

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)
на тему:

«Розробка веб-додатку для доставки продуктів»

Виконав студент групи КН-19-1
КРАВЧУК Денис Олегович

Керівник роботи:
ГРАФ Марина Сергіївна

Рецензент:
ПЕТРОСЯН Руслан Валерійович

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

Марина ГРАФ

«05» січня 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)

Здобувач вищої освіти: **КРАВЧУК Денис Олегович**

Керівник роботи: **ГРАФ Марина Сергіївна**

Тема роботи: **«Розробка веб-додатку для доставки продуктів»**,
затверджена наказом закладу вищої освіти від **«05» січня 2023 р., №10/с**

Термін здачі закінченої роботи: **«10» червня 2023 р.**

Вихідні дані роботи (зазначається предмет і об'єкт дослідження): **розробка функціонального додатку для можливості доставки їжі.**

Консультанти випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв
1	Граф М.С.	02.02.2023р.	02.02.2023р.
2	Граф М.С.	03.04.2023р.	03.04.2023р.
3	Граф М.С.	15.05.2023р.	15.05.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання завдання. Опрацювання літературних джерел	05.01.23 – 12.01.2023 13.01.23 – 19.01.2023 20.01.23 – 25.01.2023 25.01.23 – 06.02.2023	Виконав
2	Проектування структури	07.02.23 – 20.02.2023	Виконав
3	Написання програмного коду	21.02.23 – 10.04.2023	Виконав
4	Тестування системи	11.04.23 – 15.05.2023	Виконав
5	Оформлення пояснювальної записки	16.05.23 – 01.06.2023	Виконав
6	Попередній захист	10.06.2023	Виконав

Здобувач вищої освіти **Денис КРАВЧУК**

Керівник **Марина ГРАФ**

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра складається з переліку умовних позначень, вступу, трьох розділів, висновків та списку використаної літератури, а також наявних додатків.

Пояснювальна записка до випускної роботи містить 47 сторінок, 18 ілюстрацій та 5 таблиць. Список використаних джерел містить 10 найменувань і розміщується на 2 сторінках. Загальний об'єм роботи – 60 сторінок.

У першому розділі було проведено аналіз технологій та особливостей створення додатку для доставки продуктів.

У другому розділі було проведено проектування додатку для доставки продуктів з використанням обраних технологій.

У третьому розділі описується розроблений інтерфейс та порядок роботи додатку для доставки продуктів.

Розділ Висновки вміщує всі результати роботи, що було виконано в кваліфікаційній роботі з розробки додатку.

В роботі визначено основні завдання на розробку системи, проаналізовано аналоги розробленого додатку. Наведено сценарії роботи програмного комплексу, загальна структура програмного комплексу, опис алгоритмів взаємодії окремих модулів системи, опис об'єктної структури системи (діаграма класів). Також наводиться результати тестування додатку.

КН.КР.Б – 122 – 23 – ПЗ

					КН.КР.Б – 122 – 23 – ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Д.О. КРАВЧУК			Веб-додаток для доставки продуктів	Літ.		Арк.		Аркушів	
Керівник		М.С. ГРАФ						3		60	
Реценз.		В.Л. ЛЕВКІВСЬКИЙ				Пояснювальна записка					
Н. Контр.											
Зав. каф.		М.С. ГРАФ									
						Житомирська політехніка, група КН-19-1					

ABSTRACT

The explanatory note to the bachelor's thesis consists of a list of notations, an introduction, three sections, conclusions and a list of used literature, as well as available appendices.

The explanatory note to the final thesis contains 47 pages, 18 illustrations and 5 tables. The list of used sources contains 10 titles and is placed on 2 pages. The total volume of the work is 60 pages.

In the first section, an analysis of technologies and features of creating an application for product delivery was carried out.

In the second section, the application for product delivery was designed using the selected technologies.

The third section describes the developed interface and the operating procedure of the product delivery application.

The Conclusions section contains all the results of the work that was performed in the qualification work on the development of the application.

The work defines the main tasks for the development of the system, analyzes the analogues of the developed application. Scenarios of the software complex, the general structure of the software complex, a description of the interaction algorithms of individual system modules, a description of the object structure of the system (class diagram) are given. The results of application testing are also given.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	7
ВСТУП	8
РОЗДІЛ 1. ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ОСОБЛИВОСТЕЙ СТВОРЕННЯ ДОДАТКУ ДЛЯ ДОСТАВКИ ПРОДУКТІВ	10
1.1 Аналіз задачі, засобів та методів її вирішення	10
1.2 Аналіз існуючого пз для доставки продуктів	12
1.3 Вибір архітектури веб – додатка	17
1.4 Обґрунтування вибору інструментальних засобів.....	18
1.5 Вимоги до апаратного забезпечення	19
Висновки до першого розділу.....	20
РОЗДІЛ 2. ПРОЕКТУВАННЯ ДОДАТКУ ДЛЯ ДОСТАВКИ ПРОДУКТІВ ІЗ ВИКОРИСТАННЯМ ВИБРАНИХ ТЕХНОЛОГІЙ	21
2.1 Визначення варіантів використання та структури додатку	21
2.2 Проектування архітектури	23
2.3 Розробка БД додатку.....	25
2.4 Реалізація функціоналу додатку	29
Висновки до другого розділу	30
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ДОДАТКУ ДЛЯ ДОСТАВКИ ПРОДУКТІВ	32
3.1 Проектування інтерфейсу додатку	32
3.2Порядок роботи з інтерфейсом додатку	32
3.3Тестування розробленого додатку	40

Висновки до третього розділу:	43
ВИСНОВКИ.....	44
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	46
ДОДАТКИ.....	48
ДОДАТОК А.....	49

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – База даних

ПЗ – Програмне забезпечення

VS Code – Visual Studio Code

IDE – Integrated Development Environment

SPA – Single-Page Application

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

В сучасному світі, коли люди все більше зайняті роботою та іншими справами, важливо мати зручний та ефективний спосіб отримання продуктів. Інтернет-магазини та онлайн-сервіси доставки є одними з найпопулярніших засобів для забезпечення потребностей у їжі та інших товарах.

Доставка продуктів - це актуальна проблема для багатьох людей, які не мають можливості або бажання самотійно займатися покупками і носити важкі пакети з продуктами. Також, у зв'язку з пандемією COVID-19, збільшився попит на дистанційну доставку продуктів.

З кожним роком збільшується кількість людей, які використовують додатки для замовлення продуктів та їх доставки. Це створює величезний попит на інноваційні рішення у цій сфері. Тому, розробка додатку для доставки продуктів є актуальною темою для дослідження та розвитку.

У цій кваліфікаційній роботі буде описано процес розробки додатку для доставки продуктів, а також детально розглянуто функціональні та нефункціональні вимоги до додатку, його архітектуру та основні компоненти. Крім того, будуть проаналізовані та порівняні існуючі рішення на ринку, а також розглянуті перспективи подальшого розвитку додатку.

Результатом дослідження буде реалізація функціонального прототипу додатку, який відповідатиме вимогам та потребам користувачів. Такий додаток буде спрощувати процес замовлення та доставки для користувачів.

У цьому дослідженні будуть використані сучасні технології розробки програмного забезпечення, такі як мобільні платформи, бази даних, веб-сервіси та інші. Також буде проведений аналіз існуючих рішень та конкурентного середовища для визначення переваг та особливостей розроблюваного додатку.

Додаток забезпечить швидкий та зручний доступ до широкого асортименту продуктів кафе та ресторанів міста. Крім того, наші користувачі матимуть змогу замовляти доставку з будь-якої точки міста, що зробить процес замовлення ще більш зручним та доступним. Додаток також буде

мати можливість зберігати історію замовлень та персоналізовані рекомендації для користувачів.

Актуальність теми: додаток для доставки продуктів є актуальним і має потенціал для розвитку. Необхідно враховувати зміни у попиті та ринкові тенденції, щоб створити конкурентоспроможний додаток, який задовольнить потреби користувачів.

Мета: розробити повноцінний додаток для доставки продуктів. А також спростити процес замовлення та доставки продуктів, забезпечити розширений вибір та високу якість продуктів, а також покращити взаємодію між клієнтами та закладами харчування. Очікується, що цей додаток стане незамінним інструментом для всіх, хто шукає зручний та надійний спосіб отримання своїх улюблених продуктів без зайвих зусиль.

Об'єкт дослідження: доставка продуктів.

Предмет дослідження: розробка функціонального додатку для можливості доставки продуктів.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ОСОБЛИВОСТЕЙ СТВОРЕННЯ ДОДАТКУ ДЛЯ ДОСТАВКИ ПРОДУКТІВ

1.1 Аналіз задачі, засобів та методів її вирішення

Аналіз задачі полягає в тому, щоб зрозуміти, які основні функції та можливості повинен мати додаток для доставки продуктів. Наприклад, додаток повинен дозволяти користувачам шукати продукти, додавати їх у кошик, здійснювати замовлення та оплату, а також відслідковувати статус замовлення. Крім того, додаток повинен бути зручним та легким у використанні для користувачів. При розробці додатку для доставки продуктів необхідно мати на увазі його функціональні вимоги та можливості, а також враховувати потреби користувачів та їх зручність у використанні додатку. Важливо також забезпечити безпеку та захист особистих даних користувачів. Для цього можна використовувати різні засоби, такі як шифрування, аутентифікація та авторизація.

Мова програмування - це формальна мова, призначена для написання програмного коду, який виконується на комп'ютері. Мова програмування визначає синтаксис та семантику програмного коду, що дозволяє розробникам створювати програмне забезпечення з різною функціональністю та масштабом. Існує багато різних мов програмування, які можуть використовуватися для різних цілей, включаючи веб-розробку, наукові дослідження, розробку мобільних додатків, розробку ігор та багато інших. Розробники використовують мови програмування для створення програмного забезпечення, яке може взаємодіяти з користувачем, обробляти та зберігати дані, виконувати складні алгоритми та багато іншого. Крім того, мови програмування постійно розвиваються, що дозволяє розробникам створювати все більш потужне та ефективне ПЗ.

База даних – це сукупність даних організованих відповідно до концепції яка описує характеристику цих даних і взаємозв'язки між їх елементами. Задачою БД є зберігання інформації у зручному та доступному

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

вигляді. Для додатку, що містить в собі велику кількість інформації, як і раніше було сказано варто використовувати БД, структурувати всю інформацію яка знаходиться в додатку.

Firebase - це платформа розробки мобільних та веб-додатків, розроблена компанією Google. Вона надає набір інструментів та сервісів для розробників з метою полегшення процесу створення, розгортання та управління додатками.

Framework - це набір засобів, бібліотек, правил та практик, які допомагають розробникам швидше і ефективніше створювати програмне забезпечення. Фреймворки зазвичай містять базову інфраструктуру для створення програм, таку як архітектура програми, засоби для роботи з базами даних, веб-інтерфейси користувача та багато іншого. Вони дозволяють зосередитися на розробці функціональності програми, не турбуючись про низькорівневі деталі.

Visual Studio Code - це безкоштовний відкритий кодовий редактор, розроблений Microsoft, який можна використовувати для розробки програмного забезпечення в різних мовах програмування, включаючи JavaScript, Python, Java та багато інших.

VS Code має багато корисних функцій для покращення продуктивності розробника, таких як автодоповнення коду, підсвічування синтаксису, підказки, інструменти для роботи з Git, вбудований термінал та інші.

Angular - це відкрита JavaScript платформа для розробки веб-додатків. Angular базується на TypeScript, мові програмування, яка є синтаксичним розширенням JavaScript і дозволяє писати більш складні програми з більшим рівнем безпеки та ефективності. Angular пропонує ряд функцій та інструментів для розробки веб-додатків, включаючи шаблони для створення компонентів, сервісів та маршрутизації. Angular також має вбудовану підтримку для AJAX запитів, обробки форм та валідації даних.

TypeScript - це відкрита мова програмування, яка є синтаксичним розширенням мови JavaScript. Вона була розроблена компанією Microsoft і випущена в 2012 році. Однією з ключових переваг TypeScript є підтримка

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

сучасних функцій ECMAScript (стандарт JavaScript) та нових функцій, які ще не включені в стандарт. TypeScript також надає можливість створювати інтерфейси та абстрагувати код, що полегшує розробку складних додатків.

JavaScript - мова програмування, яка використовується для розробки веб-сайтів та веб-додатків. Вона є однією з найпоширеніших мов програмування і має широкий спектр застосувань.

1.2 Аналіз існуючого пз для доставки продуктів

Аналіз існуючих програмних засобів для доставки продуктів допоможе зрозуміти, які функції та можливості є на ринку та від чого можна відтіснитись. Для цього можна дослідити різноманітні додатки та веб-сайти, які дозволяють здійснювати доставку продуктів. Наприклад, серед існуючих платформ для доставки продуктів можна виділити такі, як Uber Eats, Glovo, Bolt Food та інші. Усі вони мають схожий функціонал: дозволяють здійснювати пошук та замовлення продуктів з різноманітних закладів, оплачувати замовлення, відслідковувати статус доставки тощо.

При аналізі можна звернути увагу на такі аспекти, як інтерфейс користувача, зручність у використанні додатку, швидкість та стабільність роботи, а також наявність додаткових функцій, таких як знижки, рекомендації тощо. Для врахування потреб та особливостей цільової аудиторії важливо звернути увагу на особливості ринку на Житомирщині, так як додаток буде використовуватися локально для невеликого міста. Також варто дослідити негативні відгуки користувачів про існуючі додатки, щоб уникнути помилок та недоліків при розробці власного додатку.

Аналіз існуючих платформ для доставки продуктів може надати корисну інформацію та ідеї для розробки власного додатку, що буде конкурентоспроможним на ринку.

Сервіси з доставки продуктів стали надзвичайно популярними в останні роки завдяки ряду переваг, які вони надають. Ось декілька з найбільш очевидних переваг сервісів з доставки продуктів:

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

1. Зручність: сервіси з доставки продуктів дозволяють замовити продукти зі свого смартфона або комп'ютера, і отримати її прямо до дверей. Це особливо зручно для тих, хто має обмежений час на приготування їжі або не має бажання відвідувати ресторани чи магазини.
2. Широкий вибір: сервіси з доставки продуктів пропонують великий вибір продуктів з різних кухонь світу, що дозволяє знайти їжу, яку ви любите, навіть якщо вона не доступна в вашому регіоні або місті.
3. Інформація про страви: сервіси з доставки їжі зазвичай містять інформацію про складові страв, калорійність, алергени та інші корисні дані, що дозволяють зробити більш освідомлені вибори щодо харчування.
4. Економія часу: замовлення їжі з сервісів доставки дозволяє зекономити час на приготування їжі та поїздки до ресторану. Це особливо корисно для тих, хто має зайнятий графік і не може відвідувати ресторани часто.

Не завжди такі сервіси мають тільки переваги. Ось кілька найпоширеніших недоліків сервісів з доставки їжі:

1. Незручності при замовленні: у деяких випадках може виникати незручність під час замовлення продуктів через неповний або неточний опис страв, або через те, що ви не можете змінити складові страви.
2. Якість продуктів: сервіси з доставки їжі не завжди можуть гарантувати якість їжі, оскільки вона має бути перевезена до вашого місця. Крім того, страви можуть охолоджуватися під час доставки, що може впливати на їх смак та текстуру.
3. Вплив на довкілля: сервіси з доставки продуктів можуть мати негативний вплив на довкілля через використання пластикових упаковок та пакувального матеріалу, які потрібно викидати після кожного замовлення.

Отже, після аналізу переваг та недоліків можна оцінити наявні сервіси, які будуть схожими до обраного додатку:

1) **Foodpanda** є глобальною платформою для доставки продуктів, яка співпрацює з тисячами ресторанів у багатьох країнах. Вони пропонують широкий вибір кулінарних страв і можуть задовольнити різні смакові уподобання користувачів.

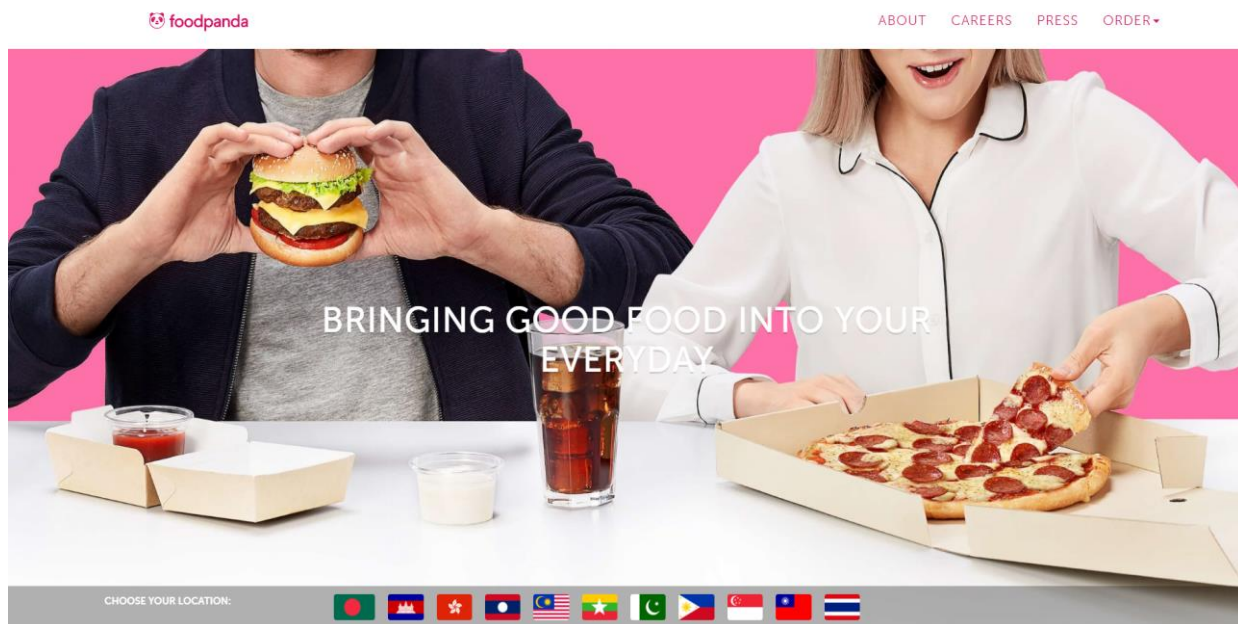


Рис.1.1 Додаток “Foodpanda”

Переваги: мобільний додаток Foodpanda простий та зручний у використанні, що робить процес замовлення та доставки їжі швидким і легким.

Недоліки: час доставки їжі може залежати від обсягу замовлень та навантаження на кур'єрську службу. Іноді можуть виникати затримки, особливо в пікові години.

2) **“Glovo”**- є одним з провідних сервісів доставки продуктів на світовому ринку. Вони співпрацюють з ресторанами, супермаркетами та іншими закладами, щоб надати широкий вибір страв та продуктів для замовлення. Glovo також пропонує доставку різних товарів, таких як ліки або продукти з крамниць.

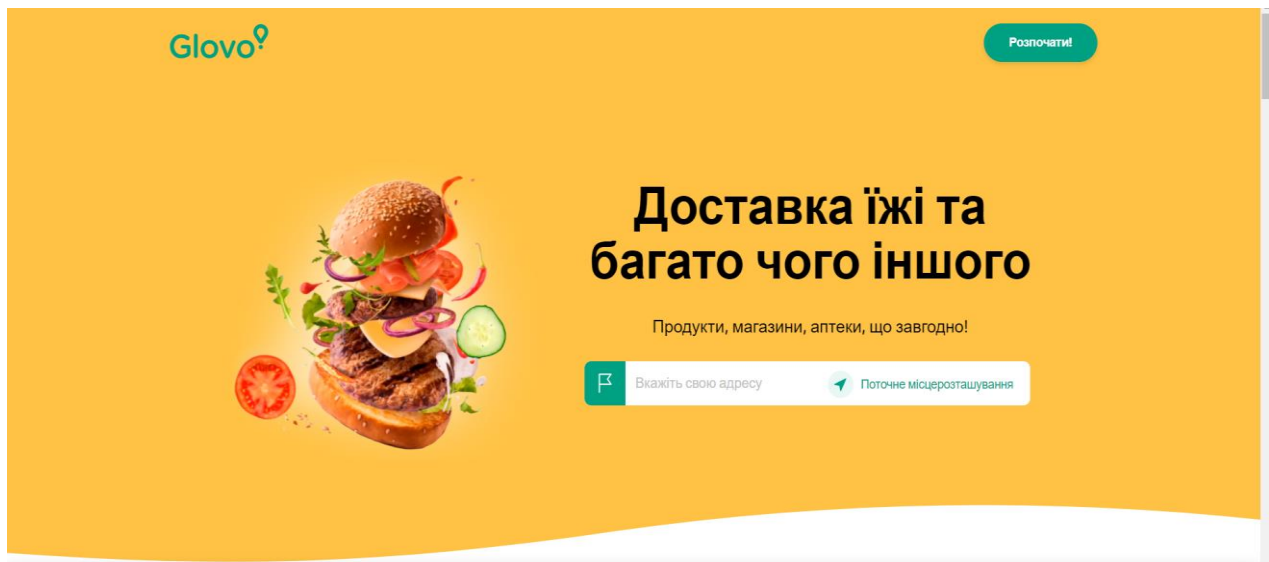


Рис.1.2 Додаток “Glovo”

Переваги: Glovo співпрацює з багатьма ресторанами та магазинами, що дозволяє користувачам мати доступ до різних видів їжі, продуктів та інших товарів. Glovo дозволяє користувачам замовляти їжу та товари з будь-якого місця, а також просувати спеціальні запити, такі як вибір конкретних інгредієнтів або додаткових побажань.

Недоліки: у деяких регіонах можуть бути обмеження на зону доставки, що може обмежувати доступність сервісу для деяких користувачів. Деякі ресторани або магазини можуть стягувати додаткові плати за доставку або мінімальний обсяг замовлення.

3) **Bolt Food** є частиною популярної платформи Bolt, яка відома своїми послугами таксі. Bolt Food спеціалізується на доставці продуктів з різних ресторанів та закладів харчування. Вони надають швидку та зручну доставку за допомогою своїх власних кур'єрів. Надають зручність і широкий вибір для користувачів, які бажають замовити їжу з різних закладів та доставити її безпосередньо до свого місця перебування.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

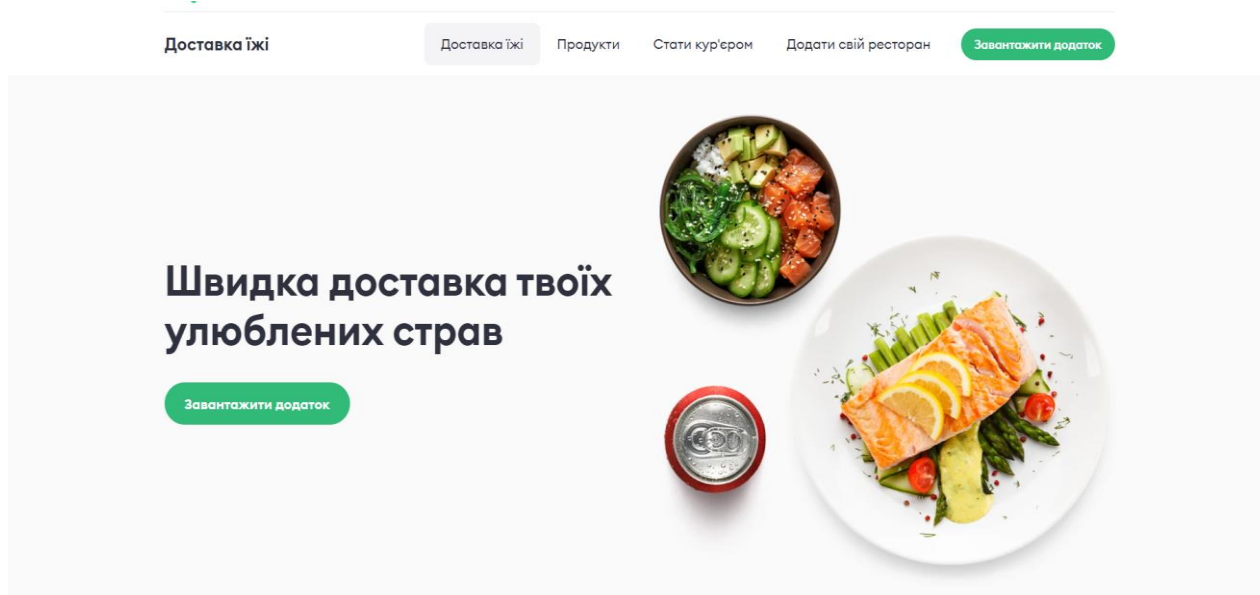


Рис.1.3 Додаток “Bolt Food”

Переваги: Bolt Food відомий своєю швидкою доставкою їжі. Вони використовують ефективну систему маршрутизації, що дозволяє зменшити час доставки та забезпечити задоволення користувачів. Даний сервіс має партнерські угоди з багатьма ресторанами та кафе, що дозволяє користувачам мати великий вибір різноманітних страв та кухонь.

Недоліки: іноді можуть виникати помилки у замовленнях, такі як неправильний склад страв або незадовільна якість доставки. Залежно від вибраного ресторану та місця доставки, можуть бути додаткові витрати на доставку їжі.

Отже, було вирішено створити власний додаток для доставки продуктів. Важливим є те, щоб він відповідав таким критеріям:

1. Простий та зручний інтерфейс
2. Додавання та редагування ресторанів
3. Запуск на пристроях різної потужності
4. Приємна графіка
5. Постійне оновлення меню ресторанів

1.3 Вибір архітектури веб – додатка

Для веб-додатка для доставки продуктів можна вибрати різні архітектури в залежності від конкретної задачі та потреб користувачів. Однією з найпопулярніших архітектур для веб-додатків є Single-Page Application (SPA), яка дозволяє створювати додатки, які не потребують перезавантаження сторінки при взаємодії з користувачем. Це забезпечує більш швидку та зручну роботу додатка, оскільки користувач може взаємодіяти з ним без затримок та перезавантажень сторінки.

Можна розглянути архітектуру Model-View-Controller (MVC) - дозволяє відокремити логіку додатка від його інтерфейсу, що забезпечує легкість розробки та підтримки. У MVC архітектурі модель відповідає за обробку та зберігання даних, контролер здійснює управління діями користувачів та забезпечує зв'язок між моделлю та представленням, а представлення забезпечує відображення даних та інтерфейс користувача.

Також можна розглянути інші архітектури, такі як Model-View-ViewModel (MVVM) - архітектурний патерн для розробки програмного забезпечення, який використовується для розробки графічних інтерфейсів користувача. Цей патерн дозволяє розділити компоненти додатку на три основні частини: модель (Model), представлення (View) та модель представлення (ViewModel).

Model-View-Presenter (MVP) - архітектурний патерн (шаблон проектування) для розробки програмного забезпечення. Він використовується для розділення логіки програми, що відповідає за представлення даних та користувацький інтерфейс, від логіки програми, що відповідає за опрацювання даних та виконання бізнес-логіки.

Основні переваги Single-Page Application над іншими архітектурами:

1. Швидкодія: оскільки SPA не потребують перезавантаження сторінок, користувачі можуть отримувати відповіді на запити миттєво, що дозволяє досягти високої швидкодії додатку.

2. Легкість розробки: SPA можуть бути розроблені за допомогою фреймворків, таких як Angular, React або Vue.js, що забезпечує більш просту і швидку розробку.
3. Краща масштабовність: SPA дозволяє розробникам більш легко масштабувати додаток, оскільки вони можуть додавати нові функції без перезавантаження сторінок.
4. Менше навантаження на сервер: SPA використовують AJAX-запити для взаємодії з сервером, що дозволяє зменшити навантаження на сервер.
5. Наявність кешування: більшість сучасних браузерів підтримують кешування SPA, що дозволяє користувачам переглядати додаток без Інтернет-з'єднання.
6. Кращий UX: SPA забезпечує більш зручний і плавний інтерфейс користувача, так як взаємодія з додатком відбувається без перезавантаження сторінок, що дозволяє користувачам відчувати себе більш комфортно.

1.4 Обґрунтування вибору інструментальних засобів

Для розробки веб-додатку на Angular для доставки продуктів можна використовувати різні інструментальні засоби. Основні з них:

1. Firebase - це інструмент, який дозволяє створювати та керувати базами даних, хостингом, автентифікацією та іншими сервісами в хмарі. Firebase підтримує Angular та надає можливість швидко та легко налаштовувати додаток на потреби користувачів.
2. Angular Material - це бібліотека компонентів та директив, розроблена Google, для розробки веб-додатків на Angular. Angular Material використовує Material Design - концепцію дизайну, яка була розроблена Google і має за мету створити єдиний дизайн-мову для всіх їх продуктів.
3. PWA (Progressive Web Apps) - це веб-додатки, які поєднують в собі переваги веб-сайтів та мобільних додатків. Вони створені з використанням сучасних веб-технологій, таких як HTML, CSS та

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

JavaScript, і можуть працювати на будь-якому пристрої з веб-браузером, включаючи мобільні пристрої, планшети та комп'ютери.

4. VSC (Visual Studio Code) - це безкоштовний відкритий текстовий редактор, розроблений компанією Microsoft для підтримки різноманітних мов програмування та технологій веб-розробки. VSC є потужним та зручним редактором для роботи з різноманітними проектами та технологіями веб-розробки.

Обрані інструментальні засоби дозволяють ефективно розробляти та підтримувати веб-додаток на Angular для доставки продуктів, забезпечуючи швидкість та якість розробки, легкість налаштування та масштабування додатку.

1.5 Вимоги до апаратного забезпечення

Для користування додатком для доставки продуктів на Angular також необхідно відповідне апаратне забезпечення. Основні вимоги до апаратури користувача:

1. Процесор - мінімум Intel Core i3 або еквівалентний процесор.
2. Оперативна пам'ять - мінімум 4 ГБ.
3. Жорсткий диск - мінімум 20 ГБ вільного простору для зберігання додатку та даних.
4. Монітор з можливістю відображення мінімум 1280x720 пікселів.
5. Клавіатура та миша - стандартні комп'ютерні пристрої.
6. Операційна система - Windows 7 або вище, або Linux або Mac OS.

Також для користування додатком потрібен доступ до Інтернету та веб-браузер, який підтримує Angular, такий як Google Chrome, Mozilla Firefox або Safari. Залежно від розміру та складності додатку може знадобитися швидкісний Інтернет-з'єднання. Загалом, вимоги до апаратного забезпечення для користування додатком для доставки продуктів на Angular не є дуже високими і задовольняють стандартні вимоги для більшості сучасних комп'ютерів та мобільних пристроїв.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до першого розділу

В першому розділі було сформульовано поставлену задачу для написання випускової кваліфікаційної роботи, також було проведення визначення основних вимог для можливості розробки проекту.

Було проведено аналіз задачі створення веб-додатку для доставки продуктів та визначено вимоги до його апаратного та програмного забезпечення. Також було проведено аналіз існуючих програмних засобів для доставки продуктів та вибрано архітектуру веб-додатка. Було обґрунтовано вибір інструментальних засобів для розробки додатку та встановлено мінімальні вимоги до апаратного забезпечення для телефонів, які планується використовувати для роботи з додатком доставки продуктів.

Спираючись на проведений аналіз, було обрано архітектуру, необхідну для розробки проекту, проведено визначення стеку технологій для можливості реалізації поставлених задач: Angular, FireBase, TypeScript та Javascript.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ДОДАТКУ ДЛЯ ДОСТАВКИ ПРОДУКТІВ ІЗ ВИКОРИСТАННЯМ ВИБРАНИХ ТЕХНОЛОГІЙ

2.1 Визначення варіантів використання та структури додатку

Варіанти використання додатку для доставки продуктів можуть бути різними, в залежності від потреб користувачів та бізнес-потреб. Основними варіантами використання можуть бути:

1. Замовлення продуктів додому або в офіс: користувачі можуть замовляти їжу з різних ресторанів та кафе та отримувати доставку додому або до робочого місця.
2. Замовлення продуктів на винос: користувачі можуть замовляти їжу з різних ресторанів та кафе та забирати її самостійно.
3. Відстеження замовлень: користувачі можуть відстежувати статус своїх замовлень та знаходити інформацію про доставку.
4. Оцінювання та відгуки: користувачі можуть залишати відгуки та оцінювати ресторани та кафе, щоб допомогти іншим користувачам при виборі місця для замовлення їжі.

Структура додатку для доставки продуктів може складатись з наступних компонентів:

1. Головна сторінка: це сторінка, на якій користувач першим бачиться після запуску додатку. Вона може містити пропозиції дня, рекомендації та іншу корисну інформацію для користувача.
2. Сторінка меню: це сторінка, на якій користувач може переглянути меню ресторану або магазину, з якого він замовляє продукти.
3. Сторінка замовлення: це сторінка, на якій користувач може перевірити своє замовлення, внести необхідні зміни та оформити замовлення.
4. Сторінка оплати: це сторінка, на якій користувач може внести платіжну інформацію та завершити оплату свого замовлення.

5. Сторінка профілю: це сторінка, на якій користувач може переглянути свій профіль, змінити інформацію про себе та переглянути історію своїх замовлень.
6. Адміністративна панель: це сторінка, на якій адміністратор може керувати замовленнями, меню, користувачами та іншими аспектами додатку.

Крім того, додаток може включати різні компоненти, такі як пошук продуктів, фільтри, сповіщення про статус замовлення, систему рейтингів та відгуків, інтеграцію з соціальними мережами та багато іншого.

Як видно з діаграми (рис.2.1), додаток може складатися з основних класів, таких як: авторизація, корзина, доставка та інші. Призначення усіх компонентів легко зрозуміти з їх назв. Наприклад, клас авторизації відповідає за реєстрацію або вхід в особистий профіль користувача.

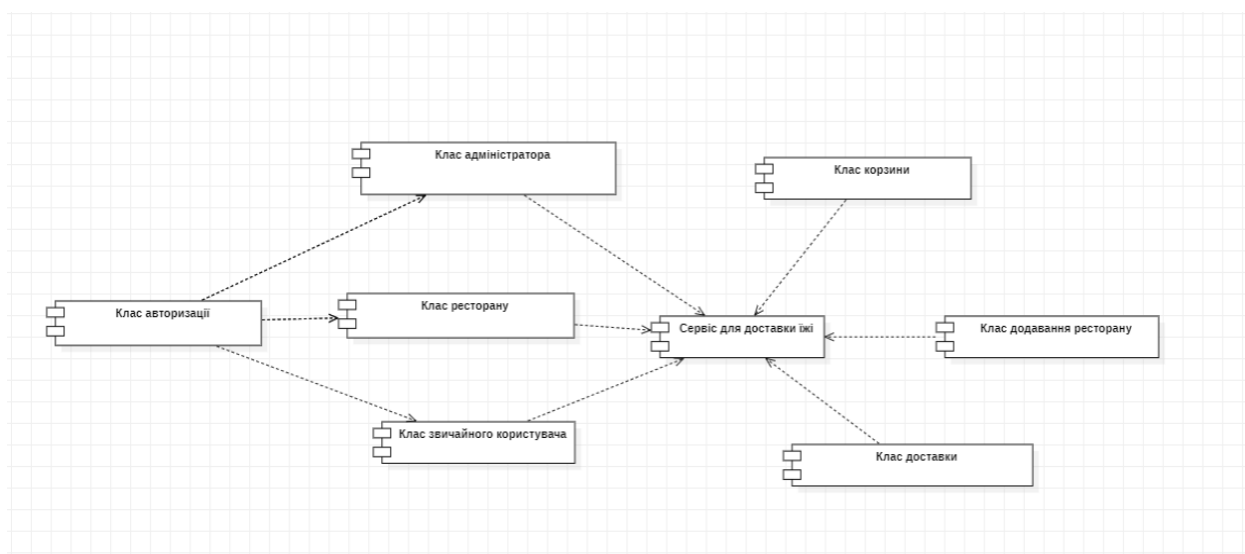


Рис.2.1 Діаграма класів

Наведено діаграму можливих варіантів використання додатку (рис.2.2). Користувач повинен зареєструватися або увійти в наявний особистий профіль. Після чого він може вибрати ресторан та страви в ньому перемістивши їх в корзину, потім з корзини користувач може видалити страви або їх оплатити та очікувати доставку.

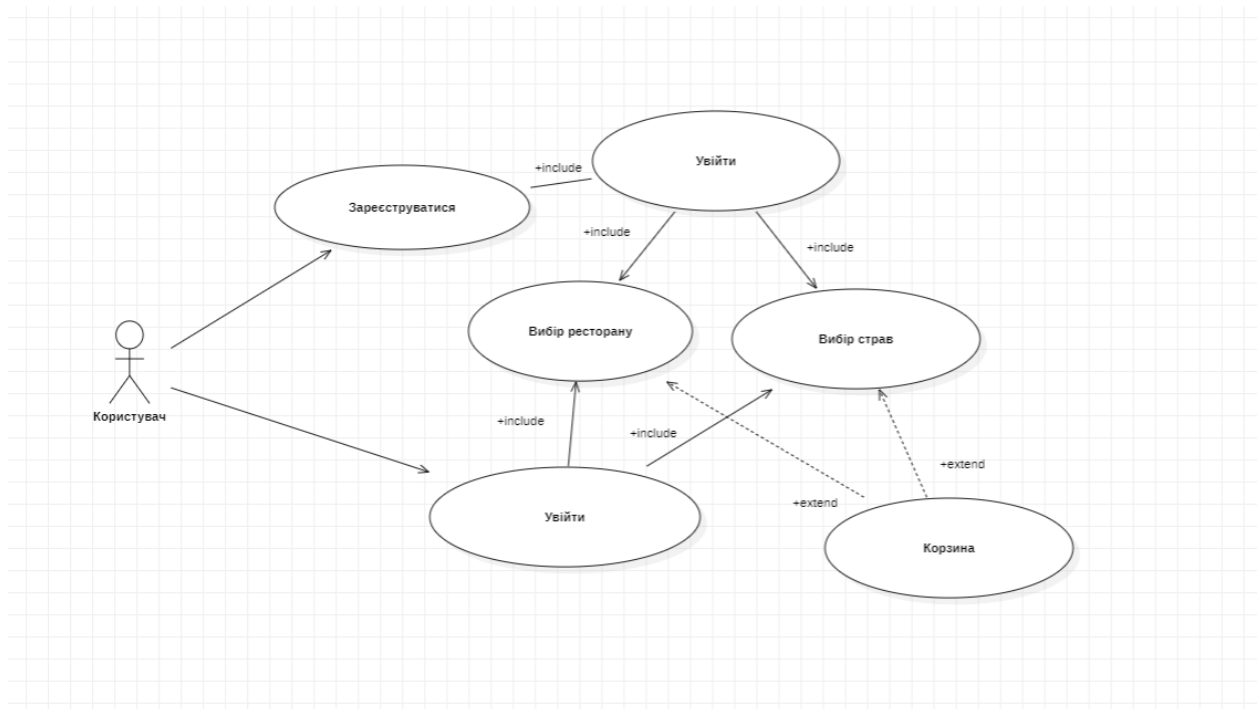


Рис.2.2 Діаграма варіантів використання

2.2 Проектування архітектури

Для розробки SPA додатку можна використовувати різні фреймворки та бібліотеки, такі як Angular, React або Vue. Я надаю перевагу Angular, тоді можна спроектувати архітектуру наступним чином:

1. Модулі:

- Головний модуль, що містить компоненти загальної інформації про додаток та їх взаємодію.
- Модуль користувача, що містить компоненти та сервіси для реєстрації, входу, відновлення паролю, налаштувань профілю.
- Модуль замовлення, що містить компоненти для перегляду, додавання, видалення та редагування замовлень.
- Модуль кафе та ресторанів, що містить компоненти для перегляду ресторанів, їх меню та інших деталей.
- Модуль кошика, що містить компоненти та сервіси для додавання та видалення товарів з кошика та оформлення замовлення.

2. Компоненти:

- Компонент головної сторінки, що містить інформацію про додаток та посилання на інші сторінки.
- Компонент реєстрації/входу, що містить форми для введення даних користувача та кнопки для відправлення запиту на сервер.
- Компонент замовлень, що містить таблицю з замовленнями та можливість додавання/редагування/видалення замовлень.
- Компонент кафе та ресторанів, що містить список ресторанів та можливість перегляду меню та деталей кожного ресторану.
- Компонент кошика, що містить список товарів у кошику, можливість їх видалення та оформлення замовлення.

3. Сервіси:

- Сервіс авторизації - використовується для авторизації користувачів в системі та збереження даних про них.
- Сервіс категорій їжі - використовується для отримання списку категорій їжі та продуктів, що відносяться до них.
- Сервіс кошика - використовується для збереження та управління замовленнями користувачів.
- Сервіс ресторанів - використовується для отримання списку кафе та ресторанів, їх адреси та режиму роботи.
- Сервіс доставки - використовується для організації доставки замовлень.
- Сервіс оплати - використовується для організації оплати замовлень.

Компоненти: вони будуть створені для розширення функціональності додатку. Наприклад, компонент "Кошик" буде відображати товари, які користувач вибрав для замовлення.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		24

Сервіси: це окремі класи, які будуть відповідати за роботу з даними. Наприклад, сервіс "Замовлення" буде відповідати за зберігання інформації про замовлення користувачів.

Моделі: вони будуть відповідати за зберігання даних про об'єкти в додатку. Наприклад, модель "Товар" буде містити інформацію про кожен товар у каталозі.

Маршрутизація: вона буде відповідати за навігацію по додатку. Наприклад, якщо користувач натисне на кнопку "Кошик", то він буде перенаправлений на сторінку з кошиком.

Крім того, для розробки додатку можна використовувати різноманітні бібліотеки та фреймворки, такі як Primeng, який надає готові компоненти для створення інтерфейсу користувача.

2.3 Розробка БД додатку

Розробка бази даних для додатку доставки продуктів є важливим етапом процесу розробки. База даних виконує роль центрального зберігання і організації інформації, пов'язаної з різними аспектами доставки продуктів, замовленнями, клієнтами та іншими важливими даними.

Основна мета розробки бази даних полягає у забезпеченні ефективного та надійного зберігання даних, забезпеченні швидкого доступу до них і здійсненні необхідних операцій обробки і аналізу. Така база даних використовується для зберігання інформації про страви, меню ресторанів, користувачів, адреси доставки, статуси замовлень, оплату та багато іншого.

При розробці бази даних для додатку доставки їжі необхідно врахувати декілька ключових аспектів:

- Необхідно визначити, які дані будуть зберігатися та як вони будуть організовані. Наприклад, таблиці для користувачів, ресторанів, замовлень тощо.

- Дані пов'язані між собою, тому потрібно визначити зв'язки між таблицями. Наприклад, зв'язок між користувачем і його замовленнями.
- Для ефективної роботи бази даних рекомендується використовувати нормалізацію даних, щоб уникнути дублювання та забезпечити цілісність даних.
- Безпека даних є критичним аспектом розробки бази даних. Потрібно встановити механізми автентифікації та авторизації, шифрування даних та захист від несанкціонованого доступу.

Таблиця 2.1

Таблиця “menus”

Поле	Тип	Призначення
id	string	Ідентифікатор
ingredients	string	Інгредієнти страви
name	string	Назва страви
photoUrl	string	Підтягує фотографії до бази даних з сайту https://unsplash.com/
price	number	Ціна страви
restaurantId	string	Підтягує унікальне id ресторану, страву з якого було замовлено

<https://unsplash.com/> - це безкоштовна платформа для пошуку та завантаження високоякісних фотографій та зображень. Вона надає доступ до широкого спектру фотографій, які можуть бути використані для різних цілей, включаючи веб-дизайн, рекламу, блоги, презентації та інші проєктів.

Таблиця “oders”

Поле	Тип	Призначення
id	string	Ідентифікатор
date	Data	Дата замовлення страви
name	string	Назва страви
price	string	Ціна страви
status	string	Показує статус замовлення
userId	string	Підтягує унікальне id користувача
menus	string	Показує замовлену страву, куди входить назва, ціна, інгредієнти, унікальне id користувача та ресторану з якого було замовлено страви
restaurantId	string	Підтягує унікальне id ресторану, страву з якого було замовлено

Таблиця 2.3

Таблиця “preorders”

Поле	Тип	Призначення
id	string	Ідентифікатор
name	string	Назва страви
price	string	Ціна страви
ingredients	string	Показує інгредієнти страви
userId	string	Підтягує унікальне id користувача
menus	string	Показує замовлені страви, куди входить назва, ціна, інгредієнти, унікальне id користувача та ресторану з якого було замовлено страви
restaurantId	string	Підтягує унікальне id ресторану, страву з якого було замовлено

Таблиця “restaurants”

Поле	Тип	Призначення
id	string	Ідентифікатор
name	string	Назва страви
description	string	Опис ресторану
userId	string	Підтягує унікальне id користувача, який створює власний ресторан

Таблиця 2.5

Таблиця “users”

Поле	Тип	Призначення
id	string	Ідентифікатор
email	string	Дані для реєстрації або входу в особистий профіль
password	string	Дані для реєстрації або входу в особистий профіль
role	string	Надає користувачу право вибору ролі під час входу в особистий профіль

У процесі розробки бази даних для додатку доставки продуктів, було враховано такі аспекти, як структура даних, взаємозв'язки між таблицями, нормалізація даних, запити до бази даних та безпека даних. Це дозволило створити ефективну та надійну базу даних, яка задовольняє потреби додатку і його користувачів.

Однією з основних переваг використання бази даних для додатку доставки продуктів є здатність швидко зберігати та отримувати інформацію про користувачів, ресторани, меню, замовлення та інші дані, що дозволяє забезпечити оперативність процесу доставки продуктів. Крім того, оптимізована база даних сприяє швидкому виконанню запитів і забезпечує користувачам зручний досвід використання додатку.

Проте, при розробці бази даних також виявлено деякі виклики і недоліки. Наприклад, необхідно було ретельно продумати структуру даних та взаємозв'язки між таблицями, а також забезпечити безпеку даних, щоб уникнути можливих порушень. Крім того, необхідно враховувати масштабованість бази даних, оскільки зі зростанням обсягу даних можуть виникати проблеми з продуктивністю та швидкодією.

Загалом, розробка бази даних для додатку доставки продуктів є важливим кроком у створенні успішного та функціонального додатку. Правильне проектування та оптимізація бази даних дозволяють забезпечити ефективну роботу додатку, задовольнити потреби користувачів та забезпечити конкурентоспроможність на ринку доставки продуктів.

2.4 Реалізація функціоналу додатку

Реалізація функціоналу додатку для доставки продуктів може бути розділена на декілька модулів і компонентів, які взаємодіють між собою. Основні модулі та їх функціонал можуть бути наступними:

1. Модуль авторизації і реєстрації:
 - Реєстрація нового користувача;
 - Авторизація зареєстрованого користувача.
2. Модуль каталогу продуктів:
 - Перегляд списку продуктів;
 - Фільтрація продуктів за категорією, ціною, рейтингом, тощо;
 - Пошук продуктів за назвою, описом, тощо;
 - Додавання продуктів до кошика.
3. Модуль кошика:
 - Додавання і видалення продуктів з кошика;
 - Зміна кількості продуктів у кошику;
 - Розрахунок вартості замовлення;
 - Оформлення замовлення.

4. Модуль замовлення:

- Отримання інформації про замовлення;
- Перегляд історії замовлень;
- Зміна статусу замовлення (наприклад, "очікує на оплату", "очікує на доставку", "доставлено", тощо).

5. Модуль доставки:

- Визначення місцезнаходження користувача;
- Відслідковування стану доставки;
- Зміна статусу доставки (наприклад, "очікує на виїзд кур'єра", "відправлено кур'єром", "доставлено", тощо).

6. Модуль профілю користувача:

- Редагування особистих даних;
- Зміна паролю;
- Видалення облікового запису.

Такий веб-додаток дозволяє користувачам замовляти продукти з різних ресторанів, кафе та забирали її в точках самовивозу або отримувати доставку за допомогою кур'єра. Для реалізації такого функціоналу було обрано архітектуру SPA з використанням Angular та Primeng.

Висновки до другого розділу

В даному розділі було проаналізовано різні варіанти використання додатку для доставки продуктів, визначена структура та функціональні можливості додатку. Була запропонована SPA архітектура з використанням Angular та сервіс-орієнтованого підходу для забезпечення розширюваності та підтримки коду.

Була спроектована база даних з урахуванням потреб користувачів та функціоналу додатку. Були визначені сутності та зв'язки між ними.

Було реалізовано основний функціонал додатку, такий як реєстрація та авторизація користувачів, перегляд меню, оформлення замовлення, оплата та відстеження статусу замовлення.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

В результаті проектування та реалізації додатку для доставки продуктів були вирішені проблеми, які стосуються зручності та швидкості процесу доставки продуктів. Було визначено оптимальну архітектуру та використано відповідні технології для забезпечення ефективної роботи додатку. Результатом є функціональний та зручний додаток для користувачів та постачальників.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ДОДАТКУ ДЛЯ ДОСТАВКИ ПРОДУКТІВ

3.1 Проектування інтерфейсу додатку

Проектування інтерфейсу додатку є важливою складовою розробки програмного забезпечення для доставки продуктів. Головна мета проектування інтерфейсу полягає в створенні зручного, привабливого та ефективного користувацького досвіду. Інтерфейс додатку повинен бути інтуїтивно зрозумілим, логічно організованим та спроектованим з урахуванням потреб та очікувань користувачів.

При проектуванні інтерфейсу додатку для доставки продуктів було враховано наступні елементи та принципи:

- мінімалістичний дизайн, який дозволяє зосередитись на головних функціях та спрощує навігацію додатку;
- інтерфейс додатку повинен мати чітку та логічну навігацію, яка дозволяє користувачам легко переміщатись між різними сторінками та функціями додатку;
- важливо забезпечити простоту використання додатку, розміщуючи елементи інтерфейсу та функціональність таким чином, щоб користувачам було зручно взаємодіяти з ними;
- інтерфейс додатку повинен бути адаптивним і зручно відображатися на різних пристроях, таких як смартфони, планшети та настільні комп'ютери;
- додаток повинен мати зручну та безпечну систему оплати, яка дозволяє користувачам легко здійснювати оплату за замовлення;
- для ресторанів було створено систему рейтингу, щоб користувач міг оцінити даний ресторан та його меню.

3.2 Порядок роботи з інтерфейсом додатку

Після завантаження, користувач опиняється у головному меню (рис. 3.1). Де знаходяться ресторани, їх опис та рейтинг від користувачів.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

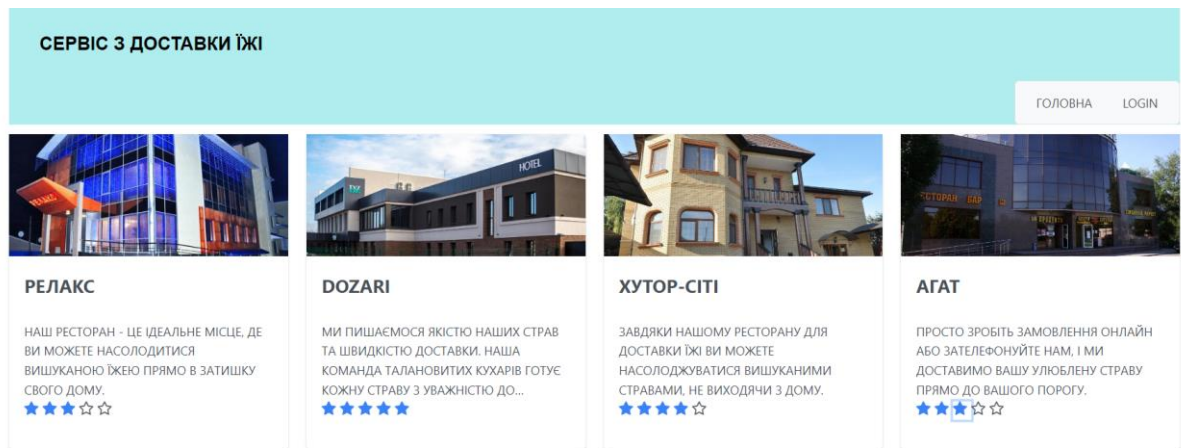


Рис. 3.1 Головне меню

Для роботи з додатком необхідно увійти у наявний профіль або створити новий. Для цього натискаємо кнопку “Login” на головній сторінці (рис.3.2). Для створення особистого профілю необхідно натиснути на кнопку “Реєстрація”(рис.3.3) у формі “Login”.

Рис.3.2 Форма “Login”

Для створення даної форми (рис.3.2) було використано такий метод:

```

userLogin() {
  if (this.form.invalid) {
    return
  }
  this.submitted = true
  const user: User = {

```

```

    email: this.form.value.email,
    password: this.form.value.password,
    returnSecureToken: false
  }
  this.authService.login(user).subscribe( () => {
    this.form.reset()
    this.router.navigate(['/'])
    this.submitted = false
  })
}

```

СЕРВІС З ДОСТАВКИ ЇЖІ

ГОЛОВНА LOGIN

РЕЄСТРАЦІЯ

ВВЕДІТЬ EMAIL

ВВЕДІТЬ ПАРОЛЬ

ПІДТВЕРДІТЬ ПАРОЛЬ

Зареєструвати аккаунт

Рис.3.3 Форма “Реєстрація”

Метод для реєстрації користувача зі створенням ролі:

```

signupWithRole(){
  if (this.form.invalid) {
    return
  }
  const user: User = {
    email: this.form.value.email,
    password: this.form.value.password,
    role: this.form.value.role,

```

```

returnSecureToken: false
}
this.authService.signupWithRole(user).subscribe( () => {
  this.form.reset();
  this.router.navigate(['/profile', 'dashboard']);
})
}

```

Після входу в особистий профіль на головній сторінці додадуться кнопки “Корзина”, “Профіль” “Logout”(рис.3.4).

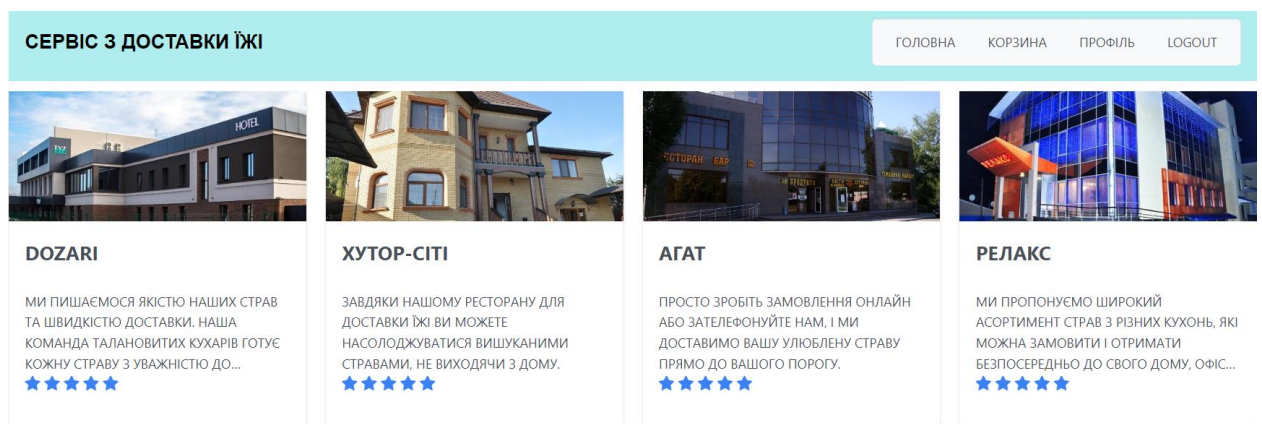


Рис.3.4 Додаткові кнопки головного меню

Кнопка “Корзина” відповідає за страви, які користувач перемістив у корзину для подальшого їх замовлення. Кнопка “Logout” відповідає за вихід з особистого профілю користувача (рис.3.5).

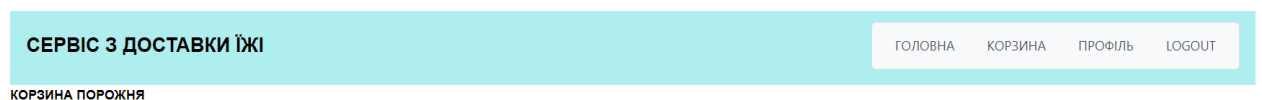


Рис.3.5 “Корзина”

Метод, що відповідає за додавання страв у кошик та їх оновлення у системі:

```

buyMenu(menu: Menu) {
  let currentUserPreorder: Preorder =
this.restaurantService.preorders.find((preorder: Preorder) =>
{

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

```

return preorder.userId === this.authService.currentUser?.id;
})
if (currentUserPreorder) {
  currentUserPreorder.menus.push(menu);
  this.restaurantService.updatePreOrder(currentUserPreorder).subscribe();
} else {
  currentUserPreorder = {
    menus:[menu],
    userId: this.authService.currentUser?.id
  };
  this.restaurantService.createPreOrder(currentUserPreorder).subscribe();
}
}

```

Кнопка “Профіль” відповідає за особистий профіль користувача, де знаходиться підменю з кнопками: “Мої замовлення”, “Ресторани” (рис.3.6), для користувача з роллю “Admin” і “Restaurant” наявні додаткові кнопки: “Створити ресторан”, “Реєстрація”.

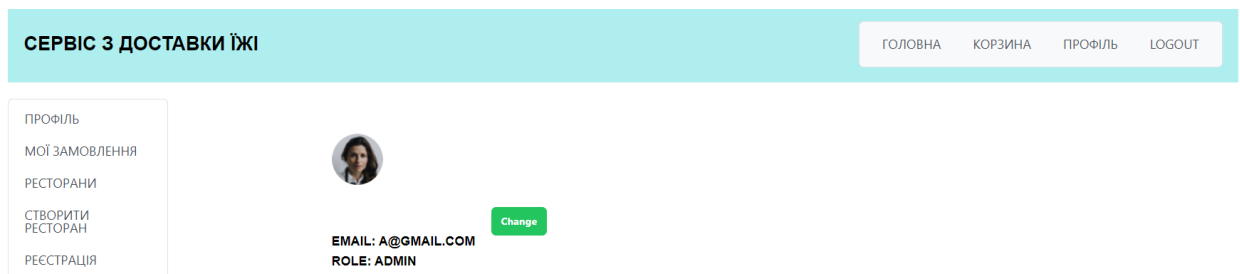


Рис.3.6 “Профіль”

Користувач може перейти в “Ресторани” (рис.3.7), де знаходяться ресторани та їх меню. Користувач з роллю “Admin” і “Restaurant” може додавати або ж видаляти ресторани та їх меню.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

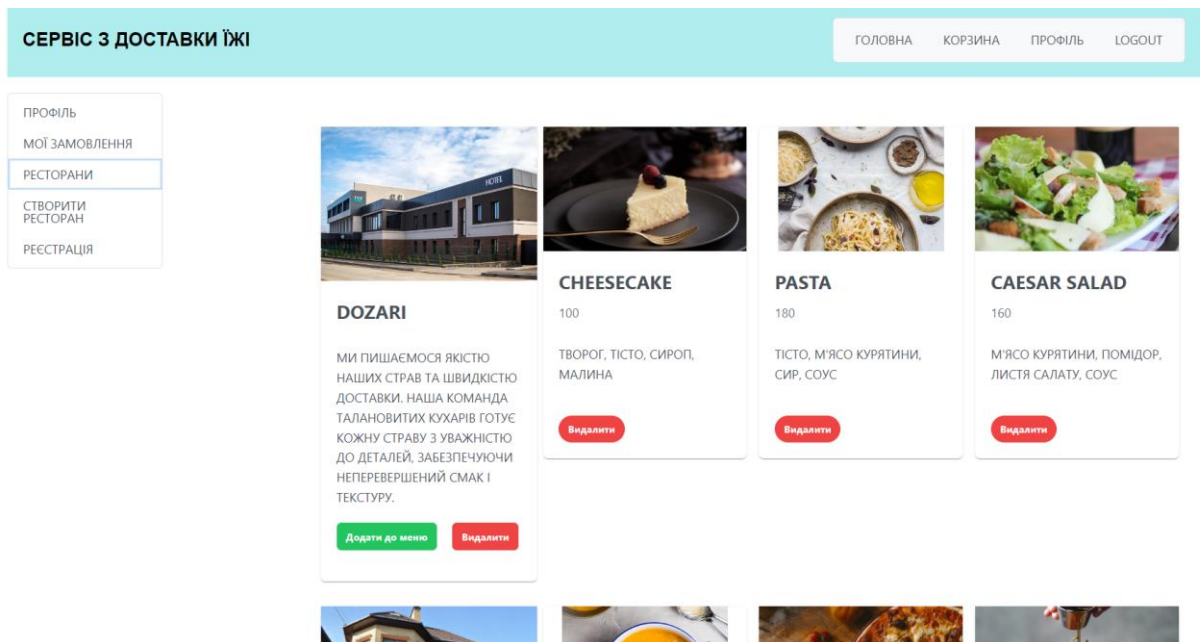


Рис.3.7 Ресторан та його меню

Якщо користувач хоче замовити якусь страву він повинен у головному меню вибрати ресторан та страви, які йому сподобались та натиснути кнопку “Додати до кошика”, після чого страва, яку обрав користувач переміститься до кошика.

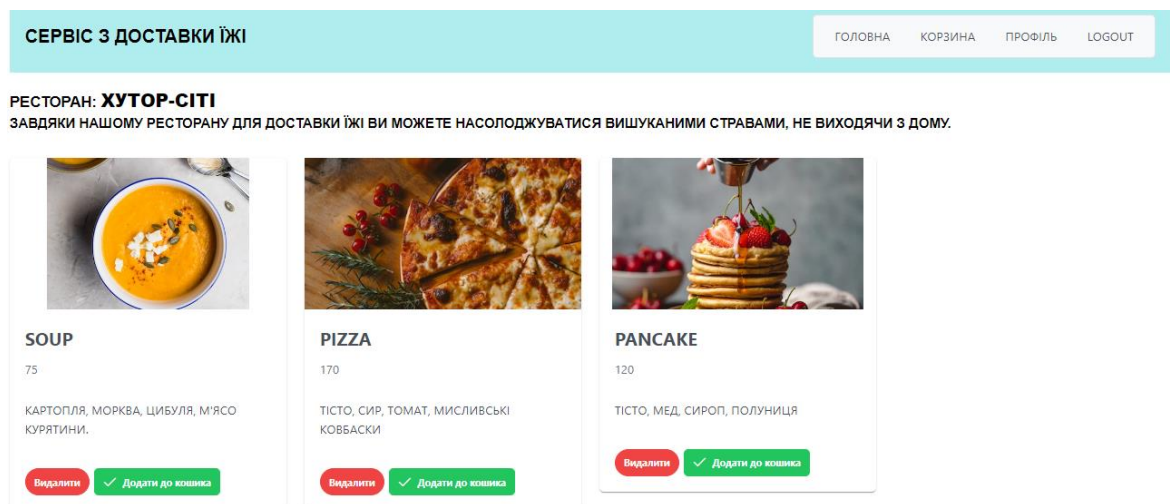


Рис.3.8 Додавання до кошика

Кошик після того, як користувач додав страви до нього (рис.3.9).

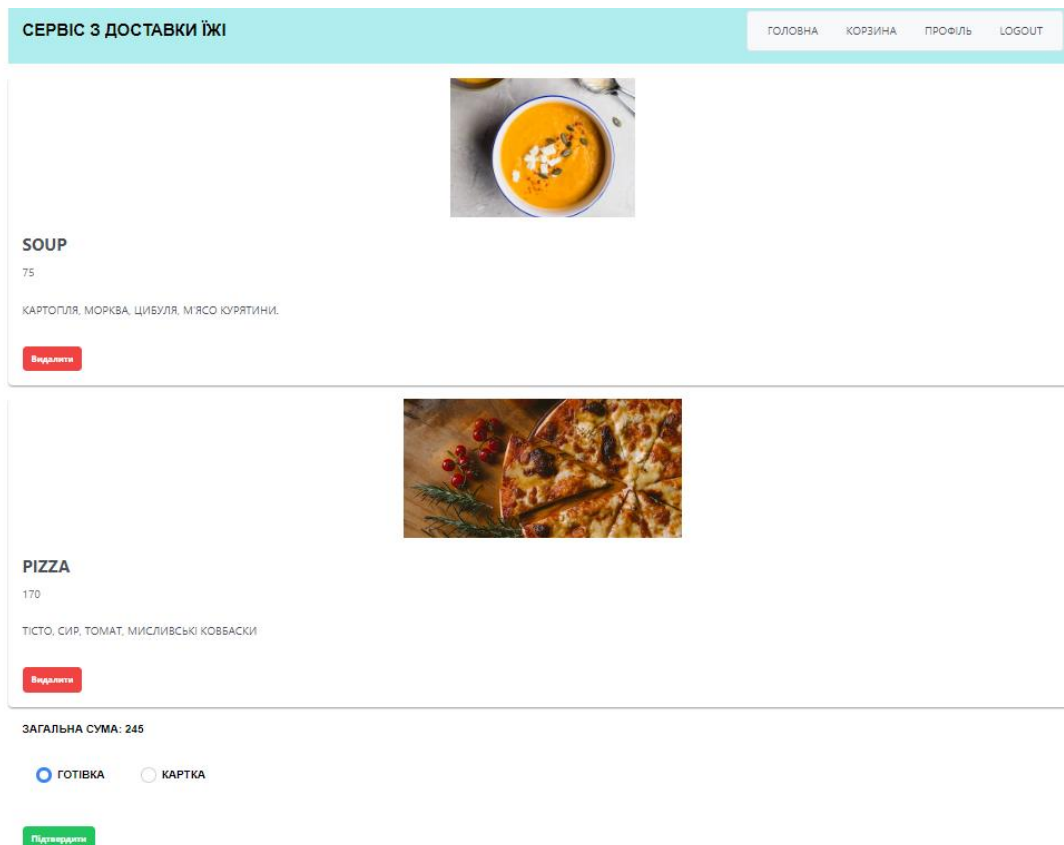


Рис.3.9 Кошик з продуктами

Після підтвердження замовлення користувач може побачити його в меню “Мої замовлення”(рис.3.10). На цій сторінці відображаються страви, які замовив користувач, а також статус замовлення та загальна сума.

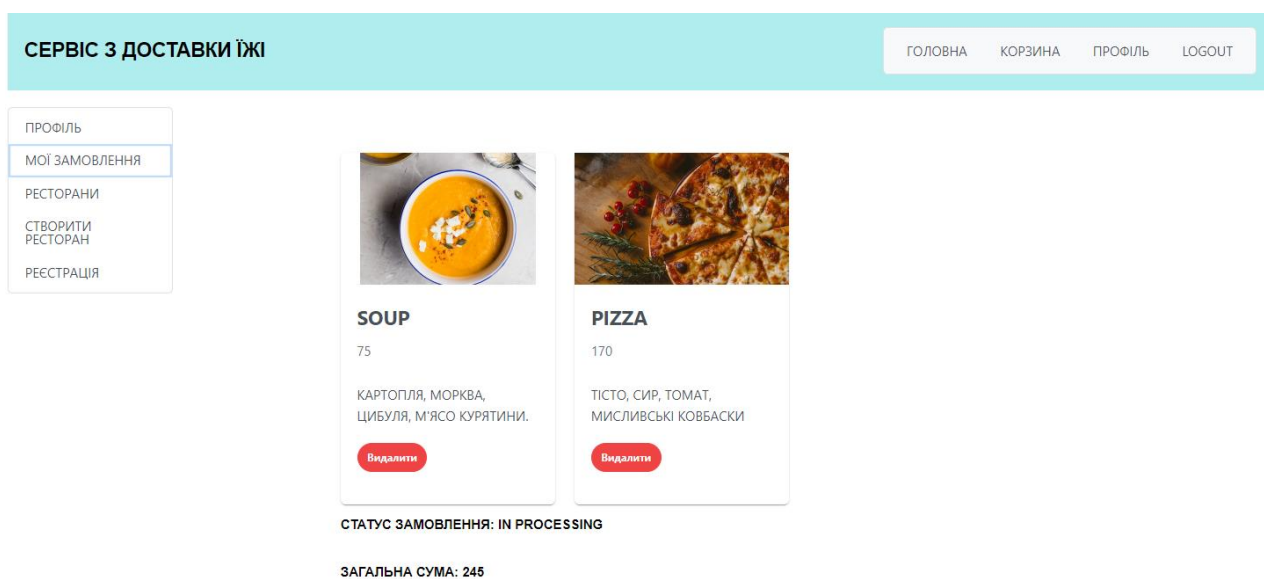
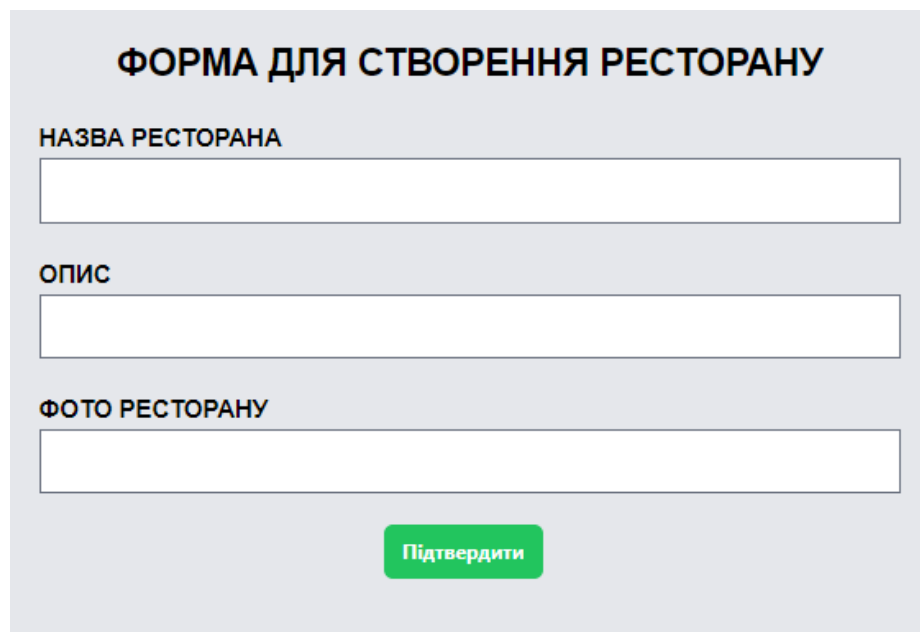


Рис.3.10 “Мої замовлення”

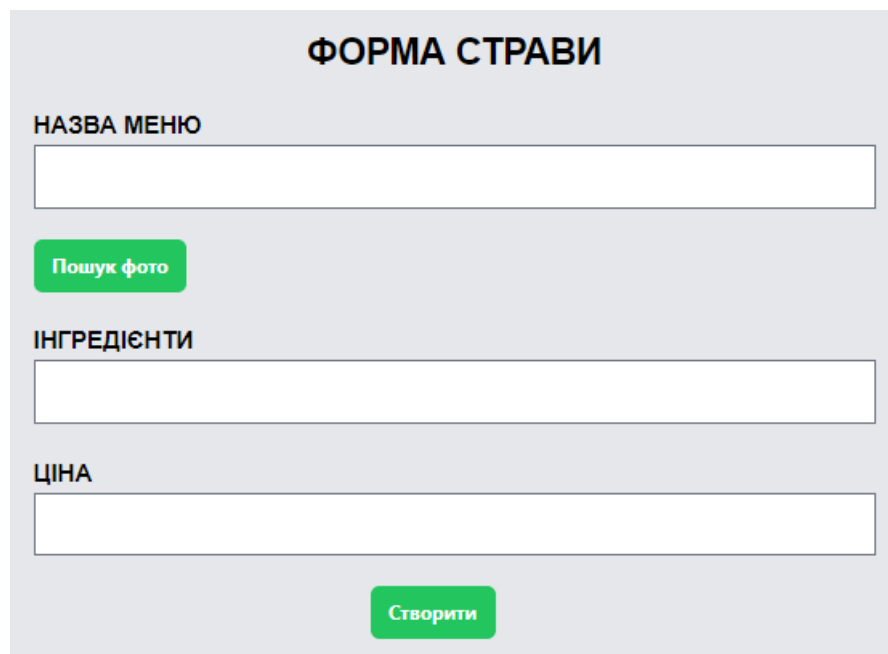
Для створення ресторану користувач з роллю “Admin” або “Restaurant” повинен заповнити відповідну форму (рис.3.11).



The form is titled "ФОРМА ДЛЯ СТВОРЕННЯ РЕСТОРАНУ" (Form for creating a restaurant). It contains three input fields: "НАЗВА РЕСТОРАНА" (Restaurant Name), "ОПИС" (Description), and "ФОТО РЕСТОРАНУ" (Restaurant Photo). Below these fields is a green button labeled "Підтвердити" (Confirm).

Рис.3.11 Форма для створення ресторану

Також користувач з роллю “Admin” і “Restaurant” може створювати страви ресторану заповнивши відповідну форму (рис.3.12).



The form is titled "ФОРМА СТРАВИ" (Form for creating a dish). It contains three input fields: "НАЗВА МЕНЮ" (Menu Name), "ІНГРЕДІЄНТИ" (Ingredients), and "ЦІНА" (Price). Between the first and second fields is a green button labeled "Пошук фото" (Search photo). Below the third field is a green button labeled "Створити" (Create).

Рис.3.12 Форма для створення страв

3.3 Тестування розробленого додатку

Перевірка працездатності є необхідним етапом під час розробки програмного коду системи. Тестування додатку має свої особливості порівняно з тестуванням інтернет-ресурсів, ігрових систем та інших продуктів. Воно включає наступні основні види тестування:

- тестування функціональності, яке перевіряє, чи працюють усі функції додатку правильно;
- тестування продукту, яке перевіряє взаємодію користувача з додатком та його можливості;
- користувацьке тестування, яке включає залучення реальних користувачів для оцінки вражень та зворотного зв'язку.

Тестування функціональності. Нижче наведено тест-кейси роботи головного меню.

Кнопка «Реєстрація» відповідає за реєстрацію користувача:

передумови: користувач повинен знаходитись у формі «Login»;

кроки: натискання кнопки «Реєстрація»;

очікуваний результат: після внесення особистих даних користувач зареєстрував профіль;

фактичний результат: після внесення особистих даних користувач зареєстрував профіль;

Кнопка «Login» відповідає за вхід користувача в особистий профіль:

передумови: користувач повинен знаходитись у головному меню;

кроки: натискання кнопки «Login»;

очікуваний результат: перехід до форми для входу в особистий профіль;

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

фактичний результат: перехід до форми для входу в особистий профіль.

Після входу в особистий кабінет з'являється кнопка «Профіль», яка відповідає за інформацію в особистому профілю користувача:

передумови: користувач повинен увійти в особистий профіль;

кроки: натискання кнопки «Профіль»;

очікуваний результат: користувач опиняється в особистому профілі, де може переглядати власні дані, замовлення, ресторани та їх меню;

фактичний результат: користувач опиняється в особистому профілі де може переглядати власні дані, замовлення, ресторани та їх меню.

Кнопка «Мої замовлення», показує замовлення користувача, їх статус та дату доставки:

передумови: користувач повинен знаходитись в особистому профілі;

кроки: натискання кнопки «Мої замовлення» для перегляду замовлень користувача;

очікуваний результат: перехід до списку замовлень, їх статусу та дати доставки;

фактичний результат: перехід до списку замовлень, їх статусу та дати доставки.

Кнопка «Ресторани» в особистому профілі користувача:

передумови: користувач повинен знаходитись в особистому профілі;

кроки: натискання кнопки «Ресторани» для перегляду ресторанів та їх меню;

очікуваний результат: перехід до списку ресторанів та їх меню;

фактичний результат: перехід до списку ресторанів та їх меню.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

Кнопка «Logout» на головній сторінці додатку, відповідає за вихід користувача з особистого кабінету:

передумови: користувач повинен знаходитись на головній сторінці додатку;

кроки: натискання кнопки «Logout»;

очікуваний результат: вихід користувача з особистого кабінету;

фактичний результат: вихід користувача з особистого кабінету.

Для легкої роботи додатку при реєстрації користувач має вибір ролей (рис.3.13). Роль «Buyer» відповідає за користувача, який може тільки робити покупки. Роль «Admin» надається розробникам додатку. Роль «Restaurant» відповідає за користувача, який зі згоди адміністратора додатку може додавати власний ресторан та його меню на головну сторінку додатку.

РЕЄСТРАЦІЯ

ВВЕДІТЬ EMAIL

ВВЕДІТЬ ПАРОЛЬ

ПІДТВЕРДІТЬ ПАРОЛЬ

☒ **BUYER**

☐ **ADMIN**

☐ **RESTAUEATEUR**

Зареєструвати аккаунт

Рис.3.13 Реєстрація користувача

Тестування продукту. Було проведено тестування особливостей додатку, включаючи основну – замовлення продуктів з різних ресторанів користувачем, який має роль «Вуцег». Крім того, було перевірено точність обрахування кількості страв та їх ціни. Також було налагоджено базу даних, за допомогою якої додаток функціонує без помилок. Під час тестування доступного функціоналу не було виявлено жодних недоліків чи помилок у роботі.

Висновки до третього розділу:

У даному розділі було описано порядок встановлення та налаштування додатку, а також визначена мінімальна конфігурація системи для комфортного користування.

Після проведення функціонального тестування додатку були виявлені помилки, які були успішно виправлені, і зараз система працює коректно та виконує всі поставлені завдання.

Цей розділ дозволяє налаштувати та запустити додаток з мінімальними зусиллями, забезпечуючи комфортне користування. Зроблені зміни та виправлення на основі проведеного тестування гарантують надійну та ефективну роботу системи.

Загалом, цей розділ підкреслює важливість належного налаштування та тестування додатку, щоб забезпечити задоволення користувачів та оптимальний досвід взаємодії з системою.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		43

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було спроектовано та реалізовано додаток для доставки продуктів, для цього було вирішено наступні завдання:

Було проведено детальний аналіз вимог, які стосуються функціональності та характеристик додатку. Це включало визначення основних функцій, можливостей замовлення та оплати, управління меню, а також вимог щодо інтерфейсу та дизайну. Огляд наявних аналогів та виявлення протиріч між можливостями та потребами користувачів показав, що головними функціями додатку має бути: зручний інтерфейс, реалізація системи авторизації користувача, наявність оплати всередині додатку та реалізація рейтингу ресторанів. Визначившись з клієнт-серверною архітектурою, обрано необхідний та достатній набір програмних продуктів для реалізації поставлених задач: VS Code, Angular, TypeScript, JavaScript, FireBase, Primeng.

Був розроблений інтуїтивно зрозумілий та привабливий інтерфейс для додатку, який сприяє зручній навігації та забезпечує зручність використання для користувачів.

Було реалізовано основні функції додатку, включаючи реєстрацію та аутентифікацію користувачів, перегляд меню ресторанів, замовлення продуктів, оплату, відстеження статусу замовлення та зворотний зв'язок з користувачами.

Було забезпечено інтеграцію з платіжною системою, щоб користувачі могли зручно та безпечно оплачувати замовлення.

Були проведені функціональні тести, під час яких виявлені помилки та недоліки були виправлені для забезпечення належної працездатності та якості додатку.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

Була підготовлена документація, яка включала опис функціональності, архітектури додатку, а також пояснення основних аспектів розробки та реалізації додатку.

В результаті виконання цих завдань було успішно реалізовано додаток для доставки продуктів, який відповідає вимогам, надає зручний та корисний досвід користування.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Оформлення кнопок за допомогою Primeng [Електронний ресурс] // Бібліотека Primeng. – 2023. – Режим доступу до ресурсу: <https://primeng.org/>
2. Гайд по розробці сайту на Angular [Електронний ресурс] // Udemy Course. – 2019 – Режим доступу до ресурсу: <https://www.udemy.com/join/login-popup/?next=/course/angular-complete-guide/learn/lecture/15772038>.
3. Розробка БД за допомогою FireBase [Електронний ресурс] // База даних FireBase. – 2023. – Режим доступу до ресурсу: <https://console.firebase.google.com/u/0/project/delivery-2f81b/database/delivery-2f81b-default-rtdb/data>.
4. Оформлення стилю додатку за допомогою tailwindcss [Електронний ресурс] // Бібліотека Tailwindcss. – 2021 – Режим доступу до ресурсу: <https://tailwindcss.com/>.
5. Приклад створення стилів [Електронний ресурс] // The Net Ninja. – 2021 – Режим доступу до ресурсу: <https://www.youtube.com/watch?v=bxmDnn7lrnk&list=PL4cUxeGkcC9gpXORlEHjc5bgnli5HEGhw>.
6. Підключення Firebase [Електронний ресурс] // Code Logic. – 2022 – Режим доступу до ресурсу: <https://www.youtube.com/watch?v=O0uVYhRE850>.
7. Підключення Angular модулів до додатку [Електронний ресурс] // Angular guide. – 2022 – Режим доступу до ресурсу: <https://angular.io/guide/lazy-loading-ngmodules>.
8. Гайд по розробці сайту за допомогою JavaScript [Електронний ресурс] // GreatStack. – 2021 – Режим доступу до ресурсу: <https://www.youtube.com/watch?v=IBfshkPlMW8>.
9. Розробка сайту на Angular [Електронний ресурс] // Червоний Хамелеон. – 2020 – Режим доступу до ресурсу:

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

<https://redchameleon.com.ua/ua/individualnaya-razrabotka/razrabotka-saytov-na-angular/>.

10. Сервіс для пошуку зображень в мережі Інтернет [Електронний ресурс]
// Unsplash Editor. – 2018 – Режим доступу до ресурсу:
<https://unsplash.com/>.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

Метод для реєстрації користувача зі створенням ролі:

```
signupWithRole(){
  if (this.form.invalid) {
    return
  }
  const user: User = {
    email: this.form.value.email,
    password: this.form.value.password,
    role: this.form.value.role,
    returnSecureToken: false
  }
  this.authService.signupWithRole(user).subscribe( () => {
    this.form.reset();
    this.router.navigate(['/profile', 'dashboard']);
  })
}
```

Метод для входу в особистий профіль користувача:

```
userLogin() {
  if (this.form.invalid) {
    return
  }
  this.submitted = true
  const user: User = {
    email: this.form.value.email,
    password: this.form.value.password,
    returnSecureToken: false
  }
```

```

this.authService.login(user).subscribe( () => {
  this.form.reset()
  this.router.navigate(['/'])
  this.submitted = false
})
}

```

Метод для створення ресторану користувачем з роллю “Admin” і “Restaurant”:

```

createRestaurant() {
  console.log(this.form)
  if (this.form.invalid) {
    return
  }
  this.submitted = true
  const restaurant: Restaurant = {
    name: this.form.value.name,
    description: this.form.value.description,
    userId: this.authService.currentUser.id,
    photo: this.form.value.photo
  }
  this.restaurantService.create(restaurant).subscribe( () => {
    this.form.reset()
    this.submitted = false
  })
}

```

Метод для створення меню користувачем з роллю “Admin” і “Restaurant”:

```

createMenu() {
  if (this.form.invalid) {
    return;
  }
}

```

```

    }
    this.submitted = true
    const menu: Menu = {
      name: this.form.value.name,
      ingredients: this.form.value.ingredients,
      price: this.form.value.price,
      photoUrl: this.photoUrl,
      restaurantId: this.restaurantId as string
    }
    this.restaurantService.createMenu(menu).subscribe( () => {
      this.form.reset();
      this.submitted = false;
    })
  }

```

Метод, що створює бокове меню в особистому профілю користувача:

```

private setAdminMenu() {
  const adminMenu = [
    {
      label: 'Ресторани',
      routerLink: '/profile/dashboard',
    },
    {
      label: 'Створити ресторан',
      routerLink: '/profile/create',
    },
    {
      label: 'Рєєстрація',
      routerLink: '/registration'
    }
  ]
}

```

```

];
console.log(this.authService.currentUser);
if (this.authService.currentUser.role === ROLE.ADMIN) {
  this.mainMenu.push(...adminMenu);
}
}

```

Метод, що створює замовлення в БД і видаляє його з кошика:

```

submitCart(preorder: Preorder) {
  const preorderCopy: Preorder = JSON.parse(JSON.stringify(preorder));
  delete preorderCopy.id;
  const today = new Date();
  const order: Order = {
    ...preorderCopy,
    status: ORDER_STATUS.PROCESSING,
    payment: this.selectedPayment,
    date: new Date(today.getFullYear(), today.getMonth(), today.getDate(), 0, 0, 0)
  }
  this.restaurantService.createOrder(order).pipe(
    switchMap(() => {
      return this.restaurantService.removePreorder(preorder.id)
    }),
    tap(() => {
      this.restaurantService.preorders =
      this.restaurantService.preorders.filter((preorder: Preorder) => preorder.id !==
      preorder.id);
    })
  ).subscribe(() => {
    this.messageService.add({ severity: 'success', summary: 'Success', detail: 'Order
    Submitted' });
    this.preorder = null;
  });
}

```

```
});  
}
```

Метод, що відповідає за вихід користувача з системи:

```
private setLogoutMenu() {  
  const logoutMenu = {  
    label: 'Logout',  
    routerLink: '/login',  
    command: this.authService.logout.bind(this.authService)  
  };  
  const loginMenu = {  
    label: 'Login',  
    routerLink: '/login'  
  };  
  this.mainMenu.push(this.authService.currentUser ? logoutMenu : loginMenu);  
}
```

Метод, що відповідає за додавання страв у кошик та їх оновлення:

```
buyMenu(menu: Menu) {  
  let currentUserPreorder: Preorder =  
this.restaurantService.preorders.find((preorder: Preorder) => {  
    return preorder.userId === this.authService.currentUser?.id;  
  })  
  if (currentUserPreorder) {  
    currentUserPreorder.menus.push(menu);  
    this.restaurantService.updatePreOrder(currentUserPreorder).subscribe();  
  } else {  
    currentUserPreorder = {  
      menus:[menu],  
      userId: this.authService.currentUser?.id
```

```
};  
this.restaurantService.createPreOrder(currentUserPreorder).subscribe();  
}  
}
```

Метод, який перевіряє доступ користувача до певних сторінок:

```
canActivate(  
  route: ActivatedRouteSnapshot,  
  state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean  
{  
  if (this.auth.isAuthenticated()) {  
    return true  
  }  
  else {  
    this.auth.logout()  
    this.router.navigate(['/profile', 'login'],  
      {  
        queryParams: {  
          loginAgain: true  
        }  
      })  
    return false  
  }  
}
```

Інтерфейс меню:

```
export interface Menu {  
  id?: string;  
  name: string;  
  ingredients: string;
```

```
    price: string;
    photoUrl?: string;
    restaurantId: string;
  }
```

Інтерфейс замовлення:

```
export interface Order {
  id?: string;
  menus: Menu[];
  userId: string;
  date: Date;
  status: ORDER_STATUS;
  payment: string;
}
```

Інтерфейс ресторану:

```
export interface Restaurant {
  id?: string;
  name: string;
  description: string;
  userId: string;
  photo: string;
}
```

Метод, який бере всіх користувачів з БД:

```
getAllUsers(): Observable<User[]> {
  return this.http.get(`${environment.fbDBUrl}/users.json`).pipe(
    map((response: any) => {
      return Object.keys(response)
        .map((key: string) => {
```

```

    return {
      ...response[key],
      id: key
    } as User;
  });
}));
}

```

Метод, який бере посилання на фото за ключем із сервісу Unsplash:

```

getMenuPhoto(key: string): Observable<string> {
  return this.http.get(`https://api.unsplash.com/search/photos?
client_id=${environment.unsplashApiKey}
&query=${key}`)
  .pipe(map((response: any) => {
    return response.results[0].urls.small;
  }));
}

```

Методи, які беруть дані з БД:

```

getAll(): Observable<Restaurant[]> {
  return this.http.get(`${environment.fbDBUrl}/restaurants.json`)
  .pipe(map((response: {[key: string]: any}) => {
    return Object
      .keys(response)
      .map(key => ({
        ...response[key],
        id: key,
        date: new Date(response[key].date)
      })))
  })))
}

```

```

    }
    getById(id: string): Observable<Restaurant> {
return
this.http.get<Restaurant>(`${environment.fbDBUrl}/restaurants/${id}.json`)
    .pipe(map((restaurant: Restaurant) => {
        return {
            ...restaurant, id
        }
    })))
    }
    createMenu(menu: Menu): Observable<Menu> {
        return this.http.post(`${environment.fbDBUrl}/menus.json`, menu)
            .pipe(map((response: any) => {
                return {
                    ...menu,
                    id: response.name,
                }
            })))
    }
    getMenus(): Observable<Menu[]> {
        return this.http.get(`${environment.fbDBUrl}/menus.json`)
            .pipe(map((response: {[key: string]: any}) => {
                return Object
                    .keys(response)
                    .map(key => ({
                        ...response[key],
                        id: key
                    })))
            })))
    }

```

```

getPreorders(): Observable<Preorder[]> {
  return this.http.get(`${environment.fbDBUrl}/preorders.json`)
    .pipe(map((response: {[key: string]: any}) => {
      return response ? Object
        .keys(response)
        .map(key => ({
          ...response[key],
          id: key
        })) : [];
    })))
}

createPreOrder(preorder: Preorder): Observable<Preorder> {
  console.log(preorder)
  return this.http.post(`${environment.fbDBUrl}/preorders.json`, preorder)
    .pipe(map((response: any) => {
      const currentPreorder = {
        ...preorder,
        id: response.name,
      };
      this.preorders = [currentPreorder];
      return currentPreorder;
    })))
}

updatePreOrder(preorder: Preorder): Observable<Preorder> {
  return this.http.patch(`${environment.fbDBUrl}/preorders/${preorder.id}.json`,
    preorder)
    .pipe(map((response: any) => {
      const currentPreorder = {
        ...preorder,
        id: response.name,
      };
    })))
}

```

```

    };
    this.preorders.push(currentPreorder);
    return currentPreorder;
  )))
}

removePreorder(id: string): Observable<void> {
  return this.http.delete<void>(`${environment.fbDBUrl}/preorders/${id}.json`);
}

createOrder(order: Order): Observable<Order> {
  return this.http.post(`${environment.fbDBUrl}/orders.json`, order)
    .pipe(map((response: any) => {
      const currentOrder = {
        ...order,
        id: response.name,
      };
      this.orders = [currentOrder];
      return currentOrder;
    })))
}

getOrders(): Observable<Order[]> {
  return this.http.get(`${environment.fbDBUrl}/orders.json`)
    .pipe(map((response: {[key: string]: any}) => {
      return response ? Object
        .keys(response)
        .map(key => ({
          ...response[key],
          id: key
        })) : [];
    })))
}

```

```
removeOrder(id: string):  
Observable<void> {  
    return this.http.delete<void>  
(`${environment.fbDBUrl}/orders/${id}.json`);  
}
```