МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАТИКИ ТА КОМП'ЮТЕРНОЇ ТЕХНІКИ

Рівень вищої освіти	Перший (бакалаврський)
Спеціальність	Інформаційні системи та технології
Освітня програма	Інформаційні системи та технології
Група	6.04.126.010.19.1

ДИПЛОМНИЙ ПРОЄКТ

на тему: «Розроблення веб-додатку доставки продуктів»

Виконав: студент Віктор КИСЕЛЬОВ

Керівник: к.т.н. Олена ПЕРЕДРІЙ

Рецензент: к.т.н, доц. Валентин ЛЮБЧЕНКО

Харків – 2023 рік

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

Факультет	Інформаційних технологій
Кафедра	Інформатики та комп'ютерної техніки
Освітній ступінь	Бакалавр
Спеціальність	126 "Інформаційні системи та технології"

ЗАТВЕРДЖУЮ
Завідувач кафедри
інформатики та комп'ютерної техніки
_____ проф. Сергій УДОВЕНКО
" 01 " лютого 2023 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ
Кисельову Віктору

1. Тема проєкту: «Розроблення веб-додатку доставки продуктів» та керівник проєкту: Передрій Олена, кандидат технічних наук затверджені наказом ректора від "01" лютого 2023 р. № 107-С.

2. Строк подання студентом проєкту: "01" червня 2023 р.

3. Вихідні дані до проєкту: ДСТУ щодо оформлення документації та бібліографічних посилань, літературні джерела, матеріали практики.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

Розділ 1. Постановка завдань дослідження та визначення вимог для веб-додатку доставки продуктів

Розділ 2 Проєктні та технічні рішення

Розділ 3. Експериментальні дослідження отриманих результатів

5. Перелік графічного матеріалу: Схема бази даних застосунку, діаграма А0, декомпозиція діаграми А-0, діаграма варіантів використання, архітектура

веб-додатку.

6. Консультанти розділів дипломного проєкту

Розділ	Прізвище, ім'я та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання: "01" лютого 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Розроблення плану дипломного проєкту, ознайомлення з літературними джерелами	01.02.2023	
2	Аналіз предметної області	01.03.2023	
3	Ознайомлення з аналогами та методами розробки веб-додатків	10.03.2023-31.03.2023	
4	Розроблення програмних модулів, проведення тестування продукту	01.04.2023-24.04.2023	
5	Попередня перевірка дипломного проєкту керівником	25.04.2023-30.04.2023	
6	Оформлення пояснювальної записки	01.05.2023-20.05.2023	
7	Перевірка дипломного проєкту на плагіат	26.05.2023	
8	Підготовка презентації та доповіді	29.05.2023	
9	Подання Голові Екзаменаційної комісії щодо захисту дипломного проєкту	01.06.2023	

Студент Віктор КИСЕЛЬОВ

Керівник проєкту Олена ПЕРЕДРІЙ

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту: 83 сторінки, 22 рисунка, 1 додаток, 20 джерел.

Об’єктом дослідження є інформаційна система для доставки продуктів споживачам.

Метою дипломного проєкту є розробка веб-додатку для замовлення доставки продуктів споживачам.

При виконанні дипломного проєкту використані методи порівняльного аналізу існуючих додатків доставки продуктів, синтез методів проєктування, розробки та моделювання бізнес-процесів предметної області, а також проєктування та розробки веб-додатків.

В результаті виконання дипломного проєкту на основі веб-технологій спроєктовано та розроблено інформаційну систему веб-додатку доставки продуктів

Розроблена інформаційна система буде призначена для використання у сфері доставки їжі споживачам та в майбутньому планується її тестування та впровадження у діяльність компанії.

JAVA SCRIPT, HYPERTEXT MARKUP LANGUAGE, VISUAL STUDIO CODE, веб-додаток, інформаційна система, доставка продуктів.

ABSTRACT

Explanatory note to the diploma project: 83 pages, 22 figures, 1 appendix, 20 sources.

The object of research is an information system for delivering products to consumers.

The purpose of the thesis project is to develop a web application for ordering food delivery to consumers.

In the course of the thesis project, the methods of comparative analysis of existing grocery delivery applications, synthesis of methods for designing, developing, and modelling business processes of the subject area, as well as designing and developing web applications were used.

As a result of the diploma project, an information system for a web-based product delivery application was designed and developed based on web technologies

The developed information system will be used in the field of food delivery to consumers, and in the future, it is planned to test and implement it in the company's activities.

JAVA SCRIPT, HYPERTEXT MARKUP LANGUAGE, VISUAL STUDIO CODE, web application, information system, food delivery.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ 7
ВСТУП 8
1 ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ ТА ВИЗНАЧЕННЯ ВИМОГ ДЛЯ ВЕБ-ДОДАТКУ ДОСТАВКИ ПРОДУКТІВ 10
1.1 Змістовний опис і аналіз предметної області 10
1.2 Огляд аналогів і аналіз існуючих варіантів розробки веб-додатку для доставки продуктів 15
1.3 Технології розробки мобільних застосунків 23
1.4 Постановка завдань розробки 29
2 ПРОЄКТНІ ТА ТЕХНІЧНІ РІШЕННЯ 33
2.1 Опис засобів реалізації програмного продукту 33
2.1.1. Вибір технологій розробки веб-додатків 33
2.1.2. Вибір системи керування базами даних 35
2.2 Проєктування БД 37
2.3 Розроблення архітектури веб-додатку 40
2.4 Концептуальне рішення 43
2.5 Проєктування інтерфейсу користувача 45
2.6 Опис розробки веб-додатку 46
3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ 48
3.1 Оцінка адекватності розробленого веб-додатку 48
3.2 Аналіз ефективності розробленого веб-додатку 55
3.3. Тестування програмного забезпечення 59
ВИСНОВКИ 63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 65
ДОДАТКИ 67

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

СУБД – СИСТЕМА УПРАВЛІННЯ БАЗАМИ ДАНИХ

PC – PERSONAL COMPUTER

JS – JAVA SCRIPT

ТП – ТЕХНІЧНА ПІДТРИМКА

ПЗ – ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

VSC – VISUAL STUDIO CODE

PHP – HYPERTEXT PREPROCESSOR

SQL – SCRUCTURED QUERY LANGUAGE

HTML – HYPERTEXT MARKUP LANGUAGE

CSS – CASCADING STYLE SHEETS

ADC – APPLICATION DELIVERY CONTROLLER

ВСТУП

Сьогоднішній швидкоплинний ритм сучасного життя ставить перед людьми все більше завдань, пов'язаних з ефективним використанням часу та ресурсів. Одним із важливих аспектів є швидкий та зручний доступ до необхідних продуктів та послуг. Зокрема, галузь доставки продуктів через Інтернет швидко набуває популярності, надаючи зручність та комфорт споживачам.

Для задоволення зростаючого попиту на послуги доставки продуктів, розроблення веб-додатку, який забезпечує швидкий та ефективний процес замовлення та доставки, є невід'ємною частиною інноваційних рішень в галузі електронної комерції. Такий додаток може забезпечити користувачам зручність та швидкість обслуговування, а також покращити ефективність роботи для підприємств, що надають послуги з доставки.

Метою даного дипломного проєкту є розроблення веб-додатку доставки продуктів, який буде відповідати потребам сучасного споживача та підприємств, що займаються доставкою. Головним завданням проєкту є створення зручного та інтуїтивно зрозумілого інтерфейсу для користувачів, що дозволить їм швидко та ефективно здійснювати замовлення, відстежувати процес доставки та забезпечувати обмін інформацією з доставщиками.

Для досягнення цілей дипломного проєкту передбачається проведення аналізу сучасних веб-додатків доставки продуктів, вивчення потреб та вимог користувачів, розробка архітектури та дизайну додатку, реалізація функціональних можливостей, а також тестування та валідація розробленого рішення.

Особлива увага буде приділена аспектам безпеки та захисту персональних даних користувачів, оскільки це є надзвичайно важливим у сучасному цифровому середовищі. Розробка механізмів автентифікації, шифрування та захисту інформації є невід'ємною частиною розробки додатку.

Очікується, що результатом цього дипломного проєкту буде веб-додаток доставки продуктів, який забезпечить зручність та ефективність процесу замовлення та доставки продуктів, відповідаючи потребам користувачів та

підприємств. Даний дипломний проєкт вносить свій внесок у сферу електронної комерції, розвиваючи новітні технології та покращуючи взаємодію між споживачами та підприємствами.

1 ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ ТА ВИЗНАЧЕННЯ ВИМОГ ДЛЯ ВЕБ-ДОДАТКУ ДОСТАВКИ ПРОДУКТІВ

1.1 Змістовний опис і аналіз предметної області

Мобільні додатки змінили всю індустрію роздрібної торгівлі в усьому світі, а продуктовий бізнес спостерігає велике зростання протягом останніх років. Продуктові мобільні програми стають все більш популярними, оскільки вони ефективно задовольняють потреби людей у продуктах, не виходячи з дому. Ознайомившись із загальною картиною продуктових онлайн-магазинів можна відзначити наступні переваги.

1.1. Зручно та заощаджує час

Це велика перевага, враховуючи напружене сучасне життя. За допомогою мобільних додатків клієнти можуть купувати продукти будь-де та будь-коли. Замість того, щоб витрачати час на пошук продуктів на полицях магазину, вони можуть просто шукати певні товари та додавати їх у свій онлайн-кошик. Їм не потрібно ходити з важкими продуктовими візками та стояти в довгій черзі, щоб розплатитися. Усі завдання можна легко виконати лише кількома кліками в додатках для покупки продуктів, і клієнтам це, безперечно, подобається.

1.2. Дозволяє клієнтам замовляти продукти, яких немає в наявності на поточний момент

Покупці завжди засмучуються, коли дізнаються, що всі необхідні їм товари в магазині розкуплені. Це марна трата їхнього часу і не задовільнені потреби. Завдяки додаткам для доставки продуктів, якщо продукту немає в наявності, електронні покупці можуть знати про це та або забронювати цей продукт заздалегідь, або налаштувати сповіщення про те, коли він знову з’явиться в наявності. Це допомагає клієнтам онлайн магазинів приємно та безперебійно купувати продукти, не виходячи з дому.

1.3. Регулярні та заплановані замовлення

У продуктових магазинах є певні товари, які споживачі хочуть купувати неодноразово протягом тижня чи місяця. Що стосується використання продуктових додатків електронної комерції, покупці можуть підписатися на регулярні замовлення, наприклад, замовлення певного типу їжі чи овочів кожні 2 дні. Це допоможе покупцям легко керувати своїми запасами. Вони можуть налаштувати сповіщення для продуктів, що закінчуються, щоб швидко підготувати товари на складі. Вони також можуть легко керувати всіма минулими, теперішніми та майбутніми замовленнями та легко їх шукати.

1.4. Безпечні варіанти оплати

Клієнти можуть вибрати спосіб оплати або навіть оплату після доставки. Більшість методів онлайн-платежів є безпечними, особливо через мобільні додатки електронної комерції.

1.5. Спеціальні пропозиції та програми лояльності

Лояльність клієнтів дуже важлива для розвитку та стабільності бізнесу, що підтверджується багаторічними дослідженнями. Але потрібно не просто мати лояльних клієнтів, а й підтримувати зацікавленість випадкових клієнтів, щоб перетворити їх на лояльних. Завдяки додаткам онлайн-магазину продуктів клієнт може отримувати персоналізовані пропозиції та ефективно діючі програми лояльності.

1.6. Покращені маркетингові можливості

Продуктові програми для електронної комерції забезпечують персоналізований та позитивний досвід і залучають покупців навіть після того, як вони виходять із системи після успішного розміщення замовлення. За допомогою функції Push-повідомлення, доступної в додатках, власники магазинів можуть надавати індивідуальну підтримку послуг і угод для кожного клієнта. Крім того, на основі даних користувачів, їх історії перегляду та шаблонів покупок, бізнес-адміністратори можуть проводити ефективні прямі та цифрові маркетингові кампанії для подальшого стимулювання покупок клієнтів. У мобільних додатках нові функції постійно оновлюються, тому можливості безмежні.

2. Основні функції продуктових програм електронної комерції для користувача

2.1. Вхід/Реєстрація

Сучасні користувачі програм настільки звикли до простих і швидких процесів, що трудомісткі дії вручну стали перешкодою. Процес реєстрації має бути простим і зручним для користувача. Процес входу в систему за посиланням через соціальні мережі сьогодні дуже популярний. Цю опцію можна запропонувати як процес реєстрації, оскільки це хороший спосіб для реклами в додатку доставки продуктів. Інші варіанти – через адресу електронної пошти та номер мобільного телефону, або деякі навіть працюють через автентифікацію за відбитками пальців. Покупці можуть зробити вибір на свій смак.

2.2. Розумні методи пошуку

Ця головна функція має бути ефективною, але простою. Вона має забезпечувати користувачам зручність пошуку продуктів у програмі доставки їжі. Розробник програми має додати різноманітні фільтри, щоб зробити операції гладкими та легкими для користувача. Пошук повинен дозволити їм якомога швидше дістатися до улюбленого елемента зі списку. У наш час розробник програми також повинен забезпечити можливість пошуку за допомогою голосових команд або навіть завантаження зображень, щоб повністю задовольнити клієнтів.

2.3. Рекомендовані продукти

Перевага електронної комерції полягає в тому, що певні продукти позначено як рекомендовані в додатку, наприклад, «бестселер», «з найвищим рейтингом» тощо. Це також може бути корисно для користувачів додатку. Така функція зустрічається на деяких торгових веб-сайтах, коли на основі попередньої історії пошуку користувача, внизу програми відображається рекомендований список.

2.4. Push-сповіщення

Push-сповіщення є найкориснішою функцією для користувачів програм, а також власників програм. Push-сповіщення – це спливаючі повідомлення, які надсилаються безпосередньо на пристрої користувачів програми. Вони

продемонстрували значну цінність у утриманні користувачів програми та покращенні продажів.

2.5. Підтримка в чаті

Завдяки функції підтримки в чаті покупці можуть робити свої запити та безпосередньо взаємодіяти з командою підтримки в чаті компанії, що займається продажом продуктів, щодо будь-яких проблем, які їм потрібно негайно вирішити.

2.6. Безпечні методи оплати

Онлайн оплата є важливою функцією. Можна інтегрувати програму доставки продуктів із кількома варіантами оплати, включаючи оплату за допомогою кредитної картки, дебетової картки, PayPal або Stripe або навіть варіантом готівки при доставці. Однак, щоб стимулювати бажання користувача оплатити онлайн, його потрібно переконати, що йому надано безпечні методи оплати.

3. Функції для адміністратора та власників продуктових магазинів

3.1. Спрощений бекенд

Внутрішня інформаційна панель конструктора програм електронної комерції дуже зручна. Інформаційна панель допоможе легко керувати магазином. Ця функція дозволяє адміністратору спостерігати на інформаційній панелі за всіма важливими функціями, такими як: замовлення, оплата та доставка. З ними немає потреби в кодуванні. Візуальна та інтерактивна інформаційна панель дозволяє швидко редагувати та налаштовувати, і це миттєво відображається в програмі.

3.2. Синхронізація в реальному часі та керування вмістом

Ця функція дає змогу оптимізувати вміст додатка та, у свою чергу, покращити взаємодію з користувачем. Для версії від кількох постачальників вміст оновлюється та керується декількома власниками магазину. Зазвичай доступні такі варіанти керування вмістом:

Редагувати інформацію про продуктовий магазин

Редагувати список товарів (приклад – видалення товарів, зміна ціни)

Якщо уже створений продуктовий веб-сайт електронної комерції, і на його основі необхідно створити програму, то повинна бути реалізована синхронізація веб-сайту та програми в режимі реального часу.

3.3. Налаштування домашньої сторінки своїми руками

Завдяки цьому плагіну часи, коли поправка вимагала програмування або кодування, давно минули. Візуальний редактор на серверній частині дозволяє адміністратору налаштовувати кожну деталь домашньої сторінки за допомогою простого методу перетягування. Вони можуть переміщувати та змінювати фотографії, додавати релевантні дії переспрямування на кожну деталь/банер/повзунок і навіть мати банери із зворотним відліком для програми.

4. Додаткові функції

Для того, щоб виділитися серед численних аналогів, існує багато функцій, за допомогою яких можна покращити процес купівлі товарів. Ось деякі з цих додаткових функцій:

4.1. Відгуки та коментарі

Потрібно постійно пам’ятати про концепцію розвитку та вдосконалення, щоб мати популярний додаток для покупки продуктів. Тому коментарі та огляди є чудовим джерелом для аналізу слабких сторін додатка, якщо вони є. Розробники та замовники не стримуються і дають оцінки всьому – і самому сервісу, і якості придбаного товару, і сервісу експедиційного персоналу тощо.

4.2. Спрощення повторних замовлень

Знайти та перевірити повторні замовлення стає легше та швидше. Це відбувається, коли клієнти можуть систематично вибирати продукти, які купують, використовуючи історію своїх попередніх покупок. Використання цієї функції в додатку для покупки продуктів, безумовно, покращить досвід онлайн-покупок користувачів, а процес повторного замовлення буде гладким і швидким.

4.3. Список бажань

Можливість збереження продуктів у кошику є важливою функцією продуктового додатку. Клієнти використовують список побажань, щоб зберегти товар, який вони хотіли б придбати пізніше. Безперечно, мати цю опцію зручно. Важливо відзначити, що покупці обов’язково захотять мати можливість давати назви своїм спискам, а також мати можливість встановлювати нагадування.

1.2 Огляд аналогів і аналіз існуючих варіантів розробки веб-додатку для доставки продуктів

Доставка – це ринок послуг з доставки невеликих поштових відправлень або пакетів практично в будь-яку точку міста, в якому він працює.

Останнім часом кількість малих і середніх підприємств, які займаються експрес-доставкою, різко зросла. Проте, поза сумнівом, проблеми з цією перспективною, потрібною та популярною справою є.

Однією з головних проблем аналітиків є те, що присутність великих іноземних компаній у цій галузі рішуче вийшла на внутрішній ринок і не здасть своїх позицій головних конкурентів малих компаній.

Друга проблема – це зниження товарообігу, оскільки вартість обслуговування постійно зростає. Це пов’язано з деякими фінансовими кризами та збільшенням вартості доставки, які вплинули на компанії, що працюють у цій сфері, з підвищенням тарифів.

Також слід зазначити відсутність чіткої законодавчої бази, яка б регулювала роботу даного сегменту ринку. Наразі, щоб заповнити цю прогалину та виробити єдині правила, проводяться консультації між експертами та власниками кур’єрських компаній.

Незважаючи на певні ризики, перспективи зростання та процвітання бізнесу набагато більші. Так, наприклад, за оцінками експертів сервісу Glovo, ринок доставки вантажів в Україні у 2020 році зріс у 6-7 разів порівняно з попереднім роком. Крім того, у 2020 році поставки тільки в супермаркети зросли в 15-18 разів через те, що omni-channel став новою нормою в роздрібній торгівлі через пандемію [6].

Замовлення їжі в Інтернеті означає, що клієнт шукає ресторан в Інтернеті та перенаправляється на веб-сайт або в додаток, за допомогою якого він може замовити їжу в цьому ресторані. Клієнти зазвичай мають можливість отримати їжу з доставкою або забрати її в магазині, а також можуть оплатити кредитною карткою

або готівкою. Багато систем онлайн-замовлень також розроблено для ведення облікових записів для постійних клієнтів, щоб полегшити замовлення.

Онлайн-замовлення вперше з’явилося наприкінці 1990-х років, коли вперше зародився Інтернет, і повільно набирав обертів на початку 2000-х років, коли великі компанії з виробництва піци почали надавати онлайн-замовлення та послуги з доставки. Однак лише в поточному десятилітті розвиток цифрових технологій, таких як смартфони, різні програми та соціальні мережі, призвело до того, що онлайн-замовлення стали нормальною частиною нашого повсякденного життя. Послуги з доставки більше не обмежується використанням лише великими компаніями, але також все частіше використовується невеликими місцевими магазинами.

Щоб зрозуміти популярність онлайн-замовлень, потрібно розглянути сучасну поведінку клієнтів. Ключ до зростання цього способу купівлі – це зручність, яку він створює для споживачів. Ми живемо зайнятим життям і працюємо довше, тому нас приваблюють системи, які економлять час і енергію. Завдяки мобільним технологіям онлайн-замовлення стало ще простіше, тому багато компаній запровадили зручні для мобільних пристроїв веб-сайти та програми, які можна завантажувати безпосередньо на смартфони, що дозволяє клієнтам замовляти їжу з будь-якого місця. Business Insider Intelligence показує, що мобільне замовлення їжі, як очікується, стане індустрією вартістю 38 мільярдів доларів США з 2020 року та пізніше.

Поява компаній-агрегаторів, таких як Uber Eats і Menulog, які пропонують доступ до кількох ресторанів через єдиний онлайн-портал, дала клієнтам додаткову можливість порівнювати меню та ціни, читати відгуки і знаходити місцеві ресторани.

Онлайн-замовлення також має переваги для бізнесу. Наприклад, такий веб-сайт-агрегатор, як Menulog, може залучити ще більше трафіку до компаній завдяки розширеному охопленню та, у свою чергу, збільшити продажі та впізнаваність бренду. Підприємствам також надається можливість збільшити утримання клієнтів. Завдяки персоналізації процесів замовлення, завдяки збереженню таких даних

клієнтів, як адреси, дані картки та звичайні замовлення, клієнти економлять час щоразу, коли роблять замовлення.

Наприклад, Domino's заохочує своїх клієнтів створити «Профіль піци», де вони зможуть зберігати своє улюблене замовлення піци та контактну інформацію, що дозволить їм замовляти свою улюблену піцу, просто надіславши текстове повідомлення емодзі. Було встановлено, що як тільки клієнт створює обліковий запис для компанії, 80% не переходять до іншої компанії, і з такою зручністю та ефективністю легко зрозуміти чому. Якщо додати до цього рекламні пропозиції та ремаркетинг електронною поштою, то легко зрозуміти, чому онлайн-замовлення стало виграшом для ресторанів та їхніх голодних шанувальників.

Онлайн-замовлення ще відносно нове направлення, тому деякі компанії все ще не наважуються перенести свої послуги в Інтернет. Однак через швидке зростання популярності цієї системи компаніям незабаром стане неможливо не стати частиною онлайн-світу. Наведемо кілька прогнозів щодо майбутнього онлайн-замовлень і того, що вони означатимуть для харчової промисловості.

1) Аутсорсинг послуг доставки

Як обговорювалося вище, веб-сайти-агрегатори, такі як Uber Eats і Menulog, виникли завдяки успіху онлайн-замовлень. Однак було зроблено ще крок вперед завдяки впровадженню нових служб доставки, таких як DoorDash і Deliveroo, які дозволяють споживачам порівнювати ресторани, замовляти онлайн або через додаток і доставляти їжу з місць, які традиційно не пропонують доставку.

Нові служби доставки зараз знаходяться в прямій конкуренції з веб-сайтами-агрегаторами, оскільки вони розширюють ринок до нової групи ресторанів і клієнтів, надаючи як можливості онлайн-замовлень, так і послуги доставки з додатка або веб-сайту.

2) Всебічне впорядкування

Незважаючи на те, що онлайн-замовлення вже трансформувалося з комп’ютерних систем замовлення на більш зручні для мобільних пристроїв варіанти, такі як веб-сайти, що реагують на смартфони, і спеціальні додатки для замовлення їжі, прогнозується, що ця тенденція до надання клієнтам можливості

замовляти та оплачувати з будь-якого місця буде розвиватися. Це стало помітним через впровадження послуг із замовлення їжі на платформах соціальних мереж. Дозволивши клієнтам робити замовлення з вашого веб-сайту, мобільного додатка, додатків-агрегаторів і сторінок у соціальних мережах, клієнту справді забезпечать максимальну зручність замовляти й оплачувати, бронювати столик, переглядати меню та читати відгуки в одному місці, будь-де, з легкістю на будь-якому пристрої.

3) Прогнозування замовлень

Дані, які можна зібрати через онлайн-замовлення, все частіше використовуватимуться підприємствами для повторних продажів. У майбутньому ресторани зможуть використовувати алгоритми у своїх системах онлайн-замовлень, щоб передбачити наступне замовлення клієнта на основі попередніх замовлень. Це включає тип і кількість замовленої їжі, а також час, день і місце замовлення.

4) Онлайн-замовлення випереджатимуть замовлення по телефону

Однією з ключових переваг онлайн-замовлень є менша ймовірність людської помилки. Під час замовлення по телефону інколи мовний бар'єр або погане з'єднання та фоновий шум можуть призвести до того, що ресторан отримає неправильне замовлення клієнта. Онлайн-замовлення мінімізує цю проблему та може допомогти скоротити час очікування. Оскільки все більше компаній впроваджують системи онлайн-замовлень, замовлення по телефону буде майже повністю замінено.

5) Ціни можуть впасти в ресторанах

Із запровадженням нових служб доставки багато клієнтів вирішують обідати вдома. Через це ресторани намагатимуться конкурувати та залучати людей у свої двері, пропонуючи спеціальні акції на обід у закладі та знижуючи ціни на страви та напої.

6) Доставка без водіїв і безпілотників

Харчова промисловість наразі швидко адаптувалася до нових технологій, тому після 2021 року, коли було створено й удосконалено ще більше технологій, можна побачити, що харчова промисловість швидко капіталізується. Google уже деякий час працює над безпілотним автомобілем, і хоча ці автомобілі спрямовані на

підвищення безпеки на дорогах, коли це стане нормою, одного дня ми зможемо побачити безпілотні автомобілі, які доставляють їжу, щоб скоротити витрати на оплату праці для підприємства. Подібним чином технологія дронів стає все більш нормалізованою, і Amazon планує використовувати дрони для доставки посилок клієнтам менш ніж за 30 хвилин. Якщо ця ідея запрацює, ми можемо зрештою побачити, що їжу також доставляють за допомогою дронів.

Ще однією позитивною зміною є те, що під час карантину користувачі звикли замовляти продукти харчування та інші товари через інтернет. Незважаючи на те, що темпи зростання зменшилися в міру послаблення обмежень, вони залишаються високими. Як приклад перспективності програмної системи доставки товарів на замовлення можна навести зростаючий попит на існуючі кур'єрські служби.

Таким чином, у 2020 році у «Glovo» кількість активних користувачів зросла на 30%, а частота замовлень – на 20%, порівняно з 2019 роком середня перевірка замовлення зросла на 24 %; кількість активних кур'єрів на платформі зросла на 20% [6].

Zakaz.ua повідомляв, що темпи зростання сервісу досягли 200% порівняно з 70% у минулі роки; кількість замовлень з початку карантину зросла на 74,5%, сервіс запрацював у семи нових містах України; сервіс вийшов на ринок Узбекистану; вдвічі збільшив кількість користувачів мобільних додатків; в середньому збільшив кількість співробітників у центральному офісі на 30%, у магазинах на 37%, у кур'єрів на 39%[6].

За шість місяців 2020 року «Нова пошта» доставила понад 128 млн пакунків і відправлень, що на 32% більше, ніж за відповідний період 2019 року. Збільшення обсягів пов'язане з тим, що клієнти почали купувати за нижчими цінами, але частіше. Ця тенденція була помічена на початку карантину і спостерігалася до травня, коли карантин було знято.

Особливістю другого кварталу стало зростання обсягів кур'єрської доставки на 35% [7].

Glovo

Glovo – це програма, яка дозволяє отримати будь-який товар, який поміститься в кошику кур'єра у вашому місті, за досить короткий термін. Вони з’єднують користувачів, компанії та кур’єрів і роблять можливим: підписання договорів з ресторанами та супермаркетами, створення комфортних умов для роботи тощо.

Проєкт народився з метою змінити спосіб, у який користувачі отримують усе необхідне, зробивши купівлю товарів у місті швидшою та легшою. У Glovo прагнуть полегшити кожному доступ до будь-чого в своєму місті та мати стійкий економічний, соціальний та екологічний вплив. Компанія займає перше місце в галузі доставки та використання, на неї припадає близько 75-80% усіх поставок продуктів харчування та продуктів харчування з ресторанів і супермаркетів.

Реєстрація відбувається при вході за допомогою соціальної мережі «Facebook» або за номером телефону та є обов’язковою для всіх користувачів. Всі переходи між вікнами супроводжуються плавними анімаціями. На рис. 1.1 зображено головне вікно програми, що з’являється одразу після реєстрації нового аккаунту користувача або вхід у вже існуючий.

Рисунок 1.1 – Головне вікно додатку

«Zakaz.ua»

Одна з найбільших компаній, що спеціалізується на доставці продуктів додому. Ви можете оформити замовлення за допомогою програми на своєму пристрої з операційною системою Android, IOS або через звичайний браузер будь-якої системи. Співпрацюють з такими супермаркетами, як «Метро», «Ашан», «Новус», «Мегамаркет», «Фуршет», «Варус». Одночасно проводяться різноманітні рекламні заходи.

Тобто в додатку «Zakaz.ua» (рис. 1.2) також можна отримати всі знижки та пропозиції, що діють у продуктовій мережі. Також вони можуть доставити будь-яку побутову хімію, посуд, одяг тощо, які є в магазинах-партнерах компанії [9].

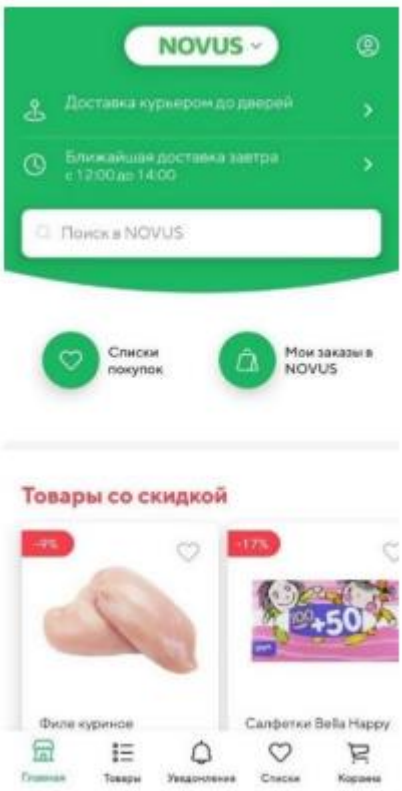

Рисунок 1.2 – Головне вікно додатку

При першому запуску програми необхідна реєстрація користувача за номером телефону, інакше програма не працюватиме. На відміну від Glovo, тут немає

можливості авторизуватися або реєструватися через соціальні мережі. Є меню, за яким можна вибрати, з якого супермаркету відправляти.

На відміну від аналогів, існує графік доставки, згідно з яким не завжди вдається отримати товар максимально швидко.

«Bolt Food»

Це наймолодша компанія на українському експрес-ринку, заснована 29 жовтня 2020 року. Як випливає з назви, він спеціалізується на доставці з різних супермаркетів і ресторанів, і наразі їх близько 150. На початку послуги доставка буде безкоштовною, але в подальшому обіцяють, що ціни будуть на 10-15% нижчими, ніж у конкурентів, кажуть у компанії. Наразі він працює лише в Києві, але цього року планують запустити ще в кількох великих містах. Під час карантину доставка здійснюватиметься лише безконтактним способом, а оплата – банківською картою.

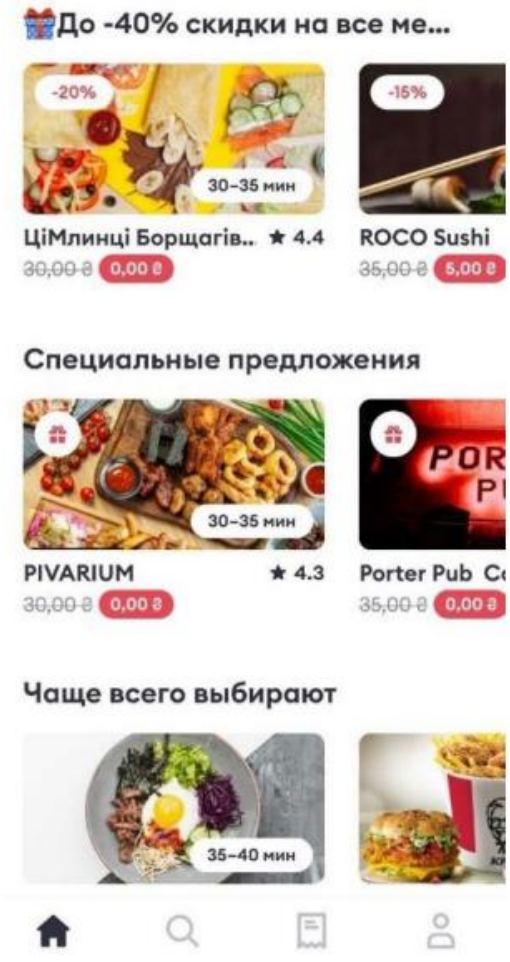

Рисунок 1.3 – Головне вікно додатку

Додаток має зручний великий інтерфейс і панель навігації, що дозволяє легко шукати і фільтрувати товари за типом і приналежністю до певного типу (рис. 1.3). Під час першого запуску програма просить ввести номер телефону, прізвище, ім’я та електронну адресу для реєстрації.

Рисунок 1.3 – Головне вікно додатку

1.3 Технології розробки мобільних застосунків

Технологія розробки мобільних застосунків відноситься до різних інструментів, фреймворків, компонентів, бібліотек та інших технологій, які використовуються для створення програм для мобільних пристроїв. Прикладами мобільних пристроїв є смартфони, ноутбуки, планшети та розумні годинники. Дві домінантні мобільні платформи на ринку – Android та iOS.

При розробці застосунків для мобільних пристроїв необхідно звернути особливу увагу на обмеження кожного пристрою.

Ці обмеження варіюються від того, як користувач взаємодіє з пристроєм, його обчислювальної потужності, розміру екрана, унікальних функцій і можливостей конкретного пристрою, платформи (Android чи iOS) і чи буде програма використовуватися на кількох платформах.

Останній момент є важливим, оскільки слід визначити, чи планується розроблення для однієї чи кількох платформ – зараз чи незабаром.

1. Програми jQuery [1, 20] завантажуються швидко

Мабуть, найбільшою перевагою створення програм за допомогою jQuery Mobile є швидкість завантаження програм (рис. 1.4).

Рисунок 1.4 – Середовище розробки jQuery

Як легкий фреймворк, він заснований на бібліотеці jQuery і має сильний акцент на сумісності, тому його можна використовувати для всіх наступних продуктів:

- Android
- Windows
- iOS

jQuery усуває потребу в мовах, що стосуються пристрою, тому розробники можуть використовувати такі стандарти, як JavaScript, HTML5 і CSS3. Це також

пом’якшує проблеми між браузерами та підтримує ряд розмірів і роздільної здатності екрана.

jQuery також можна використовувати для створення користувацьких тем за допомогою веб-програми ThemeRoller. Він надає різні інструменти, які спрощують редагування тем, у тому числі перетягування кольорів, макети сторінок і шаблони заголовків.

2. React Native інтегрується з іншими програмами

Як фреймворк з відкритим вихідним кодом, React Native можна використовувати для створення кросплатформних нативних програм (рис. 1.5). У поєднанні з JavaScript програми React Native не відрізняються від програм, створених з використанням інших мов, таких як Swift, Java та Objective-C.

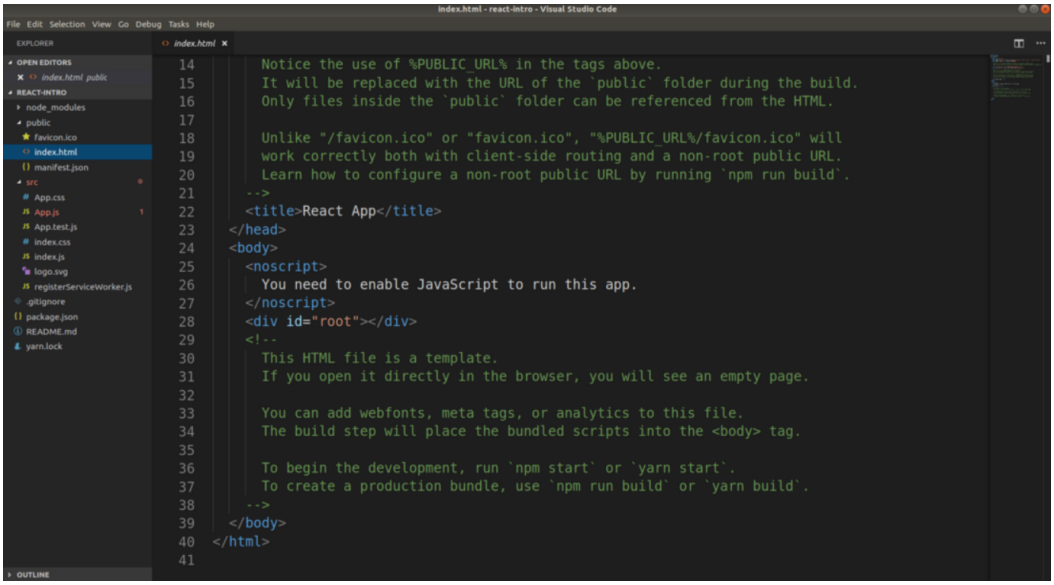

Рисунок 1.5 – Середовище розробки React Native

React Native має перевагу над іншими фреймворками завдяки своєму декларативному стилю програмування та компонентам для багаторазового використання інтерфейсу користувача.

Його можна поєднувати з рідним кодом, написаним іншими мовами програмування, що робить його дуже гнучким і є хорошим вибором для додавання нових функцій до існуючих програм.

З появою нової функції під назвою Hooks популярність фреймворка серед служб розробки JavaScript, ймовірно, продовжиться. Ця функція дозволяє розробникам отримувати доступ до функцій React без написання класу, що робить підхід більш інтуїтивним.

3. IntelliJ IDEA – це інтегроване середовище розробки (IDE) для мов JVM, розроблене для максимальної продуктивності розробників (рис. 1.6). Він виконує рутинні й повторювані завдання за вас, забезпечуючи розумне завершення коду, статичний аналіз коду та рефакторинг, а також дозволяє зосередитися на яскравій стороні розробки програмного забезпечення, роблячи це не тільки продуктивним, але й приємним.

IntelliJ IDEA – це кросплатформна IDE, яка забезпечує послідовну роботу в Windows, macOS та Linux.

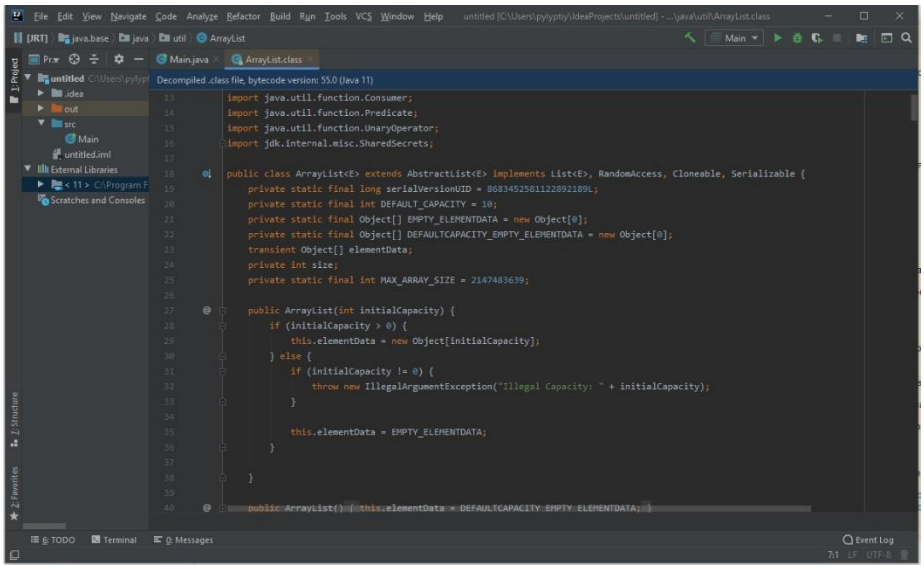

Рисунок 1.6 – Середовище розробки IntelliJ IDEA

Розробка сучасних застосунків передбачає використання кількох мов, інструментів, фреймворків і технологій. IntelliJ IDEA розроблена як IDE для мов JVM, але численні плагіни можуть розширити її, щоб забезпечити поліглот код.

IntelliJ IDEA випускається в трьох виданнях:

IntelliJ IDEA Ultimate: комерційне видання для JVM, веб- та корпоративного розвитку. Він включає в себе всі функції видання Community, а також додає

підтримку мов, на яких зосереджені інші IDE на платформі IntelliJ, а також підтримку різноманітних серверних і інтерфейсних фреймворків, серверів застосунків, інтеграцію з базою даних і профілювання, інструменти та інше.

IntelliJ IDEA Community Edition: безкоштовна версія на основі відкритого коду для розробки JVM та Android.

IntelliJ IDEA Edu: безкоштовне видання з вбудованими уроками для вивчення Java, Kotlin і Scala, інтерактивними завданнями програмування та спеціальними можливостями для вчителів для створення власних проєктів.

Плагіни в комплекті з IntelliJ IDEA, і доступні з коробки, додають підтримку деяких з найпопулярніших мов, а саме:

- Python (повна функціональність PyCharm);
- Ruby (повна функціональність RubyMine);
- PHP (повна функціональність PhpStorm);
- SQL [6] (повна функціональність DataGrip);
- Go (повна функціональність GoLand);
- JavaScript (повна функціональність WebStorm);
- TypeScript (повна функціональність WebStorm);
- JSON;
- HTML і XHTML;
- XML і XSL;
- XPath і XSLT;
- Таблиці стилів (CSS, Less, Sass).

Однією з найкращих речей IntelliJ IDEA є можливість налаштування. Можна налаштувати практично все: зовнішній вигляд IDE, розташування вікон інструментів і панелей інструментів, підсвічування коду тощо. Існує також безліч способів налаштувати редактор і налаштувати його поведінку, щоб прискорити навігацію та позбутися будь-яких додаткових елементів, які відволікають від коду. Також можна налаштувати кольори та шрифти для вихідного коду, виводу консолі, інформації про налагодження, результатів пошуку тощо. Можна вибрати з ряду

попередньо визначених колірних схем або налаштувати схему, щоб створити унікальне робоче середовище.

Саме тому середовищем розробки програмного забезпечення було вибрано IntelliJ IDEA [11].

В дипломному проєкту було вирішено обрати мову JavaScript. Давайте розглянемо її специфіку та переваги.

JavaScript є найбільш широко використовуваною мовою в області розробки в усьому світі. Вже більше 25 років JavaScript Frameworks керує Інтернетом.

JavaScript є однією з найпопулярніших мов і в даний час є вибором як для розробки front-end частини сайту, так і для серверної веб-розробки.

JavaScript – це проста об’єктно-орієнтована мова програмування, яка використовується для написання сценаріїв веб-сторінок на веб-сайтах. При застосуванні до HTML-документа JavaScript забезпечує динамічну інтерактивність веб-сторінки.

JavaScript дає змогу користувачам створювати сучасні веб-застосунки без перезавантаження сторінки. JavaScript часто використовується для оновлення інтерфейсу користувача DOM API шляхом динамічної зміни HTML і CSS. Він широко використовується в онлайн-застосунках. Нижче наведено деякі варіанти використання мови програмування JavaScript.

1. Веб-застосунки. JavaScript стає все більш популярним для розробки потужних веб-застосунків.

2. Веб-розробка. Для розробки веб-сторінок часто використовується JavaScript, щоб додати веб-сайту динаміки та надати конкретні функції. Мова JavaScript використовується в основному для перевірки введення даних на веб-сайтах, але також дозволяє виконувати складні завдання і надає користувачам можливість взаємодіяти з сайтом.

3. Мобільні застосунки [4, 5]. Мобільні телефони широко використовуються для доступу до Інтернету. Фреймворк JavaScript часто використовується при створенні мобільних застосунків. За допомогою React Native можна розробляти мобільні застосунки для кількох операційних систем. Він виявився одним із

найкращих фреймворків JavaScript для мобільних застосунків, тому що не потрібно писати різні коди для операційних систем iOS і Android.

4. Ігри. Для створення ігор також використовується JavaScript. Він має різні бібліотеки та рамки для створення 2D або 3D ігр. Механізми JavaScript, такі як PhysicsJS і Pixi.js, дозволяють створювати веб-ігри. WebGL, JavaScript API, також можна використовувати для відтворення 2D та 3D зображень у браузерах.

5. Презентації. З JavaScript можна розробляти презентації. За допомогою HTML фреймворка reveal.js можна створювати інтерактивні та привабливі презентації, які підтримують CSS 3D-перетворення, вкладені слайди та низку інших можливостей.

6. Серверні програми. Значна частина веб-програм мають серверну частину. Для створення вмісту та обробки HTTP-запитів потрібен JavaScript. NodeJS надає середовище з можливостями для запуску JavaScript на сервері.

7. Веб-сервери. Node.js найчастіше застосовують як веб-сервер. Сервери Node.js працюють швидко і не потребують буферизації або передачі фрагментів даних. Серверна частина веб-додатку, побудованому з Node.js, зможе працювати стабільно та ефективно, обслуговуючи навіть велику кількість звернень.

1.4 Постановка завдань розробки

Функціональним призначенням програми є автоматизація інформаційної системи доставки продуктів харчування користувачам. Хоча ця програма має обмежені функції, вона забезпечує вхід в обліковий запис адміністратора, клієнта, кур’єра, реєстрацію та збереження даних та інші функції.

Обліковий запис адміністратора дозволяє додавати потрібні товари, завантажувати зображення, описи товарів, кількість товарів на складі, змінювати написи та дизайн сторінок і веб-сайтів у режимі реального часу.

Обліковий запис кур'єра дозволяє додавати замовлення, які потрібно виконати, до власної черги, щоб жоден інший кур'єр не міг прийняти те саме замовлення знову. У профілі кур'єра є додаткова панель, де можна відстежувати

виконані або незавершені замовлення. Крім того, кур'єр також може переглянути всі замовлення, виконані ним за останній час.

Аккаунт клієнта дозволяє увійти в додаток і переглянути вивантажені товари на сайті, вибрати потрібний товар відповідно до категорії, також можна додатково вибрати кількість товару, яку клієнт хоче замовити.

Функціонал сайту та платформи дуже простий, але ефективний, тому що з точки зору замовника він ідеальний для роботи, оскільки є функції, які дозволяють редагувати сайт у теперішньому часі, а з боку споживача є можливість купити оптом. Платформа надає необхідні ресурси не тільки клієнтам і споживачам, а й виконавцям, а саме кур’єрам.

Система доставки їжі повинна досягати таких показників:

- Мета розробки таких систем – покращити та полегшити життя людини через доставку їжі. Інформаційна система може бути застосована до різних рівнів суспільства. Наприклад, у навчальних закладах, на масових заходах, торгових центрах, лікувально-реабілітаційних центрах тощо. Основне завдання – написати функціональний і простий сайт мовою програмування JS.

- Інформаційну систему доставки їжі можуть використовувати незалежні розробники, які не мають власної компанії та закладів, наприклад ресторанів. Ця інформаційна система буде розроблена та призначена виключно для клієнтів та користувачів. Однак, якщо ресторани не мають власного веб-сайту, вони можуть скористатися послугами таких платформ за додаткову плату.

- Перш за все, веб-додаток повинен мати яскравий і зрозумілий інтерфейс, а основні функції веб-сайту повинні відображатися перед користувачем на головній сторінці. Крім того, також важливо правильно відображати інтерфейс на мобільних телефонах і ПК. По-друге, ви можете відстежувати статус своєї заявки або сайту на основі прийнятих замовлень.

- Чіткий, правильний, структурований код, який дозволяє прийняти інформацію від користувача, тобто замовлення, обробити цю інформацію, тобто записати замовлення та відправити його в ресторан чи продуктовий магазин. Замовлення повинні оброблятися в порядку черги, тобто замовлення обробляються

від першого користувача до останнього, але без затримки, тобто неправильні замовлення автоматично виправляються. Також важливо контролювати замовлення, які надходять на сайт або платформу.

- Якщо код написаний неправильно структуровано, автоматична обробка замовлень на сайті або платформі припиниться. Сайт є більш гнучким матеріалом, тому виправляти баги легше, ніж на платформі. Тому деякі платформи навіть мають власну техпідтримку, яка обробляє звернення користувачів щодо окремих збоїв у роботі програми.

- Така інформаційна система дуже гнучка, тому її можна підключати до таких завдань, як індивідуальна роздача замовлення кожному кур’єру, який знаходиться неподалік від місця видачі замовлення. Крім того, можуть бути кур’єри, які індивідуально обробляють замовлення з ресторанів, продуктових магазинів тощо.

- Система може розподіляти між собою замовлення, надіслані різним кур'єрам. Крім того, необхідно додати функцію написання листів до ТП, які будуть автоматично сортуватися за темою написання та надсилатися у відповідний відділ, який розглядає конкретну справу. Також є досить важлива функція, яка відстежуватиме кур’єра, який має доставити замовлення користувачеві на Google Maps. Для комфортного користування необхідно додати функцію, щоб чітко відстежувати час доставки замовлення.

Веб-додаток [2], що розробляється буде призначений для використання у сфері доставки їжі споживачам.

Проблема, розглянута в даному дипломному проєкті, є актуальною та має широке практичне призначення як на сьогоднішній день, так і в майбутньому.

Метою даного проєкту є розробка системи постачання їжі для споживачів.

Для досягнення поставленої мети необхідно вирішити основні завдання:

- аналіз платформ та сайтів ресторанів, які доставляють їжу споживачам;

- уточнення вимог до написання інформаційної системи, яка дозволить без перешкод користуватися сайтом, який буде розподіляти та відправляти їжу користувачам;

- розробка алгоритму, інтерфейсу, бази даних і реалізації програми.

У відповідності до проведеного аналізу поставлені основні функціональні задачі перед системою:

- прискорення розвитку більш інноваційних систем для споживачів;
- зниження витраченого користувачем часу на покупку обраних товарів;
- збільшення ефективності ведення обліку несправностей;

В цілому, інформаційна система, що розроблюється, повинна забезпечити та підвищити рівень задоволення клієнтів та значно полегшити ресторанний бізнес через можливість доставляти їжу на задану відстань та у заданий проміжок часу.

Розроблену інформаційну систему можуть використовувати усі бажаючі споживачі, які хочуть полегшити життя та економити час.

2 ПРОЄКТНІ ТА ТЕХНІЧНІ РІШЕННЯ

2.1 Опис засобів реалізації програмного продукту

2.1.1. Вибір технологій розробки веб-додатків

В дипломному проєкті обрано мову JavaScript [3], переваги та специфіка якої детально описана в підрозділі 1.3. JavaScript є найбільш широко використовуваною мовою в області розробки в усьому світі.

CSS (Cascading Style Sheets) – це скоріше не мова програмування, а правила оформлення HTML документа. CSS дозволяє розробнику встановлювати розмітку сторінки, розмір шрифтів, колір, заголовки, а також змінювати шрифт. Можна сказати, що CSS безпосередньо відповідає за візуальну складову сайту, так як жоден HTML документ не може обійтися без CSS.

React – це фреймворк JavaScript з відкритим кодом, який насправді не є фреймворком. Але наразі це найпоширеніша технологія фронтенд розробки у світі. React, який розроблений і досі підтримується Facebook і активною спільнотою з відкритим кодом, насправді є «бібліотекою» JS.

React.js відповідає за побудову ієрархії або відтворення компонентів інтерфейсу користувача та забезпечує підтримку як інтерфейсу, так і сервера.

Як і всі фреймворки та бібліотеки JavaScript, основним випадком використання React є створення динамічних інтерфейсів користувача, які оновлюються на стороні клієнта (користувача). Сторінку не потрібно оновлювати у браузері, вона оновлюється автоматично. Точно так, як це робить стрічка Facebook. Ось чому гігант соціальних мереж інвестував у розробку власної бібліотеки JavaScript, спеціально адаптованої до потреб його власних програм.

React зарекомендував себе як найбільш використовувана фреймворк/бібліотека фронтенду JS у світі, оскільки він популярний як серед розробників програмного забезпечення, так і серед спонсорів проєктів.

Але існують інші бібліотеки та фреймворки JavaScript, які розробники «люблять» використовувати майже так само. У звіті про стан JavaScript за 2019 рік

Svelte та Vue.js вважаються майже такими ж популярними (89%, 88% і 87% відповідно), як і React на основі рівня задоволеності розробників.

Мінімальним кваліфікаційним критерієм є або має бути те, що даний фреймворк або бібліотека добре технічно відповідає специфікаціям програми. В ідеалі вибір буде гнучким для потреб широкого діапазону потенційних специфікацій. Це дозволить одній команді працювати з різними або всіма програмами роботодавця.

Також, якщо розробники хочуть працювати з мовою кодування, фреймворком або бібліотекою, це підвищує їхній рівень задоволеності роботою, що позитивно впливає на продуктивність і рівень утримання, що в кінцевому підсумку впливає на кінцевий результат.

Але є й інші причини, чому React зараз є найпопулярнішим вибором для розробки інтерфейсу. Одна з них полягає в тому, що на ринку більше розробників React, ніж спеціалістів у будь-якій іншій доступній системі/бібліотеці JS.

Розумний спонсор проєкту вирішить розвивати технології, максимально орієнтовані на майбутнє. Великий пул потенційних співробітників є одним із ключових факторів для перспективного проєкту розробки програмного забезпечення. Якщо найняти нових членів команди для збільшення ресурсів або заміни тих, хто покидає роботу, особливо складно, це може виявитися серйозною проблемою, яка зрештою стане витратою – прямо чи опосередковано. Зарплати можуть стати завищеними або проєкти затримуватися та/або поточна робота зірватися.

Існують інші причини, чому React можна вважати особливо перспективною технологією. Бібліотека підтримується Facebook, однією з найбільших компаній у світі. Він також має особливо активну спільноту з відкритим кодом, частково завдяки високій кількості розробників React. Поєднання підтримки Facebook і великої та залученої спільноти з відкритим кодом означає, що React постійно оновлюється та вдосконалюється, і в осяжному майбутньому не існує ризику застаріти.

2.1.2. Вибір системи керування базами даних

При виборі системи керування базами даних (СКБД) необхідно враховувати кілька факторів, що впливають на потреби проєкту. Ось кілька питань, на які варто відповісти для зорієнтування при виборі СКБД [17, 18]:

Тип даних. Який тип даних планується зберігати? Чи це структуровані дані (таблиці), наприклад, відносинні дані, або невстановлені дані, такі як тексти, зображення або відео?

Масштаб проєкту. Який очікується обсяг даних? Чи планується, що СКБД буде обробляти масивні об'єми даних в реальному часі, чи це менший проєкт з обмеженим обсягом даних?

Вимоги до продуктивності. Яка продуктивність потрібна? Наприклад, чи потрібно швидке виконання складних запитів, висока швидкість завантаження або швидке збереження даних?

Масштабованість. Чи очікується, що проєкт буде масштабуватися з часом? Чи потрібна можливість легкого розширення ресурсів СКБД, щоб відповідати зростаючим потребам?

Microsoft Access (повна назва Microsoft Office Access) – це система керування базами даних від Microsoft, яка входить до пакету офісних програм Microsoft Office. Він має широкий спектр функцій, включаючи пов’язані запити, сортування за різними полями, зв’язок із зовнішніми таблицями та базами даних. За допомогою вбудованої мови VBA можна писати підпрограми, які працюють з базами даних у самому Access.

ADO, OLE DB і Open Database Connectivity (ODBC) є його компонентами. Крім Microsoft Jet Database Engine, MSDASQL (постачальник ODBC для OLE DB) і Remote Data Services (RDS), існує кілька застарілих компонентів.

Система керування базами даних від Microsoft під назвою Access поєднує в собі реляційний Microsoft Jet Database Engine з графічним інтерфейсом користувача та інструментами для розробки програмного забезпечення. Крім того, він може імпортувати або посилатися безпосередньо на дані, що зберігаються в інших програмах і базах даних.

База даних Oracle – це набір даних, які розглядаються як одиниця. Метою бази даних є зберігання та отримання відповідної інформації. Сервер баз даних є ключем до вирішення проблем управління інформацією. Загалом, сервер надійно керує великою кількістю даних у багатокористувацькому середовищі, так що багато користувачів можуть одночасно отримати доступ до тих самих даних. Все це досягається з високою продуктивністю. Сервер баз даних також запобігає несанкціонованому доступу та надає ефективні рішення для відновлення збоїв.

Oracle Database – це перша база даних, розроблена для корпоративних мережевих обчислень, найбільш гнучкий та економічно ефективний спосіб керування інформацією та додатками. Корпоративні мережні обчислення створюють великі пули стандартних компонентів в галузі модульних сховищ і серверів. Завдяки цій архітектурі кожну нову систему можна швидко надати з пулу компонентів. Немає потреби в пікових навантаженнях, оскільки ємність можна легко додати або перерозподілити з пулів ресурсів за потреби.

Під час розробки застосунку було спроєктовано базу даних (рис. 2.1).

Атрибути:

Autentification – відповідає за авторизацію.

Storage – відповідає за зберігання картинок.

UserData – відповідає за зберігання id користувача, містить Data – відповідає за зберігання даних користувача та Added part/added words – відповідають за зберігання додаткового тексту.

Articles – містить id article, і в той час містить Article – відповідає за зберігання опису картинок та Level – зберігає дані про рівень користувача, Part – містить id та заголовок.

Content – відповідає за зберігання контенту. Він включає в себе Word, Sentence, articleParts, article Part – які відповідають за зберігання тексту та його id.

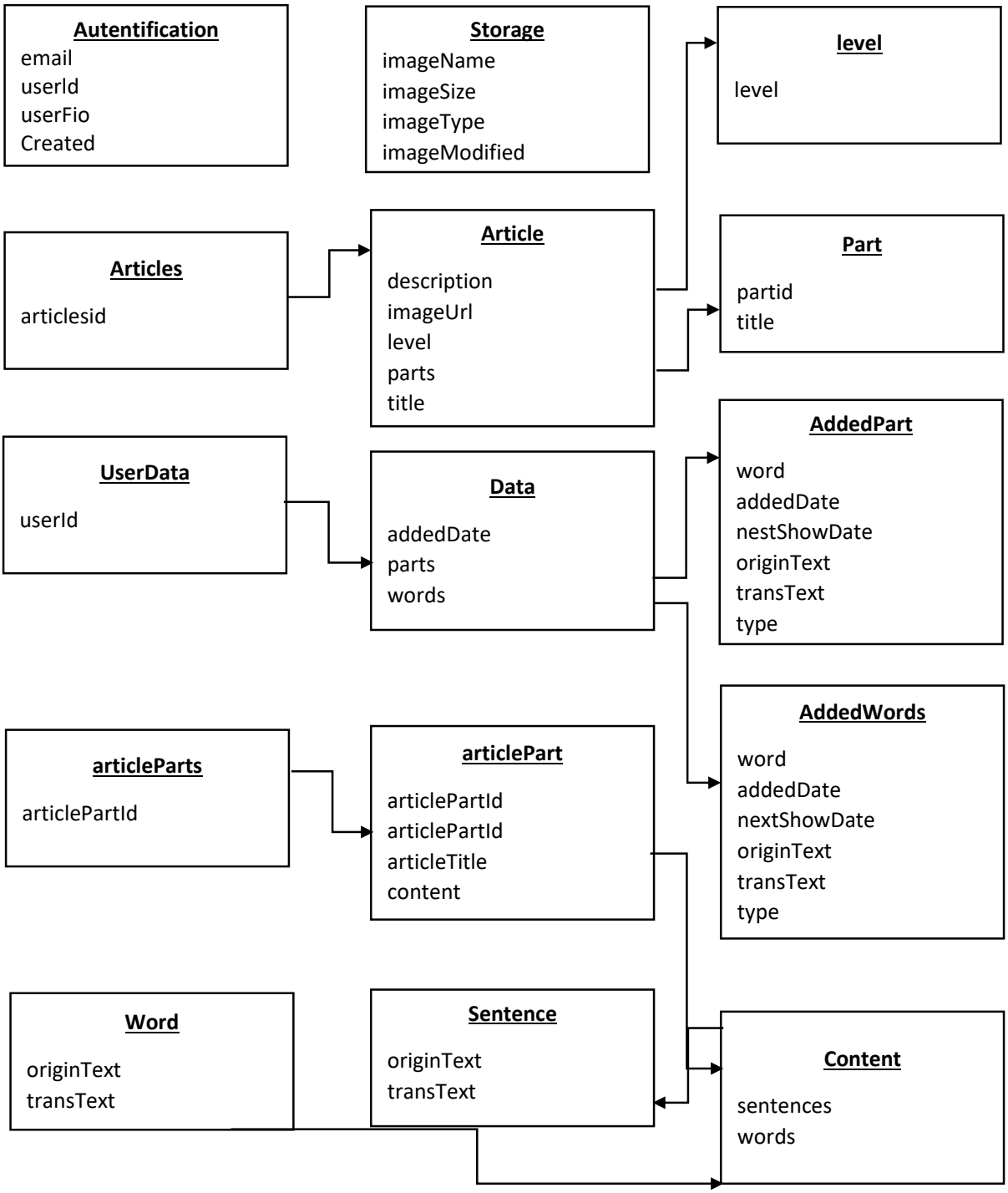

Рисунок 2.1 – Схема бази даних застосунку

2.2 Проєктування БД

Створення контекстної карти А-0 вважається початком розробки проєкту веб-додатку. Адже контекстні графи містять короткі та лаконічні дані, які

характеризують взаємодію веб-додатку з його середовищем та описують його вміст [9].

Важливі компоненти діаграми включають:

• Вхідні дані: інформація та матеріали, які існують на етапі ідентифікації продукту.

• Вихідні дані: наслідки, до яких призведе впровадження продукту.

• Управління – інформаційні та матеріальні дані, які сприяють реалізації проєкту.

• Інструменти: люди, обладнання та програмне забезпечення, які допомагають аналізувати ключові компоненти графіка.

• Вхід: замовлення клієнта у веб-додатку.

• Вихідні дані: повні замовлення та перевірки.

• Адміністрування: правила обслуговування, українське законодавство, правила оплати клієнтів.

• Механізми: адміністратори, ресторатори.

Діаграма A-0 (рис. 2.2) розроблена в програмі UML Diagram.

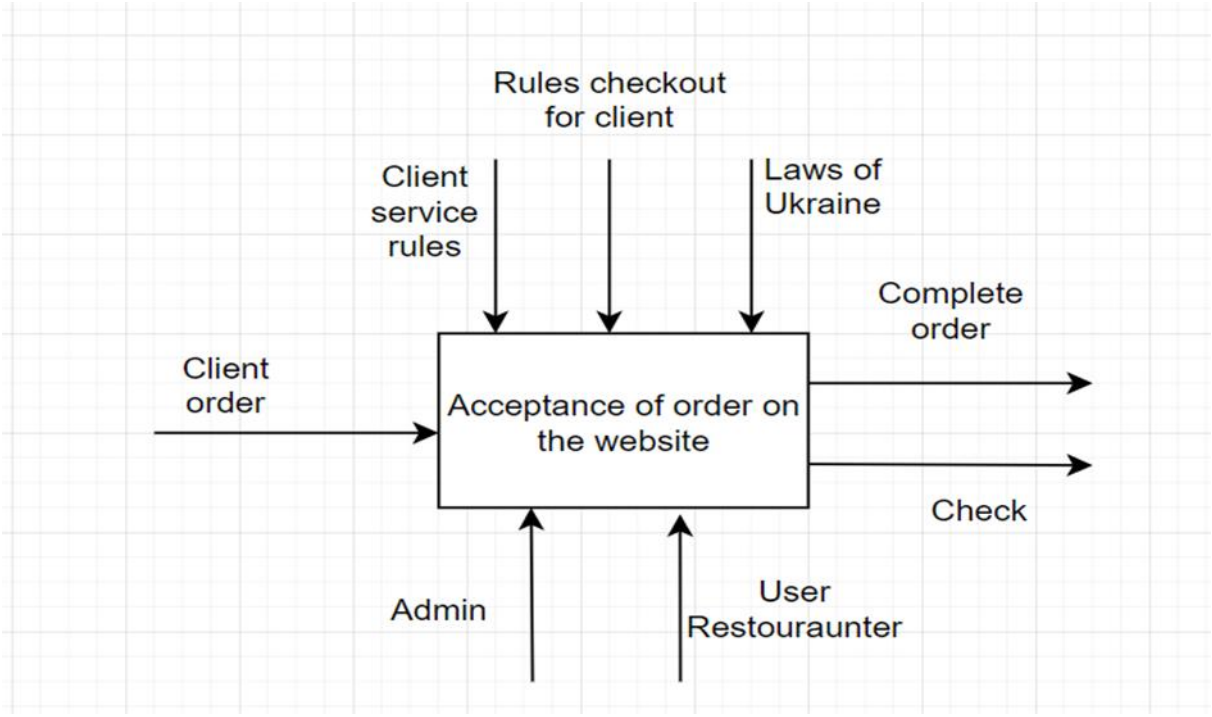

Рисунок 2.2 – Діаграма A0

Суть контекстного графа полягає в тому, щоб описати основну суть веб-додатку, що в свою чергу призводить до необхідності створення декомпозицій. Декомпозиція IDEF0 надає інформацію про функціональність веб-додатків.

Принцип розкладання:

• замовлення клієнтів

• Вибір коледжу

• Вибір меню та кухні

• Звіт про оплату: позитивний або негативний. Якщо результат позитивний, наступним кроком є оформлення замовлення та оформлення чека, якщо результат негативний, зверніться до служби підтримки.

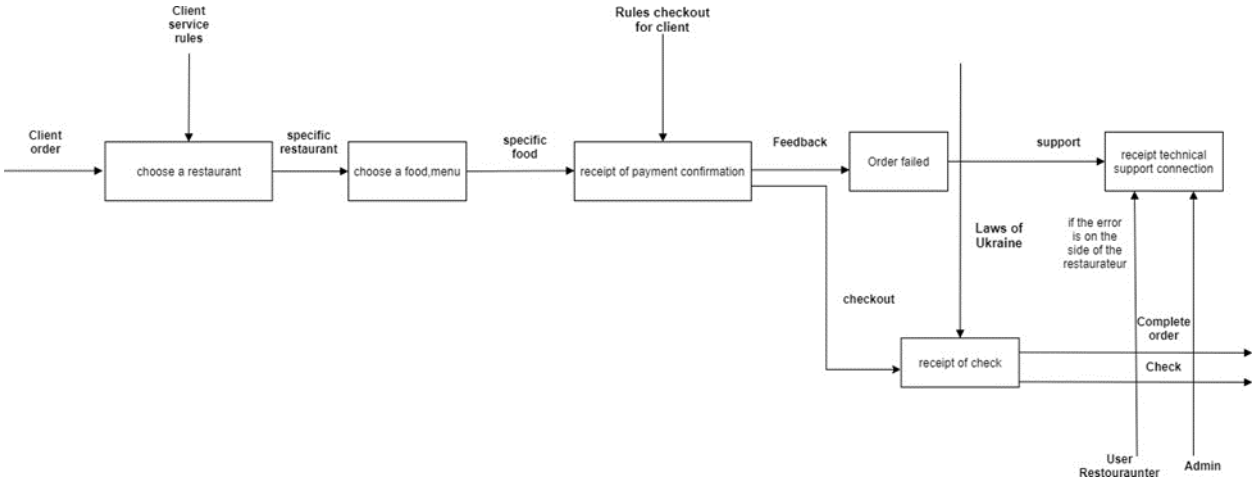

Рисунок 2.3 – Декомпозиція діаграми A-0

Для захисту ПЗ в першу чергу необхідно обмежити доступ до коду усіх, окрім системних адміністраторів та технічної підтримки, яка також працює над виправленням помилок. Також, у кожного користувача сайту або додатку буде свій власний унікальний ідентифікаційний код, який допоможе захистити облікові записи.

Почати треба з того, що сайт може працювати на будь-якій операційній системі, тобто:

- Windows 7 та вище

- Linux
- Unix

Для коректного відображення сайту потрібно мати браузер.

Мінімальні системні вимоги:

- Процесор класу Intel Core
- Не менше 2 ГБ ОЗУ
- Вільного місця на жорсткому диску приблизно від 100 МБ.

Вимоги до обладнання:

- Працюючий PC або ноутбук.

Технічні вимоги до комп'ютера, який відповідає мінімальним системним вимогам для розробки і розгортання створюваної програми, інсталяції інструментальних засобів розробки, що враховує особливості експлуатації програмного забезпечення, приведені для всіх використовуваних програмних засобів. Для розробки сайту буде використовуватись Visual Studio Code [14].

Написання коду може бути на будь-якій операційній системі, наприклад:

- Windows 7 і вище
- Linux
- Unix

2.3 Розроблення архітектури веб-додатку

Діаграма варіантів використання (Use Case Diagram) є важливою частиною планування проєкту. Вона є найпростішою з поведінкових діаграм та демонструє результат взаємодії між акторами та прецедентами. Створена діаграма дозволяє отримати інформацію щодо майбутнього функціоналу системи, а також показує яким чином актор може діяти в відповідній системі.

Для створення діаграми використовуються такі актори:

User Customer – той хто обирає певну систему та збирається взаємодіяти з нею. Після відбору всіх потенційних користувачів системи формується перелік

можливих варіантів взаємодії з нею. Варіанти обираються відповідно до вимог, яким відповідає веб-ресурс.

Діаграма Use Case [8] була розроблена на основі інформації про користувачів веб-додатку, а також можливих варіантів взаємодії з ним.

На діаграмі (рис. 2.4) представлено 3 типи юзерів. Фіолетовим кольором позначені загальні дії. Решта відповідно до кольору кожного юзера.

Варіанти використання web-додатку:

- Реєстрація
- Авторизація
- Перегляд ресторанів та меню
- Перегляд історії оплат та замовлень
- Залишати відгуки
- Розміщення свого ресторану та меню
- Перегляд заробітку та витрат свого ресторану
- Пошук по сайту
- Оплата онлайн
- Додавання в улюблене
- Підтримка юзерів
- Підтримка сайту
- Налаштування профілю в особистому кабінеті

На основі сформованих даних про акторів та всі можливі варіанти використання веб-додатку, була розроблена Use Case діаграма. Вона представлена на рис. 2.4.

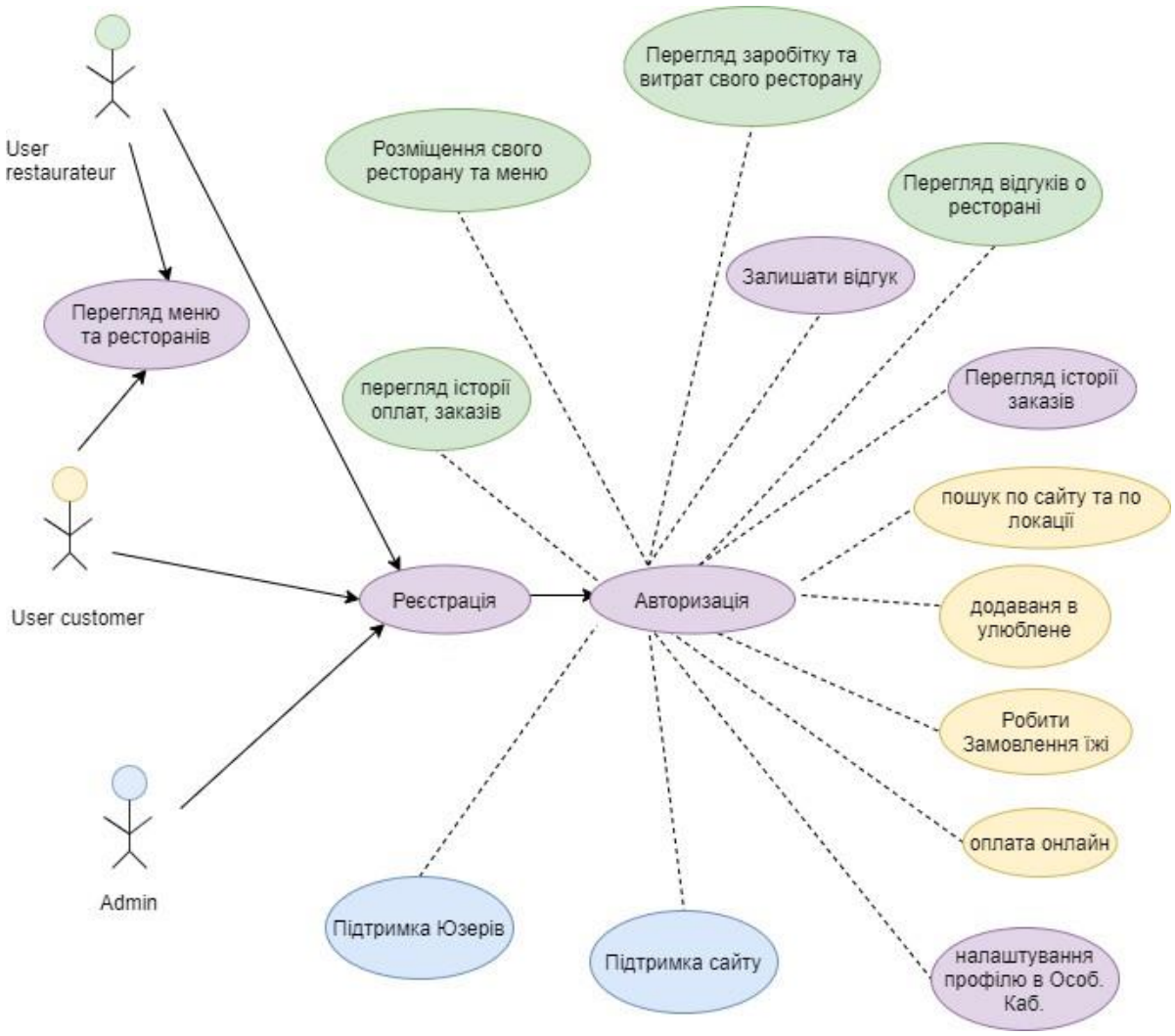

Рисунок 2.4 – Діаграма варіантів використання

У нас є 3 типи користувачів:

User restaurateur – який може переглядати меню та ресторани, реєструватись та авторизуватись, а згодом переглядати історії оплат та заказів, розміщувати свій ресторан та меню, переглядати заробіток та витрати ресторану, залишати відгук та переглядати інші відгуки.

Також User customer може переглядати історії заказів та здійснювати пошук по сайту та по локації, додавати в улюблене та замовляти їжу, оплачувати онлайн та налаштовувати свій профіль.

Роль Admin полягає у підтримці сайту та користувачів, може підтверджувати публікацію відгуків, переглядати історію заказів та налаштовувати профіль.

На рис. 2.5 зображено архітектуру веб-додатку.

Також зображено логіку та розуміння як працює веб-додаток, за допомогою чого і без чого він не може працювати.

Найголовніше – це сервер, за допомогою нього функціонує весь додаток. Далі база даних за допомогою якої відбувається обмін даними.

Все інше це компоненти, плагіни, модулі за допомогою всього цього складається наш веб-додаток

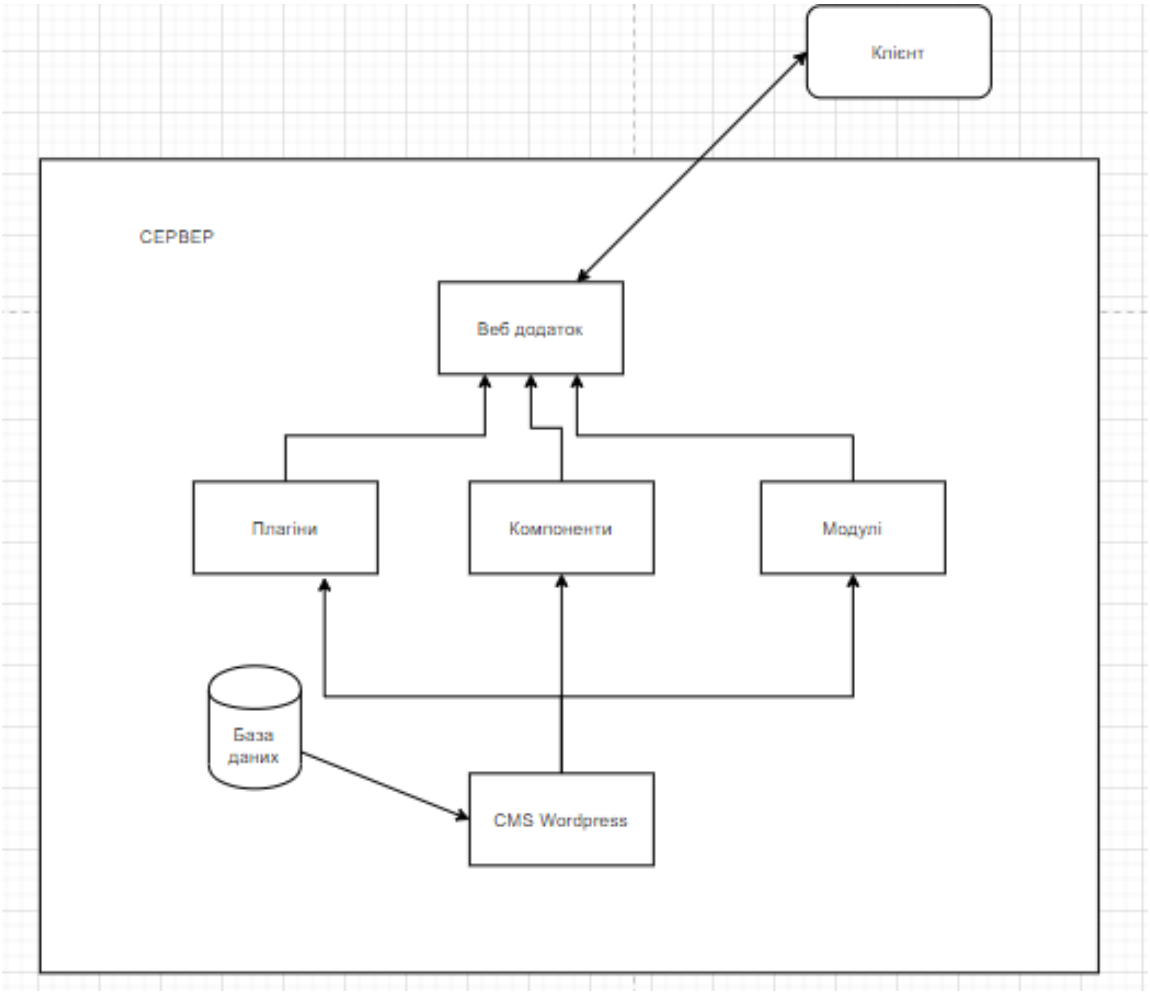

Рисунок 2.5 – Архітектура веб-додатку

2.4 Концептуальне рішення

Прийняття концептуального рішення для веб-додатку [7] включає кілька ключових кроків. Ось деякі кроки, які необхідно виконати:

Визначити мету та функціональність. Розуміння цих основних елементів допоможе зорієнтуватися при прийнятті рішень на пізніших етапах розробки.

Аналіз цільової аудиторії. Визначення потреб, очікування та користувацький досвід цільової аудиторії. Це допоможе створити додаток, який задовольняє їхні вимоги та очікування.

Властивості веб-застосунку наступні:

- Переміщує дані через мережу з підвищеною швидкістю, що забезпечує кращий досвід роботи кінцевого користувача.
- Покращує безпеку мережі за допомогою фільтрації IP-адрес, програм брандмауерів і шифрування.
- Запобігає перевантаженню серверів швидким розповсюдженням.

Контролер доставки програм (ADC, Application delivery controller) – це пристрій комп'ютерної мережі в центрі обробки даних, часто як частина мережі доставки програм. ADC розташовується між веб-серверами організації та кінцевими користувачами для керування трафіком додатків. Наприклад, ADC часто розгортають брандмауер веб-додатків на додаток до функції балансування навантаження.

Існує три ключові елементи доставки програм:

Контролер доставки програм (ADC) – це мережеві пристрої або програмно-визначені контролери (або площини керування), створені з метою підвищення безпеки, продуктивності та надійності доставки програм через Інтернет. Балансування навантаження допомагає забезпечити ефективний розподіл трафіку додатків між різними серверами.

Управління доставкою додатків (ADM, Application Delivery Management). ADM означає технологію, яка забезпечує надійне уявлення про те, як програми доставляються в багатохмарних середовищах. Потужне рішення ADM спрощує процес моніторингу для IT.

Мережа доставки додатків (ADN, Application delivery network). ADN використовує дані в реальному часі, щоб визначити пріоритетність розповсюдження програм і забезпечити надійний доступ для кінцевого користувача. Працюючи як кластер служб, ADN синхронізує розгортання в мережі, щоб забезпечити видимість, безпеку, швидкість і готовність додатків.

2.5 Проєктування інтерфейсу користувача

Проєктування інтерфейсу користувача – це процес створення взаємодії між користувачем і комп'ютерною системою через графічний і візуальний шар. Його основна мета полягає в тому, щоб зробити взаємодію з програмою або веб-сайтом якомога більш зручною, ефективною і задовольняючою для користувача.

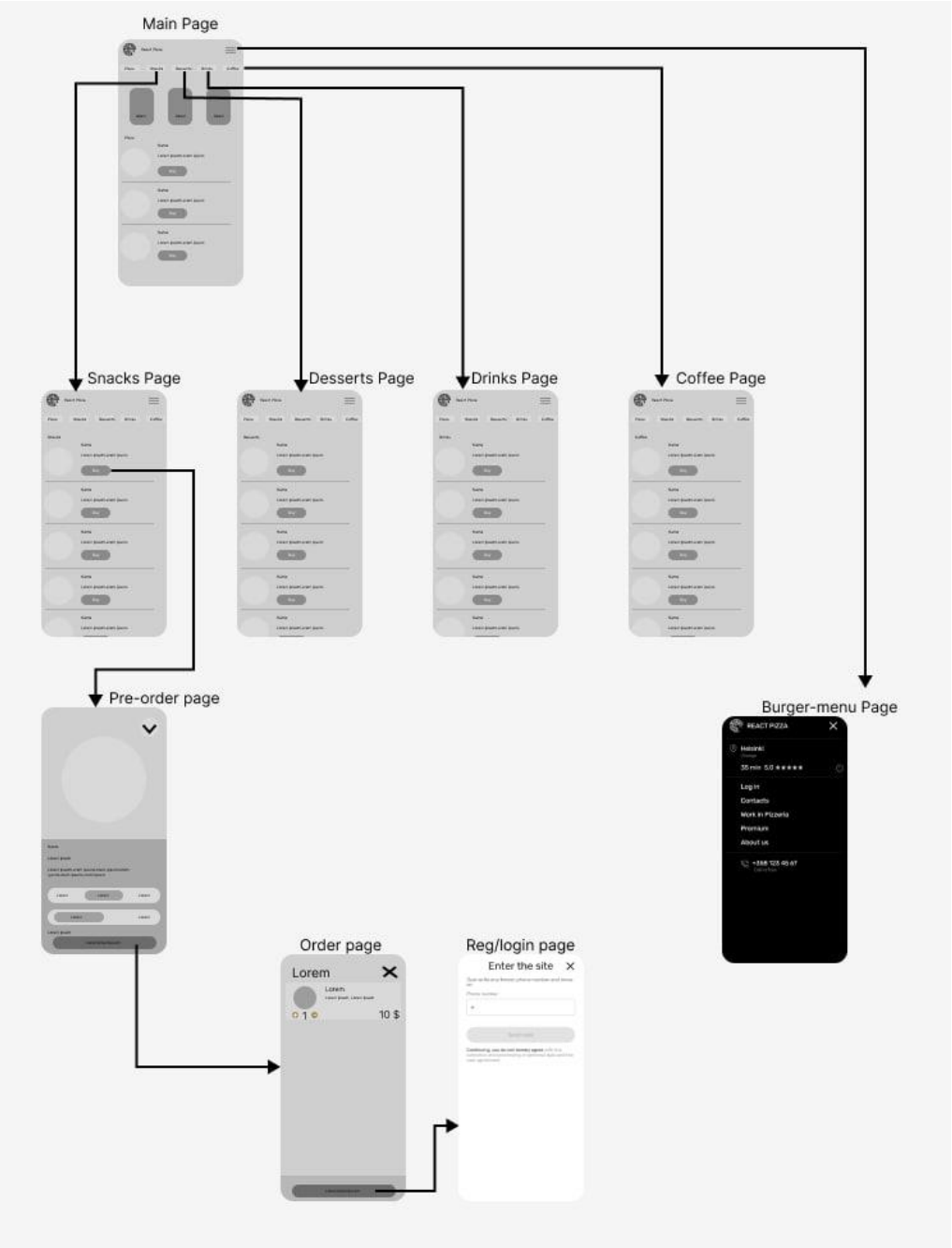

Рисунок 2.6 – Прототип веб-додатку

Задля цього процесу, використовувався веб-сервіс Figma, щоб поетапно зробити прототип додатку доставки продуктів. Бачимо, що в нас є 9 екранів, які побудовані задля переміщення по додатку (рис. 2.6).

Main Page – сторінка, яка відповідає за головну.

Snacks Page – сторінка, яка відповідає за снек частину додатку.

Desserts Page – сторінка, яка відповідає за десертну частину додатку.

Drink Page – сторінка, яка відповідає за частину напоїв додатку.

Coffee Page – сторінка, яка відповідає за частину з кавовими напоями додатку.

Pre-order Page – сторінка, яка відповідає за передзамовлення.

Order Page – сторінка, яка відповідає за замовлення.

Reg/login Page – сторінка, яка відповідає за реєстрацію та логін.

Burger-menu Page – сторінка(вікно), яка відповідає за навігацію у додатку.

2.6 Опис розробки веб-додатку

Клас додатку(App). Цей клас є основним компонентом додатку. Він містить дані про стан додатку, такі як дані про їжу, додаткові інгредієнти, дані про місто, інформацію про користувача, товари в кошику тощо. Він також містить методи додавання товарів до кошика, видалення товарів з кошика та інші функції, які використовуються в компонентах додатку.

Функція openCartModal. Ця функція відкриває модальне вікно з деталями товару, коли користувач натискає на товар.

Функція AddToCart. Ця функція додає продукт до кошика. Вона перевіряє, чи продукт вже є в кошику. Якщо так, то збільшує кількість товару в кошику. Якщо ні, то додає товар до кошика.

Функція RemoveFromCart. Ця функція видаляє товар з кошика за його ID.

Функція Notify. Ця функція відображає спливаюче повідомлення, яке сповіщає користувача про те, що товар додано до кошика.

Функція Success. Ця функція відображає спливаюче повідомлення, яке інформує користувача про те, що замовлення було успішно оформлено.

Компонент FoodCard. Цей компонент відповідає за відображення картки товару. Вона містить зображення продукту, назву, склад і ціну. Крім того, він містить кнопку "Додати в кошик", яка при натисканні викликає функцію addToCart.

3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Оцінка адекватності розробленого веб-додатку

Розглянемо основні екрани веб-додатку.

Вікно реєстрації у додатку (рис. 4.1).

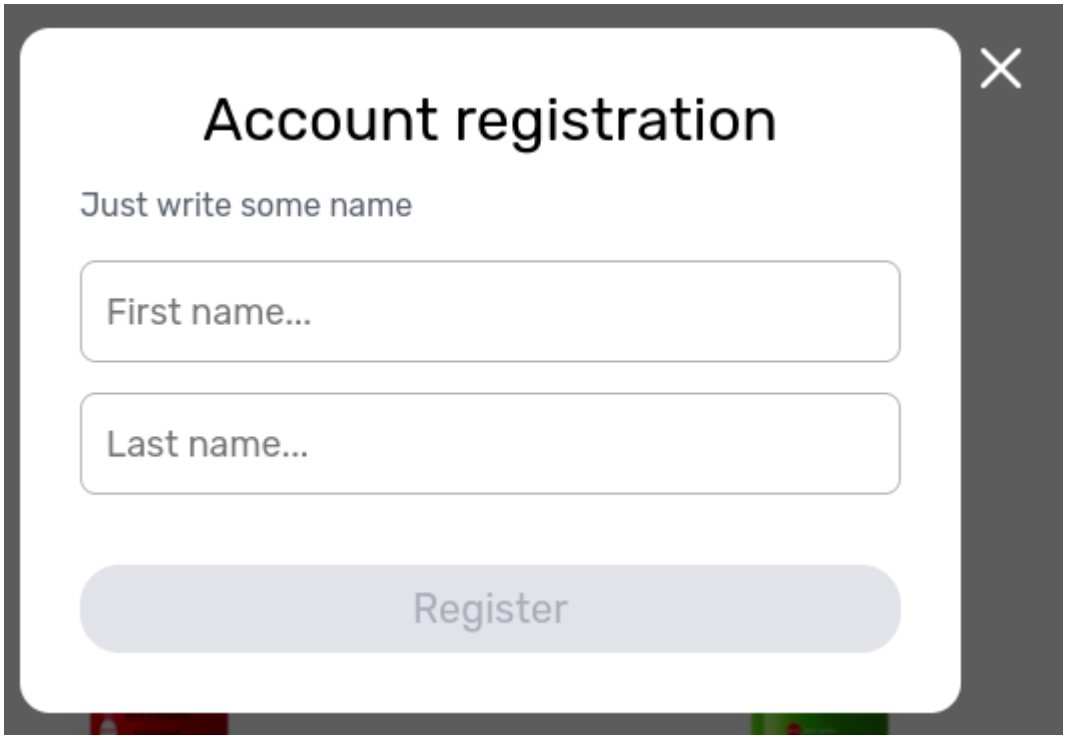

Рисунок 4.1 – Перший етап реєстрації

Далі вводимо особисті дані (рис. 4.2).

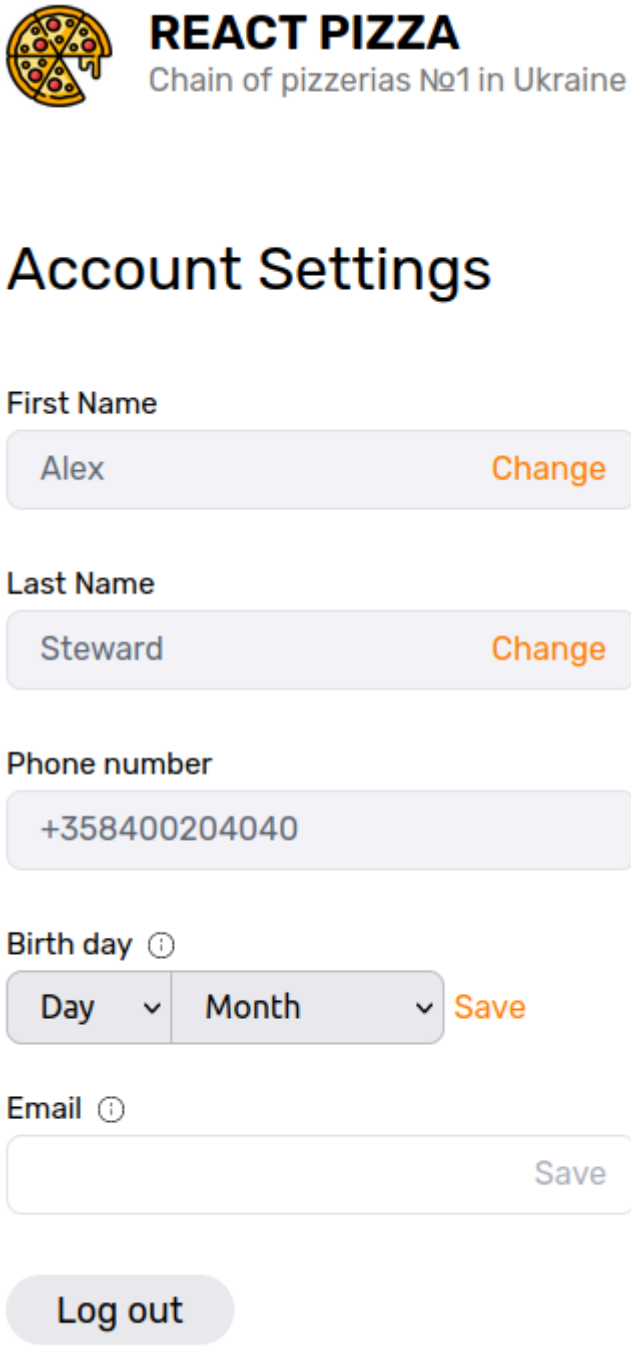

Рисунок 4.2 – Введення особистих даних при реєстрації

Якщо у Вас вже є акаунт, то можна ввести особистий номер для входу (рис. 4.3).

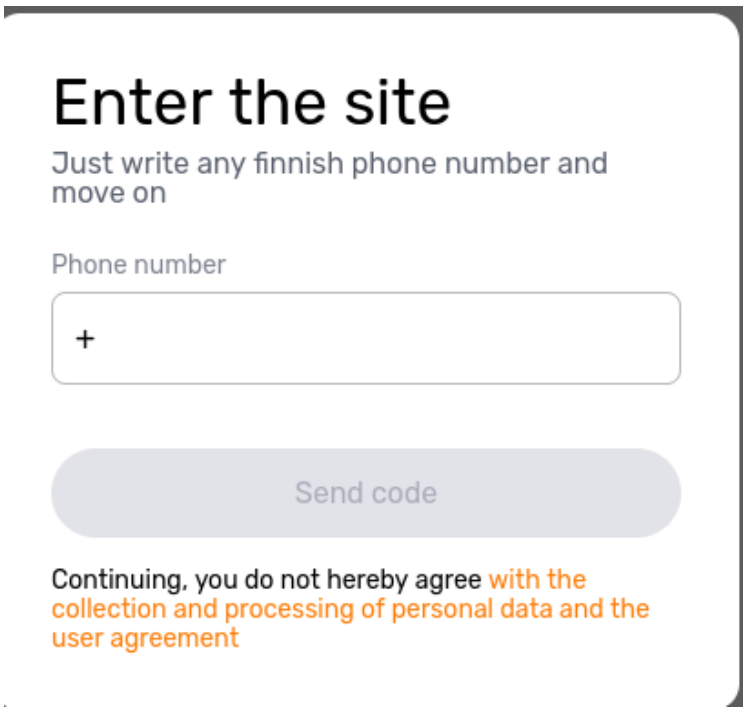

Рисунок 4.3 – Введення особистого номеру для входу

Далі надходить смс із кодом для входу на особистий номер (рис. 4.4).

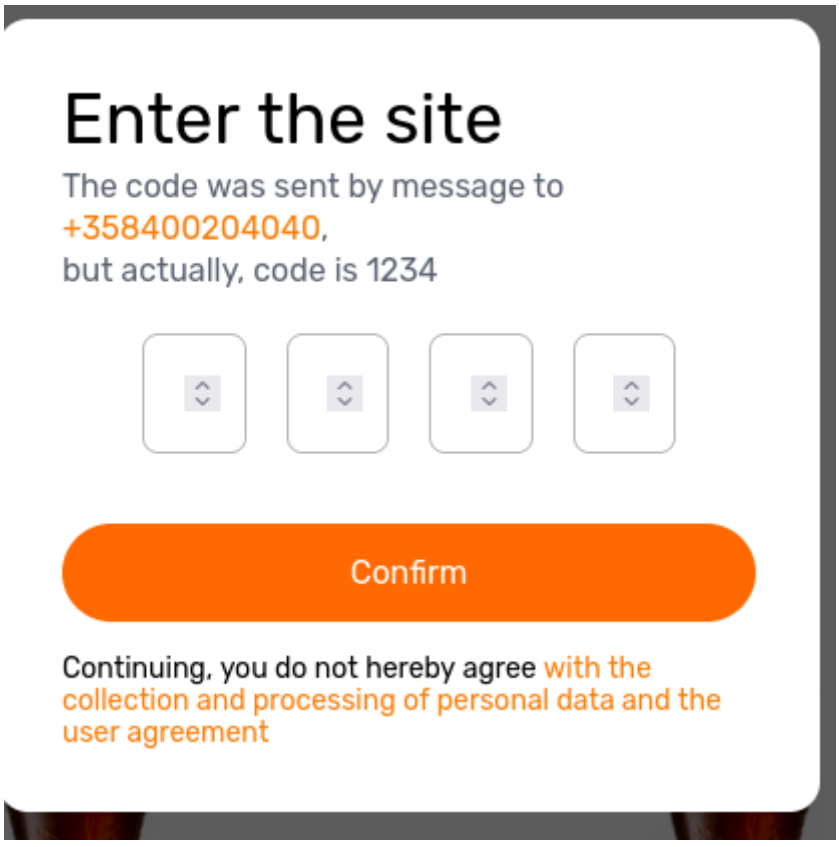

Рисунок 4.4 – Код для авторизації

Після реєстрації/входу користувач потрапляє у головне меню (рис. 4.5).

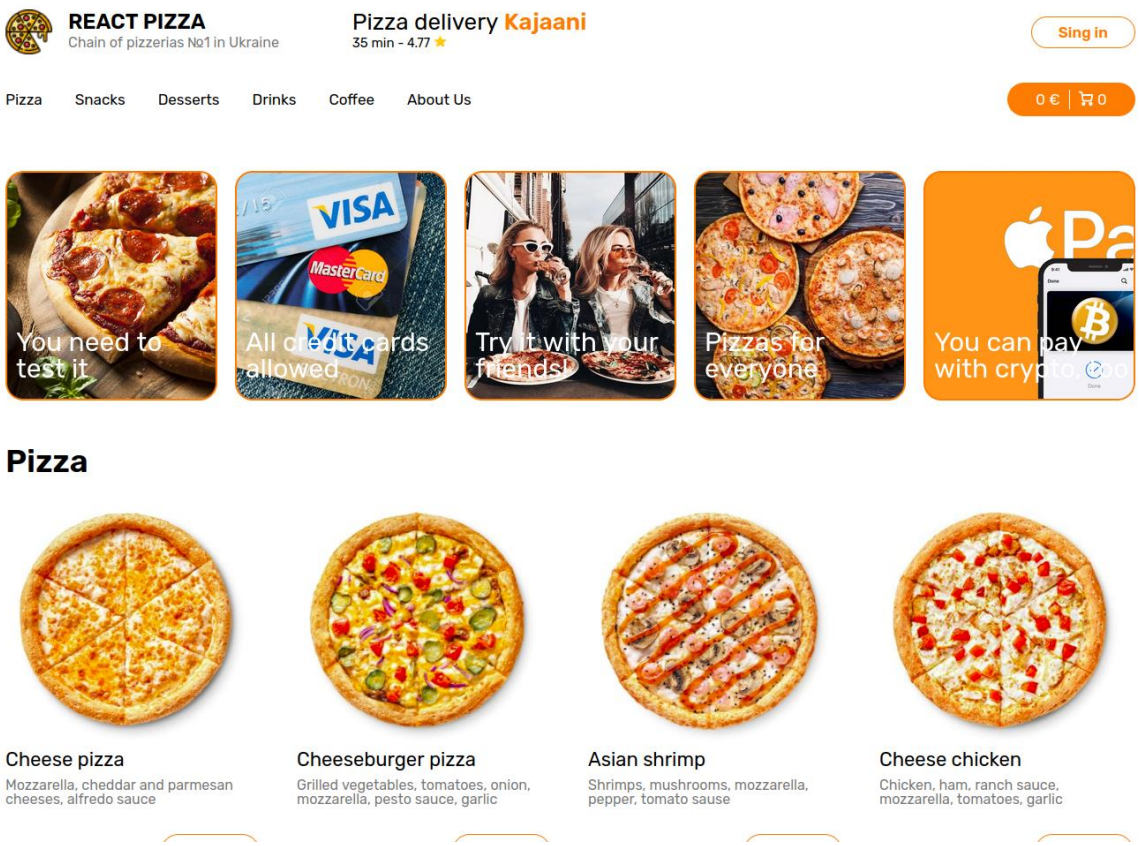

Рисунок 4.5 – Головне меню веб-додатку

Потім можна перейти у розділ «Піца» (рис. 4.6).

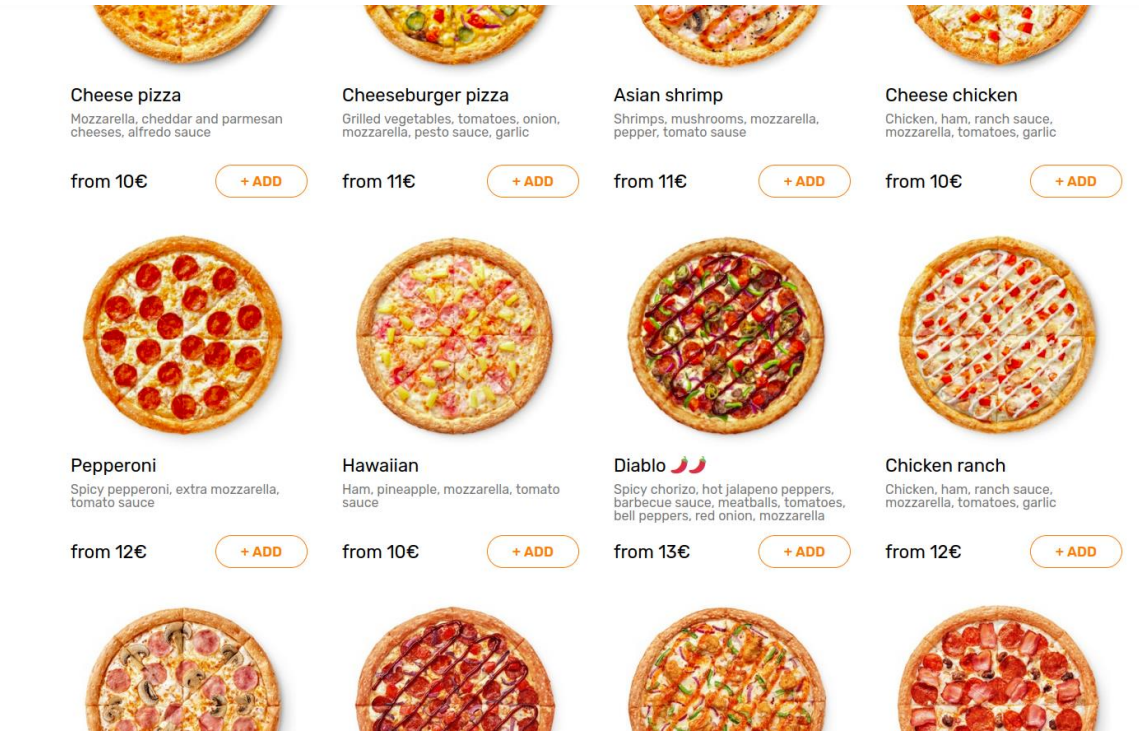

Рисунок 4.6 – Розділ «Піца»

Також можна перейти у розділ «Десерти» (рис. 4.7).

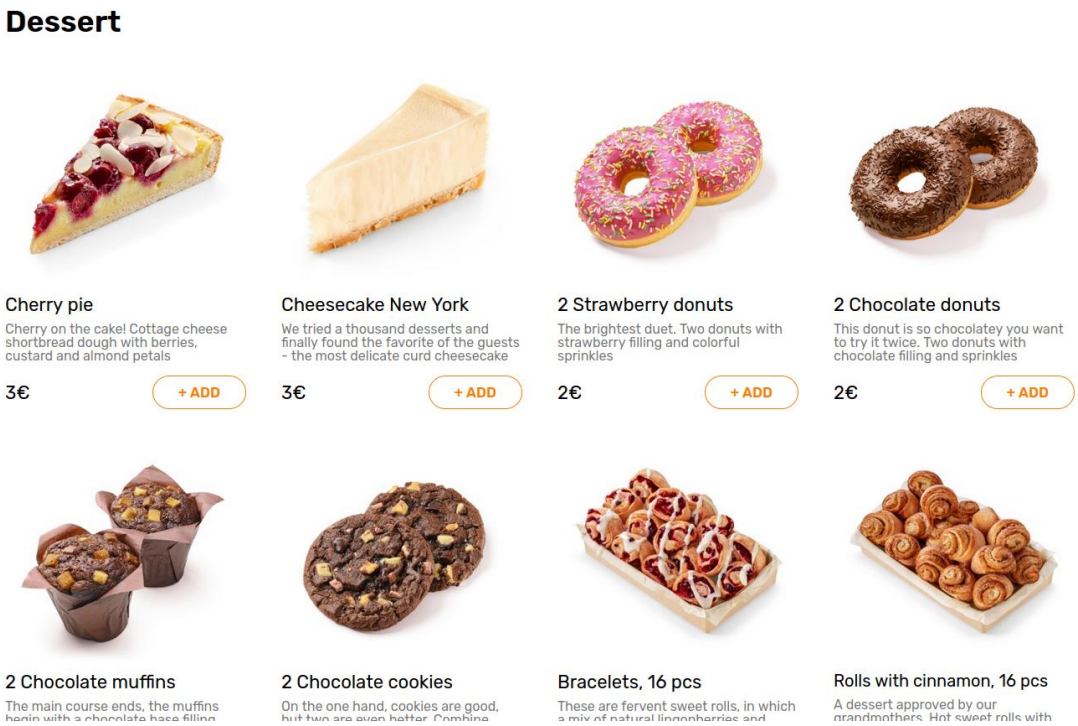

Рисунок 4.7 – Розділ «Десерти»

Також можна перейти у розділ «Напої» (рис. 4.8).

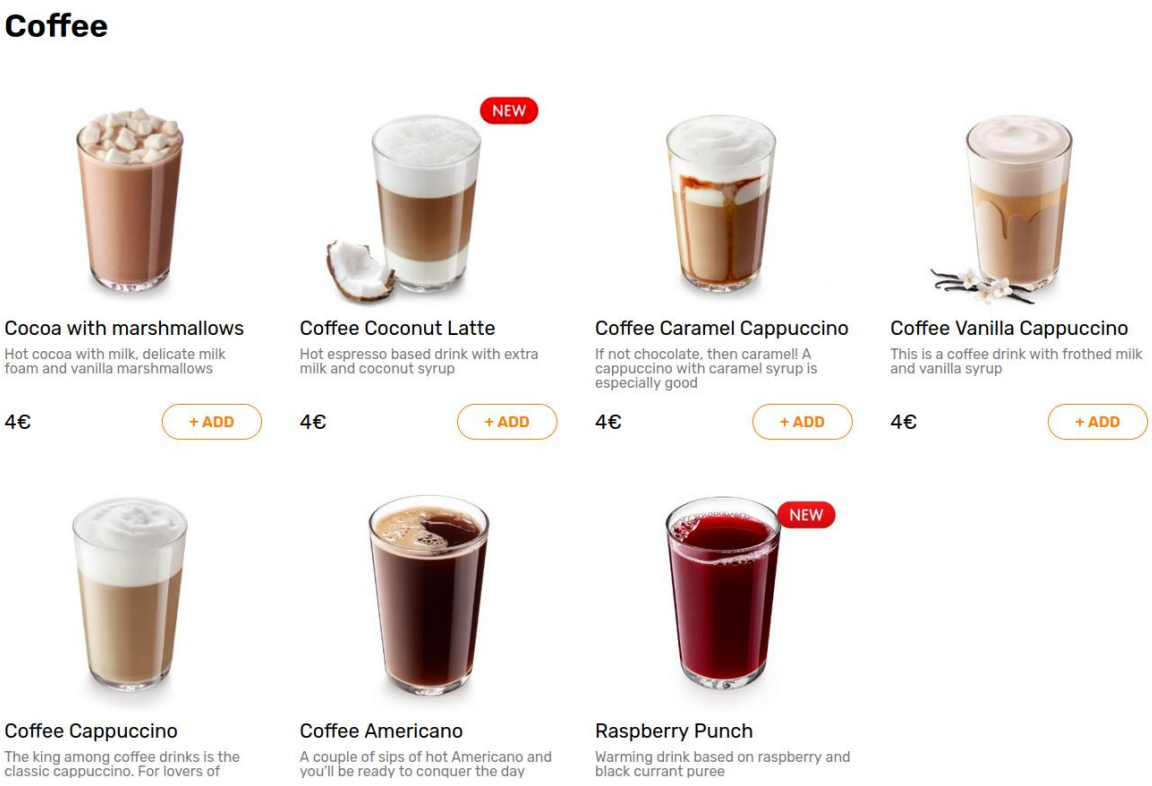

Рисунок 4.8 – Розділ «Напої»

Якщо корзина пуста, то відображається таке вікно (рис. 4.9).

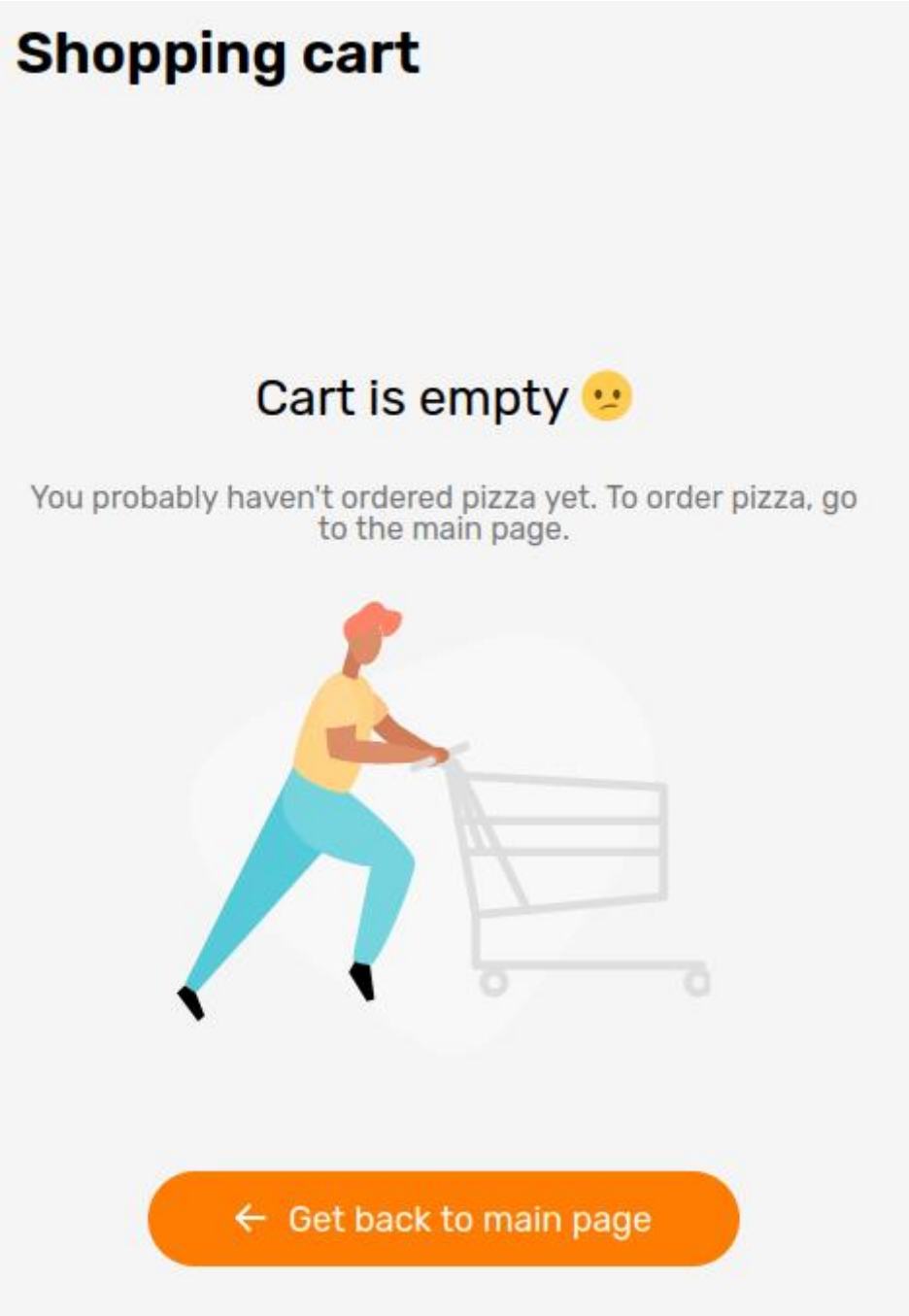

Рисунок 4.9 – Корзина

Якщо додаємо продукт у корзину, то відображається таке вікно (рис. 4.10).

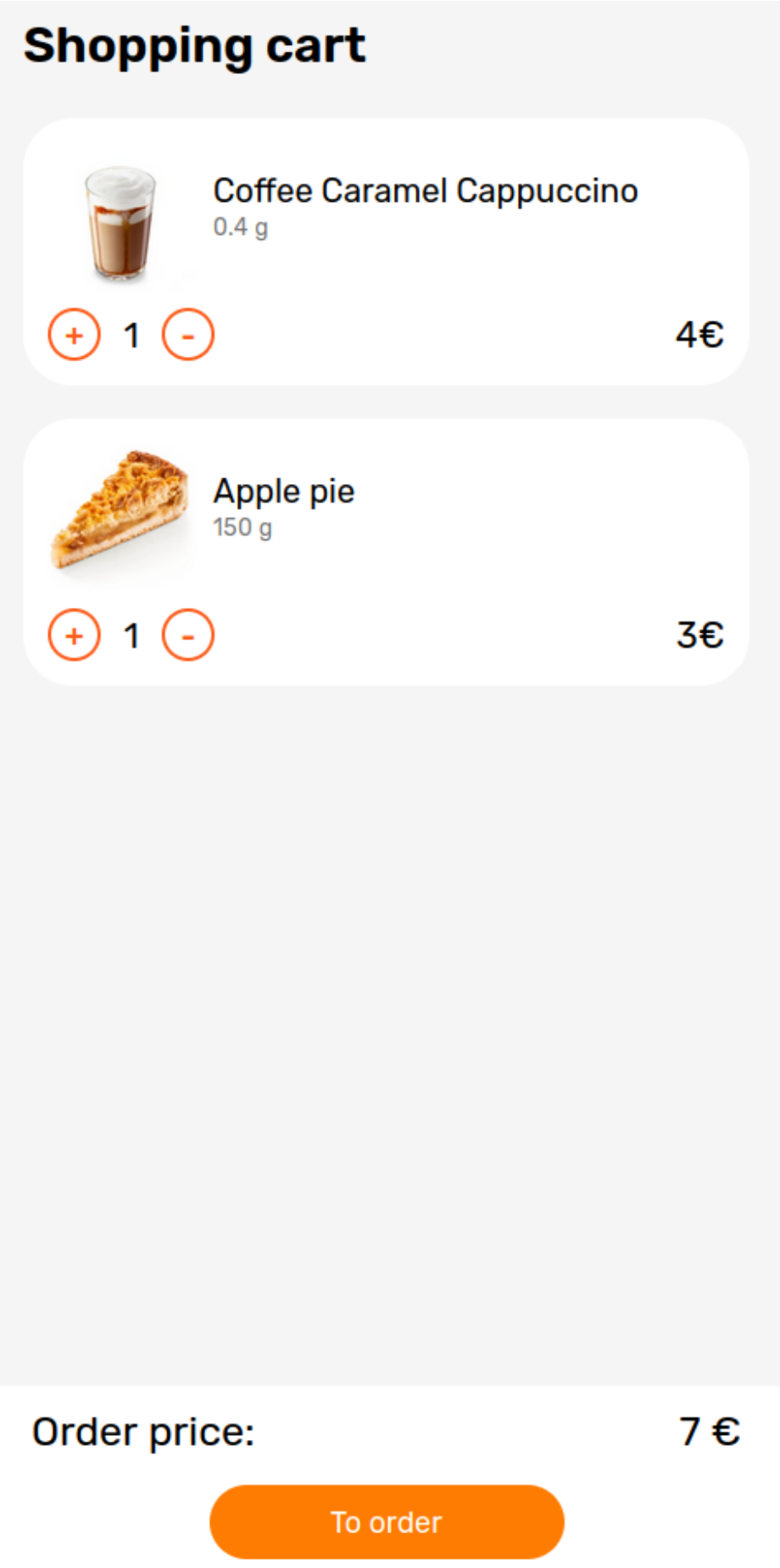

Рисунок 4.10 – Корзина із продуктами

Після оформлення замовлення надходить таке повідомлення (рис. 4.11).

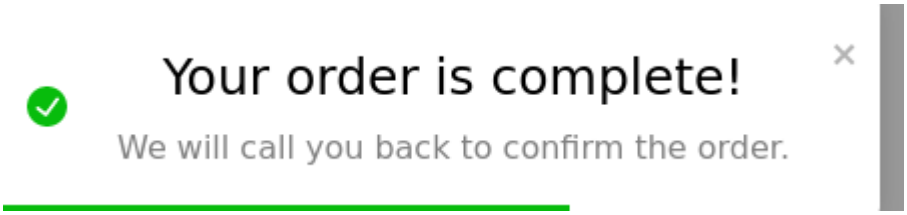

Рисунок 4.11 – Оформлення замовлення

3.2 Аналіз ефективності розробленого веб-додатку

Призначення розробки та галузь застосування – це застосування інформаційної системи для доставки продуктів споживачам. На разі, такий веб-додаток є досить актуальною у наш час та може використовуватись як з PC, так і зі смартфонів. Ця система застосовується для сортування, розподілення та розрахунку часу доставки продуктів. По-перше, будь-який користувач сайту або платформи буде спроможний використати таку систему з будь-якої точки країни, або, навіть світу. По-друге, для використання такої системи, не потрібно мати потужний гаджет через те, що функціонал мінімальний.

Система із замовлення їжі може використовуватись не тільки серед звичайних користувачів, але й мати спеціальне призначення. Наприклад, такі системи можуть використовуватись в армії для замовлення їжі між солдатами серед невеликих груп людей. Також таку систему можна використовувати для більш складних задач, наприклад для замовлення їжі в університеті або великому офісі.

При розробці програмного забезпечення (ПЗ) важливими етапами є визначення трудомісткості розробки і розрахунок витрат на створення програмного продукту.

Онлайн-замовлення дозволило багатьом ресторанам дуже ефективно керувати піковим робочим часом. Завдяки онлайн-замовленню багатьом людям вдається уникнути болісного досвіду марнування часу в довгій черзі. Завдяки

можливості замовити їжу з мобільного додатку, вони можуть легко зробити замовлення, коли вони застрягли в пробці або їдуть забрати дітей.

Мобільні додатки дають можливість робити замовлення з будь-якого місця в будь-який час, не зупиняючи все й не дзвонячи в ресторан. Досвід доставки їжі пройшов довгий шлях і став набагато більш безпроблемним для клієнтів.

Дослідження показують, що клієнти з більшою ймовірністю зроблять покупку, якщо їм надається можливість безготівкового розрахунку, тому що чим менше ми маємо справу з готівкою, тим зручніше. Хороший додаток для замовлення в ресторані матиме опції можливості приймати оплату різними способами. Таким чином, клієнт може вибрати найбільш зручний для себе варіант.

Мобільні застосунки – чудові інструменти для залучення клієнтів. Але як заохотити клієнтів завантажувати додаток ресторану та регулярно ним користуватися? Для цього потрібно забезпечити правильне поєднання вмісту. Забагато вмісту, і додаток виглядатиме захараченим. Потрібно також винагороджувати клієнтів, якщо вони замовляють через додаток. Мобільні програми мають багато вбудованих опцій, щоб підтримувати рівень залучення. Можна також звернути увагу клієнтів на ресторан за допомогою нових пропозицій або запровадити програму лояльності на основі додатка. Щоразу, коли клієнти роблять покупку через додаток, потрібно їм повернути певну вартість. Завдяки цим бонусам вони повертатимуться знов і знов.

З розширенням ринку доставки та сторонніми платформами доставки, невеликі ресторани мають можливість використовувати сектор, який був доступний лише для великих мереж із їхніми парками доставки. Фірмовий мобільний додаток діє як віртуальна вітрина, яка дає змогу клієнтам розміщувати замовлення, бачити його під час обробки та відстежувати його місцезнаходження під час доставки. Ця спеціальна функція мобільних додатків підвищує залучення клієнтів.

Клієнти вважають за краще робити замовлення на своїх мобільних пристроях. Безперебійний мобільний досвід сприятиме позитивному інтересу до бізнесу, який можна ще більше просувати, використовуючи відгуки та оцінки клієнтів. Потрібно винагороджувати клієнтів за позитивний відгук і вирішувати проблеми, якщо

отримано негативний відгук. Більшість із цих негативних відгуків можна легко вирішити швидкою відповіддю.

Розрахунок показників створення веб-застосунку відображено нижче.

Задані дані:

1) Передбачуване число операторів – 600;

2) Коефіцієнт складності програми – 1,5;

3) Коефіцієнт корекції програми в ході її розробки – 0,1;

4) Годинна заробітна плата програміста, грн/год – 250;

5) Коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,1;

6) Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,0;

7) Вартість машино-години ЕОМ, грн/год – 45.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки може бути розрахована на основі системи моделей з різною точністю оцінки. Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин.} \quad (1.1)$$

де t_o –витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів уПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \quad (1.2)$$

Q – передбачуване число операторів;C – коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$t_u = Q * B(75..85) * k$, людино-годин. (1.3)

В – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$t_a \cdot = Q/(20..25)*k$, людино-годин. (1.4)

Витрати на складання програми по готовій блок-схемі:

$t_n \cdot = Q/(20..25)*k$, людино-годин. (1.5)

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$t_{отл} \cdot = Q/(4..5)*k$, людино-годин. (1.6)

- за умови комплексного налагодження завдання:

$Tk\ отл = 1,5 * t\ отл$, людино-годин. (1.7)

Витрати праці на підготовку документації:

$t_д = t_{др} + t_{до}$, людино-годин, (1.8)

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису.

$t_{др} \cdot = Q/(15..20)*k$, людино-годин (1.9)

$t_{до}$ – трудомісткість редагування, друку й оформлення документації.

$t_{до} = 0,75 * t\ др$, людино-годин. (1.10)

Витрати на створення ПЗ $К_{пз}$ включають витрати на заробітну плату виконавця програми $З_{зп}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$К_{по} = З_{зп} + З_{мв}$, грн. (1.11)

Заробітна плата виконавців визначається за формулою:

$З_{зп} = t \cdot С_{пр}$, грн. (1.12)

t – загальна трудомісткість, людино-годин;

$С_{пр}$ – середня годинна заробітна плата програміста, грн/година.

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{мв} = t_{отл} \cdot C_{мг}, \text{ грн.} \quad (1.13)$$

$t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год. $C_{мг}$ – вартість машино-години ПК, грн/год.

Визначені в такий спосіб витрати на створення програмного забезпеченняє частиною одноразових капітальних витрат на створення АСУП.

Очікуваний період створення ПЗ:

$$T = t / B_k * F_p, \text{ міс.} \quad (1.14)$$

B_k – число виконавців;

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні F_p = 176 годин).

3.3. Тестування програмного забезпечення

Тестування програмного забезпечення – це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки. Може оцінюватись:

− відповідність вимогам, якими керувалися проєктувальники та розробники;

− правильна відповідь для усіх можливих вхідних даних;

− виконання функцій за прийнятний час;

− практичність;

− сумісність з програмним забезпеченням та операційними системами;

− відповідність задачам замовника.

Тестування програмного забезпечення традиційно відокремлювалося від решти розробки. Його часто проводять пізніше в життєвому циклі розробки програмного забезпечення після етапу створення або виконання продукту. Тестер може мати лише невелике вікно для перевірки коду – іноді безпосередньо перед

виходом програми на ринок. У разі виявлення дефектів часу на перекодування або повторне тестування може бути мало. Це не рідкість випускати програмне забезпечення вчасно, але з помилками та необхідними виправленнями. Або команда тестування може виправити помилки, але пропустити дату випуску.

Виконання тестових дій на початку циклу допомагає тримати тестування на передньому плані, а не залишатися позаду після розробки. Попередні тести програмного забезпечення також означають, що усунення дефектів обходиться дешевше.

Зараз багато команд розробників використовують методологію, відому як безперервне тестування. Це частина підходу DevOps, де розробка та операції співпрацюють протягом усього життєвого циклу продукту. Мета полягає в тому, щоб пришвидшити доставку програмного забезпечення, збалансувавши вартість, якість і ризик. Завдяки цій техніці тестування командам не потрібно чекати, поки буде створено програмне забезпечення, перш ніж почати тестування. Вони можуть проводити тести набагато раніше в циклі, щоб швидше виявити дефекти, коли їх легше виправити.

Оскільки кількість можливих тестів практично нескінченна навіть для простих програмних компонентів, стратегія тестування полягає у виконанні всіх можливих тестів у межах доступного часу та ресурсів. Тому програмне забезпечення тестується стандартним виконанням програми для виявлення помилок (або інших дефектів).

Тестування програмного забезпечення може надати користувачам і клієнтам об’єктивну та незалежну інформацію про якість програмного забезпечення та ризики збою.

Після створення виконуваного коду (навіть частково завершеного) його можна протестувати. Процес розробки зазвичай визначає, коли і як тестувати. Наприклад, у поетапному процесі більшість тестувань відбувається після того, як системні вимоги визначені, а потім реалізовані в програмі тестування. Навпаки, відповідно до вимог гнучкої розробки програмного забезпечення, програмування та тестування часто відбуваються одночасно [15].

Під час тестування програмного додатку, який використовувався для дослідження замовлення продуктів, було виявлено, що додаток однаково відображається в найпопулярніших браузерах, таких як Internet Explorer, Opera, Mozilla Firefox, Chrome.

Юзабіліті-тести програми показують, що програмне забезпечення є дружнім до користувача, має зручну панель навігації та інтуїтивно зрозумілу логічну структуру.

Тестування програмного забезпечення відбувається за стандартними процедурами. Завдання або кроки включають визначення тестового середовища, розробку тестових випадків, написання сценаріїв, аналіз результатів тестування та подання звітів про дефекти.

Тестування може зайняти багато часу. Для невеликих збірок може бути достатньо ручного або спеціального тестування. Однак для великих систем інструменти часто використовуються для автоматизації завдань. Автоматизоване тестування допомагає командам реалізовувати різні сценарії, тестувати відмінності (наприклад, переміщення компонентів у хмарне середовище) і швидко отримувати відгуки про те, що працює, а що ні.

Хороший підхід до тестування охоплює інтерфейс прикладного програмування (API), інтерфейс користувача та рівні системи. Крім того, чим більше тестів буде автоматизовано та запущено раніше, тим краще. Деякі команди створюють власні засоби автоматизації тестування. Проте рішення постачальників пропонують функції, які можуть оптимізувати ключові завдання керування тестуванням, наприклад:

Постійне тестування: команди проєкту тестують кожну збірку, щойно вона стає доступною. Цей тип тестування програмного забезпечення базується на автоматизованому тестуванні, інтегрованому в процес розгортання. Це дає змогу перевіряти програмне забезпечення в реалістичних тестових середовищах на ранніх етапах процесу, покращуючи дизайн і знижуючи ризики.

Керування конфігурацією: організації централізовано підтримують тестові активи та відстежують, яке програмне забезпечення збирається для тестування.

Команди отримують доступ до таких активів, як код, вимоги, проєктні документи, моделі, тестові сценарії та результати тестування. Хороші системи включають автентифікацію користувачів і журнали аудиту, щоб допомогти командам виконати вимоги відповідності з мінімальними адміністративними зусиллями.

Віртуалізація служби: середовища тестування можуть бути недоступними, особливо на ранніх стадіях розробки коду. Віртуалізація сервісів імітує служби та системи, які відсутні або ще не завершені, що дозволяє командам зменшити залежність і швидше тестувати. Вони можуть повторно використовувати, розгортати та змінювати конфігурацію для тестування різних сценаріїв без необхідності змінювати початкове середовище.

Відстеження дефектів або помилок: моніторинг дефектів важливий як для команд тестування, так і для команд розробників для вимірювання та покращення якості. Автоматизовані інструменти дозволяють командам відстежувати дефекти, вимірювати їх масштаб і вплив, а також виявляти пов’язані з ними проблеми.

Показники та звітність: звіти та аналітика дозволяють членам команди ділитися статусом, цілями та результатами тестування. Розширені інструменти інтегрують показники проєкту та представляють результати на інформаційній панелі. Команди швидко бачать загальний стан проєкту та можуть контролювати зв’язки між тестуванням, розробкою та іншими елементами проєкту.

ВИСНОВКИ

Метою дипломного проекту є розробка програмного забезпечення для доставки їжі споживачам, а також для користування додатком ресторанів чи власного підприємства. Програма являє собою систему, що реалізує запис інформації про замовлення користувачів, обробку та зберігання даних, відображення інформації у зрозумілій впорядкованій формі.

Мета даної дипломної роботи складалася з декількох частин, а саме:

1) Написання плану дій дипломної роботи.

2) Здійснення задуманої концепції, яка відображається коректно, а також має подальше застосування в безпосередньо важливих сферах життя людини. Також додавання декількох функцій в процесі роботи, наприклад, таких як: зміна облікових даних, додавання або зміна будь-яких продуктів, включаючи їжу, а також товари щоденного споживання.

Тема дипломної роботи є актуальною на сьогоднішній день, а також, провівши певні аналітичні спостереження за розвитком сфери діяльності людини по доставці їжі, дана тема набирає свою актуальність і популярність. Існують емпіричні дані згідно з якими послуги з доставки будуть приносити більший прибуток в найближчому майбутньому. Доставка їжі хоч і не є новою і унікальною ідеєю, однак, допускає безліч нових поправок, а також коригувань і фільтрів, які дійсно можуть зробити продукт більш унікальним і універсальним, згідно смакам користувачів.

З практичної точки зору, доставка їжі буде займати в майбутньому лідируючі позиції, так як велика частина людей замовляє вже готову їжу або продукти додому. Це пов'язано з ситуацією, яка виникла по всьому світу на даний момент, а також з тим, що велика частина роботи зараз не мануальна, а технічна, тому багато хто замовляє їжу в офіси, на роботу, або додому.

Також, дана програма не є дорогою в плані його підтримки, проте, якщо пізніше враховувати кількість витрат на підтримку програми, вони можуть зрости. У додатку є кілька пунктів економічного застосування, тобто воно буде

застосовуватися від імені незалежної компанії, або ж використовуватися ресторанами і кафе. В кожному окремому випадку, даний вид додатка буде приносити величезний прибуток, вимагаючи мінімальні витрати. Прогноз для системи по доставці їжі позитивний, так як є діаграма згідно з якою, велика частина населення переходить на готове харчування.

У цій роботі досліджується та аналізується найважливіша та актуальна інформація в розробці застосунків, а саме основні принципи створення програмних застосунків, їх структура та функції, взаємодія основних компонентів. Розглядаються також новітні та найперспективніші мережеві технології, які успішно використовуються користувачами по всьому світу.

Розроблений застосунок підтримує можливість вдосконалення, що дуже важливо в сучасному розвитку інтернет-технологій.

Також було проведено тестування програмних застосунків для замовлення їжі. Під час тестування помилок не виявлено, програма працює нормально.

Під час розробки програмних застосунків та оформлення бакалаврської роботи на практиці закріплюються теоретичні знання, удосконалюються навички програмування, готується технічна документація в текстових та графічних редакторах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. JQuery API Documentacion. URL: https://api.jqueryui.com/
2. Ionic – Cross-Platform Mobile App Development. URL: https://ionicframework.com/
3. Kukulska-Hulme A. Mobile language learning now and in the future. / A. Kukulska-Hulme- London: Swedish Net University, 2006. – P. 295-310.
4. Mobile Technology. URL: http://en.wikipedia.org/wiki/Mobile_technology
5. Mobile Technology. URL: http://www.itbusinessedge.com/topics/ show.aspx?t=738
6. MySQL. URL: https://www.mysql.com/
7. Web-технології та Web-дизайн. – 2015. URL: https://dl.sumdu.edu.ua/ textbooks/86975/413008/index.html
8. What is Use Case Diagram? URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-usecase-diagram/
9. What is Mobile Technology? URL: http://www.strategicgrowthconcepts.com/growth/mobile-technology-facts.html
10. Гнатієнко Г.М., Снитюк В.Є. Експертні технології прийняття рішень: монографія. Київ: ТОВ «Маклаут», 2008. 444 с.
11. Ілляшенко С.М. Сучасні тенденції застосування інтернет-технологій. URL: http://www.nbuv.gov.ua/portal/Soc_Gum/Mimi/2011_4_2/2_1.pdf
12. Косик В. Використання мобільних пристроїв та планшетів на базі ОС Android в навчальному процесі. URL: http://www.airo.com.ua/vikoristannya-mobilnih-pristroyiv-ta-planshetiv-na-bazi-os-android- v-navchalnomu-protsesi/.
13. Мобільне навчання стає дедалі більш популярним: новини Management.com.ua // новини від 07.04.2011. – URL: http://www.management.com.ua/news/?id=1329
14. Documentation for Visual Studio Code. URL: https://code.visualstudio.com/docs.

15. Організаційна структура проєкту (OBS). – URL: https://studopedia.com.ua/1_243503_organizatsiyna-struktura-proektuoBS.html

16. Попова Ю. В. Сутність і технічні інструменти інтернет-маркетингу. URL: http://www.nbuv.gov.ua/portal/soc_gum/Uproz/2011_6/u1106pop.pdf

17. Принципи побудови та етапи проектування баз даних. − URL: http://stud.com.ua/35671/informatika/kontseptsiya_baz_danih

18. Система керування базами даних. URL: http://uk.wikipedia.org/wiki/Система_керування_базами_даних – Назва з екрану.

19. Типи мобільних додатків. URL: https://smile-ukraine.com/ua/mobile-apps/mobile-apps-types

20. Що таке JQuery? URL: https://phpacademy.kiev.ua/uk/blog/what-is-jquery

ДОДАТКИ

App.js

```
import { Fragment, useEffect, useRef, useState } from 'react';
import { useFetching } from './hooks/useFetching';
import Header from './componets/Header/Header';
import Footer from './componets/Footer/Footer'
import ShippingPayment from './componets/ShippingPayment/shippingPayment'
import Nav from './componets/Nav/Nav';
import axios from 'axios';
import Slider from './componets/Slider/Slider';
import CardModal from './componets/Modals/CardModal/CardModal';
import './styles/_myclasses.scss'
import './styles/grid.scss'
import { AppContext, useData } from './context';
import SectionFood from './componets/SectionFood';
import SideCart from './componets/SideCart/SideCart';
import { CSSTransition } from 'react-transition-group';
import MobileMenu from './componets/MobileComponents/MobileNav/MobileMenu';
import Media from 'react-media';
import MobileCityMenu from './componets/MobileComponents/MobileCityMenu/MobileCityMenu';

import MobileCartIcon from './componets/MobileComponents/MobileCartIcon/MobileCartIcon';
import RegistrationRoutes from './componets/RegistrationForms/RegistrationRoutes';
```

```
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Route, Routes } from 'react-router-dom';
import Profile from './componets/Profile/Profile';
import CityModal from './componets/Modals/CityModal/CityModal';

function App() {
  const [foodData, setFoodData] = useState({});
  const [additivesData, setAdditivesData] = useState([]);
  const [citiesData, setCitiesData] = useState([])
  const [clientInfo, setClientInfo] = useState({})
  const [currentItem, setCurrentItem] = useState([])
  const [cartItems, setCartItems] = useState([])
  const { data, setValues } = useData();

  const [cartOpen, setCartOpen] = useState(false)
  const [cardModalOpen, setCardModalOpen] = useState(false)
  const [cityModalOpen, setCityModalOpen] = useState(false)
  const [mobileMenuOpen, setMobileMenuOpen] = useState(false)
  const [mobileMenuCitiesOpen, setMobileMenuCitiesOpen] = useState(false)
  const [phoneRegOpen, setPhoneRegOpen] = useState(false)
  const [checkoutModalOpen, setCheckoutModalOpen] = useState(false)

  /* authentification */
  const [isAuth, setIsAuth] = useState(false);

  const [priceListener, setPriceListener] = useState(false)

  const [selectedCity, setSelectedCity] = useState(localStorage.getItem('react-pizza-city') ? localStorage.getItem('react-pizza-city') : 'Helsinki')
```

```
/* Displayed on the shopping cart button */
const [totalCount, setTotalCount] = useState(0)
const [totalPrice, setTotalPrice] = useState(0)

const [fetching, isLoading, error] = useFetching(async () => {
  const responseFood = await axios.get('./db.json')
  setFoodData(responseFood.data[0])
  const responseAdditives = await axios.get('./additives.json')
  setAdditivesData(responseAdditives.data)
  const responseCity = await axios.get('./city.json')
  setCitiesData(responseCity.data)
})

useEffect(() => {
  fetching();
  /* Loading cart items from sessionStorage */
  for (let i = 0; i < sessionStorage.length; i++) {
    let key = sessionStorage.key(i);
    if (key !== 'nav') {
      cartItems.push(JSON.parse(sessionStorage.getItem(key)))
    }
  }
  /* User */
  if (JSON.parse(localStorage.getItem('react_user'))) {
    setIsAuth(true)
    setValues(JSON.parse(localStorage.getItem('react_user')))
  }
}, [])
```

```
useEffect(() => {
  let count = 0;
  let price = 0;

  cartItems.forEach(item => {
    price = price + Number(item.price) * Number(item.count)
    count = count + Number(item.count)
  })
  setTotalCount(count)
  setTotalPrice(price)

}, [priceListener, cartItems])

const openCartModal = (pizza) => {
  setCardModalOpen(true)
  setCurrentItem(pizza)
}

const addToCart = (item) => {
  setCurrentItem(item)

  setPriceListener(prev => !prev)

  const findItem = cartItems.find(el => el.id === item.id)

  if (findItem) {
    /* We are looking for an existing item in the basket and add +1 quantity to it */
    if (!item.pizza) { // if it is not pizza
      cartItems.map(item => {
        if (item.id === findItem.id) {
```

```
            return item.count += 1
          }
        })
        /* Delete an element from sessionStorage and add it back with a new count */
        cartItems.forEach(el => {
          if (el.id === findItem.id) {
            sessionStorage.removeItem(item.id)
            sessionStorage.setItem(el.id, JSON.stringify({ ...el, count: findItem.count
}))
          }
        })
      } else {
        /* Checking if there is an identical pizza for all parameters in the basket */
        const findPizza = cartItems.find(el =>
          el.pizza &&
          el.title === item.title &&
          el.cm === item.cm &&
          el.dough === item.dough &&
          el.additives.sort((a, b) => a.localeCompare(b)).join('') ===
item.additives.sort((a, b) => a.localeCompare(b)).join(''))

        if (findPizza) {
          /* If we got same pizza just add +1 count */
          cartItems.map(item => {
            if (item.id === findPizza.id) {
              return item.count += 1
            }
          })
          /* Delete an pizza from sessionStorage and add it back with a new count */
          cartItems.forEach(el => {
```

```
            if (el.id === findPizza.id) {
              sessionStorage.removeItem(item.id)
              sessionStorage.setItem(el.id, JSON.stringify({ ...el, count:
findPizza.count }))
            }
          })
        } else {
          /* If there is no identical pizza, then add a new product to the cart */
          let obj = {
            'count': 1,
            ...item,
            id: Date.now()
          }
          sessionStorage.setItem(obj.id, JSON.stringify(obj))
          setCartItems([...cartItems, obj])
        }
      }
    } else {
      /* If there is no identical element, then add a new product to the cart */
      let obj = {
        'count': 1,
        ...item
      }
      sessionStorage.setItem(obj.id, JSON.stringify(obj))
      setCartItems([...cartItems, obj])
    }
  }
  const removeFromCart = (id) => {
    setCartItems(prev => prev.filter(item => item.id !== id))
  }
```

```
const notify = (item) => {
  /* Popup on mobile screen when you added something */
  toast(`${item.title}, ${item.weight}g`, {
    position: "bottom-center",
    autoClose: 600,
    hideProgressBar: true,
    closeOnClick: false,
    pauseOnHover: true,
    draggable: true,
    theme: 'dark',
    containerId: 'notify'
  })
};
const success = () => {
  /* Success buy */
  toast.success(
    <>
      <h1 style={{ textAlign: 'center', fontSize: '25px', color: 'black' }}>Your order is complete!</h1>
      <p style={{ display: 'block', textAlign: 'center', color: 'grey', marginTop: '15px', fontSize: '16px' }}>We will call you back to confirm the order.</p>
    </>, {
    position: "top-center",
    autoClose: 4000,
    hideProgressBar: false,
    closeOnClick: true,
    pauseOnHover: true,
    draggable: true,
    containerId: 'success'
```

```
  });
 }

 return (
  <AppContext.Provider value={{
   success,
   clientInfo,
   setClientInfo,
   setPriceListener,
   currentItem,
   cartItems,
   removeFromCart,
   addToCart,
   additivesData,
   openCartModal,
   setCartItems
  }}>
   <div className="wrapper">

    <ToastContainer enableMultiContainer containerId={'success'} />
    <RegistrationRoutes checkoutModalOpen={checkoutModalOpen}
setCheckoutModalOpen={setCheckoutModalOpen} setIsAuth={setIsAuth}
phoneRegOpen={phoneRegOpen} setPhoneRegOpen={setPhoneRegOpen} />

    <SideCart
     setPhoneRegOpen={setPhoneRegOpen}
     setCheckoutModalOpen={setCheckoutModalOpen}
     isAuth={isAuth} cartOpen={cartOpen}
     setTotalPrice={setTotalPrice}
     totalPrice={totalPrice}
```

```
          setCartOpen={setCartOpen}
          cartItems={cartItems}
        />
        <div className="container">
          <Header
            isAuth={isAuth}
            setPhoneRegOpen={setPhoneRegOpen}
            setMobileMenuOpen={setMobileMenuOpen}
            setSelectedCity={setSelectedCity}
            citiesData={citiesData}
            selectedCity={selectedCity}
            setCityModalOpen={setCityModalOpen}
          />
          <Routes>
            <Route path="/" element={<>
              <Nav
                setCartOpen={setCartOpen}
                totalPrice={totalPrice}
                totalCount={totalCount} />
              <Slider />
              <main>
                <SectionFood error={error} notify={notify} foodData={foodData} isLoading={isLoading} />
                <ShippingPayment />
              </main>
            </>} />
            <Route path="/profile" element={<Profile isAuth={isAuth} setIsAuth={setIsAuth} />} />
          </Routes>
```

```
            </div>
            <Media queries={{ tablet: "(max-width: 950px)", phone: "(max-width: 750px)" }}>
              {matches => (
               <Fragment>
                 {matches.tablet
                  && <Fragment>
                   <CSSTransition
                    in={mobileMenuOpen}
                    timeout={300}
                    classNames="cart"
                    unmountOnExit
                    onEnter={() => setMobileMenuOpen(true)}
                    onExited={() => setMobileMenuOpen(false)}>
                    <MobileMenu isAuth={isAuth} setPhoneRegOpen={setPhoneRegOpen} selectedCity={selectedCity} setMobileMenuCitiesOpen={setMobileMenuCitiesOpen} setMobileMenuOpen={setMobileMenuOpen} />
                   </CSSTransition>
                   <CSSTransition
                    in={mobileMenuCitiesOpen}
                    timeout={300}
                    classNames="cart"
                    unmountOnExit
                    onEnter={() => setMobileMenuCitiesOpen(true)}
                    onExited={() => setMobileMenuCitiesOpen(false)}>
                    <MobileCityMenu setSelectedCity={setSelectedCity} citiesData={citiesData} setMobileMenuCitiesOpen={setMobileMenuCitiesOpen} />
                   </CSSTransition>
```

```
                    <ToastContainer enableMultiContainer containerId={'notify'} />
                    {cartItems.length >= 1 && <MobileCartIcon totalCount={totalCount}
setCartOpen={setCartOpen} />}
                  </Fragment>
                 }
                 {matches.phone
                  ? <CSSTransition
                    in={cardModalOpen}
                    timeout={200}
                    classNames="mobile-card-modal"
                    unmountOnExit
                    onEnter={() => setCardModalOpen}
                    onExited={() => setCardModalOpen}
                   >
                    <CardModal          notify={notify}          addToCart={addToCart}
setCardModalOpen={setCardModalOpen} item={currentItem} />
                   </CSSTransition>
                   : <Fragment>
                    <CSSTransition
                      in={cityModalOpen}
                      timeout={200}
                      classNames="alert"
                      unmountOnExit
                      onEnter={() => setCityModalOpen}
                      onExited={() => setCityModalOpen}
                    >
                      <CityModal                  setCityModalOpen={setCityModalOpen}
setSelectedCity={setSelectedCity} citiesData={citiesData} />
                    </CSSTransition>
                    <CSSTransition
```

```
                    in={cardModalOpen}
                    timeout={200}
                    classNames="alert"
                    unmountOnExit
                    onEnter={() => setCardModalOpen}
                    onExited={() => setCardModalOpen}
                  >
                    <CardModal notify={notify} addToCart={addToCart}
setCardModalOpen={setCardModalOpen} item={currentItem} />
                  </CSSTransition>
                </Fragment>


              }

            </Fragment>
          )}
        </Media>

        <Footer />
      </div>
    </AppContext.Provider>
  );
}

export default App;
```

FoodCard.jsx

```
import React, { useContext } from 'react'
import './foodcard.scss'
```

```
import BuyButton from '../UI/Buttons/BuyButton/BuyButton'
import { AppContext } from '../../context'
function FoodCard({ item }) {

  const { addToCart, openCartModal } = useContext(AppContext)

  return (
    <article
      onClick={() => { item.pizza ? openCartModal(item) : addToCart(item) }}
      style={item.category === 'drink' ? { height: '370px' } : null}
      className='food-card'
      id='foodcard'
    >
      <div>
        <picture>
          <img width={260} height={260} src={item.img} alt={item.title} />
        </picture>
        <h3 className={item.title.length >= 26 ? 'food-card__longtitle' :
null}>{item.title}</h3>
        <p style={{}}>{item.composition}</p>
      </div>
      <div className='food-card__price d-a-j'>
        <p>{item.pizza ? 'from' : ''} {item.price}€</p>
        <BuyButton>+ ADD</BuyButton>
      </div>

    </article>


  )
```

```
}

export default FoodCard
```

FoodCard.scss

```
.food-card {
  width: 288px;
  height: 405px;
  cursor: pointer;
  margin-right: 42px;
  margin-bottom: 45px;
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  .food-card__longtitle {
    font-size: 20px;
  }
  &:nth-child(4n + 0) {
    margin-right: 0;
  }

  picture {
    display: flex;
    justify-content: center;
    img {
      transition: all 0.2s ease;
      &:hover {
        transform: translateY(3px);
      }
    }
```

```
  }
  h3 {
    font-size: 22px;
    margin: 9px 0px 10px 0px;
    position: relative;
    z-index: 1;
  }
  p {
    color: #767676;
    font-size: 16px;
  }
  .food-card__price {
    p {
      font-size: 22px;
      color: #000000;
    }
  }
}

@media (max-width: 1500px) {
  .food-card {
    width: 305px;
    margin-right: 67px;
```

```
    &:nth-child(3n + 0) {
      margin-right: 0px;
    }
    &:nth-child(4n + 0) {
      margin-right: 67px;
    }
  }
}
@media (max-width: 1230px) {
  .food-card {
    width: 249px;
    height: 385px;
    margin-right: 31px;
    &:nth-child(3n + 0) {
      margin-right: 0px;
    }
    &:nth-child(4n + 0) {
      margin-right: 31px;
    }
    picture {
      img {
        width: 235px;
        height: 235px;
      }
    }
    h3 {
      margin: 9px 0px 2px 0px;
    }
    .food-card_longtitle {
      font-size: 19px ;
```

```
      margin: 5px 0px 2px 0px;
    }
    p {
      margin: 9px 0px 10px 0px;
      font-size: 14.5px;
    }
    .food-card__price {
      p {
        font-size: 20px;
      }
    }
  }
}
```