress

xUnit

xUnit є розповсюдженим фреймворком для модульного тестування застосунків на платформі .NET. xUnit надає потужний інтерфейс для створення тестових наборів, перевірки умов та автоматизації тестування.

Для роботи з xUnit в ASP.NET потрібно додати xUnit через Nuget Packege Manager. Також потрібно налаштувати посилання на проєктні рішення для тестування окремих частин застосунку. xUnit ідеально підходить для тестування серверної логіки і допомагає забезпечити надійність функціональності бізнес-сервісів. Приклад використання xUnit тестів наведено на рисунку 4.14.

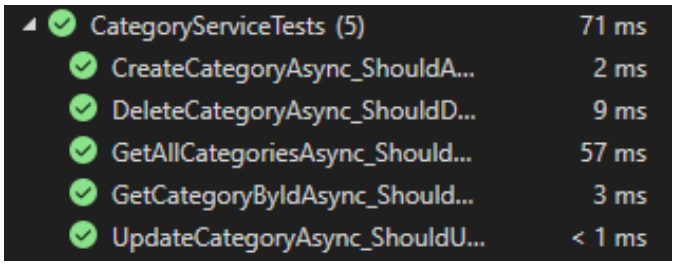

Рисунок 4.14 – Приклад тестування сервісу категорій

Інтеграція Cypress та xUnit тестування у процес розробки допомагає досягти високої якості продукту, забезпечуючи надійність і зручність використання кінцевих застосунків.

Висновки до розділу 4

Розглянуто програмну реалізацію автоматизованої системи управління рестораном.

Важливим етапом роботи стала розробка діаграм класів та діаграм сутність-зв’язок. Діаграма класів, як один з ключових інструментів UML, дозволила детально відобразити структуру програмного коду, представивши класи, їхні атрибути, методи та взаємозв'язки між ними. Це допомогло створити чітку та зрозумілу архітектуру програмного забезпечення.

Діаграма сутність-зв’язок (ERD) моделювала структуру бази даних, відобразивши сутності, їх атрибути та взаємозв'язки між ними. Це було критично важливим для розробки ефективної структури бази даних, яка забезпечує логічне та ефективне збереження й обробку даних.

Розробка діаграм послідовностей дозволила візуалізувати взаємодії між об'єктами у контексті конкретних процесів. Ці діаграми продемонстрували послідовність повідомлень, що обмінюються між об'єктами, ілюструючи, як вони співпрацюють для виконання певних завдань. Це сприяло більш глибокому розумінню процесів всередині системи та допомогло виявити потенційні проблеми на ранніх етапах розробки.

Інструкції користувача для роботи з інструментами розробки, такими як Visual Studio Code, Visual Studio 2022 та Azure Data Studio, забезпечили можливість ефективного використання розробленого програмного забезпечення. Користувачі отримали детальні рекомендації щодо налаштування робочого середовища, створення нових проєктів та управління базами даних.

Загалом, реалізація програмного забезпечення для автоматизованого робочого місця адміністратора ресторану включала ретельне проєктування архітектури системи, створення інструкцій для користувачів та забезпечення зручності у використанні кінцевого продукту. Це сприяло створенню ефективної системи управління рестораном, яка відповідає сучасним вимогам та потребам користувачів.

ВИСНОВКИ

В ході виконання КРБ було розроблено систему управління рестораном з інтегрованими функціями для адміністраторів, співробітників та клієнтів. Це дозволило підвищити ефективність операцій та поліпшити взаємодію з клієнтами. Для досягнення поставленої мети виконано наступні завдання:

1) проаналізовано специфіку діяльності ресторанів; проведено детальний аналіз існуючих систем управління ресторанним бізнесом. Вивчено функціональні можливості аналогів, що дозволило виділити їхні сильні та слабкі сторони і зрозуміти сучасні вимоги до таких систем;

2) проаналізовано вимоги користувачів до ресторанів; зібрано та проаналізовано вимоги користувачів, що стосуються функціональності та зручності користування системою; визначено специфікацію вимог, що включала ключові функції та сценарії використання для нової системи;

3) розроблено вебдизайн ресторану; створено мокапи інтерфейсу користувача, що дозволило візуалізувати кінцевий продукт; проведено оцінку з точки зору користувацького досвіду, що допомогло адаптувати дизайн до потреб ресторанного бізнесу;

4) спроєктовано базу даних; розроблено діаграми сутність-зв’язок (ERD), що детально відобразили структуру бази даних, що забезпечило чітку та ефективну організацію даних для підтримки функціонування системи;

5) розроблено інтерфейс та функціонал вебзастосунку; проаналізовано та вибрано технології для реалізації системи, включаючи Angular для фронтенду, ASP.NET Core для бекенду та SQL Server для бази даних, що дозволило забезпечити необхідний рівень продуктивності, гнучкості та підтримуваності;

6) розроблено інструкції для користувачів, що забезпечили ефективне використання розробленого програмного забезпечення, включаючи налаштування робочого середовища та управління базами даних. Практичне значення отриманих результатів полягає в тому, що розроблена система управління рестораном сприяє автоматизації операцій, покращенню

обслуговування клієнтів та підвищенню ефективності управління. Застосування сучасних технологій та інтеграція функціональних можливостей для різних типів користувачів забезпечують гнучкість і продуктивність системи, що відповідає сучасним вимогам ресторанного бізнесу. Розроблений застосунок може бути вдосконалений шляхом додавання нових функцій та інтеграції з іншими системами для покращення обслуговування та управління рестораном.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Daily Sport Menu. URL: https://dailysport.choiceqr.com/menu (date of access: 03.04.2024)

2. Vareniki Menu. URL: https://vareniki.mk.ua/ (date of access: 03.04.2024)

3. Dacha Menu. URL: https://savva-libkin.com/restaurant/odessa/dacha (date of access:: 03.04.2024)

4. How to Write a Software Requirements Specification. URL: https://stfalcon.com/uk/blog/post/How-to-Write-a-Software-Requirements-Specification (date of access: 04.03.2024).

5. UML Diagram Types Guide. URL: https://creately.com/blog/diagrams/uml-diagram-types-examples/ (date of access: 05.03.2024).

6. Making a website mockup in Figma. URL: https://blog.logrocket.com/ux-design/making-website-mockup-figma/ (date of access: 20.03.2024)

7. Freeman A. Pro Angular 9: Build Powerful and Dynamic Web Apps. Apress, 2020. 324 c.

8. React. The library for web and native user interfaces. URL: https://react.dev/ (date of access: 01.04.2024)

9. Vue.js. The progressive javascript framework. URL: https://vuejs.org/ (date of access: 01.04.2024)

10. Freeman A. Pro ASP.NET Core 3: Develop Cloud-Ready Web Applications Using MVC 3, Blazor, and Razor Pages. Apress, 2019. 287 c.

11. Node.js. URL: https://nodejs.org/en (date of access: 01.04.2024)

12. Express.js. Fast, unopinionated, minimalist web framework for node.js. URL: http://expressjs.com/ (date of access: 01.04.2024)

13. Django. The web framework for perfectionists with deadlines. URL: https://www.djangoproject.com/ (date of access: 01.04.2024)

14. SQL Server docs. URL: https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15 (date of access: 14.02.2024).

15. MySQL. The world’s most popular open source database URL: https://www.mysql.com/ (date of access: 01.04.2024)

16. PostgreSQL. The world’s most advanced open source relational database. URL: https://www.postgresql.org/ (date of access: 01.04.2024)

17. Onion-izing your multi-layer architecture. URL: https://www.incredible-web.com/blog/the-onion-architecture/ (date of access: 01.05.2024)

18. Visual Studio Code. Code Editing. Redefined. URL: https://code.visualstudio.com/ (date of access: 01.02.2024)

19. Visual Studio 22. URL: https://visualstudio.microsoft.com/ (date of access: 01.05.2024)

20. Azure Data Studio. URL: https://learn.microsoft.com/en-us/azure-data-studio/ (date of access: 01.05.2024)

21. Ben-Gan I. та ін. T-SQL Fundamentals. Microsoft Press, 2016. 432 с.

22. Amrit T. Entity Framework Core in Action. Manning Publications, 2018. 350 с.

23. Fluent Validation for .NET. URL: https://fluentvalidation.net/ (date of access: 04.05.2024).

24. Pilone D., Pitman N. UML 2.0 in a Nutshell. O'Reilly Media, Inc., 2005. 236 с.

25. Angular docs. URL: https://angular.io/docs (date of access: 04.05.2024).

26. ASP.NET Core docs. URL: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-5.0 (date of access: 04.04.2024).

27. Entity Framework Core docs. URL: https://docs.microsoft.com/en-us/ef/core/ (date of access: 24.03.2024).

28. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional, 2018. 208 с.

29. Fain Y., Moiseev A. Angular Development with TypeScript. Manning Publications, 2018. 279 с.

30. Brock A., Freeland S. IdentityServer4 in Action. Manning Publications, 2021. 365 с.

31. Enabling Cross-Origin Requests (CORS) in ASP.NET Core. URL: https://docs.microsoft.com/en-us/aspnet/core/security/cors?view=aspnetcore-5.0 (date of access:14.02.2024).

32. Fritchey G. SQL Server Query Performance Tuning. Apress, 2020. 504 с.

33. Lardinois R. Building Web APIs with ASP.NET Core 2. Apress, 2018. 376 с.

34. Nagel C. та ін. Professional C# 7 and .NET Core 2.0. Wiley, 2018. 1098 с.

35. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall, 2004. 736 с.

36. Cawood S., O'Brien M. Mastering Entity Framework Core 3.0: An advanced guide to handling data in modern web applications. Packt Publishing, 2020. 622 с.

37. Taylor J. Entity Framework Core Cookbook - Second Edition. Packt Publishing, 2020. 318