МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МАРІУПОЛЬСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЕКОНОМІКО-ПРАВОВИЙ ФАКУЛЬТЕТ
КАФЕДРА СИСТЕМНОГО АНАЛІЗУ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

До захисту допустити:
В.о. зав. кафедри

Ганна МАРТИНЮК

квітня 2024 р.

РОЗРОБКА ВЕБ-СЕРВІСУ ДЛЯ ОРГАНІЗАЦІЇ ДОСТАВКИ ЗАМОВЛЕНЬ»

Кваліфікаційна робота
здобувача вищої освіти першого (бакалаврського) рівня
освітньо-професійної програми
«Комп'ютерні науки»
Уракова Данила Олександровича
Науковий керівник:
Мнацаканян Марія Сергіївна,
кандидат технічних наук, доцент,
доцент кафедри системного аналізу та інформаційних технологій
Рецензент:
Лукашенко Вікторія Вікторівна,
кандидат технічних наук, доцент,
заступник декана факультету
комп'ютерних наук та технологій
Національного авіаційного університету

Кваліфікаційна робота захищена
з оцінкою відмінно 90 (А)
Секретар ЕК

» червня 2024 р.

Київ - 2024

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи на тему «Розробка веб-сервісу для організації доставки замовлень». Дипломна робота присвячена розробці веб-сервісу для організації доставки замовлень. В умовах стрімкого розвитку цифрових технологій та зростаючих вимог до ефективності логістичних операцій, актуальність даної теми беззаперечна. Робота включає аналітичний, дослідницький та проектний розділи, кожен з яких вирішує відповідні задачі в контексті створення високоефективного сервісу доставки. Робота спрямована на створення зручного і функціонального сервісу, який дасть змогу користувачам ефективно управляти процесом вибору і замовлення продукції з можливістю реального відстеження статусу замовлень. Проєкт передбачає використання сучасних технологічних рішень, як-от React і Node.js, а також інтеграцію з платіжними системами та методами оптимізації логістики через аналіз великих даних і машинне навчання для підвищення рівня клієнтського сервісу й оперативності виконання замовлень.

Основною метою даної роботи є створення зручного і функціонального сервісу, що дозволяє користувачам ефективно управляти процесом вибору товару і оформлення замовлення з можливістю відстежувати статус замовлень в режимі реального часу. Пропоноване рішення спрямоване на підвищення рівня обслуговування клієнтів і оперативності виконання замовлень. Результати досліджень і розробок підтверджують важливість і актуальність створення такого веб-сервісу, який допомагає задовольняти зростаючі потреби ринку і забезпечувати високий рівень задоволеності клієнтів.

SUMMARY

Explanatory note to the qualification work on the topic “Development of a web service for organizing order delivery”. The thesis is dedicated to the development of a web service for organizing order delivery. Given the rapid development of digital technologies and growing demands on the efficiency of logistics operations, the relevance of this topic is undeniable. The work includes analytical, research, and design sections, each of which solves relevant tasks in the context of creating a highly efficient delivery service. The work is aimed at creating a convenient and functional service that will allow users to effectively manage the process of selecting and ordering products with the ability to actually track the status of orders. The project involves the use of modern technological solutions, such as React and Node.js, as well as integration with payment systems and logistics optimization methods through big data analysis and machine learning to improve customer service and order fulfillment.

The main goal of this work is to create a convenient and functional service that allows users to effectively manage the process of product selection and ordering with the ability to track the status of orders in real time. The proposed solution is aimed at improving the level of customer service and the efficiency of order fulfillment. The results of research and development confirm the importance and relevance of creating such a web service that helps to meet the growing needs of the market.

ЗМІСТ

ВСТУП

У сучасному світі, де швидкість життя невпинно зростає, а час стає все більш цінним ресурсом, послуги доставки їжі набувають особливої актуальності. Пандемія COVID-19 значно прискорила перехід споживачів до онлайн-покупок та замовлення їжі через Інтернет. Збільшення попиту на онлайн-сервіси доставки викликало потребу в оптимізації та розширенні цих сервісів. Розробка ефективного веб-сервісу для організації доставки замовлень їжі дозволить задовольнити ростущі потреби ринку та забезпечити високий рівень задоволеності споживачів. Таким чином, розробка веб-сервісу для доставки їжі є актуальним і значущим завданням, що відповідає сучасним тенденціям розвитку ринку послуг.

Метою є розробка веб-сервісу для організації доставки замовлень, який би задовольняв сучасні вимоги користувачів до функціональності, швидкості обробки замовлень та зручності інтерфейсу. Веб-сервіс повинен забезпечити користувачам можливість легко вибирати страви з різних ресторанів, замовляти їх з доставкою на дім або в офіс, а також відстежувати статус свого замовлення в реальному часі.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Провести аналіз ринку доставки їжі, включаючи огляд існуючих веб-сервісів, сучасних трендів та інновацій у цій сфері.
2. Визначити основних конкурентів та проаналізувати їх бізнес-моделі та пропозиції.
3. Розробити технологічні аспекти веб-сервісу, включаючи вибір технологічного стеку, проектування архітектури та визначення функціональних вимог.
4. Створити дизайн користувацького інтерфейсу, який забезпечить зручність та інтуїтивність користування сервісом.

Об’єктом дослідження є створення веб-сервісу для організації доставки замовлень. Предметом дослідження є огляд існуючих сайтів та додатків для доставки: для замовників та для кур’єрів.

Предмет дослідження: Існуючі веб-сервіси для доставки їжі, їх функціональні можливості, технологічні рішення та бізнес-моделі.

Робота складається з вступу, трьох основних розділів, кожен з яких містить підрозділи, висновків та списку використаної літератури. У першому розділі проведено аналіз ринку доставки їжі, сучасних трендів та конкурентного середовища. У другому розділі розглянуто технологічні аспекти розробки веб-сервісу, включаючи вибір технологічного стеку, проектування архітектури та визначення функціональних вимог. Третій розділ присвячено проектуванню веб-сервісу, включаючи розробку дизайну користувацького інтерфейсу (?доповнити?).

РОЗДІЛ 1.
АНАЛІЗ РИНКУ ДОСТАВКИ ЗАМОВЛЕНЬ

1.1 Огляд існуючих веб-сервісів для доставки

В останні роки ринок веб-сервісів для доставки замовлень значно розширився, що сприяло зростанню конкуренції та вдосконаленню технологічних рішень у цій сфері. У 2023 році його обсяг оцінювався приблизно у $254.52 мільярда, а прогнозується, що він досягне $505.50 мільярда до 2030 року, з річним темпом зростання близько 10.3% [1]. Серед ключових факторів, що сприяють цьому зростанню, можна вказати збільшення кількості смартфонів, поліпшення інтернет-зв'язку та зростаючу популярність платформ онлайн-замовлень.

Основні гравці на цьому ринку включають міжнародні та локальні платформи, кожна з яких пропонує унікальні функції для забезпечення зручності та ефективності процесів доставки.

Glovo був заснований в 2014 році в Іспанії і зараз працює в багатьох містах по всьому світу[2]. Цей сервіс пропонує широкий вибір продуктів, включаючи їжу, подарунки, товари останньої хвилини та багато іншого[3]. «Доставка їжі та багато чого іншого» - цитата з головної сторінки їхнього сайту(див.рис.1.2.1). Користувачі можуть замовляти товари з більш ніж 170000 магазинів[2].

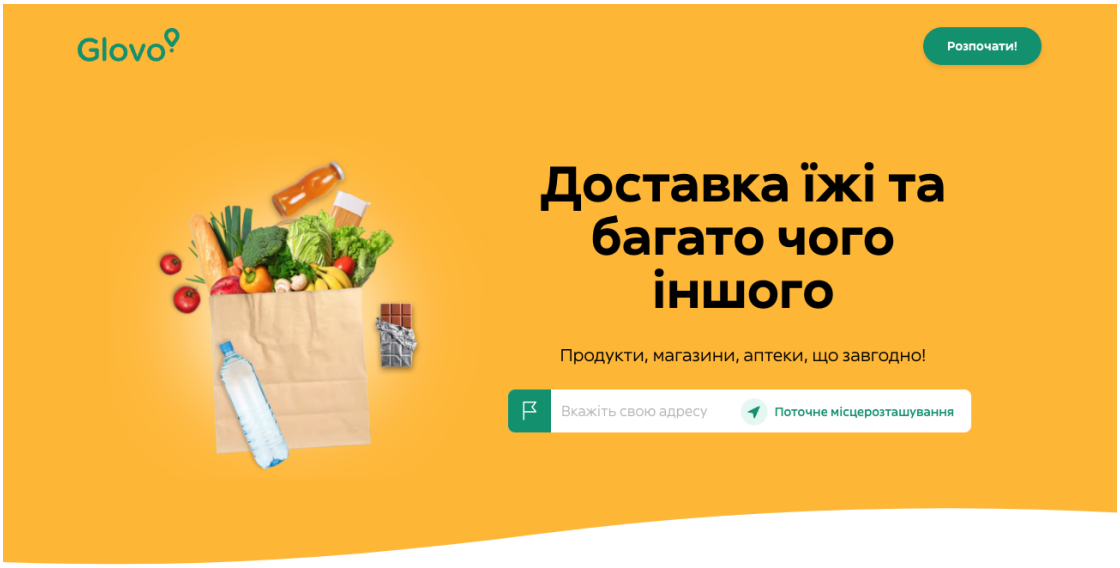

Рис. 1.2.1. Головна сторінка сервісу Glovo

Uber Eats (див.рис.1.2.2) пропонує широкий вибір ресторанів і дозволяє користувачам слідкувати за своїм замовленням в реальному часі[4] Сервіс також пропонує функцію “розумного кураторства”, яка використовує штучний інтелект для рекомендації ресторанів на основі минулих замовлень і поточної інформації[5]

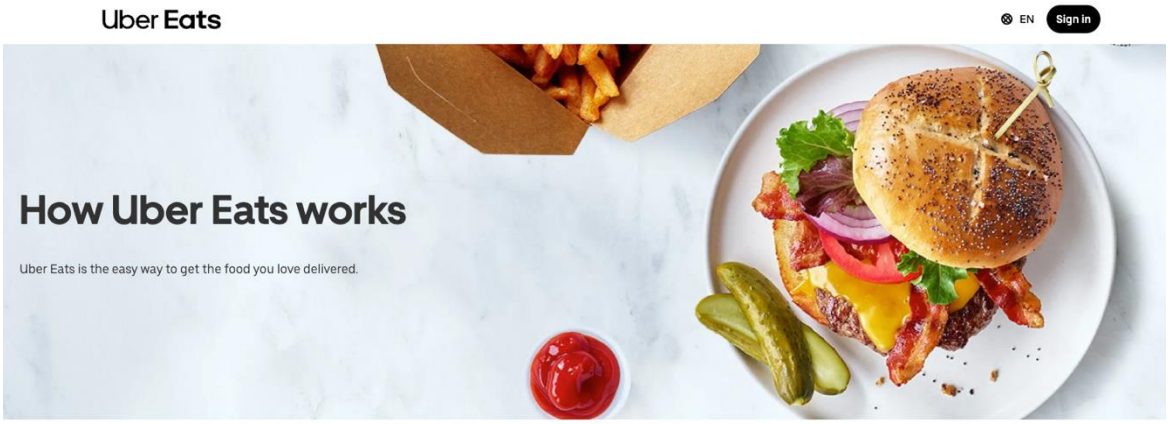

Рис. 1.2.2. Головна сторінка сервісу UberEats

Deliveroo. Ще один сервіс доставки їжі, який акцентує увагу на якості обслуговування та швидкості доставки. Deliveroo відомий своїми стратегічними партнерствами з вищими ресторанами та кафе(див.рис.1.2.3), а також інвестиціями в технологічний розвиток своїх платформ[6]. Сервіс заснований у 2013 році в Лондоні, працює в багатьох країнах, включаючи Великобританію, Францію, Бельгію, Ірландію, Італію, Сінгапур, Гонконг, ОАЕ, Кувейт і Катар[7].

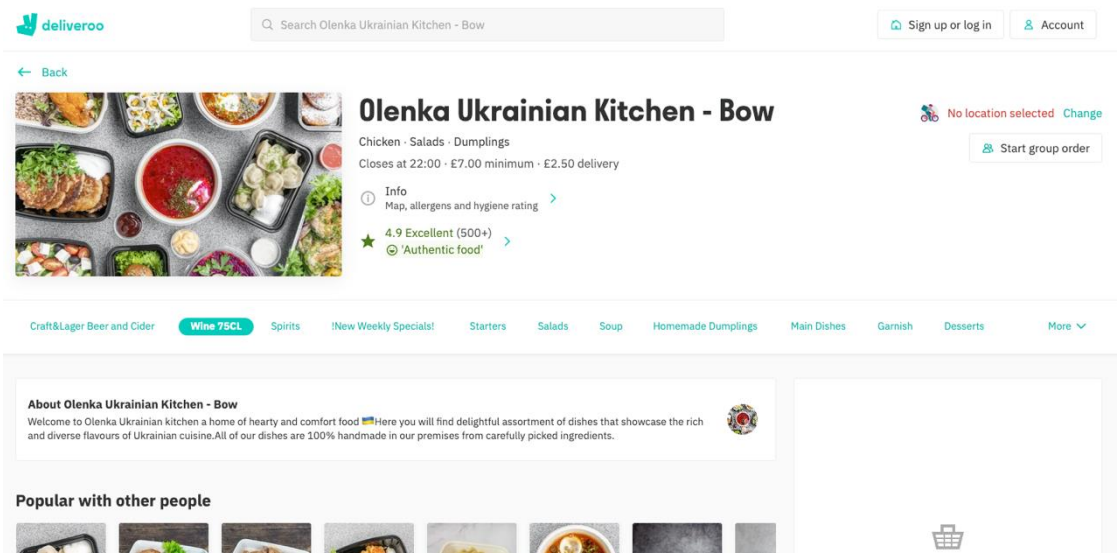

Рис. 1.2.3. Сторінка ресторану у Deliveroo

Окрім традиційних платформ доставки, з'являються нові технологічні стартапи, які використовують штучний інтелект та машинне навчання для оптимізації логістичних ланцюжків і покращення досвіду користувачів[8]. Такі інновації включають передбачення часу доставки з високою точністю, персоналізовані рекомендації та автоматизацію обслуговування клієнтів[9].

Важливо відзначити, що кожен сервіс має свої сильні та слабкі сторони. Деякі платформи можуть пропонувати ширші можливості щодо вибору закладів і товарів, тоді як інші вирізняються кращою якістю підтримки користувачів або швидшою доставкою.

1.2 Сучасні тренди та інновації в сфері доставки їжі

Сучасна сфера доставки їжі постійно розвивається, залучаючи новітні технології для покращення якості обслуговування. Використання мобільних додатків і платформ є не лише зручністю для клієнтів, а й могутнім інструментом для ресторанів, що дозволяє їм збільшити свої доходи і покращити впізнаваність. Наприклад, незважаючи на швидкий розвиток сектору, багато платформ доставки продовжують зазнавати збитків, але вони відкривають нові можливості для зростання за рахунок впровадження нових категорій товарів, таких як алкоголь та продукти [10]

Штучний інтелект (AI) відіграє ключову роль у оптимізації логістики доставки, прогнозуванні обсягів замовлень та персоналізації досвіду споживачів, що стає все більш значущим у сучасному світі доставки [11]

Доставка за допомогою дронів та роботів-кур'єрів широко використовується у великих містах, дозволяючи здійснювати швидкі та ефективні доставки без необхідності втручання людей, що особливо корисно для густонаселених або важкодоступних районів [12]

Одним із останніх трендів є зосередження на екологічності послуг доставки, що включає використання біорозкладних матеріалів для упаковки та застосування електротранспорту, який допомагає знижувати вплив на довкілля. Це відображає зростаючу увагу до сталого розвитку та екологічної відповідальності як з боку бізнесу, так і з боку споживачів [13]

1.3 Аналіз основних конкурентів та їх пропозицій

Розуміння конкурентного середовища на ринку доставки їжі є ключовим для успішної роботи веб-сервісу. До основних гравців на ринку належать як місцеві, так і міжнародні компанії, кожна з яких пропонує унікальні послуги та маркетингові стратегії. Наприклад, DoorDash та Uber Eats є лідерами у формуванні партнерств з великими ресторанними мережами, що допомагає їм розширювати свою присутність і збільшувати лояльність клієнтів через спеціальні пропозиції та знижки [14]

Основні конкуренти, такі як Domino's Pizza і PizzaExpress, відомі своєю сильною маркою та інноваціями у сфері доставки. Domino's, наприклад, має дуже ефективну систему доставки і широкий асортимент меню, що дозволяє їм задовольняти різноманітні клієнтські вподобання[15]

Вивчення відгуків клієнтів на платформах соціальних медіа і оглядових сайтах виявляє сильні та слабкі сторони конкурентів, що може допомогти у вдосконаленні власних сервісів. Наприклад, незадоволення клієнтів якістю продукції або обмеженими можливостями меню може стати ареною для інновацій і покращень.

Технологічні інновації, які використовують конкуренти, включають використання штучного інтелекту для оптимізації маршрутів доставки та впровадження дронів для покращення логістики. Ці інновації можуть надати компаніям конкурентну перевагу на ринку[16]

1.4 Визначення цільової аудиторії веб-сервісу

Ефективне визначення та аналіз цільової аудиторії є ключовим фактором успіху будь-якого веб-сервісу, особливо у галузі доставки їжі. Розуміння хто є потенційними клієнтами, їх потреби, переваги та поведінка допомагає приймати обгрунтовані рішення щодо маркетингу, дизайну продукту та стратегії обслуговування. Ключові аспекти визначення цільової аудиторії для веб-сервісу доставки замовлень(див.табл. 1.4.1):

Таблиця 1.4.1. Ключові аспекти цільової аудиторії

Категорія	**Параметри**	**Приклади**
Демографічні характеристики	Вік, стать, освіта, професія, сімейний статус	Молоді, працевлаштовані, 18-34 років, вища освіта
Психографічні характеристики	Цінності, інтереси, стиль життя	Швидке обслуговування, технологічність
Географічне розташування	Регіон	Міські райони, великі міста
Поведінкові фактори	Частота замовлень, улюблені типи їжі	Часті замовлення на вечерю, люблять італійську кухню

Демографічні характеристики: Зосередження на молодих та працевлаштованих у великих містах, які цінують швидкість та зручність доставки. Статистика DoorDash показує, що особливо активні у використанні послуг доставки молоді люди віком від 18 до 34 років[17]

Психографічні характеристики: Важливо зрозуміти цінності, інтереси та спосіб життя споживачів. До прикладу, покоління мілленіалів може віддавати перевагу технологічність та швидкість. [17].

Географічне розташування: Поняття, де користувачі фізично знаходяться, допоможе адаптувати послуги до специфіки різних регіонів, наприклад, враховувати різницю в потребах між мешканцями міст та сільських районів[18]

Поведінкові фактори: Аналізування частоти замовлень, улюблених типів їжі, часу доставки, та відданості брендам може надати важливу інформацію про звички та переваги вашої цільової аудиторії. Використання інструментів аналітики та соціальних мереж може допомогти в отриманні цих даних.

Висновки до розділу 1

У першому розділі було проведено глибокий аналіз ринку послуг доставки їжі, що охоплює декілька важливих аспектів: від загального стану та тенденцій на ринку до детального дослідження конкурентного середовища та визначення цільової аудиторії. Аналіз сучасних трендів та інновацій виявив, що ринок доставки їжі стрімко розвивається за рахунок технологічних нововведень, таких як мобільні додатки, використання штучного інтелекту та автоматизованих систем доставки. Ці зміни спрямовані на підвищення ефективності послуг та покращення користувацького досвіду.

Ретельний аналіз конкурентів дозволив виявити ключові стратегії та унікальні пропозиції, що використовуються на ринку, що є важливим для розробки стратегій ведення власного бізнесу. Вивчення цільової аудиторії

підтвердило необхідність глибокого розуміння потреб і переваг клієнтів для ефективного позиціонування на ринку і максимальної адаптації пропозицій.

Таким чином, перший розділ закладає фундамент для подальшого розроблення веб-сервісу, надаючи важливі дані для планування продукту, маркетингу та стратегій залучення клієнтів. З урахуванням отриманих даних можна ефективно вирішувати завдання проектування і розвитку веб-сервісу, що буде розглянуто в наступних розділах.

РОЗДІЛ 2.
ТЕХНОЛОГІЧНІ АСПЕКТИ РОЗРОБКИ ВЕБ-СЕРВІСУ

2.1. Основні принципи розробки веб-застосувань

Структура веб-сайту розробляється відповідно до клієнт-серверної архітектури та застосування MVC (Model-View-Controller) патерну. Браузер як клієнт ініціює запит до сервера, який, у свою чергу, надає відповідь (див. рис. 2.1.1). В якості клієнта може виступати будь-який пристрій користувача, такий як комп'ютер, смартфон або інший гаджет. Цей процес взаємодії можна відобразити наступним чином(рис.2.1.1):

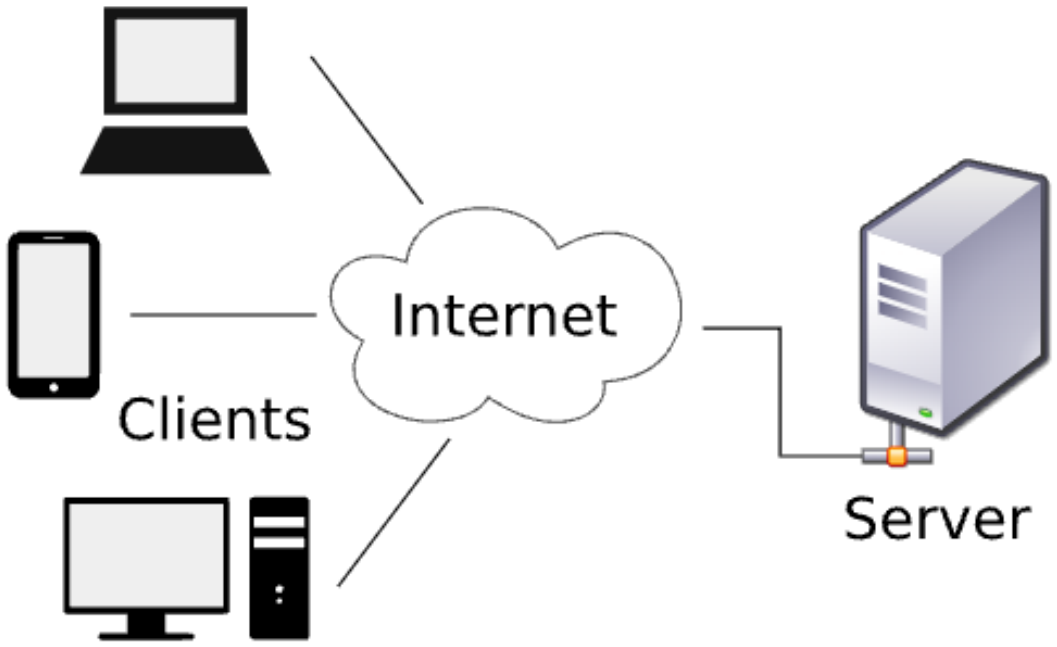

Рис 2.1.1. Схема клієнт-серверної архітектури

MVC, що розшифровується як Model View Controller, є шаблоном проектування та розробки програмного забезпечення [19]. Основними компонентами програми(див.рис.2.1.2) є моделі, контролери та представлення, кожен із яких представляє собою групу файлів.

Контролер діє як логічний центр додатку, створюючи зв'язки між моделлю та представленням. Він керує URL-адресами та діями, які можуть

бути виконані над об'єктами, або ж тими, що виконують об'єкти, наприклад обробку запиту на адресу domain_name/shops/2. Контролерні файли часто містять умовні оператори або конструкції switch-case.

Представлення визначає візуальний інтерфейс програми, тобто те, як програму сприймає користувач.

Модель забезпечує взаємодію з базою даних.

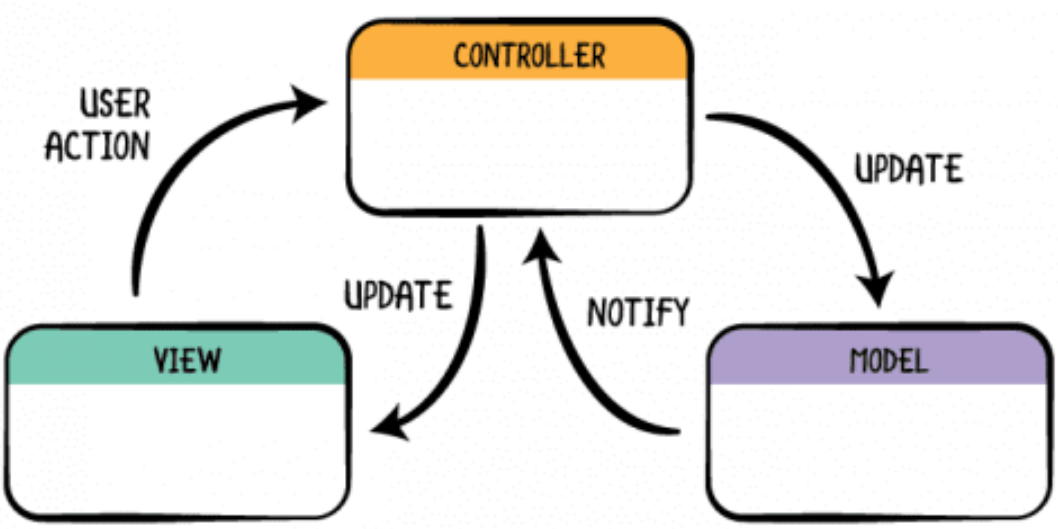

Рис 2.1.2. Принцип роботи MVC – паттерну

Життєвий цикл веб-додатка починається із введення користувачем URL-адреси в браузер, який направляє запит до контролера. Контролер взаємодіє з моделлю, яка звертається до бази даних. Після отримання даних, інформація передається з бази даних у модель, з моделі до контролера, а потім з контролера до представлення. Нарешті, представлення відображається в браузері через контролер.

Контролер виступає в ролі посередника між користувачем та базою даних, оскільки безпосередній доступ до бази вважається небезпечною практикою. Він не тільки розподіляє навантаження на систему, але й підвищує рівень безпеки даних. Об’єкти, що представляють дані, мають властивості, які відображають атрибути сутностей бази даних, та методи для їх обробки.

Основні операції, що виконуються за запитом контролера, включають вставку, вибірку, видалення та оновлення даних. Ці дії відомі під абревіатурою

CRUD [20] — create, read, update, delete, які є фундаментальними для управління даними.

Для ілюстрації, як ці операції відображаються на запити SQL та HTTP, нижче представлено таблицю [21] (див. рис.2.1.3).

Operation	SQL	HTTP
Create	INSERT	PUT / POST
Read (Retrieve)	SELECT	GET
Update (Modify)	UPDATE	PUT / POST / PATCH
Delete (Destroy)	DELETE	DELETE

Рис 2.1.3. Співвідношення операцій та запитів

2.2 Вибір технологічного стеку для розробки веб-сервісу

Для розробки ефективного та сучасного веб-сервісу важливо обрати технологічний стек, який би відповідав сучасним вимогам ринку та потребам користувачів. У випадку нашого сервісу для доставки їжі, ми обираємо комбінацію React для фронтенду та Node.js для бекенду з наступних причин:

React (Клієнт):

- ***Компонентний підхід:*** React дозволяє створювати перевикористовувані UI компоненти, що спрощує розробку та підтримку інтерфейсу.
- ***Швидкість та ефективність:*** Використання віртуального DOM дозволяє React швидко вносити зміни у веб-інтерфейс без перезавантаження сторінки, що забезпечує високу швидкість роботи сервісу.
- ***Сильна спільнота та підтримка:*** React має велику спільноту розробників та багатий набір готових рішень, що допомагає швидко знаходити відповіді на виникаючі питання та інтегрувати нові бібліотеки.

Node.js (Сервер):

- ***Єдиний мовний стек:*** Використання JavaScript на фронтенді та бекенді спрощує розробку, оскільки команда може використовувати одну мову на всіх етапах розробки.
- ***Неблокуюча асинхронна архітектура:*** Node.js використовує асинхронні операції вводу-виводу, що дозволяє обробляти велику кількість запитів без затримок.
- ***Масштабованість:*** Node.js ефективно масштабується завдяки своїй подієво-орієнтованій архітектурі, що дуже важливо для високонавантажених систем, таких як веб-сервіси доставки їжі.

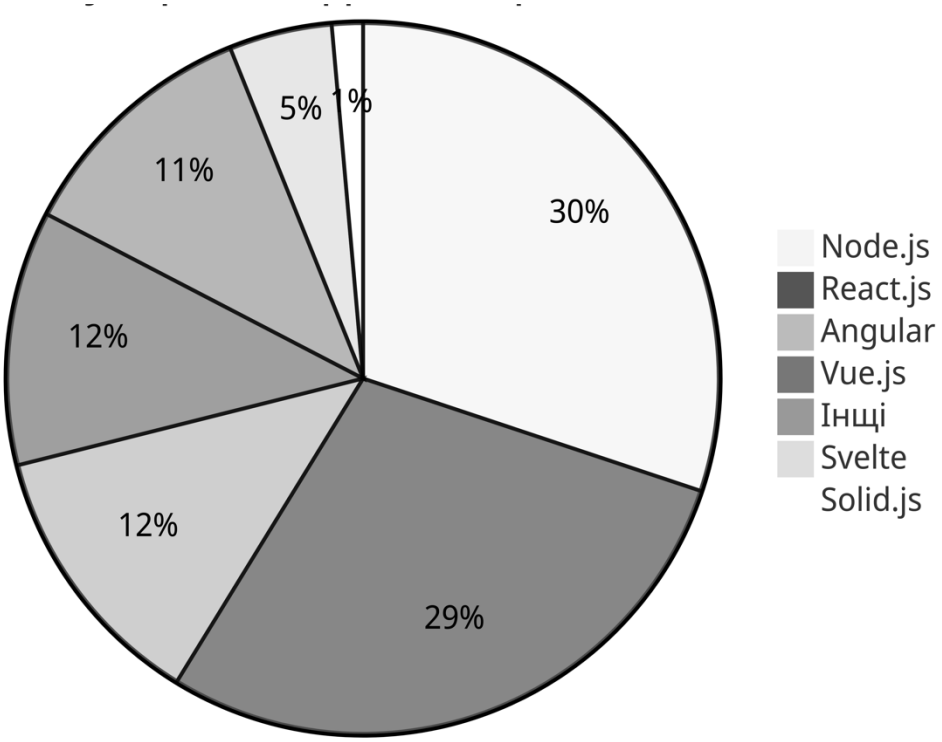

Рис. 2.2.1. Популярні веб-фреймворки 2023[22]

Ці технології забезпечать не тільки високу продуктивність та швидкість роботи сервісу, але й дадуть можливість гнучко реагувати на зміни вимог та масштабувати проект з ростом клієнтської бази. Вибір цих технологій є

відображенням сучасних тенденцій у розробці веб-додатків і забезпечить довгострокову підтримку та розвиток веб-сервісу.

2.3 Проектування архітектури веб-сервісу

При проектуванні архітектури веб-сервісу для доставки їжі, особлива увага приділяється масштабованості, безпеці та високій доступності. Мікросервісна архітектура вважається оптимальною для досягнення цих цілей через свою гнучкість та можливість швидкої адаптації до змін, дозволяючи незалежне розгортання і тестування компонентів. Такий підхід дозволяє кожному мікросервісу виконувати свою конкретну бізнес-функцію, сприяючи більш ефективній розробці та легшому впровадженню нових функцій без впливу на решту системи [24].

Використання NoSQL баз даних, таких як MongoDB(див.табл.2.2.1), забезпечує швидкий доступ до даних і їхнє масштабування, особливо в умовах великої кількості неструктурованої інформації. Кешування даних з використанням рішень, як-от Redis, допомагає знижувати навантаження на бази даних, підвищуючи швидкість відгуку системи[25].

Для інтеграції з зовнішніми системами, такими як платіжні шлюзи або сервіси логістики, використовуються відповідні API. Це сприяє автоматизації процесів обробки транзакцій та відстеження статусу доставок. Інтеграція таких сервісів з мікросервісною архітектурою дозволяє забезпечити високу надійність і доступність сервісу[25].

У сфері безпеки важливо впровадити надійні механізми аутентифікації та авторизації, наприклад, використовуючи стандарти, такі як OAuth(див.табл.2.2.1). Це забезпечує захист користувацьких даних від несанкціонованого доступу. Також критично важливим є шифрування даних, які передаються та зберігаються, для забезпечення їх конфіденційності [25].

Клієнтська частина має включати адаптивний дизайн, що забезпечує коректне відображення та зручність користування на різних пристроях. Такий підхід дозволяє користувачам вик овачувати на веб-сайтах і мобільних

додатках. Застосування технологій, таких як CSS Media Queries, сприяє адаптації контенту до різних розмірів екранів, що підвищує зручність користувачів і покращує їхній досвід користування [25].

Табл. 2.2.1. Приклад технологій та пояснення їх вибору [23]

Мікросервіс	**Технологія**	**Обґрунтування вибору**
Управління замовленнями	MongoDB	Гнучкість у схемах даних, масштабованість, висока швидкість обробки великих об'ємів даних
Авторизація	OAuth 2.0	Стандарт безпеки для делегування доступу, широко підтримується
Кешування	Redis	Висока швидкість, підтримка різних типів даних, використовується для зниження навантаження на бази даних
Логістика	RabbitMQ	Надійне та ефективне міжсервісне взаємодія, підтримка асинхронної обробки повідомлень
Платіжні системи	Stripe API	Висока безпека транзакцій, простота інтеграції, широка підтримка валют та платіжних методів

Загалом, проектування архітектури веб-сервісу для доставки замовлень з використанням мікросервісної архітектури дозволяє досягти високої гнучкості, швидко вносити зміни і забезпечувати безперервну роботу системи, навіть у випадку відмов окремих сервісів(див.табл.2.2.1). Такий підхід визнаний ефективним багатьма великими компаніями, які успішно впроваджують мікросервіси для підтримки своїх глобальних платформ, таких як Netflix і Amazon [26].

2.4 Розробка функціональних вимог до веб-сервісу

Функціональні вимоги визначають, що саме повинен робити веб-сервіс і які задачі вирішувати для своїх користувачів. Розробка цих вимог є важливим етапом, що забезпечує зрозумілість цілей проекту для всієї розробницької команди і замовника. Ось основні функціональні вимоги для веб-сервісу доставки їжі(див.рис.2.4.1):

Інтерфейс користувача:

- ***Реєстрація та авторизація користувачів***: Можливість створення особистого профілю з авторизацією через електронну пошту, соціальні мережі чи телефон.
- ***Перегляд меню***: Наявність зручного та інтуїтивно зрозумілого інтерфейсу для перегляду різноманітних страв та напоїв, які пропонує сервіс.
- ***Функція пошуку та фільтрації***: Можливість швидкого пошуку страв за категоріями, інгредієнтами, ціною або іншими параметрами.

Управління замовленнями:

- ***Корзина для покупок***: Функціональність для додавання страв у корзину та редагування замовлення перед оформленням.
- ***Оформлення замовлення***: Можливість вибору способу доставки, введення адреси доставки та вибору часу доставки.
- ***Система оплати***: Інтеграція з безпечними платіжними шлюзами для обробки онлайн-платежів через банківські картки, електронні гаманці та інші платіжні системи.

Адміністрування веб-сервісу:

- ***Управління контентом***: Можливість додавання, редагування та видалення інформації про страви, включаючи фотографії, описи, ціни та інгредієнти.
- ***Моніторинг замовлень***: Інструменти для перегляду активних, завершених та відмінених замовлень, а також управління статусами замовлень.
- ***Звітність та аналітика***: Функціональність для генерації звітів про продажі, популярні страви, зворотній зв'язок клієнтів і ефективність маркетингових акцій.

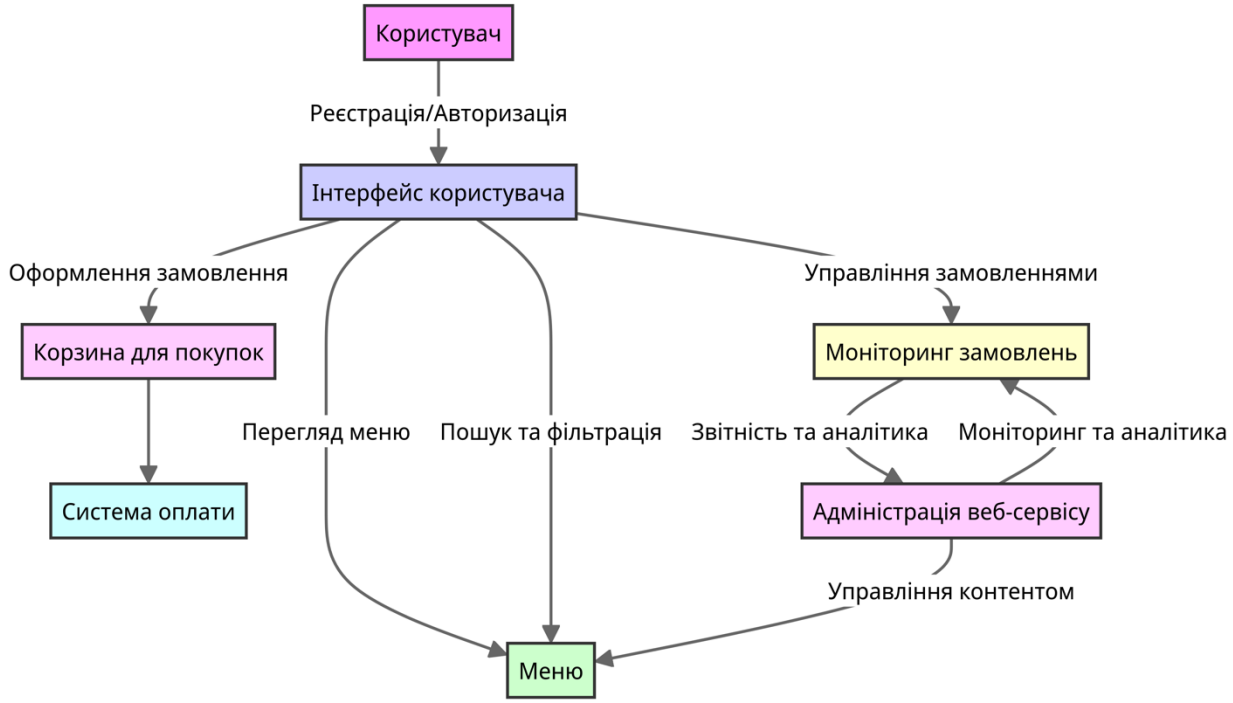

Рис.2.4.1. Основні компоненти та їх взаємозв'язки в рамках веб-сервісу доставки замовлень.

Ці функціональні вимоги визначають основні компоненти системи та відображають зобов'язання сервісу перед користувачами і адміністраторами. Вони повинні бути чітко сформульовані та документовані з урахуванням усіх аспектів користувацького досвіду та технічної експлуатації веб-сервісу. Прозоре визначення цих вимог забезпечує ефективне спілкування між розробниками, стейкхолдерами та користувачами, що є критично важливим для успішного розгортання та експлуатації системи[27].

РОЗДІЛ 3.
ПРОЕКТУВАННЯ ВЕБ-СЕРВІСУ ДЛЯ ДОСТАВКИ ЇЖІ

3.1. Архітектура веб-сервісу

Архітектура веб-сервісу для доставки їжі базується на мікросервісній архітектурі, яка забезпечує гнучкість, масштабованість та ефективність системи. Основні компоненти системи включають фронтенд, бекенд, бази даних та інтеграційні сервіси (див.рис.3.1.1).

Основні компоненти системи включають:

- ***Фронтенд***, який відповідає за відображення інтерфейсу користувачів.
- ***Бекенд***, який обробляє логіку програми, включно з замовленнями, управлінням даними користувачів та інтеграцією з внутрішніми і зовнішніми API.
- ***База даних***, яка забезпечує зберігання всіх необхідних даних, таких як інформація про користувачів, замовлення, меню, тощо.

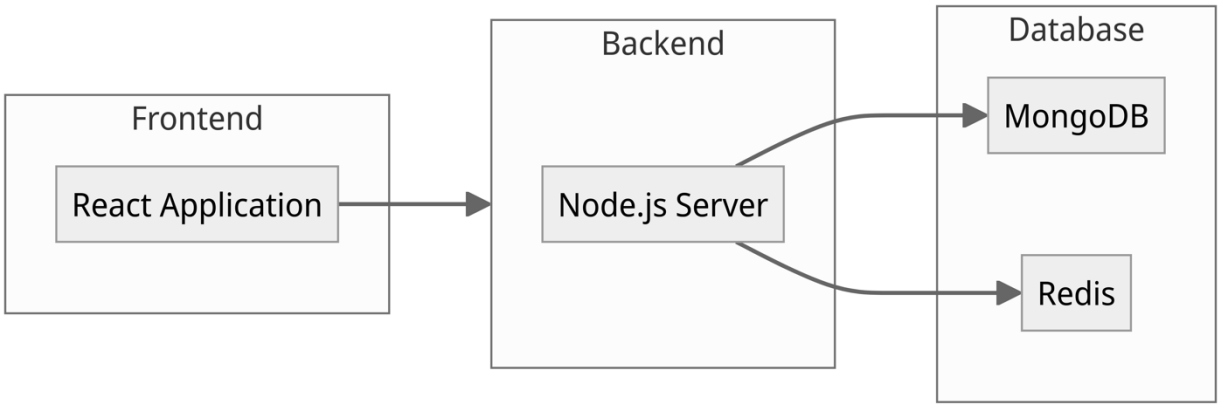

Рисунок 3.1.1.

Таким чином ми отримуємо організовуємо систему, яка взаємодіє з різними середовищами (див.рис.3.1.2).

Фронтенд розробляється на основі React, завдяки його ефективності у створенні інтерактивних користувацьких інтерфейсів. Бекенд будується на Node.js, що дозволяє використовувати JavaScript для всіх складових проекту, що спрощує розробку і підтримку. Для баз даних використовується MongoDB через її гнучкість у схемах даних і відмінну підтримку великих об'ємів неструктурованої інформації.

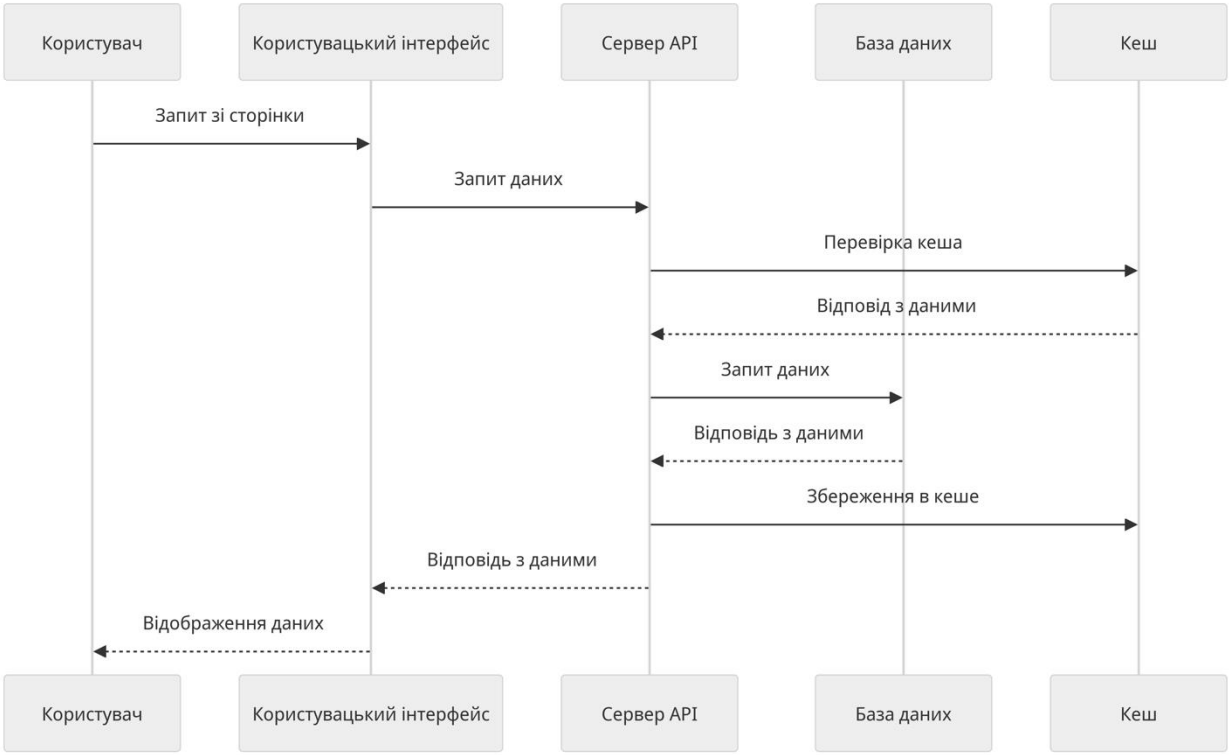

Рисунок 3.1.2.

Основою архітектури фронтенду є бібліотека React, яка дозволяє створювати інтерактивні та динамічні користувацькі інтерфейси. Використання React сприяє модульному підходу в розробці, де інтерфейс розділяється на незалежні компоненти, що можуть бути легко перевикористані та тестовані окремо.

Ключові аспекти фронтенд архітектури:

1. **Компонентний підхід:** Вся інтерфейсна логіка і представлення розділені на менші компоненти (кнопки, форми, списки тощо), що дозволяє легше управляти станами і взаємодією в межах аплікації(див.рис.3.1.3). Компоненти React пишуться як функціональні або класові компоненти з використанням хуків для управління станом та життєвим циклом.
2. **SEO-оптимізація:** Пошукові системи не зможуть коректно розпізнавати весь вміст сайту через особливість рендерінгу веб-сторінки React бібліотекою, тому ми будемо використовувати фреймворк Next.js, який дозволить серверу відправляти браузеру сторінку з вже готовим HTML.
3. **Стан додатку:** Управління станом (state management) є критичним для динамічних веб-сервісів. Використовується Redux або Context API для глобального управління станом, що дозволяє забезпечити синхронізацію даних між компонентами без прямої передачі властивостей(див.рис.3.1.4).
4. **Маршрутизація:** Для навігації між сторінками(див.рис.3.1.6) і збереження стану UI використовується бібліотека React Router. Це дозволяє виконувати лазивне завантаження компонентів, оптимізуючи час завантаження і ресурси браузера.
5. **Запити до сервера:** Взаємодія з сервером для отримання або відправки даних здійснюється через асинхронні запити, використовуючи Axios. Це забезпечує роботу з промісами і асинхронний контроль над станами завантаження, помилок та результатів(див.рис.3.1.5).
6. **Адаптивний дизайн:** Стилізація з використанням CSS-in-JS бібліотеки, такої як Styled Components або Emotion, забезпечує кастомізацію компонентів під різні розміри екранів і пристроїв без

використання зовнішніх CSS файлів. Такий підхід забезпечує ізоляцію стилів, уникнення конфліктів і легшу підтримку.

7. **Безпека:** Впровадження механізмів аутентифікації і авторизації, таких як OAuth 2.0 або JWT (JSON Web Tokens), для забезпечення безпечного доступу до ресурсів та обмеження несанкціонованого доступу.

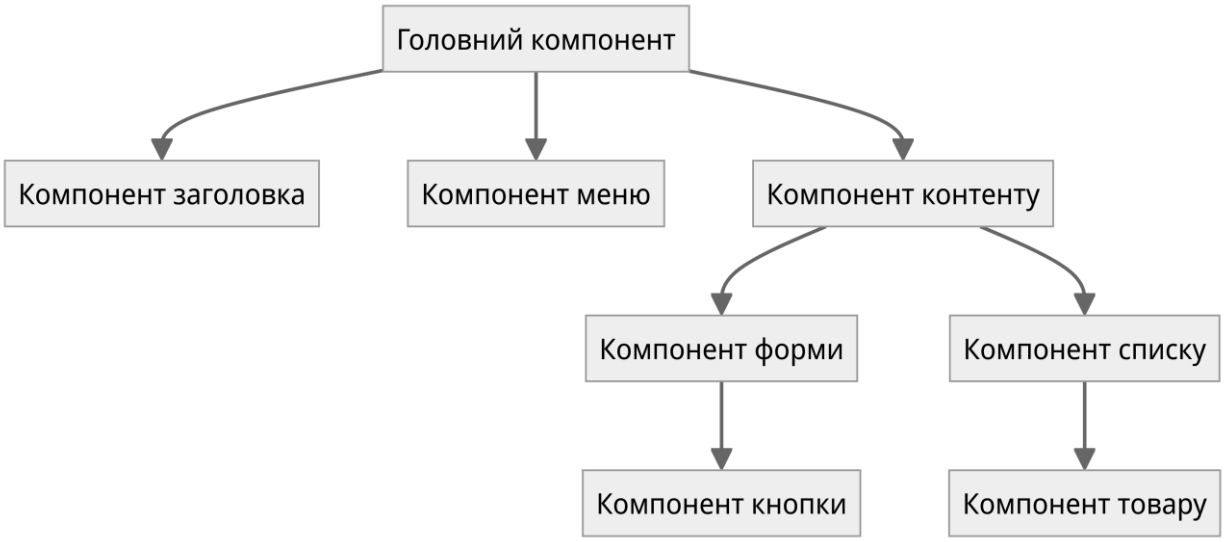

Рисунок 3.1.3.

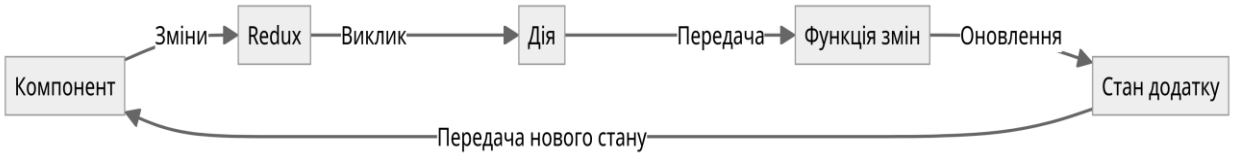

Рисунок 3.1.4.

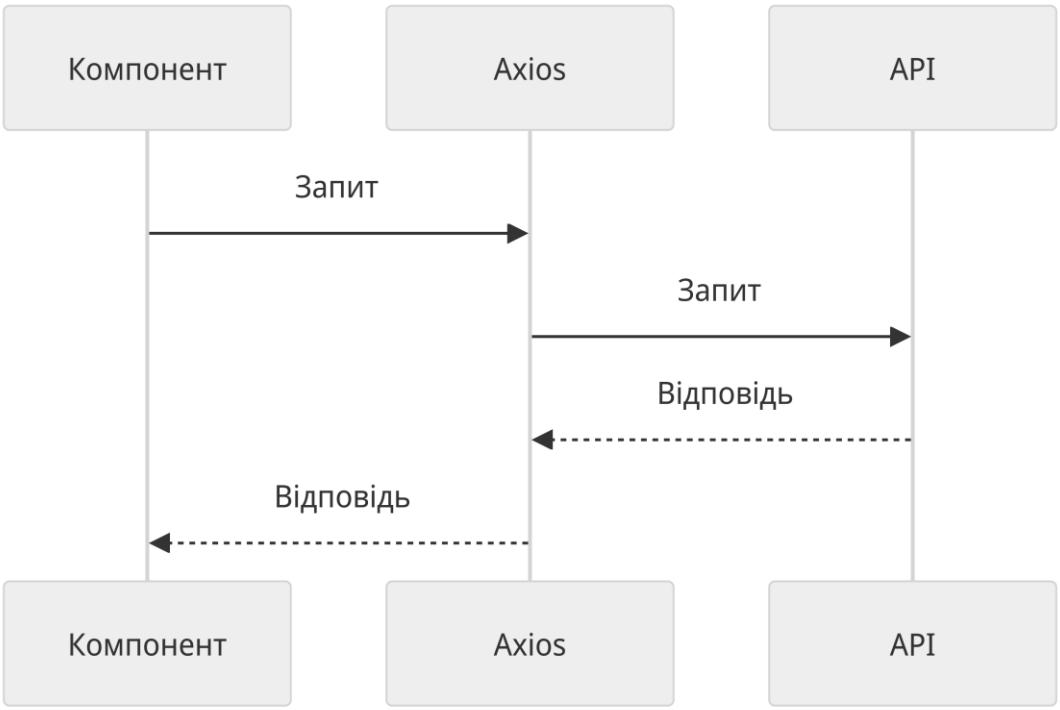

Рисунок 3.1.5.

Розробка фронтенду розпочинається з планування архітектури додатку та дизайну користувацького інтерфейсу. Основні етапи включають:

1. ***Компонентний дизайн:*** Визначення структури компонентів за допомогою бібліотеки React. Компоненти плануються таким чином, щоб вони були незалежними і перевикористовуваними, що сприяє модульності і легкості тестування(див.рис.3.1.3).
2. ***Створення макетів:*** Прототипування макетів інтерфейсу для візуалізації взаємодії користувача з сервісом. Використання інструментів типу Figma для створення деталізованих макетів, які будуть перетворені в компоненти React.
3. ***Розробка з використанням Next.js:*** Налаштування SSR(Server-side-rendering) для оптимальної SEO-оптимізації. Конфігурація роутінгу з динамічними маршрутами для підтримки SEO-дружніх URL(див.рис.3.1.6).
4. ***Розробка компонентів:*** Створення базових компонентів (кнопки, форми, картки продуктів). Розробка складніших компонентів, що об’єднують кілька базових компонентів.

5. ***Інтеграція з бекендом:*** Впровадження асинхронних запитів до API за допомогою фреймворку Axios. Створення сервісів для обробки даних, отриманих від API(див.рис.3.1.5).
6. ***Тестування:*** Перевірка взаємодії між компонентами. Відлагодження коду і оптимізація продуктивності.
7. ***Деплой та підтримка:*** Деплой додатку на сервер і перевірка його роботи в реальних умовах. Отримання зворотнього зв'язку від користувачів та внесення змін.

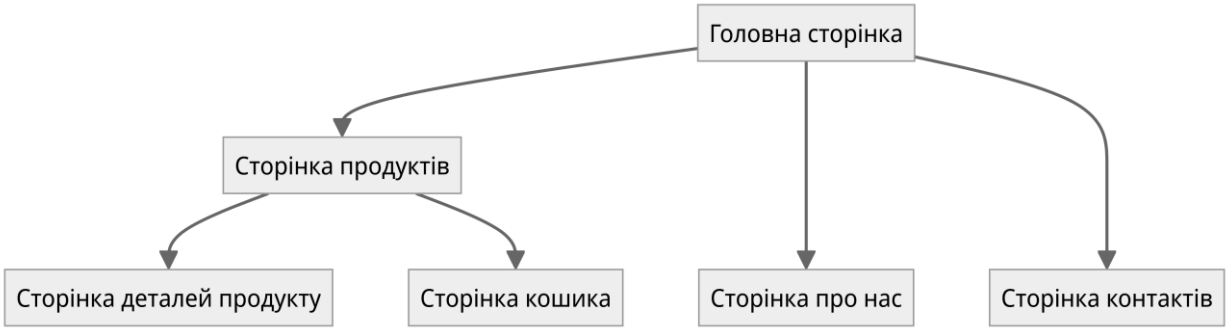

Рисунок 3.1.6

Компоненти React взаємодіють між собою за допомогою пропсів і контексту, що забезпечує передачу даних і станів з верхніх рівнів до нижніх. Запити до сервера виконуються асинхронно з використанням Axios, що дозволяє обробляти дані в реальному часі без перезавантаження сторінки.

Маршрутизація в Next.js забезпечує легку навігацію між сторінками, що дозволяє користувачам швидко переміщатися по додатку. Серверний рендеринг забезпечує попереднє завантаження даних і їх рендеринг на сервері, що покращує SEO і швидкість завантаження сторінок.

Бекенд веб-сервісу також розробляється з використанням сучасних технологій для забезпечення високої продуктивності та надійності. Основні компоненти архітектури бекенду включають:

- Node.js: Основна платформа для розробки серверної частини. Node.js дозволяє використовувати JavaScript на сервері, що забезпечує єдність мовного стека та високу продуктивність завдяки неблокуючій архітектурі.

- Express: Фреймворк для розробки веб-додатків на Node.js. Express забезпечує простий та гнучкий інтерфейс для створення API та управління маршрутизацією(див.рис.3.1.7).
- MongoDB: База даних NoSQL, яка використовується для зберігання даних про замовлення, користувачів та ресторани. MongoDB забезпечує гнучкість у схемах даних та високу швидкість обробки запитів(див.рис.3.1.7).
- Mongoose: Бібліотека для роботи з MongoDB у Node.js. Mongoose забезпечує об'єктно-документне відображення (ODM), що дозволяє зручно працювати з документами у базі даних(див.рис.3.1.7).
- JWT (JSON Web Tokens): Технологія для аутентифікації та авторизації користувачів. JWT дозволяє створювати безпечні токени для керування доступом до ресурсів(див.рис.3.1.7).

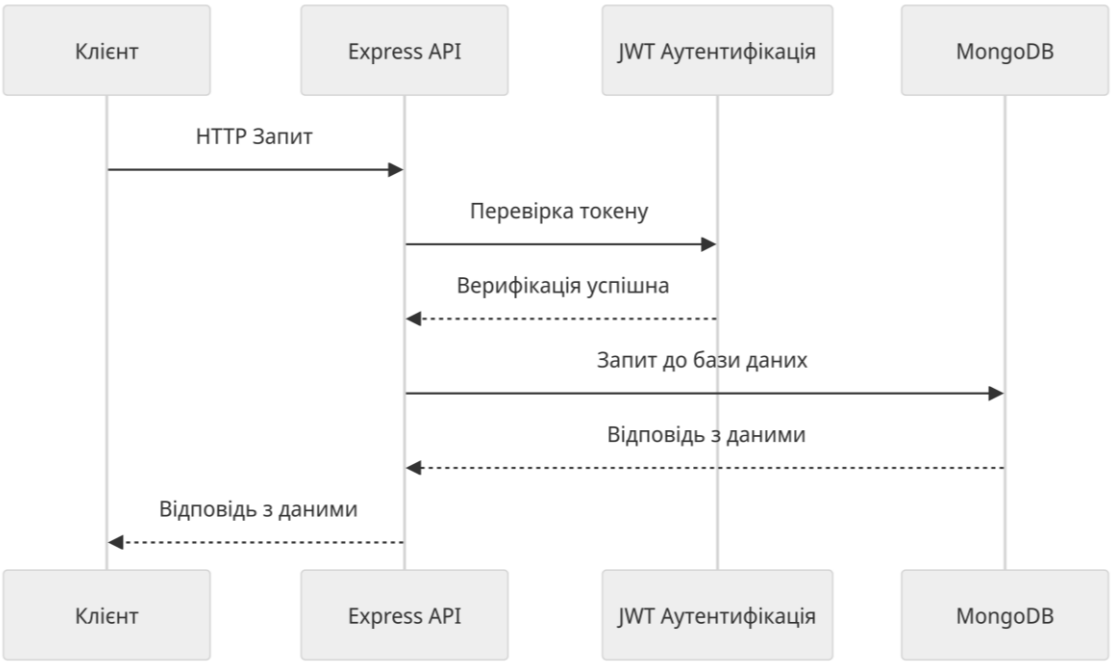

Рисунок 3.1.7.

Бекенд структура розроблена з використанням мікросервісної архітектури, де кожен мікросервіс відповідає за певну бізнес-логіку (наприклад, управління замовленнями, аутентифікація користувачів, обробка

платежів). Ці мікросервіси взаємодіють один з одним через RESTful API, забезпечуючи високу масштабованість та надійність системи.

3.2 Дизайн користувацького інтерфейсу

Розробка дизайну користувацького інтерфейсу базується на принципах зручності та інтуїтивності. Вибір кольорової гами, шрифтів та компоновки елементів інтерфейсу здійснюється на основі психології споживача та аналізу статистики використання подібних сервісів. Основні аспекти стратегії дизайну включають:

- **Вибір кольорів:** Використання приємних та ненав'язливих кольорів, що викликають довіру та комфорт у користувачів.
- **Шрифти:** Використання чітких та легко читабельних шрифтів для забезпечення зручності сприйняття інформації.
- **Компоновка елементів:** Логічне розташування елементів інтерфейсу для забезпечення інтуїтивного користувацького досвіду.

Прототипування є важливим етапом у розробці користувацького інтерфейсу, оскільки дозволяє візуально представити концепцію дизайну та перевірити її на зручність і функціональність(див.рис.3.1.8). Основні прототипи сторінок включають:

- **Головна сторінка:** Вітальна сторінка з коротким описом сервісу, навігаційним меню та основними функціями.
- **Сторінка меню:** Інтерфейс для перегляду та вибору страв, з можливістю фільтрації за категоріями та цінами.
- **Сторінка замовлення:** Кошик для перегляду обраних страв, оформлення замовлення та вибору способу оплати.
- **Сторінка відстеження замовлення:** Інтерфейс для відстеження статусу замовлення в режимі реального часу.

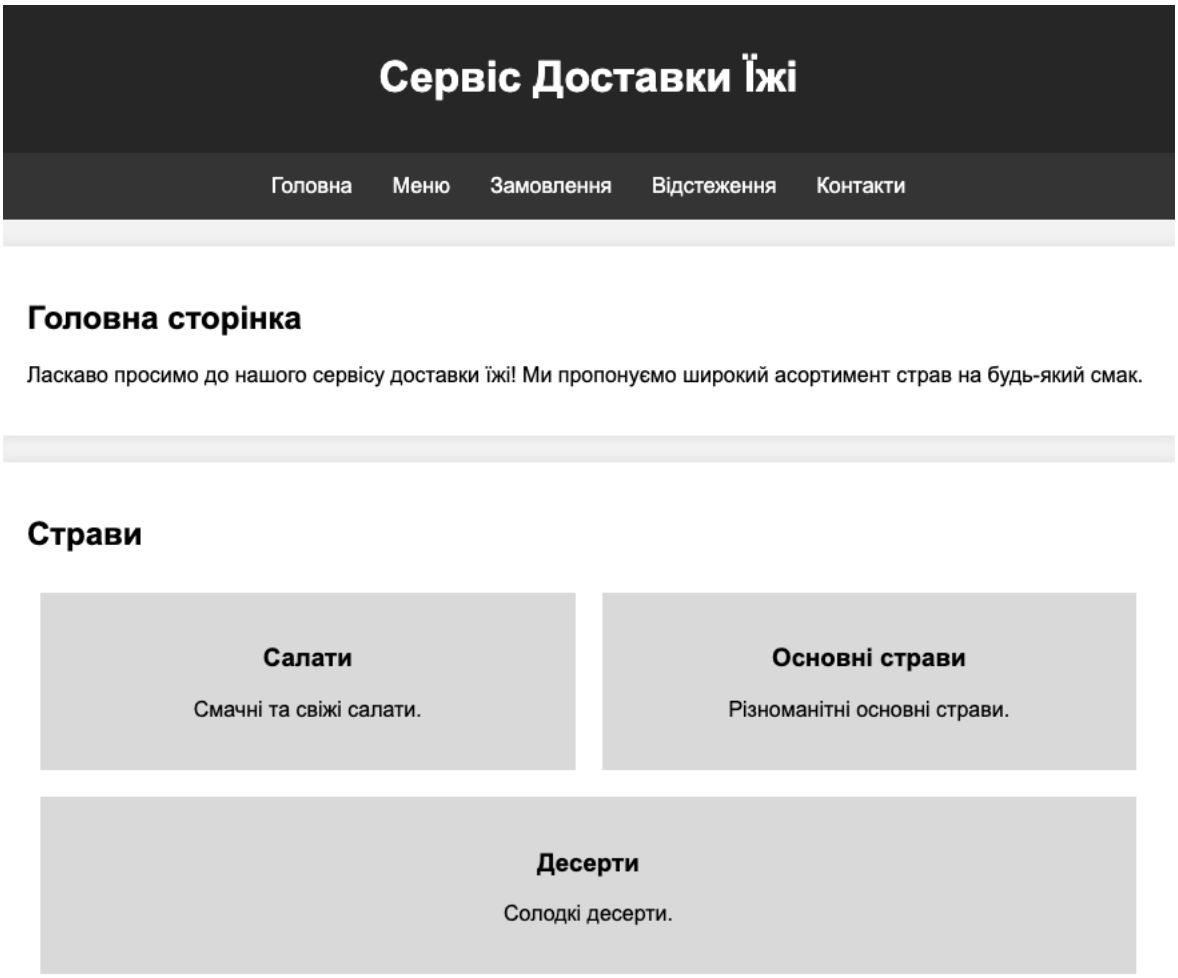

Рисунок 3.1.8. Прототип головної веб-сторінки для комп’ютерів.

Адаптивний дизайн забезпечує коректне відображення веб-сервісу на різних пристроях та платформах(див.рис.3.1.9). Це включає:

- **Мобільна версія:** Оптимізація інтерфейсу для мобільних пристроїв з урахуванням обмеженого екранного простору.
- **Доступність:** Забезпечення доступності для користувачів з обмеженими можливостями, включаючи підтримку навігації за допомогою клавіатури та використання спеціальних підказок для екранних зчитувачів.
- **Тестування:** Перевірка адаптивності дизайну на різних пристроях та браузерах для забезпечення узгодженого користувацького досвіду.

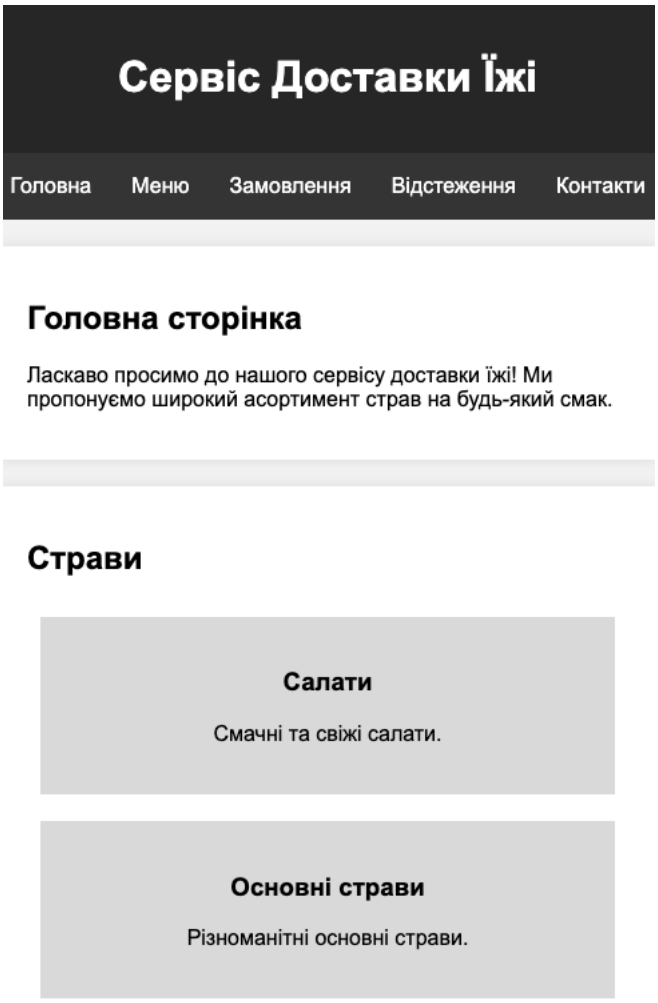

Рисунок 3.1.9. Прототип головної веб-сторінки для смартфонів.

ВИСНОВКИ

У ході виконання роботи було досягнуто поставленої мети — розробити сучасний, ефективний та зручний веб-сервіс для організації доставки замовлень, який відповідає сучасним вимогам користувачів і ринковим тенденціям.

На основі глибокого аналізу ринку доставки їжі, який включав огляд існуючих веб-сервісів, сучасних трендів та інновацій у цій сфері, було визначено основних конкурентів і проаналізовано їх бізнес-моделі та пропозиції. Ці дані стали основою для подальшого проектування та розробки веб-сервісу.

Вибір технологічного стеку, зокрема використання React для фронтенду та Node.js для бекенду, забезпечив високу продуктивність та масштабованість сервісу. Проектування архітектури на основі мікросервісної моделі дозволило забезпечити гнучкість, незалежне розгортання компонентів і високу надійність системи.

Створений дизайн користувацького інтерфейсу забезпечив зручність та інтуїтивність користування сервісом. Використання адаптивного дизайну дозволило забезпечити коректне відображення веб-сервісу на різних пристроях та платформах, що підвищило зручність для користувачів.

Таким чином, виконана дипломна робота підтвердила актуальність та значущість створення веб-сервісу для доставки їжі. Впровадження розробленого сервісу дозволить задовольнити зростаючі потреби ринку та забезпечити високий рівень задоволеності споживачів, сприяючи подальшому розвитку ринку онлайн-замовлень та підвищенню якості послуг у сфері доставки їжі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Global Online Food Delivery Market Size & Share Report, 2030. URL: https://www.grandviewresearch.com/industry-analysis/online-food-delivery-market-report (дата звернення: 24.04.2024)

2. Glovo [Wikipedia]. URL: https://en.wikipedia.org/wiki/Glovo (дата звернення: 24.04.2024)

3. Home - Glovo. URL: https://about.glovoapp.com (дата звернення: 24.04.2024)

4. UberEats URL: https://about.ubereats.com/ (дата звернення: 22.04.2024)

5. 6 Top Uber Eats Features to Include in Your Food Delivery App // Space Technologies URL: https://www.spaceotechnologies.com/blog/food-delivery-app-uber-eats-features/ (дата звернення: 22.04.2024).

6. Top 3 Deliveroo App Features to Create On-demand Food Delivery App URL: https://www.spaceotechnologies.com/blog/food-delivery-app-deliveroo-features/ (дата звернення: 22.04.2024).

7. Deliveroo URL: https://en.wikipedia.org/wiki/Deliveroo (дата звернення: 22.04.2024).

8. João Reis. Artificial Intelligence in Service Delivery Systems: A Systematic Literature Review. - 2020. - 233 с.

9. Using AI To Improve Delivery Applications URL: https://www.analyticsinsight.net/using-ai-to-improve-delivery-applications/ (дата звернення: 22.04.2024).

10. Ordering in: The rapid evolution of food delivery URL: https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/ordering-in-the-rapid-evolution-of-food-delivery (дата звернення: 22.04.2024).

11. 10 Food Delivery Trends For 2023 URL: https://idealmagazine.co.uk/10-food-delivery-trends-for-2023/ (дата звернення: 22.04.2024).

12. Restaurant technology trends to watch in 2024 URL: https://hospitalityinsights.ehl.edu/restaurant-technology-trends (дата звернення: 22.04.2024).

13. The top restaurant trends in 2023 URL: https://www.restaurantdive.com/spons/the-top-restaurant-trends-in-2023/650297 (дата звернення: 22.04.2024).

14. Which company is winning the restaurant food delivery war? URL: https://secondmeasure.com/datapoints/food-delivery-services-grubhub-uber-eats-doordash-postmates/)] (дата звернення: 22.04.2024).

15. Domino's Pizza: Business Model, SWOT Analysis, and Competitors 2023 URL: https://pitchgrade.com/companies/dominos-pizza (дата звернення: 22.04.2024).

16. United States Online Food Delivery Market URL: https://www.imarcgroup.com/united-states-online-food-delivery-market (дата звернення: 22.04.2024).

17. DoorDash Target Market Segmentation and Marketing Strategy URL: https://www.start.io/blog/doordash-target-market-segmentation-and-marketing-strategy/ (дата звернення: 22.04.2024).

18. Online Food Delivery Market Size & Share Analysis Source: https://www.mordorintelligence.com/industry-reports/online-food-delivery-market URL: https://www.mordorintelligence.com/industry-reports/online-food-delivery-market (дата звернення: 22.04.2024).

19. MVC [Wikipedia] URL: https://uk.wikipedia.org/wiki/Модель-вид-контролер (дата звернення: 22.04.2024).

20. CRUD [Wikipedia] URL: https://uk.wikipedia.org/wiki/CRUD (дата звернення: 22.04.2024).

21. Create, Read, Update, Delete [Wikipedia] URL: https://en.wikipedia.org/wiki/Create,_read,_update_and_delete (дата звернення: 22.04.2024).

22. Most used web frameworks among developers worldwide, as of 2023 URL: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/ (дата звернення: 22.04.2024).

23. GitHub Repository "go-food-delivery-microservices" URL: https://github.com/mehdihadeli/go-food-delivery-microservices (дата звернення: 22.04.2024).

24. Design a microservices architecture URL: https://learn.microsoft.com/en-us/azure/architecture/microservices/design/ (дата звернення: 22.04.2024).

25. Amazon Web. What are Microservices? URL: https://aws.amazon.com/ru/microservices/ (дата звернення: 22.04.2024).

26. Microservices architecture URL: https://www.atlassian.com/microservices/microservices-architecture (дата звернення: 22.04.2024).

27. RESTful web API design URL: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design (дата звернення: 22.04.2024).