МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри_____ Є. О. Давиденко

підпис

«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Вебзастосунок для створення онлайн меню ресторану

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 409.22010912

Здобувач _____А. А. Каракулін

підпис

«__» _____ 2024 р.

Керівник канд. техн. наук, доцент _____ Г. В. Горбань

підпис

«__» _____ 2024 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва

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

Завідувач кафедри _____ Є. О. Давиденко

підпис

«_22__» ___грудня ___ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 409 факультету комп'ютерних наук

_____Каракулін Антон Андрійович_____

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

Вебзастосунок для створення онлайн меню ресторану_____

Затверджена наказом по ЧНУ від «22» ___грудня_____ 2023р. №__269___

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є веб застосунок для створення онлайн меню ресторану_____

4. Перелік питань, що підлягають розробці

- дослідження предметної області та аналіз існуючих аналогів;
- формування специфікації вимог до програмного забезпечення;
- визначення архітектури для проєктування програмного забезпечення;
- моделювання та проєктування програмного забезпечення;

– розробка програмного забезпечення;

– тестування роботи програмного забезпечення.

5. Перелік графічних матеріалів

Презентація_____

6. Завдання до спеціальної частини

_____

_____

_____

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
А.О. Алєксеєва	Кафедра екології	

Консультант Кафедра (організація) Частина роботи

Керівник роботи __канд. техн. наук, доцент Гліб Валентинович Горбань

(посада, прізвище, ім'я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

_____Каракулін Антон Андрійович_____

(прізвище, ім'я, по батькові студента)

_____

(підпис)

Дата видачі завдання «22» _____грудня_____ 2023 р.

КАЛЕНДАРНИЙ ПЛАН

виконання бакалаврської кваліфікаційної роботи

Тема: «Веб застосунок для створення онлайн меню ресторану»

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	22.12.23	22.12.23	Виконано
2.	Огляд літератури за темою роботи	04.01.24	04.01.24	Виконано
3.	Складання календарного плану КРБ	15.01.24	15.01.24	Виконано
4.	Аналіз предметної області	16.01.24	17.01.24	Виконано
5.	Розробка проєктних рішень	21.01.24	30.01.24	Виконано
6.	Моделювання та конструювання	21.01.24	30.01.24	Виконано
7.	Кодування ПЗ	23.01.24	01.02.24	Виконано
8.	Розробка керівництва користувача	22.03.24	22.03.24	Виконано
9.	Розробка спеціальної частини з охорони праці	22.03.24	22.03.24	Виконано
10.	Оформлення КРБ та презентації	22.03.24	22.03.24	Виконано
11.	Відгук керівника КРБ	29.03.24	29.03.24	Виконано
12.	Попередній захист	03.06.24	05.06.24	
13.	Рецензування	15.06.24	18.06.24	
14.	Захист кваліфікаційної роботи			

Розробив студент _____Каракулін Антон Андрійович_____

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

«15» __січня_ 2024 р.

Керівник роботи _канд. техн. наук, доцент Гліб Валентинович Горбань

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

«16» __січня_ 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Вебзастосунок для створення онлайн меню ресторану»

Студент 409 гр.: Каракулін А.А

Керівник: канд. техн. наук, доцент Горбань Г.В

Актуальністю роботи є розробка веб-застосунку для створення онлайн-меню, що відповідає сучасним тенденціям цифровізації, підвищує доступність інформації про страви, знижує витрати на друковані матеріали, покращує ефективність маркетингу, поліпшує взаємодію з клієнтами та забезпечує зручність користувачів.

Об'єктом роботи є цифрові технології в індустрії громадського харчування.

Предметом роботи є розробка та впровадження вебзастосунку для створення, редагування та управління онлайн меню в ресторанах.

Метою роботи є створення ефективного вебзастосунку для спрощення процесу управління меню ресторанів, що включає створення, редагування та представлення меню клієнтам в онлайн форматі. Відповідно до мети визначено такі завдання:

1) провести аналіз існуючих рішень у сфері створення онлайн меню для ресторанів;

2) розробити специфікацію вимог до програмного забезпечення вебзастосунку для створення онлайн меню ресторану;

3) створити функціональну та інформаційну модель вебзастосунку;

4) розробити архітектуру програмного забезпечення та вибрати технологічний стек для реалізації;

5) здійснити розробку, тестування та впровадження вебзастосунку для створення онлайн меню ресторану.

У роботі використовуються методи аналізу, проєктування та програмування для створення вебзастосунку. Застосовуються сучасні технології та фреймворки, такі як Node.js, Express.js, React, Next.js та MongoDB.

У першому розділі розкрито об'єкт і предмет дослідження, проаналізовано структурні та функціональні особливості вебсайту для створення меню ресторану, оглянуто стан інформаційних технологій у цій сфері, проведено аналіз існуючих методів і засобів для створення вебсайтів, обґрунтовано вибір підходів та сформовано специфікацію вимог до ПЗ.

У другому розділі описано етапи реалізації проєкту, побудовано алгоритм реалізації функціоналу, описано базу даних та побудовано її фізичну схему, забезпечуючи реалізацію в коді.

У третьому розділі розглянуто архітектуру програмного забезпечення, яка поділяється на фронтенд і бекенд. Бекенд реалізовано за допомогою Node.js та Express.js, а фронтенд - за допомогою React з Next.js. Вибрано основні компоненти, бібліотеки та патерни, які забезпечують функціональність, гнучкість і масштабованість системи. Створено UML-діаграми, що відображають ключові аспекти функціонування системи.

У четвертому розділі проведено тестування ключових функцій веб-застосунку, таких як реєстрація користувачів, додавання ресторанів, створення та відображення меню, генерація QR-кодів і бронювання столиків. Це тестування підтвердило коректність і надійність роботи ПЗ. Розроблено детальне керівництво користувача, яке охоплює всі основні процеси взаємодії з застосунком.

гнучкість і масштабованість системи. Використання покрокового підходу до побудови алгоритмів забезпечує чітке розуміння логіки виконання задач та їх

КРБ викладена на __ сторінки, вона містить 4 розділи, __ ілюстрацій, 20 джерел в переліку посилань.

Ключові слова: вебзастосунок, меню ресторану, QR-код, Node.js, React, MongoDB, Next.js.

ABSTRACT

of the Bachelor's Thesis

"Web application for creating an online restaurant menu"

Student 409 gr.: Karakulin A.A

Supervisor: candidate technical of Sciences, associate professor H. V. Horban

The relevance of the work is to develop a web application for creating an online menu that meets modern digitalization trends, increases the availability of information about dishes, reduces the cost of printed materials, improves marketing efficiency, improves customer interaction, and provides user convenience.

The object of research is digital technologies in the catering industry. The subject of the research is the development and implementation of a web application for creating, editing and managing online menus in restaurants.

The purpose of the work is to create an effective web application to simplify the process of restaurant menu management, which includes creating, editing and presenting menus to customers in an online format. In accordance with the goal, the following tasks have been defined:

1) to analyze existing solutions in the field of creating online menus for restaurants;

2) to develop a specification of requirements for the software of a web application for creating an online restaurant menu;

3) create a functional and information model of the web application;

4) develop a software architecture and select a technology stack for implementation;

5) to develop, test and implement a web application for creating an online restaurant menu.

The work uses methods of analysis, design and programming to create a web application. Modern technologies and frameworks such as Node.js, Express.js, React, Next.js and MongoDB are used.

The first section describes the object and subject of the study, analyzes the structural and functional features of a website for creating a restaurant menu, reviews the state of information technology in this area, analyzes existing methods and tools for creating websites, justifies the choice of approaches, and forms a specification of software requirements.

The second section describes the stages of the project implementation, builds an algorithm for implementing the functionality, describes the database and builds its physical scheme, providing implementation in code.

The third section describes the software architecture, which is divided into frontend and backend. The backend is implemented using Node.js and Express.js, and the frontend is implemented using React with Next.js. The main components, libraries, and patterns that provide functionality, flexibility, and scalability of the system were selected. UML diagrams were created that reflect the key aspects of the system.

In the fourth section, we tested the key functions of the web application, such as user registration, adding restaurants, creating and displaying menus, generating QR codes, and making table reservations. This testing confirmed the correctness and reliability of the software. A detailed user manual was developed, covering all the main processes of interaction with the application.

The qualification work is presented on __ pages, containing 4 sections, __ illustrations and 20 sources in the reference list.

Keywords: web application, restaurant menu, QR code, Node.js, React, MongoDB, Next.js.

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій та зростаючих потреб у цифрових рішеннях для оптимізації бізнес-процесів, розробка вебзастосунків для ресторанного бізнесу стає все більш актуальною. Індустрія громадського харчування потребує інноваційних підходів для підвищення ефективності управління замовленнями, покращення якості обслуговування клієнтів та оптимізації внутрішніх процесів. Одним із таких інноваційних рішень є створення вебзастосунку для онлайн меню ресторану. Цей вебзастосунок дозволяє ресторанам створювати, редагувати та управляти своїм меню в режимі онлайн, що значно полегшує процес внесення змін та забезпечує актуальність інформації для клієнтів.

Об’єктом роботи є цифрові технології в індустрії громадського харчування.

Предметом роботи є розробка та впровадження вебзастосунку для створення, редагування та управління онлайн меню в ресторанах.

Метою роботи є створення ефективного вебзастосунку для спрощення процесу управління меню ресторанів, що включає створення, редагування та представлення меню клієнтам в онлайн форматі. Відповідно до мети визначено такі завдання:

6) провести аналіз існуючих рішень у сфері створення онлайн меню для ресторанів;

7) розробити специфікацію вимог до програмного забезпечення вебзастосунку для створення онлайн меню ресторану;

8) створити функціональну та інформаційну модель вебзастосунку;

9) розробити архітектуру програмного забезпечення та вибрати технологічний стек для реалізації;

10) здійснити розробку, тестування та впровадження вебзастосунку для створення онлайн меню ресторану.

Необхідність нової розробки обумовлена сучасними тенденціями цифровізації бізнесу, а також зростаючими вимогами до зручності та оперативності управління інформацією у сфері громадського харчування. Аналіз сучасного стану проблеми показав, що існуючі рішення не завжди відповідають потребам ринку в плані гнучкості та зручності використання. Провідні фірми та вчені в цій галузі відзначають важливість впровадження вебзастосунків, які дозволяють автоматизувати процеси управління меню та забезпечують актуальність інформації для клієнтів.

Основні проєктні рішення включають вибір технологічного стеку, який забезпечує високу продуктивність, масштабованість та зручність у розробці і підтримці вебзастосунку. Для реалізації клієнтської частини обрано технології React та Next.js, що дозволяють створювати інтерактивні користувацькі інтерфейси та забезпечують SEO-оптимізацію. Для серверної частини використано Node.js та Express.js, що дозволяє створювати масштабовані серверні застосунки. Використання MongoDB як бази даних забезпечує гнучкість у зберіганні та обробці даних меню.

Результати даного дослідження та розроблений вебзастосунок можуть бути використані ресторанами, кафе, бістро та іншими закладами громадського харчування для ефективного управління своїм меню, підвищення якості обслуговування клієнтів та оптимізації внутрішніх процесів.

1. ОПИС ФУНКЦІЙ ТА ВЛАСТИВОСТЕЙ ЗАСТОСУНКУ. ОГЛЯД АНАЛОГІВ.

1.1 Призначення та межі проєкту

Об’єкт роботи включає використання цифрових технологій у сфері громадського харчування, які спрямовані на підвищення ефективності робочих процесів, покращення якості обслуговування клієнтів, і оптимізацію управління інформацією в ресторанах. Це включає, але не обмежується, системами для управління замовленнями, системами для управління персоналом, електронними меню, системами лояльності клієнтів, та іншими цифровими рішеннями, що допомагають ресторанам бути більш ефективними і забезпечувати кращий досвід для своїх клієнтів.

Предмет роботи концентрується на конкретній цифровій технології – вебзастосунку для створення та управління онлайн меню ресторанів. Основна мета цього застосунку – спростити процес створення, оновлення та представлення меню клієнтам в онлайн форматі.

Застосунок дозволяє ресторанам:

1) створювати та редагувати меню;
2) управляти доступом та ролями користувачів;
3) аналізувати поведінку клієнтів;
4) інтегрувати з іншими системами.

Цей застосунок не лише допомагає в управлінні меню, але й забезпечує засоби для збільшення задоволеності клієнтів через покращення сервісу та зручність використання цифрових інструментів. Завдяки цьому, ресторани можуть не тільки підвищити свою конкурентоспроможність, але й адаптуватись до сучасних трендів цифровізації у сфері харчування.

Погодження, що ухвалені в програмній документації: важливість обговорення та узгодження основних функцій і дизайну застосунку з

керівниками ресторанів, кухарями, маркетологами, і технічними спеціалістами, для забезпечення відповідності потребам ринку та користувачів.

Межі проєкту ПЗ: проєкт обмежений розробкою програмного забезпечення без втручання в апаратне забезпечення ресторанів. Застосунок має фокусуватись лише на функціях, пов’язаних із меню, без інтеграції системи замовлень або POS-систем.

1.2 Опис і аналіз проєкту та технологій

Структура вебсайту для створення меню ресторану зазвичай включає кілька ключових компонентів.

Головна сторінка – це вступна сторінка, яка вітає користувачів та надає загальний огляд послуг, які пропонує сайт. Часто містить візуальні елементи, які асоціюються з брендом ресторану [21].

Панель адміністрування дозволяє ресторанам керувати своїм меню, включаючи додавання, редагування та видалення страв. Також може включати функції управління користувачами і ролями для різних співробітників ресторану.

Сторінка меню показує всі доступні страви з детальним описом, цінами і, можливо, зображеннями. Може бути організована за категоріями, наприклад, закуски, основні страви, десерти тощо.

Інструменти фільтрації та пошуку допомагають клієнтам та співробітникам легко знаходити страви за ключовими словами, категоріями або іншими параметрами.

Інтеграція з іншими системами може включати інтеграцію з системами бронювання столиків, системами доставки їжі або з соціальними мережами для ефективного просування.

Мобільна адаптивність – важливо, щоб вебсайт був повністю адаптивний до мобільних пристроїв, оскільки багато користувачів відвідують сайти ресторанів за допомогою своїх смартфонів.

Функціональні особливості

1) Керування контентом меню

Адміністратори можуть легко додавати, редагувати або видаляти страви, включаючи опис, інгредієнти, ціни та фотографії. Важливою є можливість оновлення меню в реальному часі.

2) Мульти-користувацька функціональність

Різні рівні доступу для різних типів користувачів (наприклад, менеджери, кухарі, маркетологи), що дозволяє керувати різними аспектами сайту відповідно до ролей.

3) Інтерактивність

Включення інтерактивних елементів, таких як відгуки клієнтів, рейтинги страв, можливість залишати коментарі.

4) Звітність і аналітика

Інструменти для моніторингу відвідуваності сайту, популярності страв, і замовлень, що дозволяє збирати дані для маркетингових та стратегічних рішень.

5) Безпека

Захист даних користувачів та безпечне зберігання інформації про страви і ціни, захист від несанкціонованого доступу, застосування сучасних стандартів криптографії для забезпечення безпеки транзакцій.

Аналіз особливостей

Сучасний вебсайт для створення меню ресторану має бути інтуїтивно зрозумілим та легким у використанні як для адміністраторів, так і для кінцевих користувачів [21]. Основні виклики полягають у забезпеченні високої продуктивності під час великих навантажень, адаптивності до різних пристроїв та інтеграції з іншими системами і сервісами. Важливою є також гнучкість платформи, яка має підтримувати швидкі зміни у меню і бути безпечною для зберігання та обробки даних [21].

Опис і аналіз стану інформаційних технологій у сфері створення онлайн меню для ресторанів

Інформаційні технології в області ресторанного бізнесу активно розвиваються, адаптуючись до зростаючих вимог ринку і змін у споживацьких преференціях. Центральне місце у цих технологіях займають системи для створення та управління онлайн меню, які інтегруються з ширшими платформами для управління ресторанами.

Такі системи включають:

CMS (Content Management Systems) – системи управління контентом дозволяють ресторанам легко оновлювати меню, змінювати ціни і додавати описи страв без потреби залучення розробників.

POS (Point of Sale) системи – сучасні POS системи часто інтегровані з функціоналом меню, що дозволяє синхронізувати зміни в меню з замовленнями клієнтів у реальному часі.

Мобільні застосунки та вебсайти – ресторани використовують мобільні застосунки і адаптивні вебсайти для поліпшення доступу до своїх меню та послуг, забезпечуючи зручність для користувачів на будь-яких пристроях.

Інтеграція з системами бронювання і доставки - онлайн меню часто інтегровані з системами бронювання столиків та доставки їжі, що дозволяє клієнтам легко робити замовлення одразу під час перегляду меню.

Аналіз стану інформаційних технологій

ІТ-рішення сприяють зниженню витрат на друк та допомагають уникнути помилок, пов’язаних з людським фактором. Цифрові меню можуть бути швидко оновлені, що дає можливість ресторанам оперативно реагувати на зміни в попиті та наявності продуктів. Системи можуть інтегруватися з іншими інструментами управління для створення єдиної ефективної системи управління рестораном.

З огляду на важливість персональних даних клієнтів та конфіденційної інформації ресторану, безпека стає ключовим питанням [22]. Висока залежність від ІТ може створювати ризики, пов'язані з відмовами обладнання чи програмного забезпечення. Незважаючи на зниження вартості технологій,

початкові витрати на інтеграцію та підтримку IT систем можуть бути значними, особливо для малого та середнього бізнесу.

Ці інформаційні технології відіграють критичну роль у сучасному ресторанному бізнесі, допомагаючи не тільки покращити ефективність роботи, але й забезпечити кращий досвід для клієнтів.

Загальний опис застосунку

Сфера застосування: застосунок може бути використаний різними типами закладів харчування, включаючи ресторани, кафе, бістро, бари, що потребують онлайн платформи для управління своїми меню.

Характеристики користувачів: основні користувачі – менеджери ресторанів, які відповідають за оновлення та управління меню, кухарі, які можуть вносити зміни до страв, та адміністратори, відповідальні за налаштування доступу і безпеки.

Загальна структура і склад системи: система містить два основних компоненти: бекенд для адміністрування та управління базою даних меню і фронтенд для відображення меню клієнтам. Бекенд забезпечує інтерфейси для введення та редагування інформації, тоді як фронтенд дозволяє користувачам переглядати актуальне меню.

Загальні обмеження: застосунок має підтримувати велику кількість одночасних користувачів без зниження продуктивності. Він має бути оптимізований для роботи на різних пристроях, включаючи мобільні телефони, планшети, та настільні комп'ютери.

1.3 Функції системи

Функція створення нового меню - описує процес додавання нових страв до меню. Вхідні дані: назва страви, опис, ціна, фотографія. Вихідні дані: оновлений список меню, доступний для перегляду в онлайн-режимі.

Функція редагування меню – дозволяє користувачам редагувати існуючі страви в меню. Вхідні дані: обрана страва, зміни в описі, ціні, чи фотографії. Вихідні дані: актуалізований список страв.

Функція видалення страв з меню – уможливлює видалення страви з меню. Вхідні дані: обрана страва для видалення. Вихідні дані: оновлене меню без видаленої страви.

Функція перегляду меню – надає можливість користувачам переглядати актуальне меню. Вхідні дані: запит на перегляд. Вихідні дані: відображення актуального меню.

1.4 Властивості програмного забезпечення

Доступність програмного забезпечення визначає його здатність бути доступним для використання в будь-який час без значних перебоїв. Це означає, що вебзастосунок повинен мати високий рівень надійності та відмовостійкості, особливо під час пікових навантажень, коли багато користувачів водночас можуть вносити зміни до меню або переглядати його. Це досягається за допомогою розгортання на масштабованих хмарних платформах та використання технологій, що забезпечують неперервну роботу сервісу.

Супроводжуваність відноситься до легкості, з якою програмне забезпечення може бути модифіковане для виправлення помилок, покращення функціональності або адаптації до змінених умов. Застосунок має бути розроблений таким чином, щоб код був чистим, добре структурованим та документованим, що полегшує майбутнє оновлення та обслуговування. Використання модульної архітектури [1] та стандартів кодування забезпечує, що різні розробники можуть легко розуміти та вносити зміни в проєкт.

Переносимість програмного забезпечення забезпечує його здатність функціонувати в різних середовищах, наприклад, на різних операційних системах або апаратних платформах. Для вебзастосунку це означає забезпечення його коректної роботи на різних веббраузерах і пристроях, включаючи мобільні

телефони, планшети та настільні комп'ютери. Адаптивний дизайн і крос-платформне програмування є ключовими для досягнення цієї мети.

Продуктивність визначає, наскільки ефективно програмне забезпечення виконує необхідні функції при заданих ресурсах. Для вебзастосунку створення онлайн меню ресторану це означає здатність обробляти велику кількість запитів від користувачів без затримок або падінь [22]. Оптимізація бази даних, кешування даних та асинхронне завантаження контенту можуть значно покращити продуктивність системи.

Надійність вказує на здатність програмного забезпечення надійно та стабільно функціонувати, виконуючи очікувані операції під різними умовами. Це включає обробку помилок, відсутність збоїв і втрати даних під час обробки запитів користувачів. Регулярне тестування, багаторівневе резервне копіювання даних та розробка заходів для автоматичного відновлення системи після збоїв є необхідними для забезпечення високого рівня надійності.

Безпека програмного забезпечення забезпечує захист даних користувачів від несанкціонованого доступу та інших загроз. Важливо впровадити сучасні методи захисту даних, такі як шифрування, безпечне зберігання паролів, аутентифікація та авторизація користувачів. Також важливо регулярно проводити аудити безпеки та оновлювати захисні механізми відповідно до новітніх загроз і вразливостей.

1.5 Огляд і аналіз існуючих аналогів

UpMenu [8]

Виробник: UpMenu

Архітектура: Хмарний сервіс (SaaS)

Мова реалізації: Не вказано

Основні функції та характеристики:

- інтегрована система онлайн замовлень і платежів;
- налаштування бренду для QR-кодів і цифрових меню;

– вбудована CRM для управління даними клієнтів;

– маркетингові інструменти для акцій і збору відгуків;

– сумісність з багатьма пристроями без необхідності встановлення додатків.

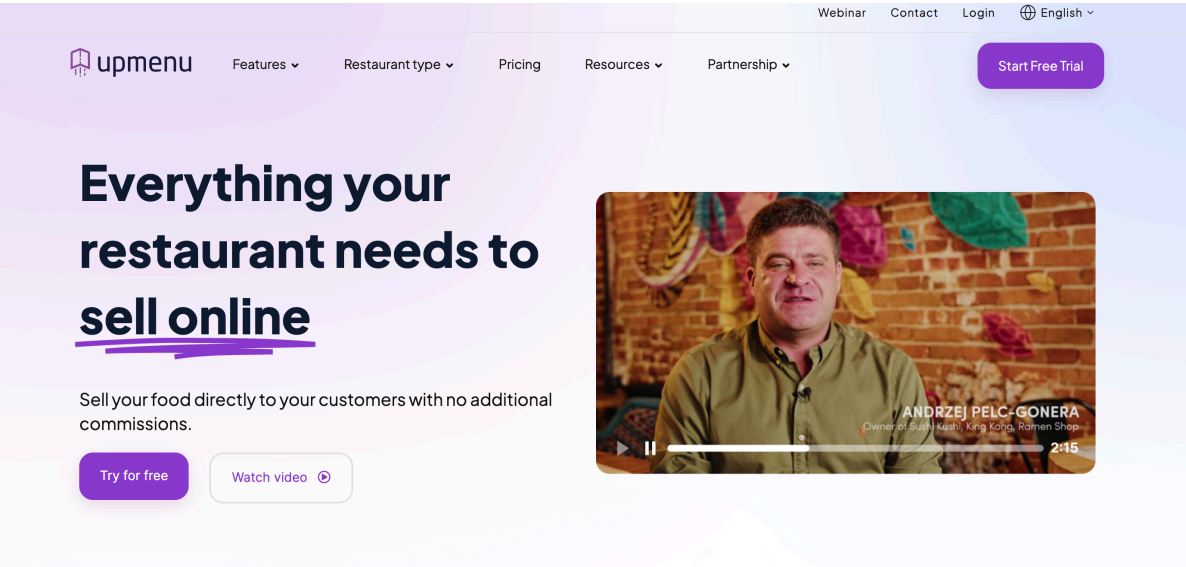

Рисунок 1.1 – Застосунок Upmenu

Переваги:

– підвищує зручність для клієнтів завдяки мобільному доступу;

– знижує витрати на друк і вплив на навколишнє середовище;

– підтримує маркетинг і залучення клієнтів.

Недоліки:

– залежність від інтернету для доступності меню;

– складний для користувачів, які не мають технічних навичок;

Greet [2]

Виробник: Greet

Архітектура: Хмарна платформа

Мова реалізації: Не вказано

Основні функції та характеристики:

– швидкі опції імпорту з фото, вебу або Excel;

– налаштування шаблонів QR-коду;

– функції для додавання рекомендацій та спільного продажу;

– дозволяє миттєво оновлювати меню;

– аналітичні інструменти для оцінки взаємодії з клієнтами.

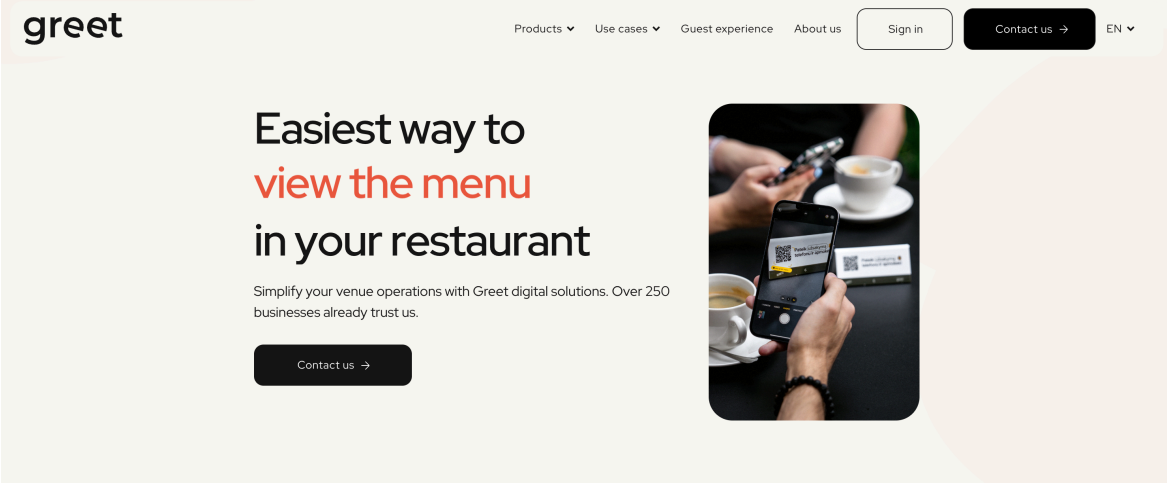

Рисунок 1.2 – Застосунок 2

Переваги:

– забезпечує швидке оновлення меню, знижуючи витрати на друк;

– мобільно-дружні дизайни покращують досвід споживачів;

– підтримує детальну інформацію про продукти, включаючи відео та алергени.

Недоліки:

– вимагає постійного управління для оновлень;

– початкове налаштування може потребувати введення даних і налаштувань.

1.6 Специфікація умов до ПЗ

Вимоги до інформаційного забезпечення:

– джерела і зміст вхідної інформації (даних) включають дані про страви, такі як назви, описи, ціни, категорії, і фотографії;

– нормативно-довідкова інформація - класифікатори та довідники, які можуть бути використані для категоризації страв за типами (наприклад, закуски, основні страви, напої);

– вимоги до способів організації, збереження та ведення інформації: інформація повинна зберігатися в базі даних із забезпеченням її цілісності та захисту від несанкціонованого доступу.

Вимоги до технічного забезпечення:

система повинна працювати на сервері, який забезпечує високий рівень доступності та відмовостійкості. Вебсервери та бази даних повинні бути налаштовані на обробку великої кількості запитів та даних.

Вимоги до інформаційного забезпечення:

– архітектура програмної системи: модульна архітектура з чітким розділенням функцій бекенду та фронтенду.

– системне програмне забезпечення: серверне програмне забезпечення повинно включати вебсервер та базу даних.

– мережне програмне забезпечення: вимоги до протоколів зв'язку та безпеки даних.

– програмне забезпечення ведення інформаційної бази: системи управління базами даних, які підтримують транзакції, резервне копіювання та відновлення.

Вимоги до інформаційного забезпечення:

– інтерфейс користувача: забезпечення інтуїтивно зрозумілого вебінтерфейсу для користувачів і адміністраторів.

– апаратний інтерфейс: налаштування для роботи зі стандартними вебпереглядачами на різноманітних пристроях.

– програмний інтерфейс: API для інтеграції з іншими системами, якщо потрібно.

– комунікаційний протокол: HTTPS для захисту даних користувачів.

Висновки до розділу 1

В першому розділі кваліфікаційної роботи бакалавра було розкрито об'єкт і предмет дослідження для вебзастосунку створення онлайн меню ресторану. Описано і проаналізовано структурні і функціональні особливості вебсайту для створення меню ресторану.

Було описано і проаналізовано стан інформаційних технологій у сфері створення онлайн меню для ресторанів. Проведено огляд і аналіз існуючих методів і засобів рішення завдань створення вебсайту для створення меню ресторанів. Обґрунтовано та обрано підхід до створення вебсайту для QR-меню ресторану. Здійснено формування специфікації вимог до ПЗ.

2. РЕАЛІЗАЦІЯ ПРОЄКТУ ТА СТВОРЕННЯ UML-ДІАГРАМ

2.1 Етапи реалізації проєкту

Реалізація проєкту вебзастосунку для QR-меню ресторану може бути розділена на декілька ключових етапів, кожен з яких має свої цілі та завдання:

1) Планування:

– визначення вимог: збір вимог від зацікавлених сторін, включаючи функціональні та нефункціональні вимоги;

– аналіз ринку та конкурентів: дослідження ринкових тенденцій і аналіз існуючих рішень;

– створення технічного завдання: формулювання детальних специфікацій проєкту та плану робіт.

2. Проєктування:

– архітектура системи: визначення архітектурного рішення для вебсайту, включаючи вибір технологій та платформ;

– проєктування бази даних: розробка схеми бази даних, яка підтримує всі необхідні дані та забезпечує їх швидкий доступ;

– проєктування інтерфейсів: створення дизайну користувацьких інтерфейсів, що відповідають вимогам зручності та доступності.

3. Розробка:

– налаштування середовища: розробницького та тестового;

– кодування: реалізація всіх компонентів системи, включаючи фронтенд та бекенд;

– інтеграція систем: підключення всіх зовнішніх сервісів та API.

4. Тестування:

– юніт-тестування: тестування окремих модулів програми на коректність роботи;

– інтеграційне тестування: перевірка взаємодії різних частин системи;

– приймальне тестування: забезпечення відповідності готового продукту вимогам замовника;

– тестування на вразливості: перевірка системи на наявність безпекових вразливостей.

5. Розгортання:

– налаштування продуктивного середовища: підготовка та налаштування серверів та інфраструктури;

– запуск системи: первинне розгортання вебсайту;

– моніторинг: встановлення систем моніторингу для відстеження роботи вебсайту.

2.2 Алгоритм реалізації системи

Для реалізації програмних моделей, які потребують використання логічних операторів (розгалуження, вибір варіантів продовження тощо), розробляються блок-схеми алгоритмів.

Алгоритм створення замовлення (покроковий спосіб)

1) Що зробити?: Створити замовлення користувача.
2) Як зробити?:
    1) перевірити, чи користувач аутентифікований;
    2) отримати дані замовлення з запиту;
    3) розрахувати загальну вартість замовлення;
    4) зберегти замовлення в базі даних;
    5) повернути відповідь користувачу з інформацією про замовлення.

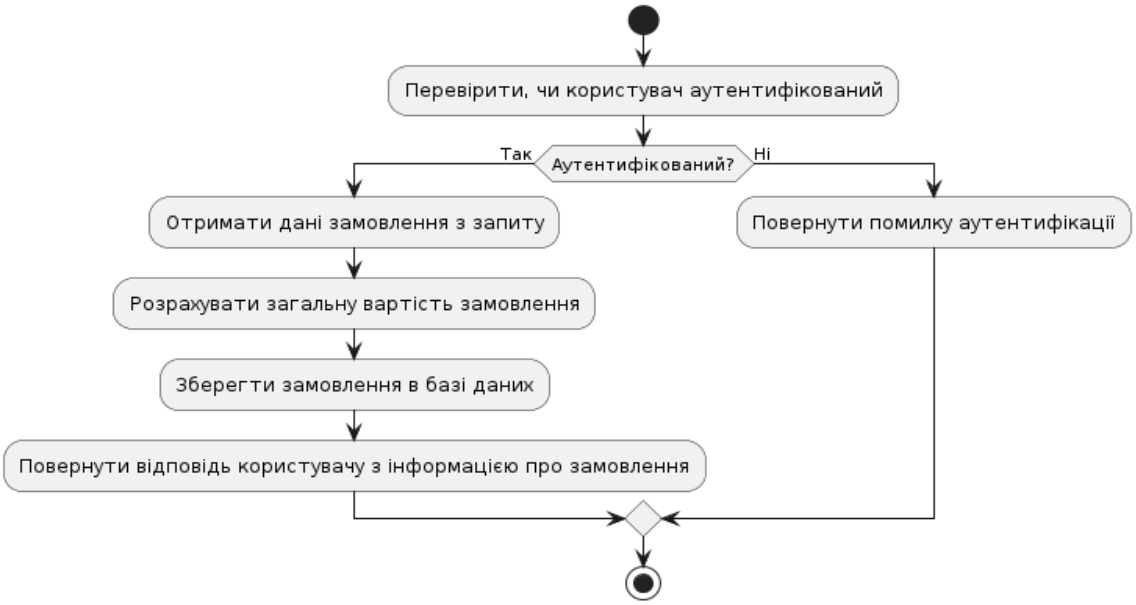

Рисунок 2.1 – Блок-схема алгоритму замовлення

Алгоритм обробки бронювання (покроковий спосіб)

1) Що зробити?: Обробити запит на бронювання столика.
2) Як зробити?:
    1) перевірити, чи користувач аутентифікований;
    2) отримати дані бронювання з запиту;
    3) перевірити наявність вільних столиків на вказаний час;
    4) зберегти бронювання в базі даних;
    5) повернути відповідь користувачу з інформацією про бронювання.

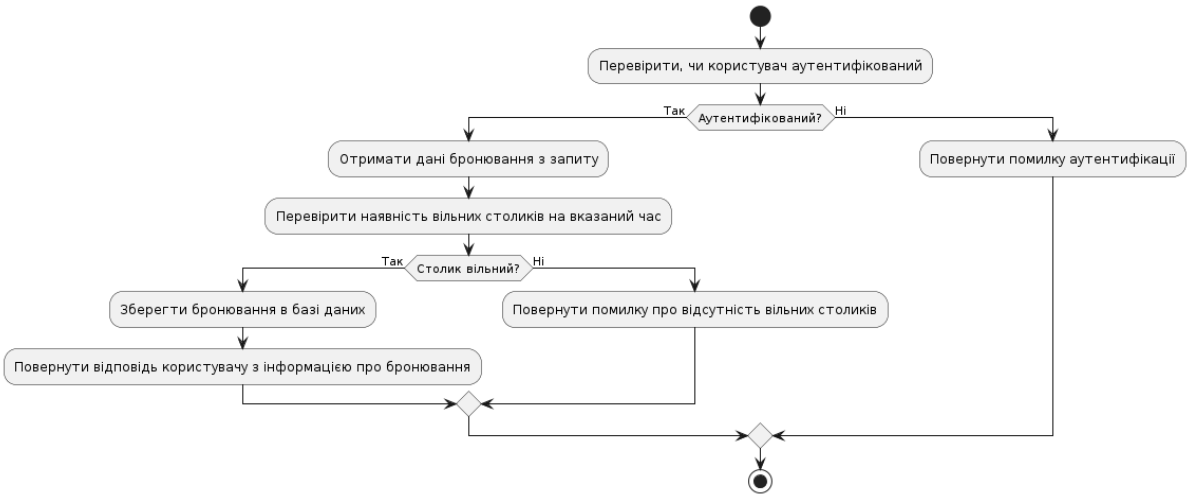

Рисунок 2.2 – Блок-схема алгоритму бронювання

Алгоритм створення QR-коду для меню ресторану передбачає декілька кроків. На основі аналізу коду з архіву, алгоритм виглядає наступним чином:

1) Що зробити?: Створити QR-код, який містить URL для доступу до меню ресторану.

2) Як зробити?:

1) перевірити аутентифікацію адміністратора;

2) отримати ідентифікатор ресторану;

3) згенерувати URL для доступу до меню ресторану;

4) створити QR-код на основі згенерованого URL;

5) зберегти QR-код та надати доступ адміністратору.

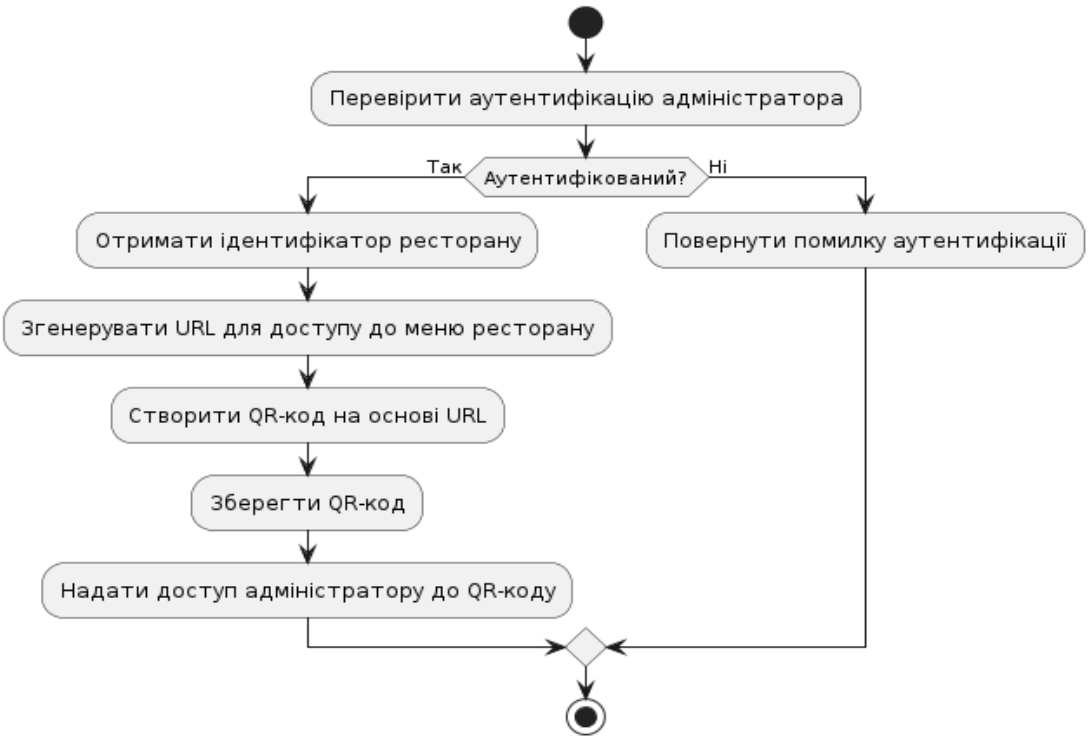

Рисунок 2.3 – Блок-схема алгоритму створення меню

Алгоритм створення QR-коду меню включає перевірку аутентифікації адміністратора, отримання ідентифікатора ресторану, генерацію URL для доступу до меню, створення QR-коду на основі цього URL та повернення згенерованого QR-коду. Цей процес забезпечує простий спосіб для

адміністраторів створити і використовувати QR-коди для доступу до меню ресторанів.

2.3 Розробка UML-діаграм

2.3.1 Діаграма використання

Діаграма використання (Use Case Diagram) – це тип діаграми в області програмної інженерії та системного аналізу, яка використовується для опису функціональних вимог до системи [20]. Ця діаграма показує взаємодію між користувачами (або "акторами") та системою, які разом визначають, як система має вести себе в різних сценаріях.

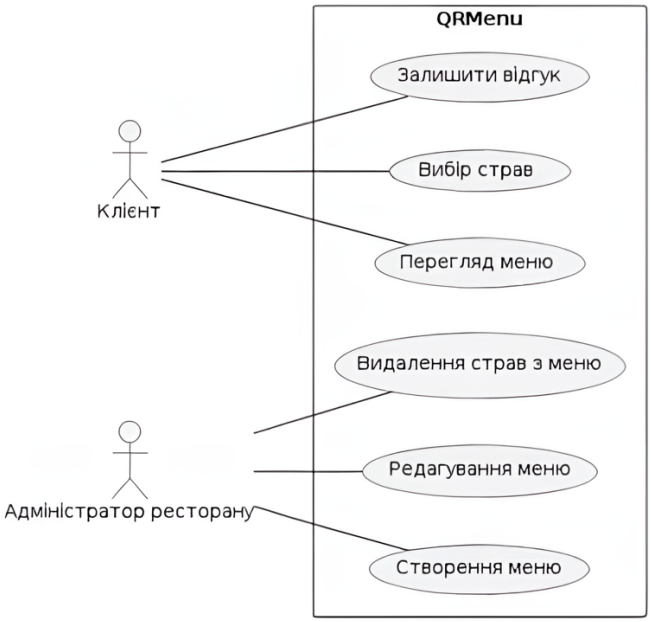

Рисунок 2.3 – Діаграма використання

Опис коротких, поверхневих та альтернативних сценаріїв використання

Сценарій "Перегляд меню"

Короткий сценарій: клієнт сканує QR-код та отримує доступ до актуального меню ресторану.

Поверхневий сценарій: клієнт відкриває меню, переглядає різні категорії страв та знайомиться з цінами і описами.

Альтернативний сценарій: якщо меню не оновлене або не вантажиться, клієнт отримує повідомлення про технічні роботи або просить персонал ресторану надати паперову версію меню.

Сценарій "Створения меню"

Короткий сценарій: адміністратор входить у систему управління меню і створює нове меню.

Поверхневий сценарій: адміністратор вибирає шаблон меню, додає страви, їх описи, ціни та зображення.

Альтернативний сценарій: якщо інформація про страви не повна, система відображає попередження та запитує підтвердження перед збереженням.

Опис функцій та ролей всіх дійових осіб системи

Клієнт:

– перегляд меню: клієнт може переглядати актуальне меню ресторану;

– вибір страв: клієнт може вибирати страви для замовлення;

– залишити відгук: клієнт може залишити відгук про страви та сервіс ресторану.

Адміністратор ресторану:

- створення меню: адміністратор має можливість створювати та публікувати нові меню.
- редагування меню: адміністратор може редагувати існуючі меню, вносячи зміни в страви, їх описи, ціни.
- видалення страв з меню: адміністратор може видаляти страви з меню.

2.3.2 Діаграма розгортання

Діаграма розгортання в UML (Unified Modeling Language) використовується для представлення архітектури розгортання програмного забезпечення [20]. Вона показує конфігурацію апаратного забезпечення (вузлів), на якому працює програмне забезпечення, і фізичні зв’язки між компонентами

апаратури. Основні компоненти цієї діаграми включають вузли, артефакти (файли програм, бібліотеки тощо) та їх взаємозв'язки.

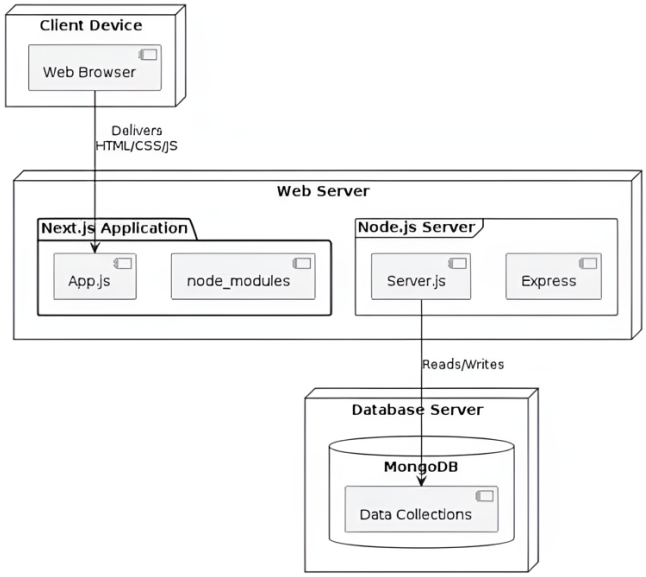

Рисунок 2.4 – Діаграма розгортання

Опис компонентів діаграми розгортання

1) Вузли (Nodes): Кожен вузол представляє апаратний ресурс, який виконує частину системи.

- Web Server: сервер, на якому розгорнуті Next.js та Node.js.
- Database Server: сервер бази даних, де зберігається MongoDB.
- Client Device: клієнтський пристрій, такий як смартфон або комп'ютер, на якому запускається веббраузер.

2) Компоненти (Components) – це програмні елементи, які розгорнуті на вузлах.

- Next.js Application включає основний файл застосунку (App.js) та залежності (node_modules).
- Node.js Server - серверний скрипт (Server.js) із використанням Express для обробки запитів.

3) Артефакти (Artifacts) – фізичні файли програми, такі як скрипти, виконувані файли, бібліотеки, які розгортаються на серверах.

4) Зв’язки – лінії, які показують взаємодію між компонентами і вузлами, наприклад, як веббраузер взаємодіє з вебзастосунком або як серверне програмне забезпечення зчитує та записує дані у базу даних.

2.3.3 Діаграма послідовності

Діаграма послідовності в UML (Unified Modeling Language) використовується для відображення порядку взаємодії між об’єктами в межах певного сценарію [20]. Ця діаграма показує, як об’єкти спілкуються між собою згідно з часовою послідовністю їхньої взаємодії.

Діаграма послідовності включає наступні основні елементи:

– адміністратор та клієнт як основні актори, які взаємодіють з системою.

– адмін панель та клієнтський інтерфейс як інтерфейси через які актори взаємодіють з системою.

– серверна логіка, яка обробляє запити від інтерфейсів і взаємодіє з базою даних.

– база даних, яка зберігає всю необхідну інформацію про меню і користувачів.

На рис.2.4 відображено діаграму послідовності створення нового меню.

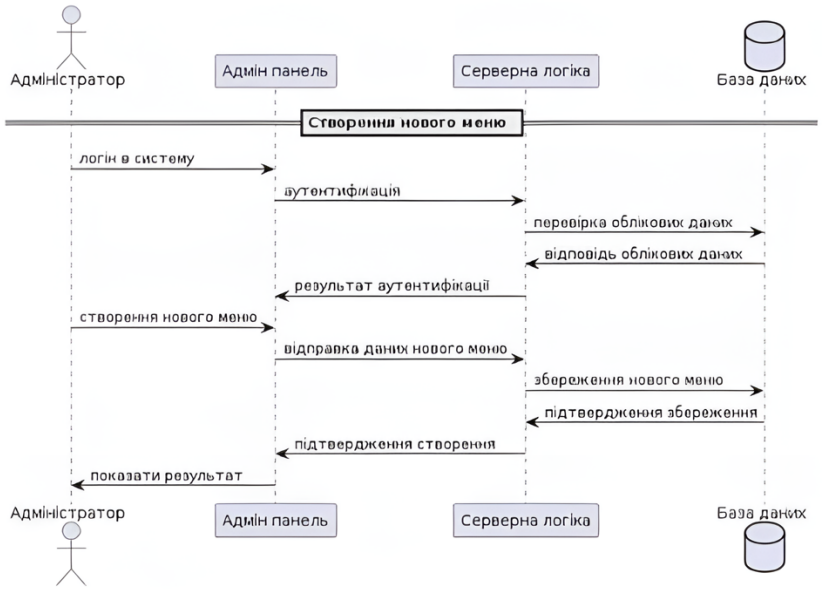

Рисунок 2.4 – Діаграма послідовності створення нового меню

Створення нового меню включає наступні етапи:

1) Автентифікація адміністратора:
   - адміністратор входить на вебсайт і вводить свої облікові дані;
   - адмін-панель надсилає облікові дані на серверну логіку;
   - серверна логіка перевіряє облікові дані в базі даних;
   - база даних повертає результат перевірки (успішний чи ні).

2) Введення даних нового меню:
   - після успішної автентифікації адміністратор отримує доступ до функції створення нового меню;
   - адміністратор вводить всі необхідні дані для нового меню, включаючи назви страв, описи, ціни і фотографії.

3) Збереження нового меню:
   - адмін-панель відправляє введені дані на серверну логіку;
   - серверна логіка обробляє дані і зберігає нове меню в базі даних;
   - база даних повідомляє серверну логіку про успішне збереження;
   - серверна логіка повертає підтвердження створення меню до адмін-панелі, яка інформує адміністратора про успіх операції.

На рис.2.5 відображено діаграму послідовності редагування меню.

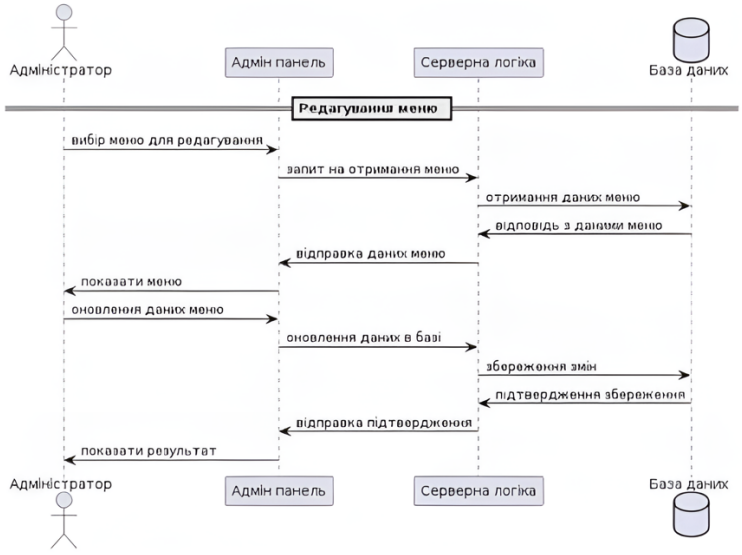

Рисунок 2.5 – Діаграма послідовності редагування меню

Редагування меню включає наступні етапи:

1) Вибір меню для редагування:

- адміністратор вибирає існуюче меню яке потрібно редагувати, у адмін-панелі;
- адмін-панель запитує відповідні дані меню з серверної логіки;
- серверна логіка отримує дані з бази даних;
- дані меню відправляються назад до адмін-панелі.

2) Внесення змін у меню:

- адміністратор робить необхідні зміни в меню через адмін-панель;
- змінені дані відправляються на серверну логіку.

3) Оновлення даних у базі даних:

- серверна логіка оновлює інформацію в базі даних;
- база даних підтверджує оновлення;
- серверна логіка відправляє підтвердження оновлення адмін-панелі.

На рис.2.6 відображено діаграму послідовності перегляду меню.

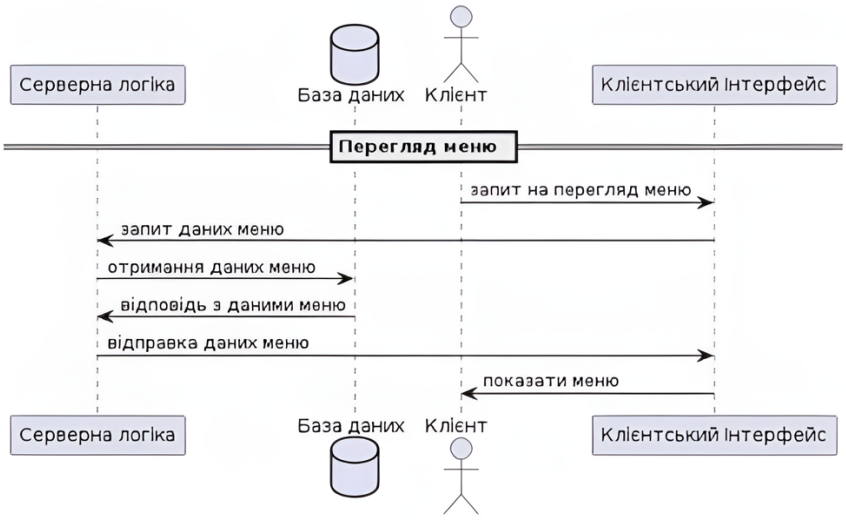

Рисунок 2.6 – Діаграма послідовності перегляду меню

Перегляд меню включає наступні етапи:

1) Запит на перегляд меню:

– клієнт відкриває вебсайт і вводить запит на перегляд меню через клієнтський інтерфейс;
– клієнтський інтерфейс відправляє запит на серверну логіку.

2) Отримання даних меню:
– серверна логіка запитує потрібні дані з бази даних.
– база даних надсилає дані меню назад на серверну логіку.

3) Відображення меню клієнта:
– серверна логіка відправляє дані меню до клієнтського інтерфейсу.
– клієнтський інтерфейс відображає меню клієнту.

Висновки до розділу 2

В другому розділі було описано етапи реалізації проєкту. Побудовано алгоритм реалізації функціоналу. Описано базу даних та побудовано її фізичну схему. База даних проєкту організована у вигляді колекцій, які зберігають інформацію про користувачів, адміністраторів, ресторани, категорії меню, пункти меню, замовлення та бронювання. Використання MongoDB забезпечує гнучкість і масштабованість, необхідні для обробки великих обсягів даних і підтримки складних взаємозв'язків між різними компонентами системи. Фізична схема бази даних та алгоритми програмної реалізації моделей визначають основні структури даних та кроки, необхідні для обробки користувацьких запитів у системі. Використання покрокового підходу для побудови алгоритмів забезпечує чітке розуміння логіки виконання задач та їх подальшу реалізацію в коді.

3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ОПИС ТЕХНОЛОГІЙ

3.1 Розробка архітектури ПЗ

Архітектура програмного забезпечення (ПЗ) визначає основні компоненти системи, їх взаємозв’язки та принципи взаємодії [1]. У випадку даного проєкту, архітектура поділяється на два основні рівні: бекенд (серверна частина) та фронтенд (клієнтська частина). Нижче наведено детальний опис архітектури кожного рівня.

Архітектура бекенду базується на використанні Node.js 3 Express.js, що дозволяє створювати швидкі та масштабовані серверні застосунки [19].

Основні компоненти бекенду включають:

1) Контролери (Controllers)

Відповідають за обробку HTT-запитів та повернення відповідей. Наприклад:|admins.controller.ts, auth.controller.ts, bookings.controller.ts.

2) Моделі (Models)

Визначають структуру даних та взаємодію з базою даних. Наприклад: admins.model.ts, bookings.model.ts.

3) Посередники (Middleware)

Проміжні обробники, які виконують певні дії під час обробки запиту. Наприклад: auth-check.ts, errorHandler.ts.

4) База даних (Database)

Використовується MongoDB для зберігання даних. Моделі визначають структуру документів у базі даних.

5) Конфігурація (Configuration)

Файли для налаштування середовища та параметрів застосунку. Наприклад: config.ts, .env.template.

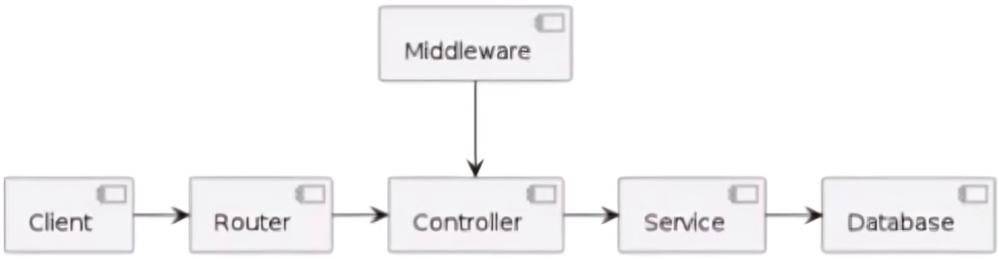

Рисунок 3.1 – Діаграма архітектури бекенду

Архітектура фронтенду базується на використанні React з Next.js, що забезпечує швидкий рендеринг сторінок та SEO-оптимізацію [10]. Основні компоненти фронтенду включають:

5) Компоненти (Components)

Відповідають за відображення інтерфейсу користувача. Наприклад: qr-code-modal.tsx, restaurant-card.tsx.

6) Сторінки (Pages)

Визначають маршрути та відповідні компоненти для відображення. Наприклад: index.tsx, _app.tsx.

7) Стан (State)

Управління станом застосунку за допомогою Redux. Наприклад: addToCartModalSlice.ts, adminSlice.ts.

8) Стилі (Styles)

Використання Tailwind CSS для створення стилів. Файл tailwind.config.js.

9) Бібліотеки та утиліти (Libraries and Utilities)

Додаткові інструменти та бібліотеки для спрощення розробки. Наприклад: hooks.ts, store.ts

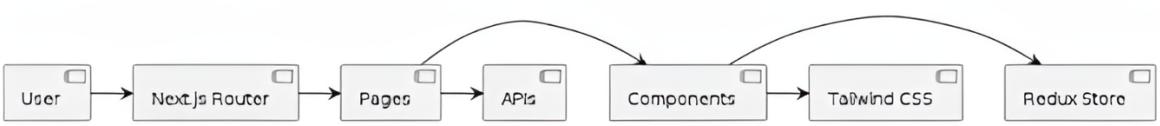

Рисунок 3.2 – Діаграма архітектури фронтенду

Взаємодія між бекендом та фронтендом здійснюється за допомогою HTTP-запитів (REST API). Фронтенд надсилає запити до серверу, який обробляє їх за допомогою контролерів, отримує дані з бази даних та повертає відповіді фронтенду.

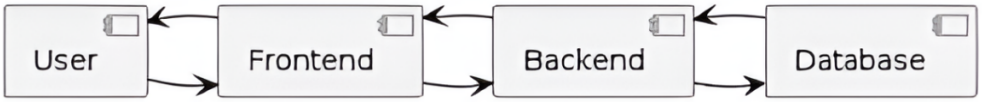

Рисунок 3.3 – Діаграма взаємодії

Архітектура ПЗ проєкту базується на сучасних технологіях, що забезпечують високу продуктивність, гнучкість та масштабованість. Використання Node js та Express.js для бекенду, а також React [17] з Next.js [10] для фронтенду дозволяє створювати ефективні вебзастосунки, що відповідають сучасним вимогам до якості та функціональності.

3.2 Вибір технології та мов програмування

3.2.1 Огляд і аналіз засобів розробки

Створення вебсайту для меню ресторанів включає використання різноманітних методів та засобів. Платформи для управління контентом (CMS), як от WordPress. Завдяки своїй гнучкості та великому вибору плагінів, WordPress є популярним вибором для ресторанів, які бажають створити налаштовувані вебсайти з інтегрованими функціями меню. Joomla та Drupal - популярні CMS, які пропонують розширені можливості кастомізації та управління контентом.

Спеціалізовані платформи для ресторанів , як от платформа GloriaFood для онлайн-замовлень, яка включає функції створення меню та прийому платежів. OpenTable часто використовується для бронювання столиків, але також може інтегруватися з вебсайтами для показу актуальних меню.

Фреймворки веброзробки, такі як React.js та Angular.js - сучасні JavaScript-фреймворки [11], які дозволяють створювати інтерактивні та динамічні вебсайти

для ресторанів. Фреймворк Bootstrap для швидкої розробки адаптивних та мобільно-дружніх вебсайтів.

Інтеграція з зовнішніми API, наприклад Google Maps API - інтеграція карти для вказівки розташування ресторанів.

Payment Gateways API - інтеграція платіжних шлюзів для прийому оплати онлайн. Інструменти для аналітики та оптимізації такі, як Google Analytics використовується для відстеження відвідуваності вебсайту та поведінки користувачів. SEO інструменти - оптимізація вебсайту для пошукових систем для залучення більшої аудиторії.

Аналіз методів і засобів

Переваги:

Більшість сучасних інструментів та технологій дозволяють створювати вебсайти, які адаптовані до різних пристроїв та потреб користувачів. Легка інтеграція з іншими системами та сервісами забезпечує ефективне управління меню та замовленнями.

Недоліки:

Складність управління через необхідність керувати різними системами та інтеграціями може стати складною задачею, особливо для малого бізнесу без відповідних технічних знань. Початкові витрати на розробку та подальше обслуговування вебсайту можуть бути значними, зокрема якщо використовуються преміум інструменти та сервіси.

Ці методи та засоби дозволяють ресторанам ефективно презентувати свої меню в онлайні, покращуючи доступність і зручність для клієнтів. З врахуванням швидкого розвитку цифрових технологій, важливо постійно оновлювати та оптимізувати використання цих інструментів для досягнення найкращих результатів.

На початку створення проєктів важливим аспектом є вибір технологічного стеку. Для Frontend частини було обрано Next.js + Typescript. Вибір Next.js є виправданим завдяки його підтримці серверного рендерингу (SSR), що значно

покращує час завантаження сторінок та оптимізацію SEO [10], важливу для просування ресторану [21]. Typescript , на відміну від ECMAScript [18] додає строгу типізацію [16], що покращує читабельність та масштабованість коду, зменшує ймовірність помилок у розробці. Використання React Bootstrap дозволяє швидко та легко створювати адаптивний дизайн, який автоматично налаштовується під різні пристрої, забезпечуючи хороший користувацький досвід як на мобільних, так і на десктопних версіях [17].

Для Backend частини було обрано Node.js + Express + Typescript. Node.js спільно з фреймворком Express становить потужне середовище для обробки запитів на сервері, легко інтегрується з MongoDB і підтримує побудову RESTful API [9]. Використання Typescript на бекенді [16] забезпечує консистентність коду і спрощує управління великими проєктами. Вибір NoSQL бази даних, як MongoDB [13], ідеально підходить для сценаріїв з великою кількістю читань та неструктурованими даними [12], що типово для систем управління меню, де кожен ресторан може мати унікальні поля в своїх стравах.

Розробка клієнтської частини (Frontend)

Розробка адміністративної панелі та клієнтського інтерфейсу за допомогою Next.js [10] та React Bootstrap забезпечить легкість візуального налаштування і швидкість реалізації стандартних компонентів (кнопки, форми введення, модальні вікна) [3]. Використання адаптивного дизайну гарантує, що вебсайт буде зручний для використання як на десктопах, так і на для мобільних пристроях.

Розробка серверної частини (Backend)

Створення API за допомогою Node.js та Express, яке дозволить адмініструвати меню (створення, редагування, видалення страв) та обробляти запити від клієнтів. Використання MongoDB як бази даних для зберігання даних меню, користувацьких налаштувань та інших необхідних даних забезпечить швидкий доступ та гнучкість у структурі даних [13].

Цей технологічний стек забезпечує високу продуктивність, масштабованість, та зручність у розробці та підтримці вебсайту. Ці технології є добре підтримуваними і широко використовуються у спільноті розробників, що забезпечує велику кількість ресурсів для навчання та вирішення можливих проблем. Крім того, використання SSR через Next.js допомагає покращити SEO, що є важливим для просування ресторану в інтернеті [10].

3.2.2 Бекенд

Технології та мови програмування:

1) **Node.js** [9]:
   - використовується для виконання серверного JavaScript коду [7];
   - файли з розширенням .ts свідчать про використання TypeScript.
2) **TypeScript:**
   - типізована надбудова над JavaScript, які додає статичну типізацію [16];
   - файли: package.json, tsconfig.json, .env.template.
3) **Express.js** [19]:
   - мінімалістичний вебфреймворк для Node.js [9], який спрощує розробку серверного коду.
   - використовується у файлах контролерів, таких як admins.controllers.ts, auth.controllers.ts, bookings.controllers.ts.
4) **MongoDB:**
   - NoSQL база даних, яка використовується для зберігання даних [12];
   - файли: admins.mongo.ts, bookings.mongo.ts.

Структура проєкту

Проєкт організовано за типом MVC (Model-View-Controller), що дозволяє чітко розділити логіку застосунку на окремі компоненти:

- **Models:** файли з моделями даних (admins.model.ts, bookings.model.ts).

– **Controllers**: файли з логікою обробки запитів (admins.controller.ts, auth.controller.ts).
– **Middleware**: файли з проміжною логікою (auth-check.ts, errorHandler.ts).

3.2.3 Фронтенд

Під час розробки веб сайту застосовувалась методика Mobile First [17]. Технології та мови програмування:

1) React [3]:

– бібліотека для створення користувацьких інтерфейсів;
– файли з розширенням tsx свідчать про використання JSX разом з TypeScript.

2) Next.js [10]:

– фреймворк для React, який додає серверний рендеринг та інші можливості [17];
– файл packagejson містить залежності, які вказують на використання Next.js.

3) Tailwind CSS:

– утилітарний CSS фреймворк для швидкого створення стилів;
– файл tailwind.config.js вказує на використання Tailwind CSS.

4) Redux:

– бібліотека для керування станом застосунку.
– файли в директорії src/lib/features свідчать про використання Redux.

Структура проєкту

Проєкт має структуру, типової для сучасного React-застосунку [17] з використанням Next.js:

– **компоненти**: файли в директорії src/components (qr-code-modal.tsx, restaurant-card.tsx).
– **сторінки**: файли в директорії src/pages (index.tsx,_app.tsx).

– **стан**: файли в директорії ste/lib/features (addToCartModalSlice.ts, adminSlice.ts).

3.2.4 Інструменти для розробки та збірки

1) npm:

– менеджер пакетів для Node.js [9], використовується для управління залежностями.

– файли: package.json.

2) Webpack:

– інструмент для збірки модулів JavaScript [11].

– можливе використання у фронтенді для збору та обробки файлів.

3) ESLint:

– інструмент для аналізу коду на наявність помилок;

– файл eslintre.js свідчить про використання ESLint.

4) Prettier:

– інструмент для форматування коду;

– файл prettierre свідчить про використання Prettier.

Проєкт використовує сучасні технології та мови програмування для реалізації бекенду та фронтенду. Для серверної частини використовується Node.js та TypeScript разом з фреймворком Express.js. Для клієнтської частини використовується React з Next.js, Tailwind CSS та Redux. Це забезпечує високу гнучкість, продуктивність та можливість швидкого розширення функціональності.

3.2.5 База даних системи

Фізична схема БД визначає структуру таблиць і їх зв'язків. У проєкті використовується MongoDB, що є NoSQL базою даних [12].

На рис. 3.4 відображено фізичну схему бази даних.

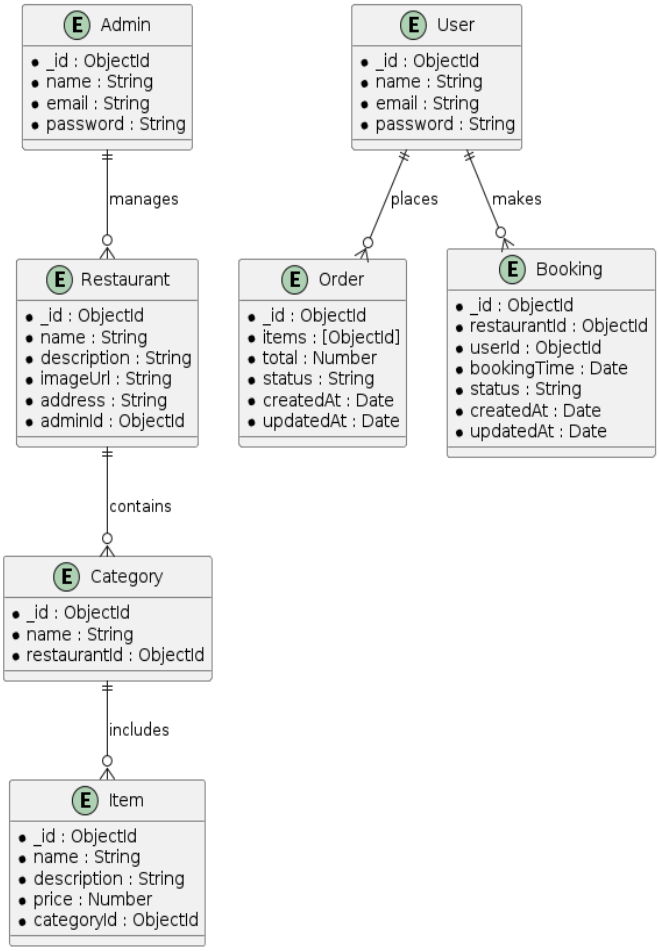

Рисунок 3.4 – Фізична схема бази даних

Основні колекції БД:

1) користувачі (Users);
2) адміністратори (Admins);
3) ресторани (Restaurants);
4) категорії (Categories);
5) меню (Items);
6) замовлення (Orders);
7) бронювання (Bookings).

3.3 Вибір компоненти ПЗ (бібліотеки, патерни, плагіни тощо)

Розробка програмного забезпечення (ПЗ) вимагає вибору відповідних компонентів, які забезпечують функціональність, гнучкість і масштабованість

системи. Нижче наведено вибір основних компонентів, бібліотек, патернів і плагінів для проєкту, виходячи з аналізу архіву.

Express.js – мінімалістичний вебфреймворк для Node.js [19], який використовується для створення серверного API. Забезпечує простий і ефективний спосіб створення маршрутів та обробки запитів. Mongoose - бібліотека для роботи з MongoDB [13] у середовищі Node.js. Забезпечує зручний API для визначення схем даних і виконання CRUD-операцій, jsonwebtoken - бібліотека для роботи з JSON Web Token (JWT). Використовується для аутентифікації та авторизації користувачів.

berypt – бібліотека для хешування паролів. Забезпечує безпечне зберігання паролів користувачів. dotenv - бібліотска для завантаження змінних середовища з env файлу. Дозволяє зберігати конфіденційні дані окремо від коду.

React – бібліотека для створення користувацьких інтерфейсів. Використовується для побудови компонентів інтерфейсу, які можна повторно використовувати [3]. Next.js - фреймворк для , який забезпечує серверний рендеринг та інші можливості. Дозволяє створювати SEO-оптимізовані вебзастосунки з високою продуктивністю [10].

Redux – бібліотека для управління станом застосунку. Дозволяє централізовано керувати станом застосунку, що спрощує його обслуговування та масштабування. Tailwind CSS - утилітарний CSS-фреймворк для швидкого створення стилів. Дозволяє створювати адаптивні та кастомізовані інтерфейси з мінімальними зусиллями.

Axios – бібліотека для виконання HTTP-запитів. Використовується для взаємодії з серверним API.

MVC (Model-View-Controller) – патерн проєктування, який розділяє логіку застосунку на три основні компоненти: модель, представлення та контролер. Забезпечує чітке розділення відповідальностей і покращує підтримку коду. Singleton – патерн, який забезпечує існування лише одного екземпляра класу. Використовується для конфігураційних класів та з’єднань з базою даних. Factory

- патерн, який використовує фабричні методи для створення об’єктів. Дозволяє створювати об’єкти без необхідності вказувати точний клас об’єкта, який буде створено.

ESLint – інструмент для аналізу коду на наявність помилок та дотримання стилю кодування. Допомагає підтримувати якість та консистентність коду. Prettier - інструмент для автоматичного форматування коду. Забезпечує єдиний стиль кодування у всьому проєкті.

Webpack – інструмент для збірки модулів JavaScript [4]. Використовується для компіляції, пакування та оптимізації ресурсів фронтенд застосунку. Babel - транспілятор для JavaScript [20], який дозволяє використовувати сучасні можливості мови у старих середовищах. Забезпечує сумісність коду з різними версіями браузерів.

3.4 Опис інтерфейсів ПЗ

Інтерфейси програмного забезпечення (ПЗ) є ключовими елементами, які визначають взаємодію між різними компонентами системи, а також між системою та користувачем. У цьому розділі описуються основні типи інтерфейсів, що використовуються у проєкті, зокрема користувацький інтерфейс (UI) та програмні інтерфейси (API).

Користувацький інтерфейс є важливим елементом фронтенд застосунку, що забезпечує взаємодію кінцевих користувачів з системою. Основні компоненти користувацького інтерфейсу включають:

Компоненти інтерфейсу:

– форми - інтерфейси для введення даних, зокрема форми реєстрації та входу, форми створення та редагування замовлень.

– модальні вікна для підтвердження дій або відображення додаткової інформації (наприклад, QR-код модальне вікно).

– картки компоненти для відображення інформації у структурованому вигляді (наприклад, картки ресторанів).

– списки – компоненти для відображення переліків елементів (наприклад, список замовлень або категорій).

Сторінки та маршрути:

– головна сторінка відображає загальну інформацію та навігацію по основним розділам.

– сторінка замовлень відображає перелік замовлень та деталі кожного замовлення.

– сторінка ресторанів відображає список ресторанів та деталі кожного ресторану.

Стилі та теми:

– Tailwind CSS використовується для створення адаптивних стилів інтерфейсу.

– компоненти стилів – визначення кольорової схеми, шрифтів та інших елементів стилю.

Програмні інтерфейси забезпечують взаємодію між клієнтською та серверною частинами системи, а також між різними сервісами бекенду.

Основні компоненти API включають:

1) REST API:

- використовується для передачі даних між клієнтом та сервером;
- визначає набір запитів для доступу до ресурсів системи.

2) Запити:

- GET – отримання ресурсів (наприклад, отримання списку ресторанів або замовлень).
- POST – створення нових ресурсів (наприклад, створення нового замовлення).
- PUT – оновлення існуючих ресурсів (наприклад, оновлення інформації про ресторан).
- DELETE – видалення ресурсів (наприклад, видалення замовлення).

Інтерфейси ПЗ проєкту забезпечують ефективну взаємодію між різними компонентами системи, а також між системою та користувачем. Використання сучасних технологій та бібліотек дозволяє створювати інтуїтивно зрозумілі та зручні користувацькі інтерфейси, а також надійні програмні інтерфейси для взаємодії клієнт-сервер.

3.5 Діаграма класів

Діаграма класів — статичне представлення структури моделі. Відображає статичні елементи, такі як: класи, типи даних, їх зміст та відношення. Взаємозв'язки між класами забезпечують логічну структуру даних, де ресторани можуть мати багато бронювань та категорій, а кожна категорія може містити багато елементів меню. Це дозволяє ефективно організувати дані та забезпечити необхідний функціонал системи для управління рестораном.

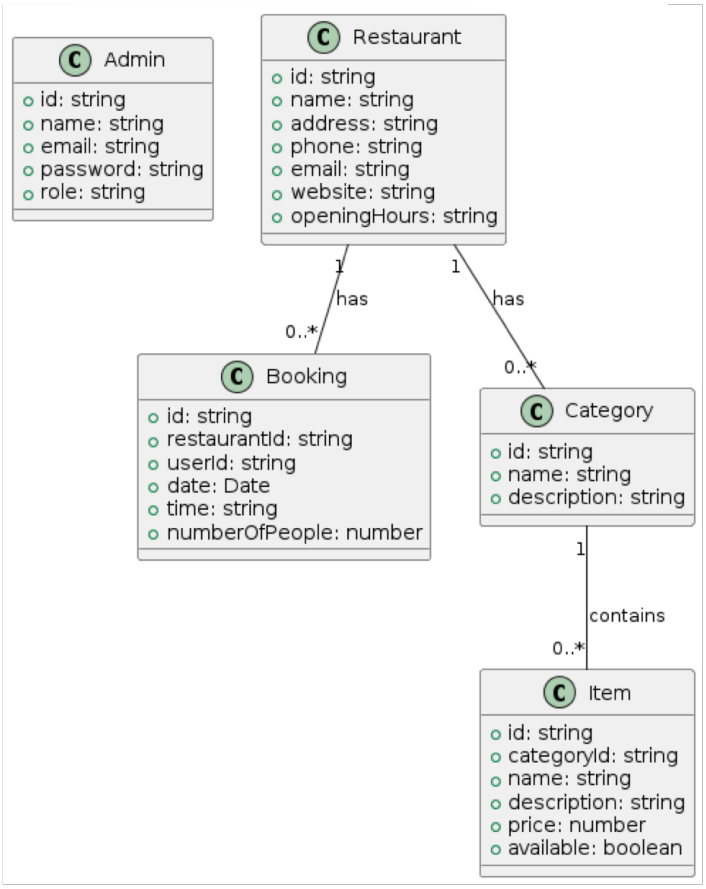

Рисунок 3.5 – Діаграма класів

Дана діаграма класів описує модель даних для системи управління рестораном. Вона включає такі основні класи, як Admin, Restaurant, Booking, Category та Item, кожен з яких має свої атрибути та відповідає за різні аспекти системи.

Admin відповідає за управління системою, включаючи адміністрування користувачів та ресторанів.

Restaurant представляє заклад з усією необхідною інформацією, такою як назва, адреса, контакти та години роботи.

Booking зберігає дані про бронювання столиків у ресторані, включаючи дату, час та кількість осіб.

Category визначає різні категорії меню, такі як напої, десерти, головні страви тощо.

Item представляє конкретні страви чи напої в меню, з інформацією про назву, опис, ціну та доступність.

Взаємозв'язки між класами забезпечують логічну структуру даних, де ресторани можуть мати багато бронювань та категорій, а кожна категорія може містити багато елементів меню. Це дозволяє ефективно організувати дані та забезпечити необхідний функціонал системи для управління рестораном [14].

Таким чином, діаграма класів надає чітке уявлення про структуру даних у системі, що є важливим етапом у процесі розробки програмного забезпечення, забезпечуючи основу для подальшої реалізації та тестування функціоналу.

Висновки до розділу 3

У третьому розділі було детально розглянуто архітектуру моделювання та проєктування програмного забезпечення вебзастосунку для створення онлайн меню ресторану. Базова архітектура проєкту складається з двох основних частин: фронтенд і бекенд. Для реалізації серверної частини (бекенд) було використано Node.js та Express.js, що забезпечує високу продуктивність та масштабованість.

Фронтенд частина реалізована за допомогою React та Next.js, що забезпечує швидке відображення сторінок та оптимізацію для пошукових систем (SEO).

Таким чином, у цьому розділі було визначено та детально описано архітектурні рішення, технологічний стек, основні компоненти програмного забезпечення та створено UML-діаграми, що відображають ключові аспекти функціонування системи. Це забезпечує чітке розуміння структури та логіки роботи вебзастосунку, що с основою для його успішної реалізації.

4. ОСНОВНІ МЕТОДИ ВЕБ-ЗАСТОСУНКУ. ІНТЕРФЕЙС ТА КЕРІВНИЦТВО КОРИСТУВАЧА. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Опис основних методів

Нижче представлено детальний опис основних методів, використаних у проєкті для забезпечення функціональності вебзастосунку. Кожен метод відіграє важливу роль у взаємодії між різними компонентами системи, такими як серверна частина (бекенд) та клієнтська частина (фронтенд), а також у забезпеченні зручного та безпечного користування застосунком для кінцевих користувачів та адміністраторів.

У табл. 4.1 наведено опис основних методів, використаних у проєкті, з логікою їх виконання та прикладами застосування.

Таблиця 4.1 – Опис методів веб - застосунку

Назва методу	**Логіка**	**Застосування**
authenticate	Перевіряє, чи користувач аутентифікований на основі токена.	Використовується як middleware для захисту маршрутів API.
getRestaurants	Отримує список усіх ресторанів з бази даних.	Викликається в контролері для відображення списку ресторанів.
createRestaurant	Створює новий ресторан на основі даних, отриманих з запиту.	Викликається в контролері для додавання нового ресторану.

updateRestaurant	Оновлює інформацію про ресторан на основі даних з запиту та ID ресторану.	Використовується для оновлення інформації про ресторан.
deleteRestaurant	Видаляє ресторан з бази даних на основі ID.	Використовується для видалення ресторану.
generateQRCode	Генерує QR-код на основі заданого URL.	Використовується для створення QR-кодів для меню ресторанів.
createOrder	Створює нове замовлення на основі даних з запиту.	Використовується для створення замовлень користувачами.
getOrders	Отримує список усіх замовлень користувача.	Викликається для відображення історії замовлень користувача.
createBooking	Створює нове бронювання на основі даних з запиту.	Використовується для бронювання столиків у ресторанах.
getBookings	Отримує список бронювань користувача.	Викликається для відображення історії бронювань користувача.

Основні методи, використані у проєкті, забезпечують функціональність, необхідну для управління користувачами, ресторанами, замовленнями та бронюваннями. Кожен метод має чітку логіку та застосування, що сприяє ефективній реалізації та підтримці проєкту.

4.2 Опис інтерфейсу користувача

Реєстрація користувача передбачає заповнення форми з полями для введення імені, email та пароля. Після заповнення форми дані надсилаються на сервер для створення нового користувача в базі даних. На рис.4.1 відображено форму реєстрації користувача.

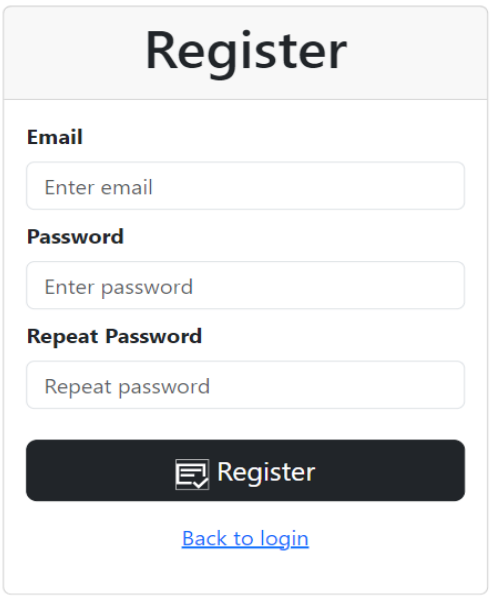

Рисунок 4.1 – Реєстрація користувача

Головна сторінка адмін панелі застосунку відображає доступні функції та навігацію до різних розділів (наприклад, створення нового ресторану та вихід). На рис. 4.2 відображено головну сторінку адмін панелі.

Рисунок 4.2 – Головна сторінка

Адміністратор може додати новий ресторан, заповнивши форму з інформацією про ресторан (назва, контакти, адреса). На рис.4.3 відображено форму створення нового ресторану.

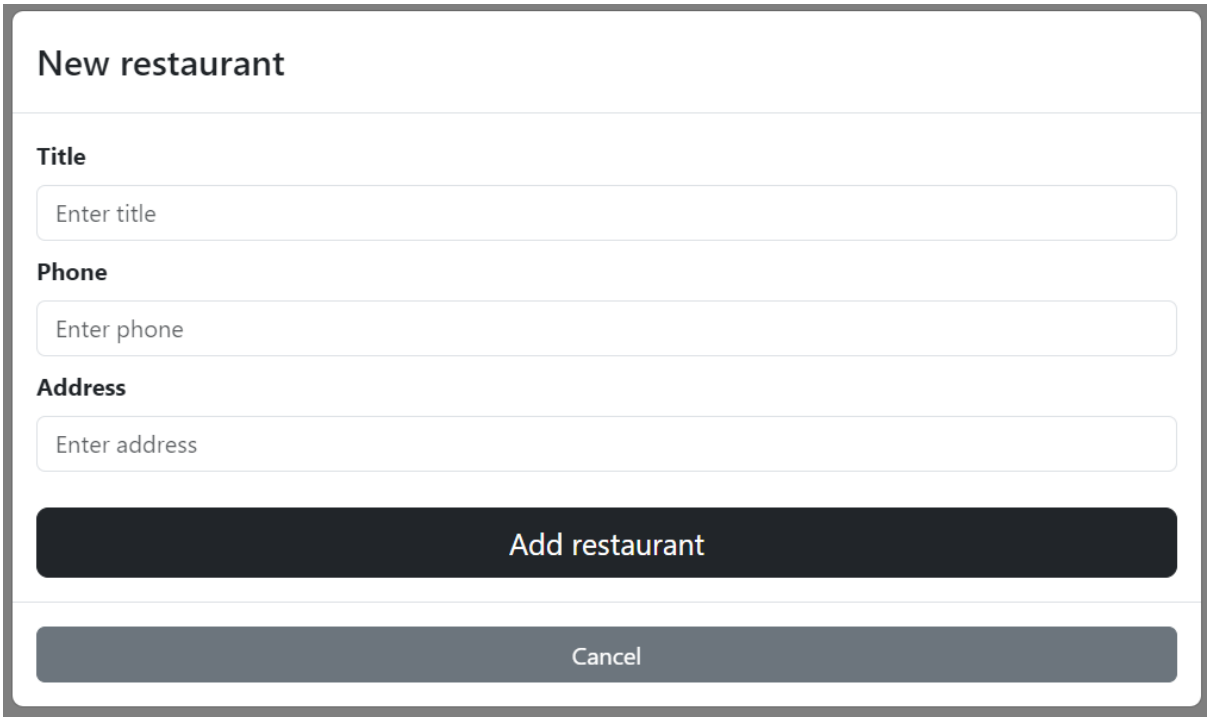

Рисунок 4.3 – Додавання ресторану

Адміністратор може виконувати різні дії з рестораном, такі як редагування, видалення або перегляд замовлень, генерація QR - коду меню ресторану. На рис.4.4 відображено можливі дії щодо створеного ресторану.

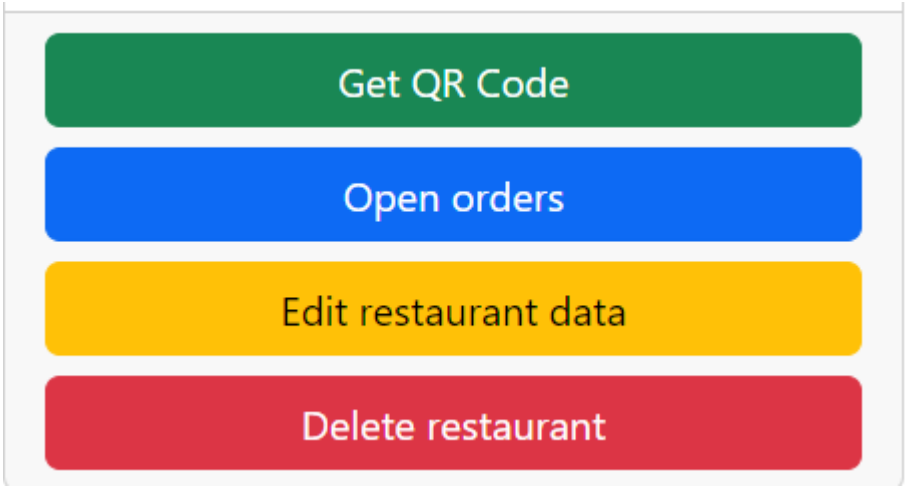

Рисунок 4.4 – Дії щодо створеного ресторану

Адміністратор може згенерувати QR-код для доступу до меню ресторану. На рис.4.5 відображено генерацію QR-коду меню ресторану.

Рисунок 4.5 - Отримання QR - коду меню ресторану

Адміністратор може додати нові пункти меню, заповнивши форму з інформацією про страву (назва, опис, ціна тощо). На рис.4.6 відображено форми додавання страв меню.

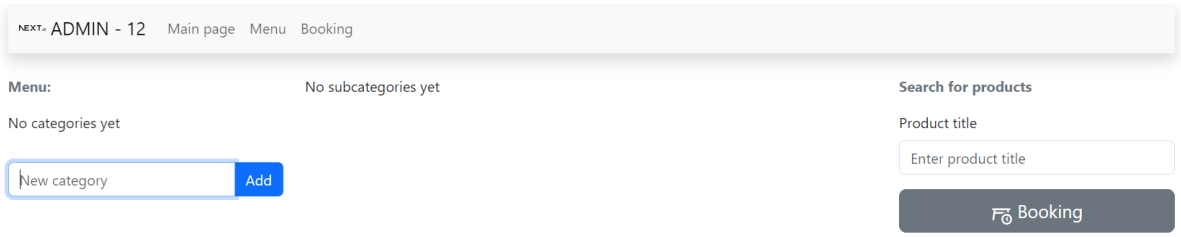

Рисунок 4.6 – Додавання елементів меню

Адміністратор може додати або оновити інформацію про ресторан (наприклад, години роботи, контактну інформацію). На рис.4.7 відображено форми для інформації про ресторан.

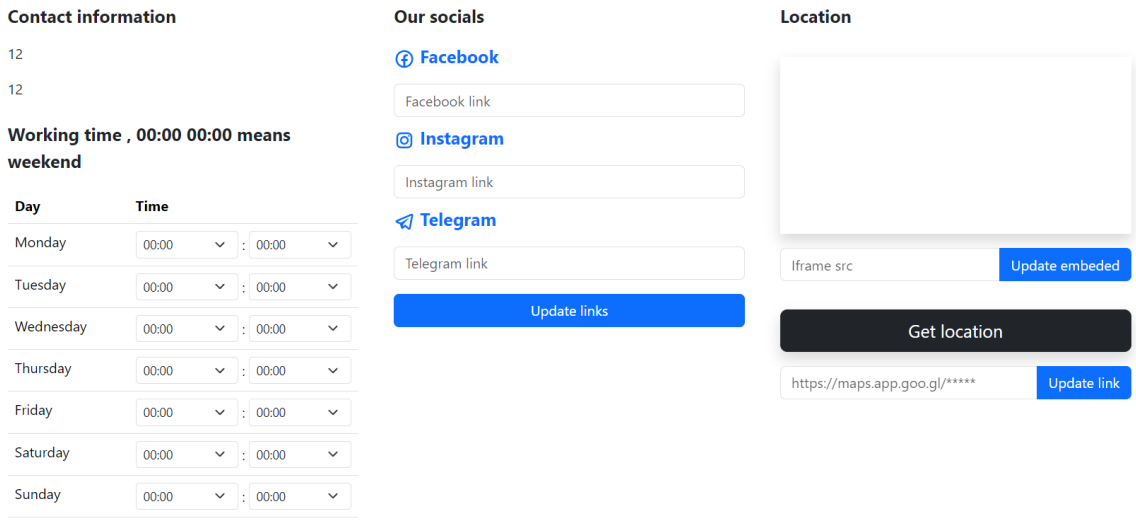

Рисунок 4.7 – Додавання інформації про ресторан

Користувач може переглянути створене меню ресторану, яке містить всі пункти меню з їх описом та ціною. На рис. 4.8 відображено створене меню ресторану.

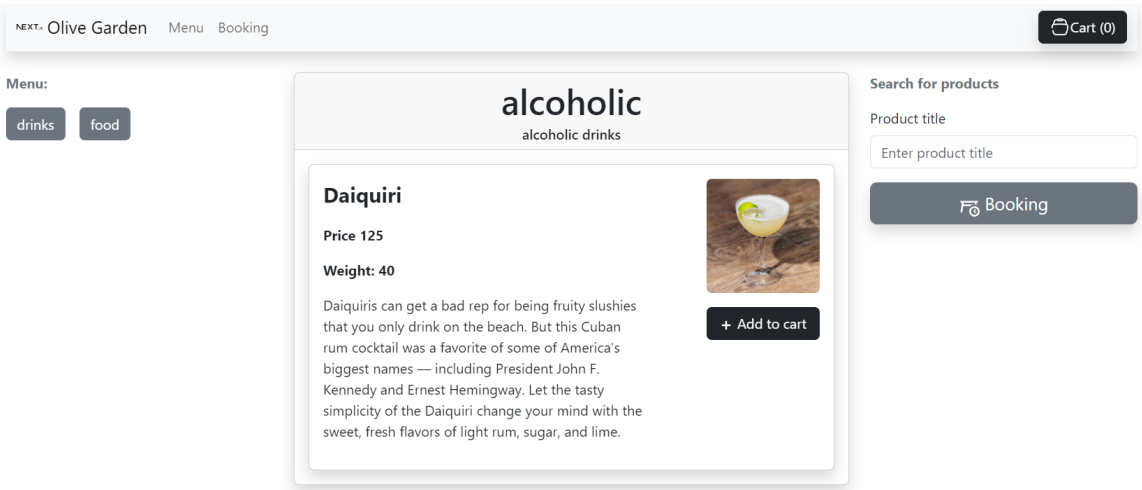

Рисунок 4.8 – Створене меню ресторану

Користувач може забронювати столик у ресторані, заповнивши форму бронювання. На рис. 4.9 відображено форму бронювання столика в ресторані.

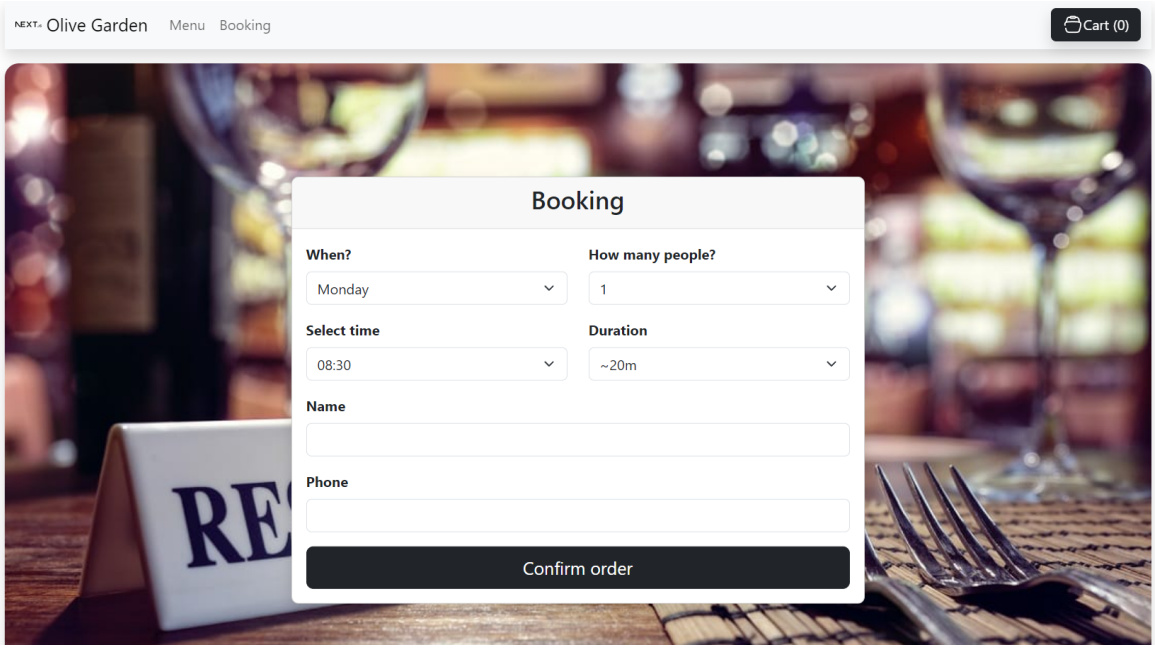

Рисунок 4.9 – Бронювання столика в ресторані

Рисунки описують ключові функції та процеси у застосунку, зокрема реєстрацію користувачів, управління ресторанами, створення та відображення меню, а також бронювання столиків. Це забезпечує чітке розуміння кожного етапу взаємодії користувача з системою.

4.3 Тестування та аналіз результатів

Для тестування застосунку було обрано технології Mocha та Chai. Mocha та Chai були обрані завдяки їх гнучкості, широким можливостям та читабельності тестів. Mocha надає потужну основу для організації та виконання тестів, тоді як Chai забезпечує зрозумілу та гнучку систему асерцій. Разом вони створюють ефективне та зрозуміле середовище для написання юніт тестів у Node.js проєктах.

Обрано процес тестування Unit Testing. Це обумовлено наступними причинами:

– ізольоване тестування: Ми можемо тестувати методи незалежно від інших частин системи, забезпечуючи, що тест функція коректно обробляє вхідні дані та відповідає очікуваній поведінці.

– легкість локалізації помилок: У разі невдачі тестів ми точно знаємо, де шукати проблему – у конкретному методі, що спрощує процес налагодження.

– швидкість виконання: Юніт тести зазвичай виконуються швидко, що важливо для безперервної інтеграції та розгортання (CI/CD).

Для кожної функції створюється окремий тестовий файл. Використовується патерн Arrange-Act-Assert (AAA) для кожного тесту, що забезпечує чітку структуру та високу читабельність тестів. Тестування нормальних випадків забезпечує правильне виконання функцій з коректними вхідними даними, підтверджуючи, що функція виконує всі необхідні дії та повертає очікуваний результат. Тестування некоректних вхідних даних перевіряє, як функції обробляють неочікувані або неправильні дані, забезпечуючи відповідні коди помилок та повідомлення.

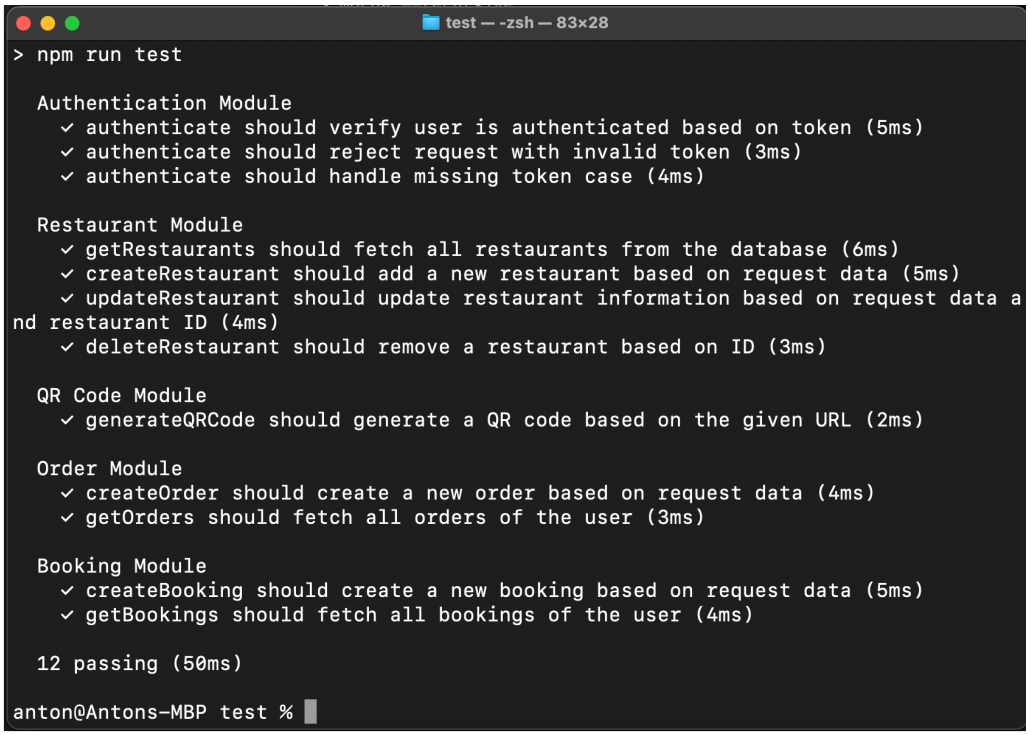

Рисунок 4.1 – Результат тестування застосунку

На рис 4.1 відображено результат тестування. Після запуску всіх юніт тестів, було виявлено, що всі вони пройшли успішно. Це означає, що кожна функція нашого застосунку виконується відповідно до очікувань як у позитивних, так і в негативних сценаріях. Процес розробки та виконання юніт

тестів для нашого застосунку продемонстрував високу якість реалізації функцій та здатність системи коректно обробляти різні сценарії використання. Успішне проходження всіх тестів підтверджує, що наш застосунок відповідає вимогам і готовий до розгортання.

4.4 Керівництво користувача

Нижче наведено детальні інструкції для користувачів щодо використання основних функцій вебзастосунку. Керівництво охоплює процеси реєстрації, входу в систему, управління ресторанами, додавання меню, генерації QR-кодів та бронювання столиків.

Реєстрація користувача

1) відкрийте головну сторінку вебзастосунку;
2) натисніть на кнопку «Реєстрація» у верхньому правому куті;
3) заповніть форму реєстрації, вказавши своє ім'я, електронну адресу та пароль;
4) натисніть кнопку «Зареєструватися»;
5) після успішної реєстрації ви отримаєте підтвердження на вказану електронну адресу.

Вхід в систему

1) відкрийте головну сторінку вебзастосунку;
2) натисніть на кнопку «Вхід» у верхньому правому куті;
3) введіть свою електронну адресу та пароль;
4) натисніть кнопку «Увійти»;
5) після успішного входу ви будете перенаправлені на головну сторінку.

Додавання ресторану

1) після входу в систему натисніть на кнопку "Додати ресторан" у навігаційному меню;
2) заповніть форму додавання ресторану, вказавши назву, опис, адресу та іншу необхідну інформацію;

3) натисніть кнопку «Додати ресторан»;

4) після успішного додавання ресторан буде відображено у списку ваших ресторанів.

Дії щодо створеного ресторану

1) Перейдіть до списку ваших ресторанів, натиснувши на відповідну вкладку у навігаційному меню.

2) Виберіть ресторан, який ви бажаєте редагувати або видалити.

3) Натисніть на кнопку "Редагувати" для внесення змін до інформації про ресторан.

4) Натисніть на кнопку "Видалити" для видалення ресторану зі списку.

Отримання QR-коду меню ресторану

1) Виберіть ресторан зі списку ваших ресторанів.

2) Натисніть на кнопку "Отримати QR-код" біля назви ресторану.

3) Система згенерує QR-код, який буде відображено на екрані.

4) Збережіть QR-код або роздрукуйте його для використання у вашому ресторані.

Додавання елементів меню

1) Виберіть ресторан зі списку ваших ресторанів.

2) Натисніть на кнопку "Додати пункт меню".

3) Заповніть форму додавання пункту меню, вказавши назву, опис та ціну.

4) Натисніть кнопку "Додати пункт меню".

5) Після успішного додавання новий пункт меню буде відображено у списку меню ресторану.

Додавання інформації про ресторан

1) Виберіть ресторан зі списку ваших ресторанів.

2) Натисніть на кнопку "Редагувати" біля назви ресторану.

3) Внесіть необхідні зміни до інформації про ресторан, такі як години роботи, контактна інформація тощо.

4) Натисніть кнопку "Зберегти зміни".

5) Після успішного збереження зміни будуть застосовані до інформації про ресторан.

Створене меню ресторану

1) Виберіть ресторан зі списку ваших ресторанів.

2) Натисніть на вкладку "Меню".

3) Список пунктів меню буде відображено на екрані.

4) Ви можете переглядати, редагувати або видаляти пункти меню за потреби.

Бронювання столика в ресторані

1) Перейдіть на головну сторінку та виберіть ресторан, у якому ви бажаєте забронювати столик.

2) Натисніть на кнопку "Бронювання".

3) Заповніть форму бронювання, вказавши дату, час та кількість гостей.

4) Натисніть кнопку "Забронювати".

5) Після успішного бронювання ви отримаєте підтвердження на вказану електронну адресу.

Керівництво користувача надає чіткі інструкції щодо використання основних функцій вебзастосунку. Дотримуючись цих інструкцій, користувачі можуть легко взаємодіяти з системою, керувати своїми ресторанами, створювати та редагувати меню, генерувати QR-коди, а також бронювати столики в ресторанах.

Висновки до розділу 4

У розділі 4 було здійснено детальний опис, кодування, тестування (верифікацію та валідацію) програмного забезпечення (ПЗ), проведення обчислень (апробація ПЗ), а також аналіз результатів обчислень і розробка керівництва користувача. Було представлено детальний опис основних методів, що використовуються у проєкті. Опис включав логіку виконання та застосування

кожного методу. Це забезпечило чітке розуміння їх функціональності та взаємодії між компонентами системи.

Було проведено тестування ключових функцій вебзастосунку, таких як реєстрація користувачів, додавання ресторанів, створення та відображення меню, генерація QR-кодів і бронювання столиків. Це тестування допомогло переконатися у коректності та надійності роботи ПЗ.

Було розроблено детальне керівництво користувача, яке охоплює всі основні процеси взаємодії з застосунком: реєстрація, вхід, додавання ресторанів, управління меню, генерація QR-кодів і бронювання столиків. Це керівництво допоможе користувачам легко і ефективно використовувати застосунок.

ВИСНОВКИ

В результаті проведеної роботи досягнуто значних результатів у реалізації веб-застосунку для створення онлайн-меню ресторанів, що забезпечує зручне та ефективне управління меню, замовленнями та бронюваннями.

Було протестовано ключові функції веб-застосунку, такі як реєстрація користувачів, додавання ресторанів, створення та відображення меню, генерація QR-кодів і бронювання столиків. Це підтвердило коректність та надійність роботи програмного забезпечення.

Розроблено детальне керівництво користувача, що охоплює всі основні процеси взаємодії з застосунком: реєстрація, вхід, додавання ресторанів, управління меню, генерація QR-кодів і бронювання столиків, що допоможе користувачам легко і ефективно використовувати застосунок.

Основні досягнення включають:

Детальний опис та аналіз об'єкту і предмету дослідження.

Опис етапів реалізації проєкту, включаючи розробку алгоритмів та фізичної схеми бази даних.

Визначення архітектурних рішень, технологічного стеку та основних компонентів програмного забезпечення.

Проведення кодування, тестування та валідації програмного забезпечення.

Розробка детального керівництва користувача.

Ці результати сприяють створенню ефективного та зручного у використанні веб-застосунку, який відповідає сучасним вимогам до якості та функціональності програмного забезпечення у сфері ресторанного бізнесу. Користувачі зможуть легко управляти своїми ресторанами, створювати та редагувати меню, генерувати QR-коди та забезпечувати зручне бронювання столиків, що покращує загальний досвід взаємодії з рестораном для кінцевих користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bass L. Software Architecture in Practice. Addison-Wesley, 2013. 662 с.
2. Greet: #1 QR code menu for restaurants. Greet. URL: https://greet.menu/en/home/ (дата звернення: 17.01.2024).
3. Hartmann M., Dziurzanski P. React 16 Essentials. Packt Publishing, 2018. 240 с.
4. Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming. No Starch Press, 2018. 472 с.
5. Hunt A., Thomas D. The Pragmatic Programmer: Your Journey to Mastery. Addison-Wesley, 2019. 352 с.
6. Newman S. Building Microservices. O'Reilly Media, 2019. 612 с.
7. Nixon R. Learning PHP, MySQL & JavaScript. O'Reilly Media, 2018. 528 с.
8. Online Ordering System for Restaurants. UpMenu. URL: https://www.upmenu.com/ (дата звернення: 17.01.2024).
9. Owens S., Housley N. Node.js Design Patterns. Packt Publishing, 2018. 664 с.
10. Pathak P. Next.js Quick Start Guide. Packt Publishing, 2020. 164 с.
11. Resig J., Bibeault B. Secrets of the JavaScript Ninja. Manning Publications, 2013. 464 с.
12. Sadalage P., Fowler M. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley, 2013. 192 с.
13. Shusterman B., Robinson A. The MongoDB 4.0 Manual. MongoDB, Inc., 2017. 610 с.
14. Simpson J., Moffat A. Introduction to Information Retrieval. Cambridge University Press, 2017. 581 с.
15. Sommer N. Mastering React Test-Driven Development. Packt Publishing, 2020. 496 с.
16. Sterr G., Hoppe H. Hands-On RESTful Web Services with TypeScript 3. Packt Publishing, 2019. 470 с.
17. Wróblewski L. Mobile First. A Book Apart, 2014. 130 с.

18. Zakas N. C. Understanding ECMAScript 6: The Definitive Guide for JavaScript Developers. No Starch Press, 2016. 352 с.
19. Побудова динамічного веб-сайту з використанням Node.js та Express.js - IT рейтинг UA. IT рейтинг України. URL: https://it-rating.ua/pobudova-dinamichnogo-veb-saytu-z-vikoristannyam-nodejs-ta-expressjs (дата звернення: 02.04.2024).
20. Простий посібник зі схем UML і моделювання баз даних. URL: https://www.microsoft.com/uk-ua/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling (дата звернення: 30.03.2024).
21. Силівейстр В. Створення сайту ресторану – яким повинен бути сайт кафе чи ресторану – Poster POS. *Poster POS – программа автоматизации* HoReCa: система учета для общепита на планшете. URL: https://joinposter.com/ua/post/sait-dlya-kafe-ta-restoraniv (дата звернення: 21.04.2024).
22. п

ДОДАТОК А

Лістинг файлу admins.controller.ts

```
import * as adminsModel from "../models/admins/admins.model";
import { Request, Response } from "express";

const getById = async(req: Request, res: Response, next) => {
   const { _id } = req.query;

   try {
      const admin = await adminsModel.id(_id as string);
      return res.status(200).json({
         ok: true,
         admin,
      })
   } catch (error) {
      next(error);
   }
}

export {
   getById,
}
```

ДОДАТОК Б

Лістинг файлу auth.controller.ts

```
import passport from "passport";
import { Request, Response } from "express";
import jwt from "jsonwebtoken";
import Admin from "../models/admins/admins.mongo";

const currentUser = async(req: Request, res: Response) => {
   // Decode the JWT token to get the user object
   return res.json({
      ok: req?.user && true,
      user: req.user,
   });
}

const localLogin = (req: Request, res: Response, next) => {
   passport.authenticate('local-login', function(err, token, userData) {
      if (err) {
         return res.status(400).json({
            ok: false,
         });
      }

      return res.json({
         ok: true,
         message: 'Logged in',
         token,
         user: userData
      });
   })(req, res, next);
}

const localSignup = (req: Request, res: Response, next) => {
   passport.authenticate('local-signup', function(err, token, userData) {
      if (err) {
         return res.status(400).json({
            ok: false,
```

```
            });
        }

        return res.json({
            ok: true,
            message: 'Logged in',
            token,
            user: userData
        });
    })(req, res, next);
}


export {
    currentUser,
    localLogin,
    localSignup,
}
```

ДОДАТОК В

Лістинг файлу bookings.controller.ts

```
import { Request, Response } from "express";
import * as bookingModel from "../models/bookings/bookings.model";


const createBooking = async(req: Request, res: Response, next) => {
   const { booking } = req.body;
   try {
      const bookingInserted = await bookingModel.create(booking);
      return res.status(201).json({
         ok: true,
         booking: bookingInserted[0],
      });
   } catch (error) {
      next(error);
   }
}

const cancelBooking = async(req: Request, res: Response, next) => {
   const { _id } = req.body;

   try {
      const booking = await bookingModel.cancel(_id);
      return res.status(200).json({
         ok: true,
         booking,
      });
   } catch (error) {
      next(error);
   }
}

const getBookingById = async(req: Request, res: Response, next) => {
   const { _id } = req.query;

   try {
```

```
        const booking = await bookingModel.id(_id as string);
        return res.status(200).json({
            ok: true,
            booking,
        });
    } catch (error) {
        next(error);
    }
}

const getBookingsByIds = async(req: Request, res: Response, next) => {
    const { ids } = req.query;

    try {
        const bookings = await bookingModel.ids(ids as string[]);
        return res.status(200).json({
            ok: true,
            bookings,
        });
    } catch (error) {
        next(error);
    }
}

const setBokingStatus = async(req: Request, res: Response, next) => {
    const { status, _id } = req.body;
    try {
        const booking = await bookingModel.setStatus(_id, status);
        return res.status(200).json({
            ok: true,
            booking,
        })
    } catch (error) {
        next(error);
    }
}

const getAllByRestaurant = async(req: Request, res: Response, next) => {
```

```
    const {restaurantId} = req.query;
    try {
        const bookings = await bookingModel.getAllByRestaurant(restaurantId as
string);
        return res.status(200).json({
            ok: true,
            bookings,
        })
    } catch (error) {
        next(error);
    }
}

export {
    createBooking,
    cancelBooking,
    getBookingById,
    getBookingsByIds,
    setBokingStatus,
    getAllByRestaurant,
}
```

ДОДАТОК Г

Лістинг файлу restaurants.controller.ts

```
import { Request, Response } from "express";
import * as restaurtantsModel from "../models/restaurants/restaurants.model";
import { TNewRestaurant } from "../models/restaurants/types";


const createRestaurant = async(req: Request, res: Response, next) => {
   const data: TNewRestaurant = req.body;

   try {
      const restaurant = await restaurtantsModel.create(data);
      return res.status(201).json({
         ok: true,
         restaurant: restaurant[0],
      });
   } catch (error) {
      next(error);
   }
}

const removeRestaurant = async(req: Request, res: Response, next) => {
   const { _id  } = req.body;

   try {
      const restaurant = await restaurtantsModel.remove(_id);
      return res.status(200).json({
         ok: true,
         restaurant,
      });
   } catch (error) {
      next(error);
   }
}

const updateRestaurant = async(req: Request, res: Response, next) => {
   const { _id, what, value } = req.body;
```

```
    try {
        const restaurant = await restaurtantsModel.update(_id, what, value);
        return res.status(200).json({
            ok: true,
            restaurant,
        });
    } catch (error) {
        next(error);
    }
}

const getRestaurantById = async(req: Request, res: Response, next) => {
    const { _id } = req.query;
    try {
        const restaurant = await restaurtantsModel.id(_id as string);
        return res.status(200).json({
            ok: true,
            restaurant,
        });
    } catch (error) {
        next(error);
    }
}

const getRestaurantsByIds = async(req: Request, res: Response, next) => {
    const { ids } = req.query;
    try {
        const restaurants = await restaurtantsModel.ids(ids as string[]);
        return res.status(200).json({
            ok: true,
            restaurants,
        });
    } catch (error) {
        next(error);
    }
}
```

```
const addToWelcomePhotos = async(req: Request, res: Response, next) => {
   const { _id, photo } = req.body;
   try {
      const restaurant = await restaurtantsModel.addToWelcomePhotos(_id, photo);
      return res.status(200).json({
         ok: true,
         restaurant,
      });
   } catch (error) {
      next(error);
   }
}

const removeFromWelcomePhotos = async(req: Request, res: Response, next) => {
   const { _id, photo } = req.body;
   try {
      const restaurant = await restaurtantsModel.removeFromWelcomePhotos(_id,
photo);
      return res.status(200).json({
         ok: true,
         restaurant,
      });
   } catch (error) {
      next(error);
   }
}

const getByAdminId = async(req: Request, res: Response, next) => {
   const { adminId } = req.query;
   try {
      const restaurants = await restaurtantsModel.getByAdmin(adminId as string);
      return res.status(200).json({
         ok: true,
         restaurants,
      });
   } catch (error) {
      next(error);
   }
```

```
}



export {
    createRestaurant,
    removeRestaurant,
    updateRestaurant,
    getRestaurantById,
    getRestaurantsByIds,
    addToWelcomePhotos,
    removeFromWelcomePhotos,
    getByAdminId,
}
```