МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Інтернет магазин замовлення доставки піци»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Шовкопляс О.А.**

Студента групи ІН – 62 — **Шерстюка А.В.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую _____

Зав. кафедрою Довбиш А.С.

«_____» _____ 20 р.

Завдання

до випускної роботи

Студента четвертого курсу, ІН – 62 спеціальності “Інформатика” денної форми навчання Шерстюк А.В.

Затверджена наказом по СумДУ

№_____ від _____ 20___ р.

Зміст пояснювальної записки: 1) аналіз предметної області;

2) моделювання та проектування; 3) програмна реалізація.

Дата видачі завдання “_____” _____ 20___ р.

Керівник випускної роботи _____ Шовкопляс О. А.

Завдання прийняв до виконання _____ Шерстюк А. В.

РЕФЕРАТ

Записка: 66 стор., 41 рис., 2 додатки, 20 джерел.

Об’єкт дослідження – процесс розробки веб-ресурсу для доставки піци.

Мета роботи – розробка веб-ресурсу для доставки піци. Завдяки якому користувач за декілька хвилин зможе замовити піцу не покидаючи власного дому, або робочого місця.

Методи дослідження – проведено аналіз аналогів, проектування і розробку додатку.

Результати – виконано розробки веб-ресурсу для доставки піци. Детально розглянуто методи створення веб-ресурсу для доставки піци мовою JS. Проведено аналіз існуючих аналогів, визначено їх переваги та недоліки. Проведено порівняння альтернатив з розроблюваним додатком. Було розглянуто обрані для створення додатка компоненти розробки. Розглянуто та обрано базу данных MySQL.

PHP, JAVASCRIPT, ПІЦА, ПРОТОТИП, СКРИПТИ, LARAVEL, САЙТ

ЗМІСТ

ВСТУП

Кожен день у нас накопичується маса справ, кроків, завдань і т.д. Необхідно все встигати вчасно. Треба робити все без помарок і промахів. Важливо створювати і пропонувати щось нове, масштабне і цікаве, до того ж швидше, ніж конкуренти. І, звичайно, ніколи не підводити керівника, партнера або співробітника. При такому щільному розподілі часу просто необхідно встигнути бути в декількох місцях одночасно. І при цьому життєво важливо залишатися при здоровому розумі і пам'ятати до якої мети ви так стрімко бігли.

З активним темпом життя все менше вистачає часу і можливостей на приготування смачної їжі, тому доводиться відвідувати заклади швидкого харчування, що не завжди зручно. Сьогодні все більшої популярності набирають служби кур’єрської доставки.

Доставка кур'єром – це передача кореспонденції, листів або невеликих вантажів через довірену особу – кур'єра. Головним завданням яких є доставка товару, або продуктів харчування за короткий проміжок часу.

Саме для розвантаження вашого часу і економії енергії так важливі кур'єрські послуги та служби.

Мета кваліфікаційної роботи є розробка веб-ресурсу для доставки піци. Завдяки якому користувач за декілька хвилин зможе замовити піцу не покидаючи власного дому, або робочого місця. Для реалізації поставленої мети потрібно вирішити такі задачі:

– Вивчити процеси організації замовлення і доставки піци та провести аналіз сайтів-аналогів для розроблення технічного завдання.

– Обрати та налаштувати інструменти реалізації.

– Визначити структуру даних та спроектувати веб-сайт.

– Реалізувати додатки та розробити інструкції користувача.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Насправді, кур'єрська доставка – це комплексний і дуже трудомісткий процес. У ньому задіяна величезна кількість ресурсів – як людських, так і інформаційних. Незважаючи на свою видиму простоту, функціонування даної служби не можливо без злагодженості між собою всіх процесів роботи кур'єрської служби доставки, які відбуваються після отримання заявки.

Робота кур'єрської фірми починається з ініціативи відправника, а саме – зі звернення до служби доставки. Замовлення, вступаючи до диспетчера, передається вільному кур'єру. Як правило, це відбувається автоматично завдяки CRM-системі. CRM-система це прикладне програмне забезпечення або сайт для організацій, призначені для автоматизації стратегій взаємодії з замовниками (клієнтами), зокрема для підвищення рівня продажів, оптимізації маркетингу і поліпшення обслуговування клієнтів шляхом збереження інформації про клієнтів і історію взаємин з ними, встановлення і поліпшення бізнес-процесів і подальшого аналізу результатів [1].

Також за допомогою автоматизованих систем, сайтів (або відділу логістики), формується найкоротший шлях доставки. Кур'єру залишається підтвердити замовлення і виїхати за посилкою.

Існують як невеликі веб-сайти, які надають коротку інформацію про компанії та послуги, так і обширний онлайн-каталог компанії, який надає найточніші характеристики товару, зображення та ціни. Зазвичай такі онлайн-каталоги створюються для того, щоб відвідувачі могли знайти детальний опис та зображення продуктів. Тобто сайт у цьому випадку є лише ілюстрованим каталогом реклами товару.

По-перше, інтернет-магазин – це веб-сайт, який містить докладний каталог товарів з описами та зображеннями. Основна відмінність від звичайного інтернет-каталогу полягає в тому, що ви не тільки можете

побачити товари, що відображаються в інтернет-магазині, але також можете замовити їх, не відхиляючись від вихідного положення.

1.2 Аналіз сайтів аналогів

Для всіх інтернет-магазинів властивий певний обов'язковий набір елементів, таких як:

– Спеціалізований каталог з підрозділами.

– Система реєстрації користувача.

– Система доставки товару.

Однак, незважаючи на загальні риси, Інтернет-магазини все ж відрізняються один від одного. Власник кожного магазину прагне зробити свій сайт максимально зручним для відвідувача, удосконалюючи систему замовлення та способи переходу від одного розділу до іншого.

Архітектура інтернет-магазину повинна бути проста і інтуїтивно зручна.

Інтернет-магазин має такі переваги:

– допомагає швидко зорієнтуватися в асортименті і знайти потрібний товар або послугу (за тематикою, назвою, ціною тощо).

– розглянути товар "з усіх боків", порівняти його характеристики, ціну, зовнішній вигляд з іншими товарами.

– розрахувати точну вартість замовлення; відібрати товар в корзину, оформити замовлення on-line, оформити доставку на будинок.

– переглядати інформацію за поточним замовленням.

Для аналізу існуючих рішень було обрано декілька сайтів зі схожою тематикою.

Один із популярних сайтів по замовленню і доставці піци в місті Суми це sumypizza.com (рис. 1.1).

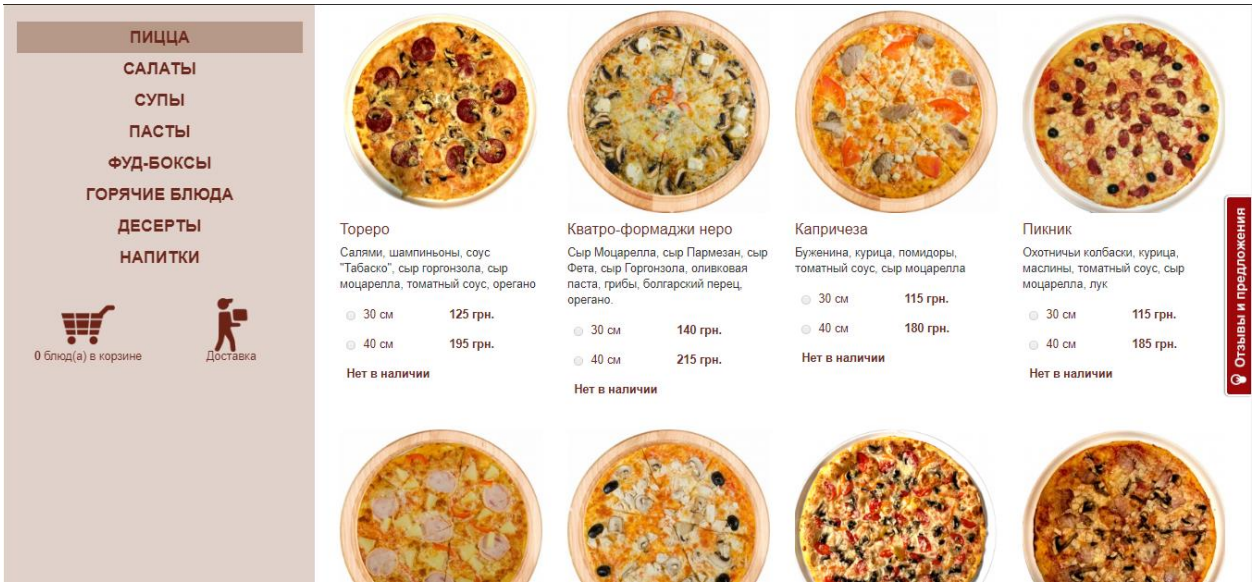

Рисунок 1.1 – Головна сторінка сайту sumypizza.com

Даний сайт має достатньо зручний інтерфейс. Проте сайт також має ряд недоліків. Наприклад відсутність зручного пошуку не дає можливості знати бажаний продукт за назвою, також немає можливості додати додаткові інгредієнти за бажанням. Основним недоліком є відсутність можливості перегляду детальної інформації про товар і залишити відгук.

Наступний сайт подібної тематики який також користується популярністю в місті Суми це safari-cafe.com.ua (рис. 1.2).

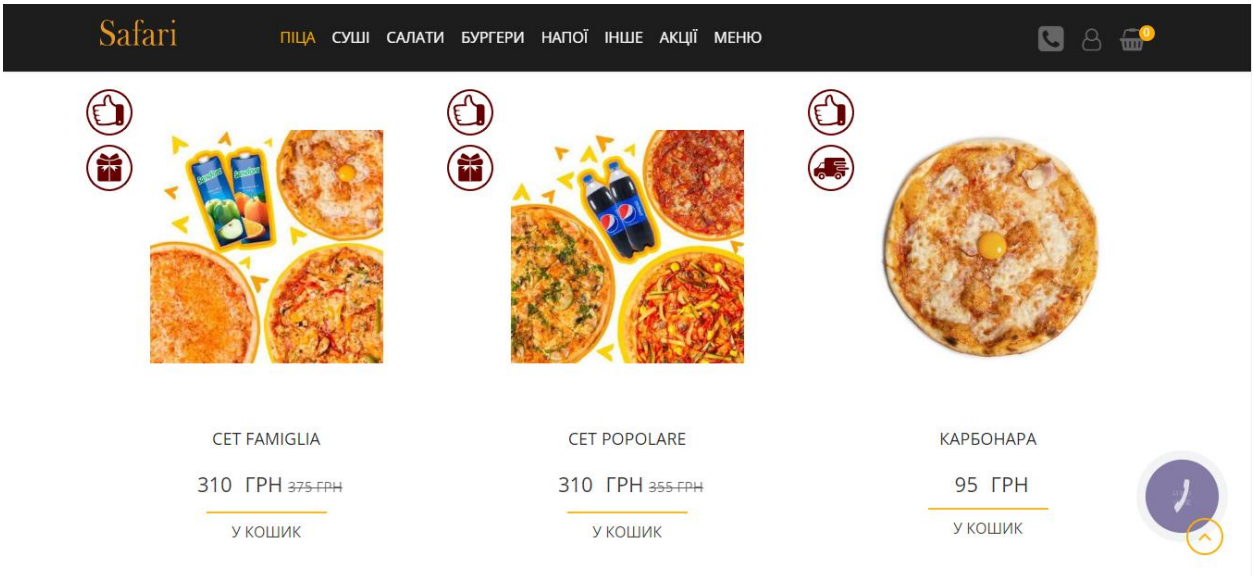

Рисунок 1.2 – Головна сторінка сайту safari-cafe.com.ua

Даний сайт також має ряд недоліків. А саме відсутність пошуку, в також вибору розміру піци, що є достатньо важливою функцією при певних умовах.

В результаті проведеного аналізу було вирішено розробити веб сайт по замовленню доставки піци який матиме необхідний набір функцій для замовлення доставки піци і досить зручний інтерфейс, що дозволяє за досить короткий час оформити замовлення.

1.3 Вибір засобів реалізації

Розробка веб-сайтів – це процес створення веб-сайтів та додатків для Інтернету або для приватної мережі, відомий як інтернет. Розробка веб-сайтів це програмування, що забезпечує функціональність веб-сайту.

Сайт – структурована інформаційна одиниця всесвітньої павутини. Він може містити як одну, так і величезну кількість сторінок. Наприклад, сайт компанії IBM містить кілька тисяч сторінок. Зазвичай на сайті виділяють таке поняття, як головна сторінка. Це та сторінка, яка відображається на сайті першою.

Сторінка веб-сайту – це набір текстових файлів з тегами HTML. Коли відвідувач завантажує ці файли на свій комп'ютер, вони обробляються браузером і виводяться на інструмент відображення користувача наприклад монітор, смартфон, принтер або принтер. HTML дозволяє відформатувати текст, виділити функціональні елементи, створити гіпертекстові посилання (гіперпосилання), а також вставити зображення, записи та інші мультимедійні елементи на сторінку. Ви можете змінити відображення сторінки, додавши стилі CSS, щоб об’єднати всі елементи форматування (розмір та колір великих літер, розмір та вигляд вставлених блоків тощо) у файлі чи сценарії, що дозволяє переглядати сторінку події [2].

1.3.1 Клієнтські мови програмування

Мови розмітки використовуються для визначення форматування текстового файлу. Іншими словами, мова розмітки повідомляє програмне забезпечення, яке відображає текст, як слід форматувати текст. Мови розмітки є повністю розбірливими для людини – вони містять стандартні слова, але теги розмітки приховані від користувача в кінцевому результаті роботи.

Дві найпопулярніші мови розмітки – HTML та XML. HTML розшифровується як мова розмітки HyperText та використовується для створення веб-сайтів. Додані до звичайного текстового документа, всі теги HTML описують, як цей документ повинен відображатися веб-браузером.

Невідмінна частина будь-якої веб сторінки – каскадні стилі розмітки сторінки або CSS. Стилі – це в основному сукупність стилістичних правил. Мови аркушів стилів використовуються, доволі буквально, для оформлення документів, написаних мовами розмітки. CSS розшифровується як каскадні таблиці стилів з акцентом на «стиль».

Хоча HTML використовується для структури веб-документа, визначає такі речі, як заголовки та абзаци, і дозволяє вставляти зображення, відео та інші медіа, використаний CSS визначає стиль створеного документа.

Сьогоднішній світ веб-сайтів важко уявити без мови JavaScript. JavaScript – це те, що робить живими веб-сторінки, які людина кожен день переглядає в своєму веб-браузері [3].

JavaScript був створений в 1995 році в компанії Netscape в якості мови сценаріїв в браузері Netscape Navigator. Спочатку мова називалася LiveScript, але на хвилі популярності в той момент іншої мови Java LiveScript був перейменований в JavaScript.

Спочатку JavaScript мав досить невеликі можливості. Його мета полягала лише в тому, щоб додати трохи поведінки на веб-сторінку. Наприклад, обробити натискання кнопок на веб-сторінці, зробити дії, пов'язані перш за все з елементами управління.

Однак розвиток веб-середовища, поява HTML5 і технології Node.js відкрило перед JavaScript набагато більші горизонти. Зараз JavaScript продовжує використовуватися для створення веб-сайтів, тільки тепер він надає набагато більше можливостей. JavaScript класифікують як прототипну, скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

З самого початку існувало кілька веб-браузерів (Netscape, Internet Explorer), які надавали різні реалізації мови. І щоб звести різні реалізації до загального стрижня і стандартизувати мову під керівництвом організації ECMA, був розроблений стандарт ECMAScript. Найвідомішими реалізаціями стандарту є мови JavaScript, JScript та ActionScript, які широко використовується у веб розробці. Самі терміни JavaScript і ECMAScript є багато в чому взаємозамінними і відносяться до одного і того ж мови. До теперішнього часу ECMA було розроблено декілька стандартів мови, які відображають його розвиток. JavaScript є мовою, що інтерпретується. Це означає, що код на мові JavaScript виконується за допомогою інтерпретатора. Інтерпретатор отримує інструкції мови JavaScript, які визначені на веб-сторінці, виконує їх (або інтерпретує) [4].

1.3.2 Серверні мови програмування

На сьогоднішній день PHP є найбільш поширеною мовою веб-програмування. Переважна більшість сайтів і веб-сервісів в інтернеті написано за допомогою PHP. За деякими оцінками, PHP застосовується більш ніж на 80% сайтів, серед яких такі сервіси, як facebook.com, baidu.com і інші. Популярність PHP обумовлена простотою мови, яка дозволяє швидко і легко створювати сайти і портали різної складності. PHP був створений в 1994 році датським програмістом Расмусом Лердорфом і спочатку був набір скриптів на іншій мові Perl. Пізніше цей набір скриптів був переписаний в

інтерпретатор на мові Сі. І з самого виникнення PHP (скорочення від PHP: Hypertext Preprocessor – PHP: препроцесор гіпертексту) представляв зручний набір інструментів для спрощеного створення веб-сайтів і веб-додатків [5].

Переваги PHP:

- для всіх найбільш поширених операційних систем (Windows, MacOS, Linux) є свої версії пакетів розробки на PHP, а це значить, що можна створювати веб-сайти на будь-який з цих операційних систем;
- PHP може працювати в зв'язці з різними веб-серверами: Apache, Nginx, IIS;
- простота і легкість освоєння. Як правило, вже маючи невеликий досвід в програмуванні на PHP, можна створювати найпростіші веб-сайти;
- PHP має схожий синтаксис з мовою Сі, тому, знаючи Сі або однією з мов з сіподобним синтаксисом, буде простіше опанувати PHP;
- PHP підтримує роботу з великою кількістю систем баз даних (MySQL, MSSQL, Oracle, Postgre, MongoDB та інші);
- поширеність хостингових послуг і їх дешевизна;
- постійний розвиток. PHP продовжує розвиватися, з'являються все нові версії, які несуть нові функції, адаптуючи мову програмування до нових викликів. І, як правило, перейти на нову версію не становить труднощів [6].

Проте для спрощення розробки і підвищення ефективності були створені фреймворки.

Фреймворк – це каркас, призначений для створення динамічних веб-сайтів, веб-додатків, служб або ресурсів. Він спрощує розробку і позбавляє від необхідності писати звичайний код. Фреймворк спрощує доступ до бази даних, розробку інтерфейсу та зменшує дублювання коду.

Розглянемо один з популярних фреймворків Yii 2. Yii – це універсальний фреймворк, і він може бути задіяний у всіх типах веб-додатків.

Завдяки своїй складовій структурі та чудовій підтримці кешу, фреймворки особливо підходять для розробки великих проектів, таких як портали, форуми, CMS, магазини або програми RESTful. Yii – це командний проект. Він підтримується і розвивається сильною командою та великою спільнотою розробників. Автор цього фреймворку стежать за тенденціями розвитку веб-сайтів та інших проектів.

Найбільш підходящі можливості і кращі практики регулярно впроваджуються в фреймворк в вигляді простих і елегантних інтерфейсів:

– Як і багато інших PHP фреймворків, для організації коду Yii використовує архітектурний патерн MVC.

– Yii дотримується філософії простого і елегантного коду не намагаючись ускладнювати дизайн тільки заради проходження будь-якими шаблонами проектування.

– Yii є full-stack фреймворком і включає в себе перевірені і добре зарекомендовані в себе можливості: ActiveRecord для реляційних і NoSQL баз даних, підтримку REST API, багаторівневе кешування та інше.

– Yii відмінно розширюємий. Можна налаштувати або замінити практично будь-яку частину основного коду. Використовуючи архітектуру розширень, легко ділитися кодом або використовувати код спільноти.

Розглянемо також не менш популярний фреймворк Laravel 7. Laravel – це фреймворк для веб-додатків з виразним і елегантним синтаксисом. Він дозволить спростити вирішення основних завдань, таких як аутентифікація, маршрутизація, сесії і кешування. Laravel – це спроба об'єднати все найкраще, що є в інших PHP фреймворк, а також Ruby on Rails, ASP.NET і Sinatra [7].

Laravel – доступний, але потужний. Має безліч відмінних інструментів для великих, надійних додатків:

– Як і багато інших PHP фреймворки, для організації коду Laravel використовує архітектурний патерн MVC.

– Laravel дотримується філософії виразного, красивого синтаксису. Він призначений для людей, які цінують елегантність, простоту і читаність.

– Функціонал Laravel є простим для розуміння і використання.

– Більшість функцій Laravel чудово працюють, не вимагаючи додаткових налаштувань. Вони спираються на загальноприйняті стандарти написання коду, роблячи його інтуїтивно зрозумілим.

– Документація Laravel зручна і постійно оновлюється.

– Чудова IoC (Інверсія управління).

– Зручна система міграцій.

– Інтегрована система модульного тестування [8].

Обидва фреймворка для організації коду використовують архітектурний патерн MVC. При детальному розгляді ці фреймворки складаються з трьох основних компонентів:

– Model – це сам додаток який нічого не знає про HTTP запитах.

– View – це HTTP запит / відповідь і уявлення даних, які вимагає клієнт від сервера.

– Controller – це кілька і більше класів, чиє завдання – абстрагування моделі від HTTP.

За архітектурою дані фрейморкі ідентичні, так як реалізують один і той же архітектурний патерн. Він дозволяє швидко реалізувати проект, а також легко супроводжувати і масштабувати його в подальшому [9].

Розширення важлива частина фреймворка, тому розглянемо їх докладніше. Розширення – це можливість підключати додаткові модулі або бібліотеки для фрейморка, які дозволять розширити його можливості. Yii і Laravel підтримують таку можливість. На даний момент, розширень набагато більше, звичайно ж, у Yii, так як фреймворк живе набагато довше, ніж Laravel, однак другий, в свою чергу, розвивається дуже великими темпами і включає в себе багато всього для роботи відразу «з коробки».

Розширення є абсолютно для будь-якого завдання, будь то панель адміністратора або ж платіжна система.

Чимало важливим фактором при виборі фреймворка є його документація. Через малу кількість років розробки або особистих переконань авторів документація Laravel дуже мізерна і неінформативна. Тому для навчання доведеться дивитися вихідний код, і як наслідок, не завжди буде зрозуміло, що робить та чи інша функція. В Yii ж є велика і корисна документація, де описана кожна функція із зазначенням її призначення і приклад використання. Так що, безсумнівно, це величезний плюс фреймворка Yii.

Для належної роботи сайту важлива підтримка REST API. REST – це архітектурний стиль взаємодії між компонентами розподілених додатків у мережі. REST – це набір послідовних обмежень, які слід враховувати при проектуванні розподіленої системи гіпермедіа. У деяких випадках (інтернет-магазини, пошукові системи, інші системи на основі даних) це може підвищити продуктивність та спростити архітектуру. Загалом, компоненти REST взаємодіють із клієнтами та серверами у всесвітній мережі. REST – це альтернатива RPC.

В Інтернеті віддалені процедурні виклики можуть бути звичайними HTTP-запитами (зазвичай "GET" або "POST"; це називається "REST-запит"), а необхідні дані передаються як параметри запиту.

Для веб-служб, які відповідають REST тобто, не порушуючи жодних обмежень, що застосовуються до нього, використовується термін "RESTful". На відміну від веб-служб на основі SOAP, веб-API RESTful не має "офіційного" стандарту. Справа в тому, що REST – це архітектурний стиль, а SOAP – протокол. Хоча сам REST не є стандартом, більшість реалізацій RESTful використовують стандарти HTTP, URL, JSON та XML.

В кінцевому рахунку, в даний час всі сучасні PHP фреймворки зобов'язані підтримувати REST архітектуру. Нижче наведено список можливостей фреймворків для роботи з REST API [10].

Yii 2:

– Підтримка JSON, JSONP і XML.
– Роутинг відповідно до REST-запитами.
– Підтримка HATEAOS.
– Кешування запитів.
– Обмеження швидкості.

Laravel:

– Налаштування роутінга REST-запитів.
– Підтримка JSON, JSONP.

На основі проведеного аналізу був зроблений вибір на сторону, що активно розвивається Laravel, його структура дозволяє легко масштабувати наш додаток. Yii є більш стабільним, але менш масштабованим.

1.3.3 Розробка бази даних

Для розробки бази даних використовують такі інструменти, як MySQL, Oracle чи SQL Server, щоб знайти, зберегти або відредагувати дані та повернути їх користувачеві за допомогою коду візуальної частини.

Вище перелічені мови використовуються не лише для створення веб-сайтів, програмного забезпечення та додатків, вони також використовуються для створення та управління базами даних.

Бази даних не розроблені для розуміння тих же мов, на яких запрограмовані програми, тому важливо мати мову, яку вони розуміють – як SQL, стандартну мову для доступу та маніпулювання реляційними базами даних. SQL означає структурована мова запитів [11].

Вона має власну розмітку і в основному дозволяє програмістам працювати з даними, що зберігаються в системі баз даних. Розглянемо детальніше різни типи, переваги та недоліки баз даних (табл.1.1).

Таблиця 1.1 – Типи баз даних

№	Назва	Переваги	Недоліки
1	Oracle	Має останні інновації та функції, що надходять від їх продукції, оскільки Oracle прагне встановити планку для інших інструментів управління базами даних.	Вартість Oracle може бути непосильною, особливо для менших організацій.
		Інструменти управління базами даних Oracle також надзвичайно надійні, і ви можете знайти той, який може робити практично все, про що ви можете подумати.	Після встановлення система може вимагати значних ресурсів, тому для впровадження Oracle може знадобитися оновлення обладнання.
2	MySQL	Безкоштовний.	Ви можете витратити багато часу і сил, щоб змусити MySQL робити те, що інші системи роблять автоматично.
		Пропонує безліч функціональних можливостей.	Немає вбудованої підтримки для XML або OLAP.

Продовження таблиці 1.1

№	Назва	Переваги	Недоліки
2	MySQL	Існують різноманітні інтерфейси користувачів.	Підтримка доступна для безкоштовної версії, але за неї потрібно заплатити.
		Це може бути зроблено для роботи з іншими базами даних, включаючи DB2 та Oracle.	
3	PostgreSQL	Ця система управління базами даних є масштабованою і може обробляти терабайти даних.	Не чітка документація.
		Підтримує JSON.	Конфігурація може бути заплутаною.
		Існують різноманітні заздалегідь визначені функції.	Швидкість може постраждати під час великих масових операцій чи запитів читання.
		Доступна низка інтерфейсів.	
4	MongoDB	Швидка і проста у використанні.	Установки за замовчуванням не захищені
		Підтримує JSON та інші документи NoSQL.	Налаштування може бути тривалим процесом.

Продовження таблиці 1.1

№	Назва	Переваги	Недоліки
4	MongoDB	Дані будь-якої структури можна зберігати та отримувати доступ до них швидко та легко.	Інструменти для перекладу SQL на запити MongoDB доступні, але вони додають додатковий крок до використання системи.
		SQL не використовується як мова запитів.	

В ході виконання даного проекту, вибір було зроблено на користь MySQL. По-перше, дана СУБД є безкоштовною. По-друге, супутній продукт MySQL Workbench дозволяє просто взаємодіяти зі встановленою БД, надаючи можливість через зручний інтерфейс користувача виконувати будь-які маніпуляції з даними.

Та нарешті, ця СУБД є однією з найбільш розповсюджених, займаючи друге місце за популярністю у світі, поступаючись лише корпоративній СУБД від Oracle. А це означає, що на випадок проблем чи збоїв в роботі цієї системи, в мережі можна знайти величезних кількість матеріалу для їх вирішення [12].

Для розробки клієнтської частини сайту було обрано мови програмування HTML, CSS і JavaScript так як вони є най популярнішими мовами для веб розробки і досить легкі у вивченні.

Розробка серверної частини буде відбуватись за допомогою мови програмування PHP з використанням фреймфорку Laravel. Ця мова активно використовується багатьма сайтами і досить легка у вивченні. Також найбільш популярні системи контролю контентом використовують саме цю

мову програмування, що надає можливість розробляти додатковий функціонал.

1.4 Постановка задачі

В результаті проведеного аналізу виявлено, що для досягнення поставленої мети, необхідно виконати такі задачі:

- обрати мову розробки програми;
- обрати метод підключення до зовнішніх даних;
- розробити адміністративну панель;
- модуль додання категорій та товару на сайт;
- модуль кошику з обраним товаром;
- модуль оформлення замовлення;
- модуль для додання коментарів;
- розробити модуль авторизації і обмеження функцій для ролей користувачів;
- провести тестування роботи сайту.

В першу чергу розробка сайту для замовлення доставки піци розрахована на користувачів з базовими навичками роботи з комп'ютером тому потребує зручного та інтуїтивно зрозумілого інтерфейсу.

Для роботи з системою було визначено декілька груп користувачів, які повинні пройти авторизацію:

- адміністратор;
- менеджер;
- користувач (замовник).

Наступним кроком, стало виявлення потреби, для чіткого визначення функціоналу програми.

Функціонал для групи користувачів (замовників):

1. Можливість перегляду товару за категоріями.
2. Можливість перегляду детальної інформації про окремий товар.

3. Можливість залишати відгуки про окремий товар.
4. Можливість додавати до продукції додаткових продуктів і послуг.
5. Пошук продукції.
6. Додання товару до кошику.
7. Можливість оформлення замовлення на сайті.

Функціонал для групи менеджерів:

1. Додання, редагування, видалення категорій продукції сайту.
2. Додання, редагування, перегляд, видалення продукції сайту.
3. Обробка, перегляд і видалення замовлень.
4. Перегляд і видалення коментарів.

Функціонал для групи адміністраторів:

1. Додання, редагування, видалення категорій продукції сайту.
2. Додання, редагування, перегляд, видалення продукції сайту.
3. Обробка, перегляд і видалення замовлень.
4. Перегляд і видалення коментарів.
5. Можливість редагувати загальну інформацію про сайт.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

2.1 Моделювання процесу роботи сайту

2.1.1 Моделювання процесу роботи сайту у нотації IDEF

На рівні 0 весь процес розглядається як функціональний блок із усіма відповідними робочими та керуючими об'єктами. Ця діаграма також відображає мету структурного аналізу та перспективу, з якої розглядається модель. Діаграма нульового рівня відображена на рисунку 2.1.

Рисунок 2.1 – Діаграма IDEF 0

Перша діаграма рівнів детально описує функцію обробки нульового рівня. Тому функціональний блок 0 розкладається на набір взаємопов'язаних під функцій [13]. Представлений варіант діаграми для даного проекту представлений на рисунку 2.2.

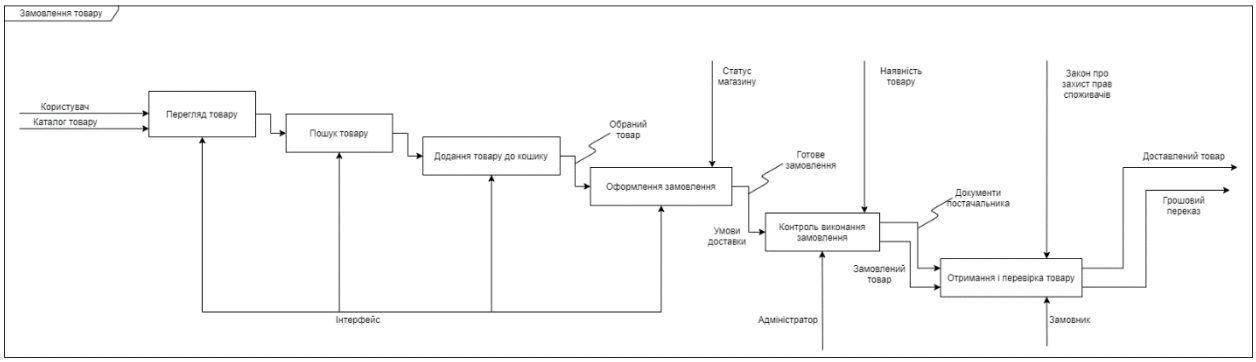

Рисунок 2.2 – Діаграма IDEF 1

2.1.2 Модель варіантів використання веб-додатку

Приклад діаграми використання може бути виражений як низхідний процес певного рівня від найбільш загальної та абстрактної концептуальної моделі вихідної системи до логіки відповідної програмної системи, а потім фізичної моделі.

Суть діаграми полягає в наступному: розроблена система представлена у вигляді сутностей або суб'єктів, які взаємодіють із системою, використовуючи так звані випадки використання. У цьому випадку учасником є будь-яка організація, яка взаємодіє із системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система і може виступати джерелом впливу на систему моделювання, визначену розробником. У свою чергу, випадки використання використовуються для опису послуг, які система надає учасникам. Іншими словами, кожен випадок використання визначає певний набір операцій, які система виконує після розмови з учасниками [14]. Приклад діаграми інтернет магазину представлено на рисунку 2.3.

Розглянемо інформацію про акторів (табл. 2.1) та опис варіантів використання (табл. 2.2).

Табл. 2.1. – Опис акторів

Назва	Опис
	Актори-користувачі
Користувач	Користувач інтернет магазину.
Адміністратор	Користувач, що використовує функції «адміністратора».
	Актори-зовнішні системи
База даних	База даних, що зберігає інформацію.

Табл. 2.2 – Опис варіантів використання

Назва	Опис
Авторизація	Функція входу в систему.
Додання відгуків	Функція, щоб надати можливість користувачу залишити відгуки про товар.
Перегляд товару	Користувач має можливість переглянути товар на сайті.
Пошук товару	Можливість виконувати пошук бажаного товару на сайті.
Додання товару в кошик	Функція для додання обраного товару в кошик для оформлення замовлення.
Редагування даних	Адміністратор може редагувати дані на сайті.
Додання товару	Адміністратор може додати новий товар на сайт.
Перегляд замовлень	Адміністратор може переглядати замовлення відвідувачів сайту.
Перегляд відгуків про товар.	Адміністратор може переглядати відгуки відвідувачів сайту.
Оформлення замовлень	Створення заявки замовлення обраного товару на сайті.

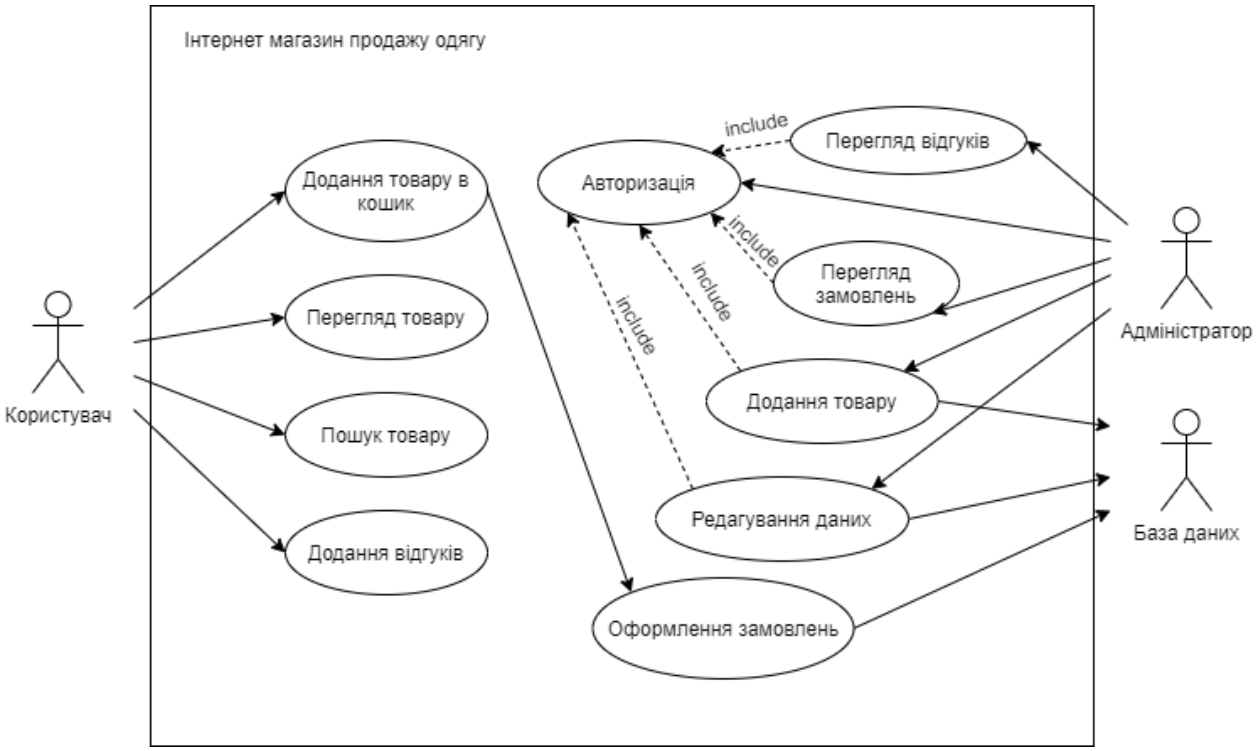

Рисунок 2.3 – Діаграма варіантів використання

2.1.3 Модель аналізу розроблюваного ПЗ

Діаграма комунікацій – це спеціальна діаграма взаємодії, яка фокусується на обміні даними між різними учасниками взаємодії.

Діаграма комунікацій не потребує відображати кожного учасника, як життєву лінію та відображати послідовність повідомлень вертикально, як діаграму послідовностей. Натомість учасників можна розміщувати за бажанням, дозволяючи комунікації показувати взаємозв'язок учасників та використовувати номери для представлення послідовностей повідомлень.

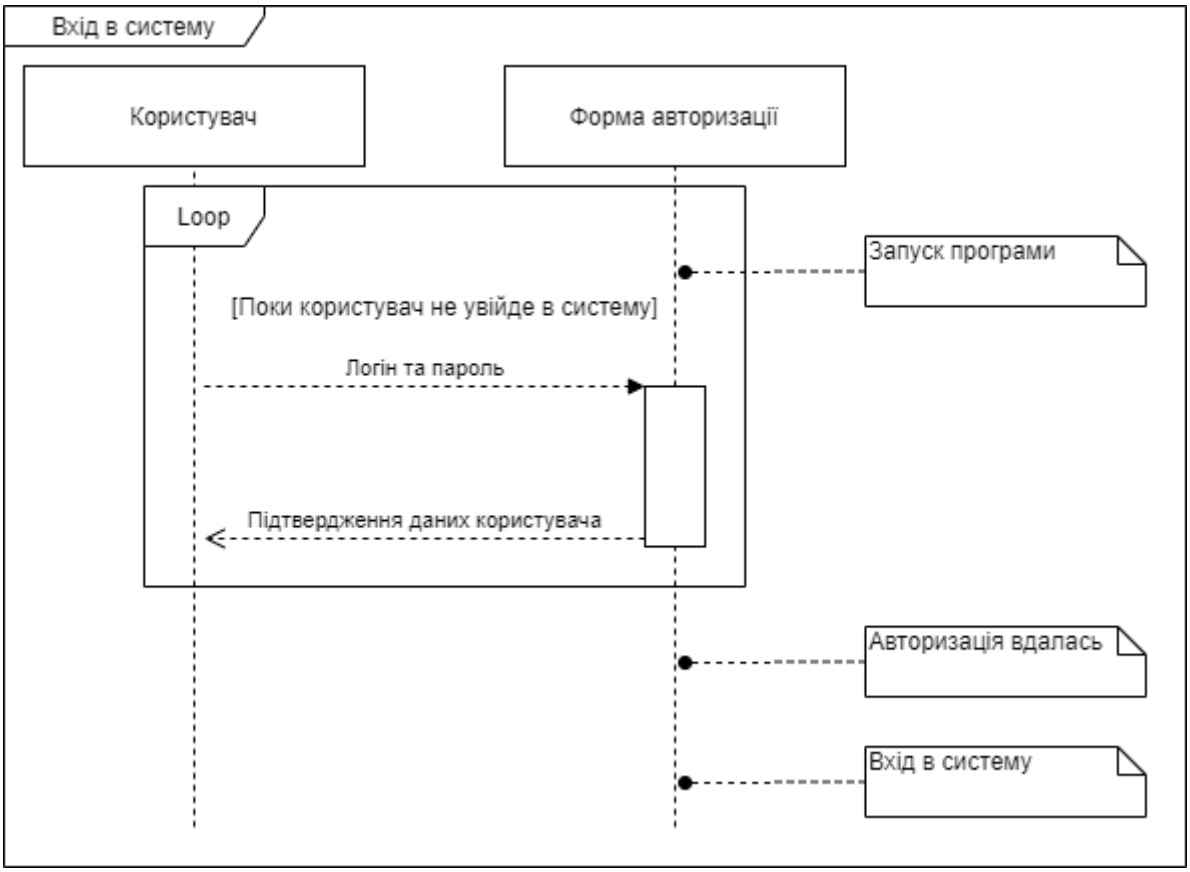

Рисунок 2.4 – Діаграма послідовності входу в систему

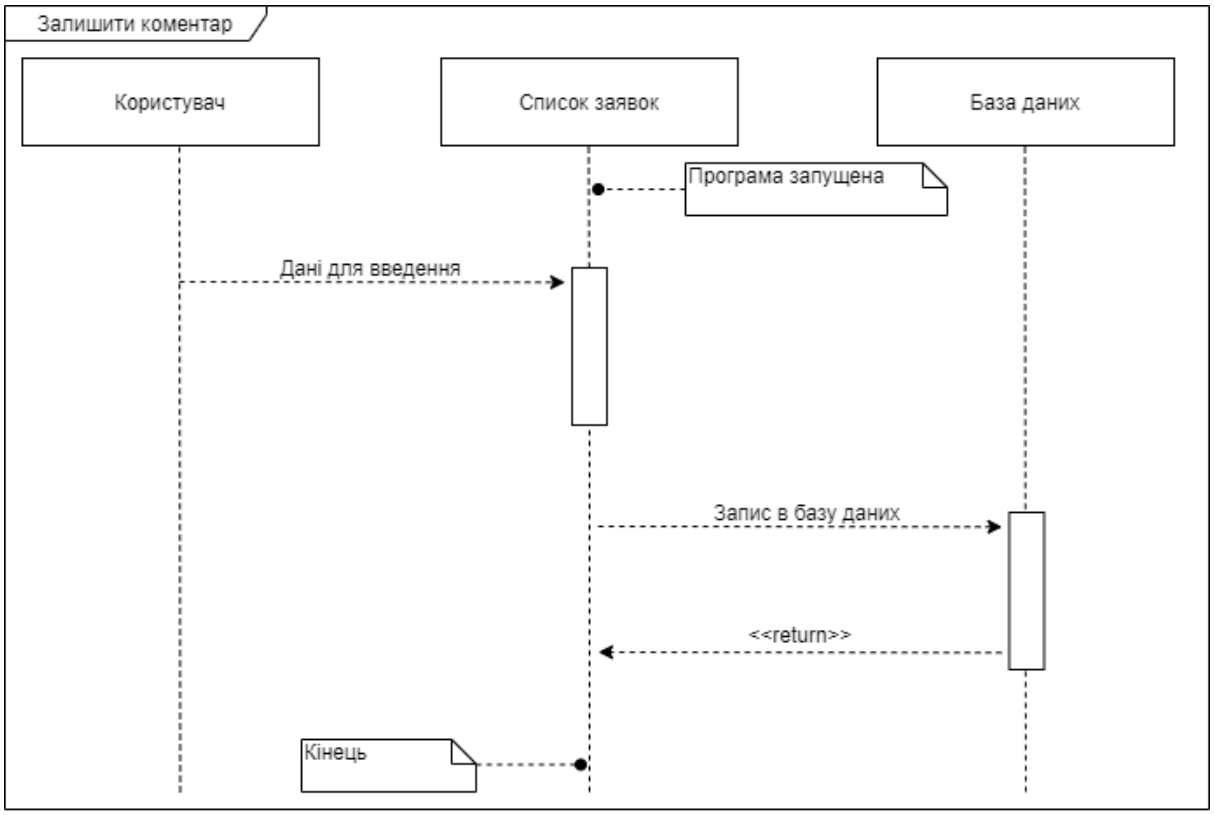

Рисунок 2.5 – Діаграма послідовності коментування

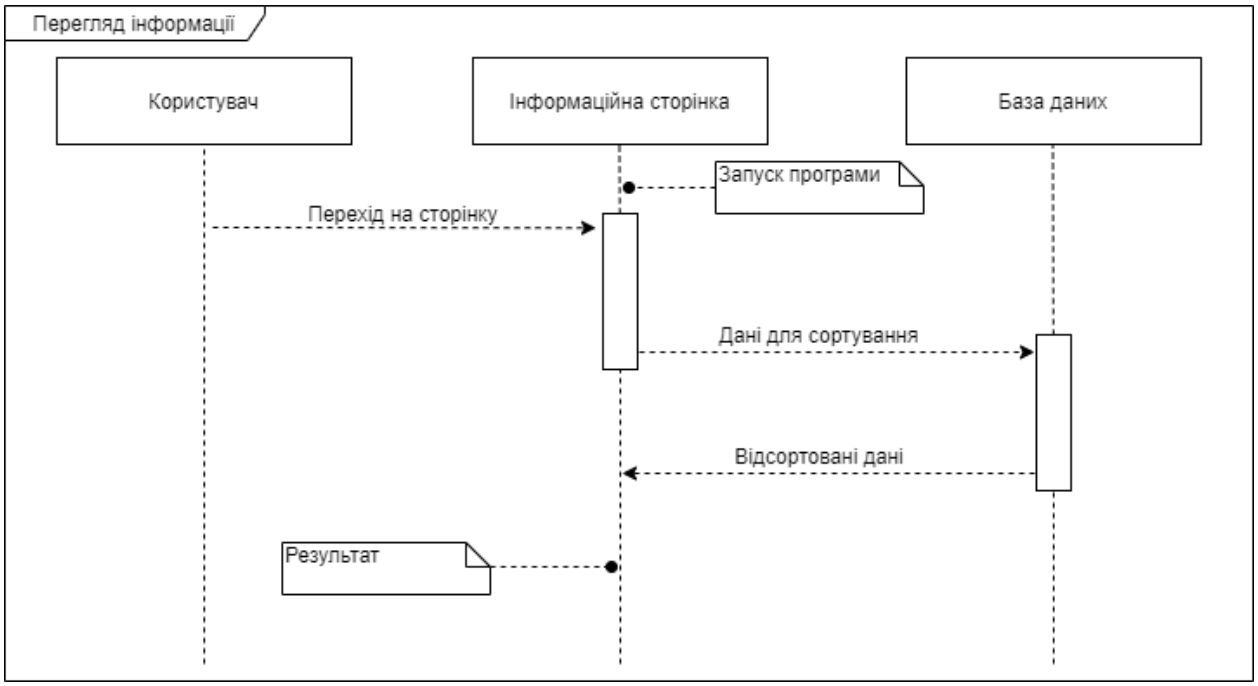

Рисунок 2.6 – Діаграма послідовності перегляду інформації

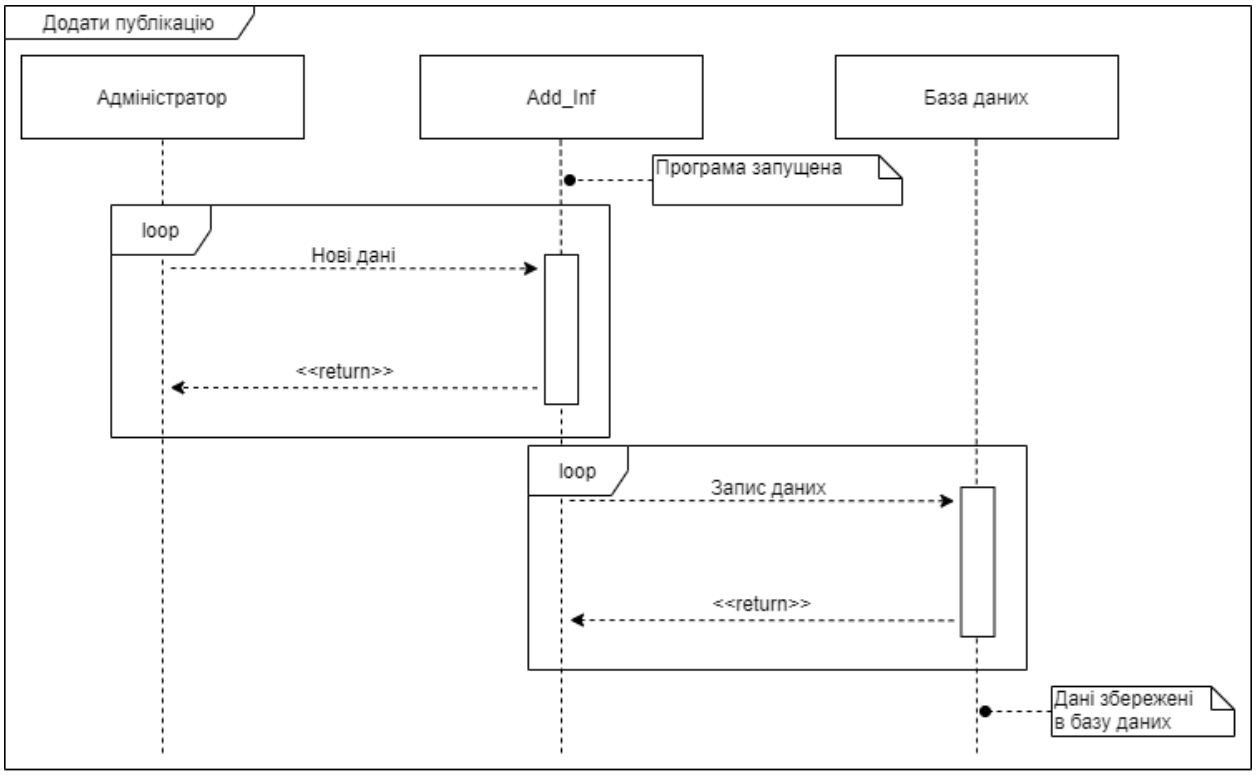

Рисунок 2.7 – Діаграма послідовності додавання інформації

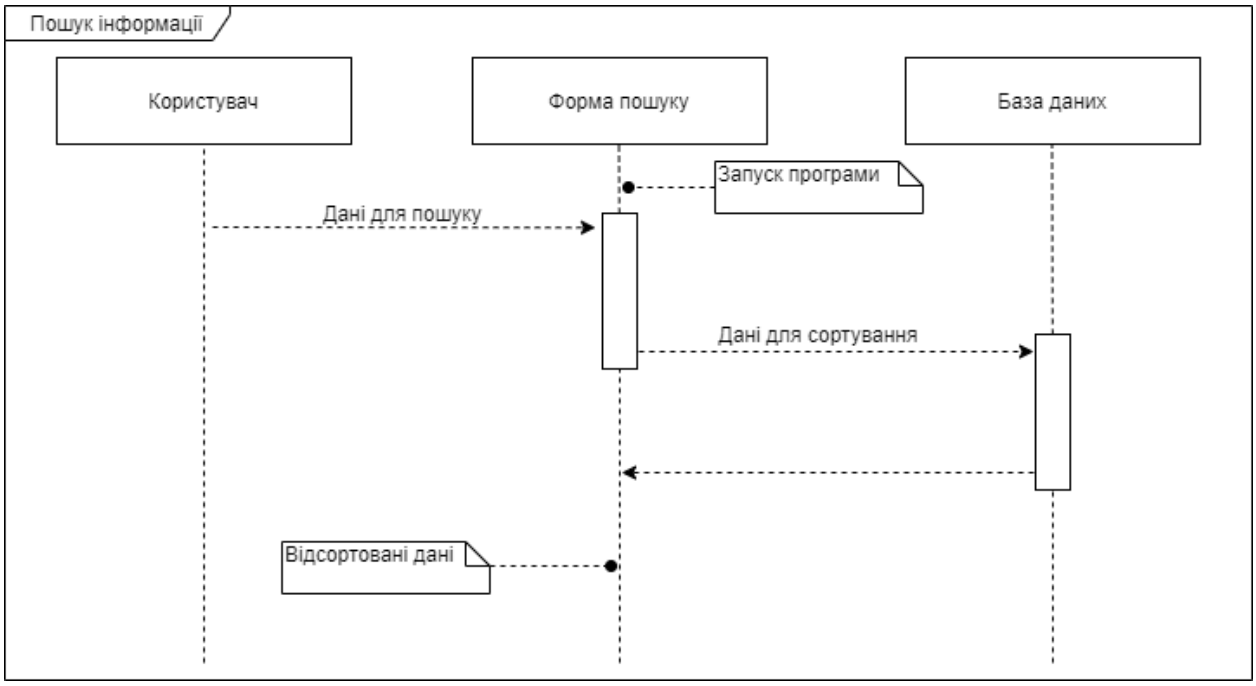

Рисунок 2.8 – Діаграма послідовності пошуку інформації

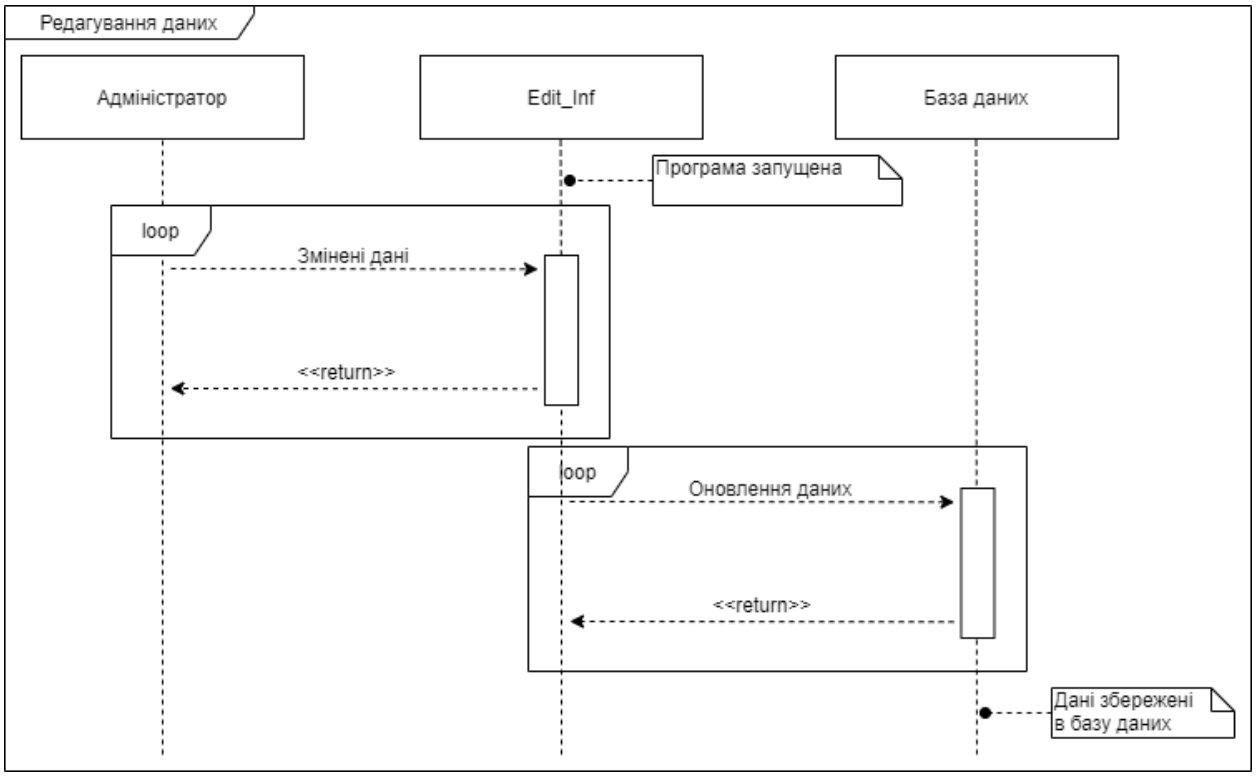

Рисунок 2.9 – Діаграма послідовності редагування інформації

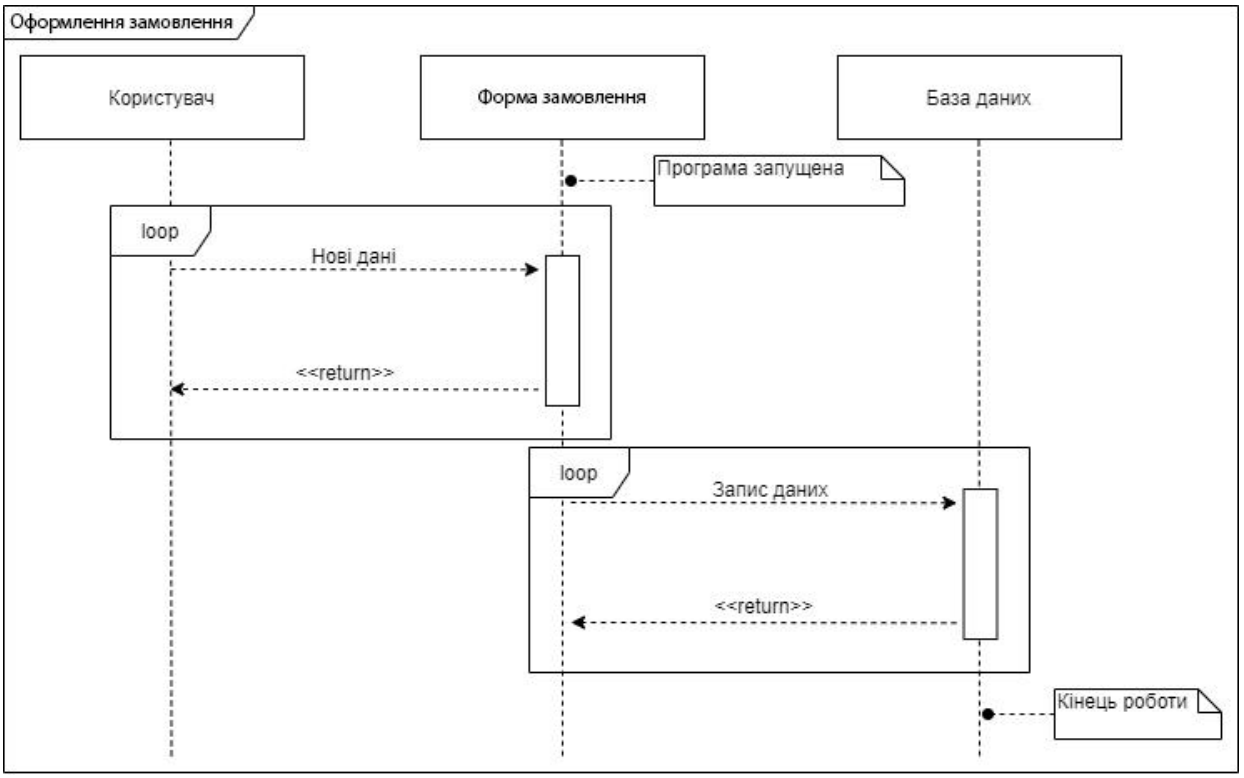

Рисунок 2.10 – Діаграма послідовності оформлення замовлення

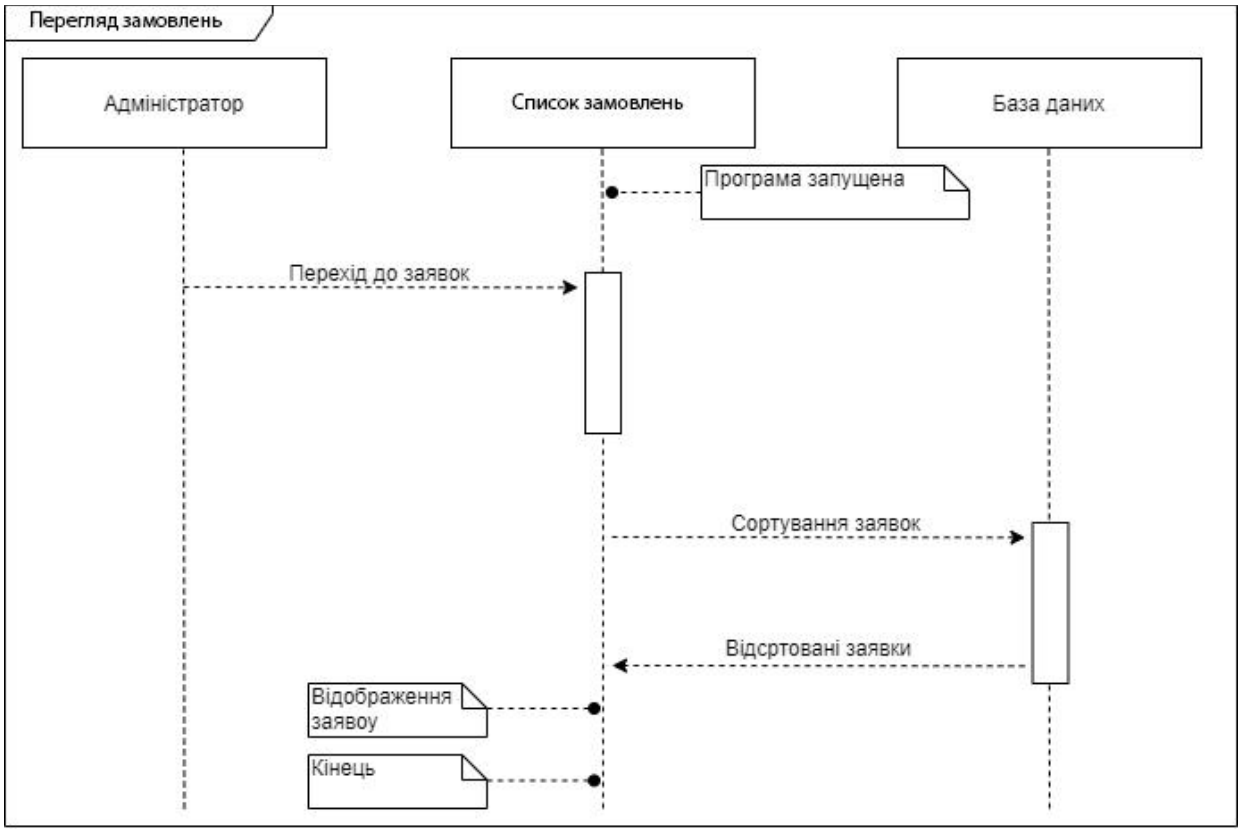

Рисунок 2.11 – Діаграма послідовності перегляду замовлень

2.2 Проектування інформаційної системи

Створення веб-ресурсу – це комплекс заходів, які об'єднують в собі розробку дизайну, інформаційне наповнення, застосування веб-технологій, спрямованих на задоволення потреб відвідувачів і власників майбутнього сайту.

Інтерфейс сайту розробляється в кілька етапів:

- визначення функцій сайту;
- вибір стилістичного оформлення;
- визначення сторінок і змісту;
- вибір мови програмування, яка задовольняє функції сайту;
- створення графічних елементів сайту;
- верстка сайту.

Після затвердження постановки завдання на розробку веб-сайту починається розробка дизайну [15]. Орієнтуючись на технічне завдання був розроблений ескіз головної сторінки (рис 2.12).

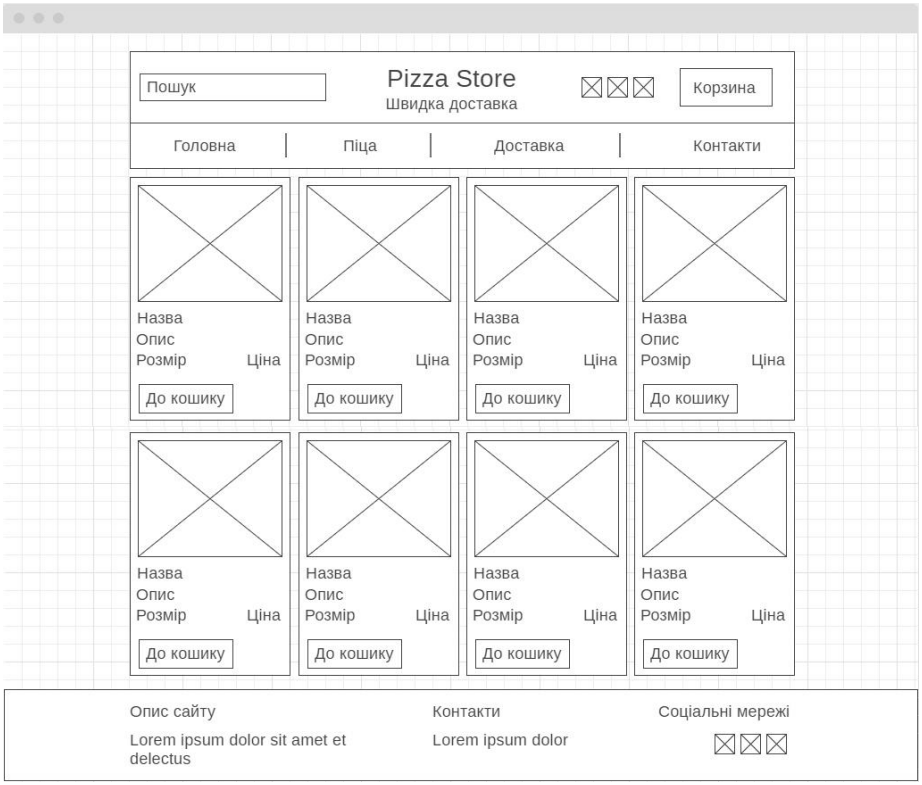

Рисунок 2.12 – ескіз головної сторінки сайту

В результаті виконання цього етапу роботи було отримано так звану "нарізку веб-сайту", тобто, готовий шаблон, який буде використаний як основа для розробки окремих сторінок.

Написання шаблону починається із створення «скелету» з використанням мови розмітки HTML. Після чого за допомогою мови каскадних стилів CSS формується зовнішній вигляд веб-сайту і надається візуальна форма елементів для зручного користування. Із використанням фреймворку Vuetify даний етап виконується в рази швидше і легше. Також даний фреймворк надає сайту адаптивність під різні платформи, а також кросбраузерність, що надає можливість користуватися сайтом використовуючи різні версії браузерів.

Сайт буде розділено на дві частини, а саме загальну частину для всіх користувачів і адміністративну доступ до якої матимуть лише адміністратори сайту та менеджери.

Інтерфейс загальної частини сайту буде розподілятися на такі частини:

- головна сторінка;
- сторінки товарів по категоріям;
- сторінки для перегляду кожного товару окремо;
- сторінка з інформацією про доставку;
- сторінка з контактами;
- сторінка для оформлення замовлення;
- Кошик з товаром.

Інтерфейс адміністративної частини сайту буде розподілятися на такі частини:

- сторінка з основною інформацією;
- сторінка зі списком товарів;
- сторінка зі списком інгредієнтів;
- сторінка зі списком замовлень;
- сторінка зі списом відгуків;
- сторінка зі списом менеджерів сайту.

2.3 Проектування моделі бази даних

Під час створення бази даних необхідно слідкувати за впорядкованістю інформації, що в ній зберігатиметься, для того щоб в процесі подальшої роботи можна було отримувати чи модифікувати дані в зручному вигляді. Для цього бажано вхідні дані структурувати. Структуризація – це сукупність угод щодо шляхів представлення інформації. Зрозуміло, що структурувати інформацію можна по-різному. Залежно від структури розрізняють ієрархічну, мережеву, реляційну, об'єктно-орієнтовану і гібридну модель баз даних. Найпопулярнішою на сьогоднішній день є реляційна структура.

Вхідні дані – це набір даних потрібних для того щоб система коректно працювала. Аналіз вимог вхідних даних полягає в визначенні потреб та умов, які висуваються щодо нового, чи зміненого матеріалу, враховуючи можливо конфліктні вимоги різних замовників.

На основі дослідження предметної області був проведений аналіз, в результаті якого була побудована ERD діаграма потоків даних в додатку MySQL Workbench 6.3 CE (рис 2.13).

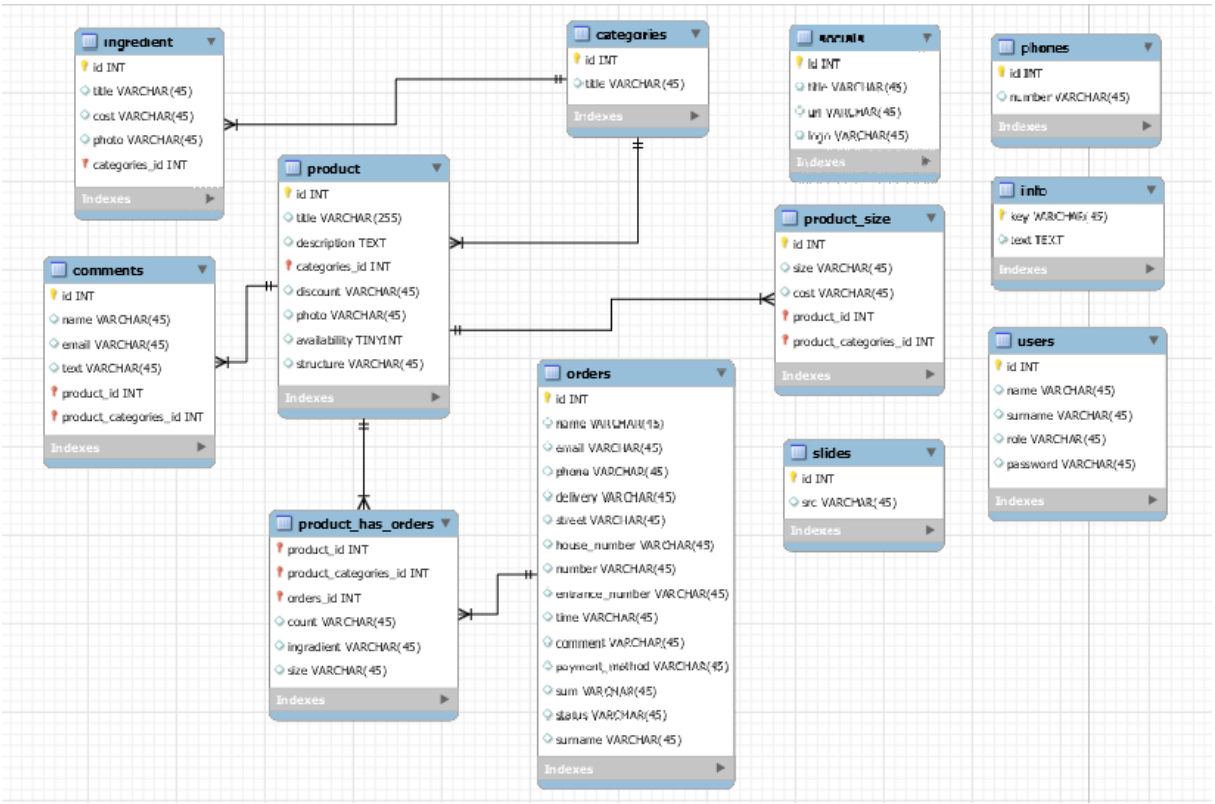

Рисунок 2.13 – ERD діаграма потоків даних

На основі ERD діаграми була згенерована база даних в додатку MySQL, що складається з одинадцяти таблиць.

В таблиці «categories» (рис. 2.14) містяться назви категорій продукції сайту. Таблиця містить 2 поля первинний ключ, назва категорії.

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
☐	1	id	bigint(20)		UNSIGNED	Нет	*Нет*		AUTO_INCREMENT
☐	2	title	varchar(255)	utf8mb4_unicode_ci		Нет	*Нет*		

Рисунок 2.14 – Таблиця «categories»

В таблиці «products» (рис. 2.15) містяться дані про товар. Таблиця містить 10 полів в яких зберігається первинний ключ, назва товару, опис, фото, статус наявності, структура, ідентифікатор категорії до якої відноситься товар, дати створення і оновлення інформації.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	id	bigint(20)		UNSIGNED	Нет	Нем		AUTO_INCREMENT
2	title	varchar(255)	utf8mb4_unicode_ci		Нет	Нем		
3	description	text	utf8mb4_unicode_ci		Да	NULL		
4	discount	int(11)			Да	NULL		
5	photo	varchar(255)	utf8mb4_unicode_ci		Да	NULL		
6	availability	tinyint(1)			Нет	1		
7	structure	text	utf8mb4_unicode_ci		Нет	Нем		
8	categories_id	bigint(20)		UNSIGNED	Нет	Нем		
9	created_at	timestamp			Да	NULL		
10	updated_at	timestamp			Да	NULL		

Рисунок 2.15 – Таблиця «products»

В таблиці «product_size» (рис. 2.16) містяться дані про різновиди кожного з товару наприклад розміри піци. Таблиця містить 4 поля в яких зберігається первинний ключ, вид товару, ціна, ідентифікатор відповідної продукції.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	id	bigint(20)		UNSIGNED	Нет	Нем		AUTO_INCREMENT
2	size	varchar(255)	utf8mb4_unicode_ci		Нет	Нем		
3	cost	varchar(255)	utf8mb4_unicode_ci		Нет	Нем		
4	products_id	bigint(20)		UNSIGNED	Нет	Нем		

Рисунок 2.16 – Таблиця «product_size»

В таблиці «ingredient» (рис. 2.17) містяться дані про додаткові інгредієнти для страв. Таблиця містить 5 полів в яких зберігається первинний ключ, назва, ціна, фото, ідентифікатор категорії.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	id	bigint(20)		UNSIGNED	Нет	Нем		AUTO_INCREMENT
2	title	varchar(255)	utf8mb4_unicode_ci		Нет	Нем		
3	cost	varchar(255)	utf8mb4_unicode_ci		Нет	Нем		
4	photo	varchar(255)	utf8mb4_unicode_ci		Нет	Нем		
5	categories_id	bigint(20)		UNSIGNED	Нет	Нем		

Рисунок 2.17 – Таблиця «ingredient»

В таблиці «orders» (рис. 2.18) міститься інформація про замовлення, а саме контактні дані замовника, адреса доставки і загальна сума замовлення. Таблиця містить 15 полів в яких зберігається первинний ключ, статус замовлення, ім’я, прізвище, email, телефон, вулиця, номер будинку, номер квартири, номер під’їзду, бажаний час доставки, коментар, спосіб оплати, загальна сума замовлення.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id	bigint(20)		UNSIGNED	Нет	Нет		AUTO_INCREMENT	
2	status	varchar(255)	utf8mb4_unicode_ci		Нет	Новий			
3	name	varchar(255)	utf8mb4_unicode_ci		Нет	Нет			
4	surname	varchar(255)	utf8mb4_unicode_ci		Нет	Нет			
5	email	varchar(255)	utf8mb4_unicode_ci		Нет	Нет			
6	phone	varchar(255)	utf8mb4_unicode_ci		Нет	Нет			
7	street	varchar(255)	utf8mb4_unicode_ci		Да	NULL			
8	house_number	varchar(255)	utf8mb4_unicode_ci		Да	NULL			
9	number	varchar(255)	utf8mb4_unicode_ci		Да	NULL			
10	entrance_number	varchar(255)	utf8mb4_unicode_ci		Да	NULL			
11	time	varchar(255)	utf8mb4_unicode_ci		Да	NULL			
12	comment	text	utf8mb4_unicode_ci		Да	NULL			
13	payment_method	varchar(255)	utf8mb4_unicode_ci		Нет	Нет			
14	sum	varchar(255)	utf8mb4_unicode_ci		Нет	Нет			

Рисунок 2.18 – Таблиця «orders»

В таблиці «product_has_orders» (рис. 2.19) міститься інформація про кожен замовлений товар. Таблиця містить 6 полів в яких зберігається первинний ключ, ідентифікатор товару, ідентифікатор замовлення, кількість замовленого товару, список інгредієнтів і вид товару.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	id	bigint(20)		UNSIGNED	Нет	Нет		AUTO_INCREMENT
2	products_id	bigint(20)		UNSIGNED	Нет	Нет		
3	orders_id	bigint(20)		UNSIGNED	Нет	Нет		
4	count	int(11)			Нет	Нет		
5	ingradient	varchar(255)	utf8mb4_unicode_ci		Нет	Нет		
6	size	varchar(255)	utf8mb4_unicode_ci		Нет	Нет		

Рисунок 2.19 – Таблиця «product_has_orders»

В таблиці «comments» (рис. 2.20) міститься інформація про коментарі залишені користувачами про товар. Таблиця містить 6 полів в яких

зберігається первинний ключ, ім’я користувача, email, текст коментаря, ідентифікатор товару, дату створення.

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
☐	1	id	bigint(20)		UNSIGNED	Нет	Нем		AUTO_INCREMENT
☐	2	name	varchar(255)	utf8mb4_unicode_ci		Нет	Нем		
☐	3	email	varchar(255)	utf8mb4_unicode_ci		Нет	Нем		
☐	4	text	text	utf8mb4_unicode_ci		Нет	Нем		
☐	5	products_id	bigint(20)		UNSIGNED	Нет	Нем		
☐	6	created_at	timestamp			Да	NULL		

Рисунок 2.20 – Таблиця «comments»

В таблиці «users» (рис. 2.21) міститься інформація про користувачів зареєстрованих на сайті. Таблиця містить 6 полів в яких зберігається первинний ключ, Прізвище та ім’я користувача, email, роль на сайті, пароль, токен для збереження сесії авторизації.

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
☐	1	id	bigint(20)		UNSIGNED	Нет	Нем		AUTO_INCREMENT
☐	2	name	varchar(255)	utf8mb4_unicode_ci		Нет	Нем		
☐	3	email	varchar(255)	utf8mb4_unicode_ci		Нет	Нем		
☐	4	role	varchar(255)	utf8mb4_unicode_ci		Нет	Нем		
☐	5	password	varchar(255)	utf8mb4_unicode_ci		Нет	Нем		
☐	6	remember_token	varchar(100)	utf8mb4_unicode_ci		Да	NULL		

Рисунок 2.21 – Таблиця «users»

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Архітектура веб додатку

Перелік головних вимог до сайту:

1. Сайт повинен складатись з двох частин, а саме користувацької та адміністративної.

2. Користувацька частина призначена для не зареєстрованих користувачів які маю можливість користуватись сайтом не виконуючи авторизацію.

Користувацька частина повинна складатися із наступних сторінок:

– Головна.

– Сторінки категорій товару.

– Інформація про доставку.

– Контакти.

– Сторінка з детальною інформацією про товар.

– Кошик.

– Сторінка замовлення товару.

Доступ до адміністративної частини матимуть лише адміністратори і менеджери сайту. Для адміністраторів повинен бути розроблений функціонал який надає наступні можливості:

- додання, редагування, видалення категорій продукції сайту.
- Додання, редагування, перегляд, видалення продукції сайту.
- Обробка, перегляд і видалення замовлень.
- Перегляд і видалення коментарів.
- Можливість редагувати загальну інформацію про сайт.

Функціонал для групи менеджерів:

- Додання, редагування, видалення категорій продукції сайту.
- Додання, редагування, перегляд, видалення продукції сайту.
- Обробка, перегляд і видалення замовлень.

- Перегляд і видалення коментарів.

Адміністративна частина повинна складатися із наступних сторінок:

- Загальні налаштування.
- Товар.
- Інгредієнти.
- Замовлення.
- Відгуки користувачів.
- Менеджери сайту.

3.2 Програмна реалізація

Розробка буде відбуватись у безкоштовному текстовому редакторі VS Code. Visual Studio Code – це крос-платформенний редактор скриптів, створений корпорацією Майкрософт. Поряд з розширенням PowerShell він надає широкі інтерактивні можливості редагування скриптів, спрощуючи написання надійних скриптів PowerShell.

Також для збереження вихідного коду, а також відслідковування версій сайту було використано систему контролю версій Git. Системи контролю версій дозволяють розробникам зберігати всі зміни, внесені в код. Тому в разі, помилки, вони можуть просто відкотити код до робочого стану замість того, щоб витрачати години на пошуки маленької помилки або помилок, які ламають весь код [16].

Системи контролю версій також дають можливість декільком розробникам працювати над одним проектом і зберігати внесені зміни, щоб переконатися, що всі можуть стежити за тим, над чим вони працюють.

Проект буде розроблятись на віртуальному веб сервері OpenServer. Open Server Panel – це портативна серверна платформа і програмне середовище, створена спеціально для веб-розробників з урахуванням їх рекомендацій і побажань. Також буде використаний менеджер модулів для PHP – Composer.

Розробка сайту починається з створення нового проекту з використанням фреймворку laravel. Для цього в терміналі Composer необхідно написати наступну команду:

```
composer create-project --prefer-dist laravel/laravel pizza
```

Де «pizza» – ім'я проекту і може бути змінено на будь-яке. Після чого буде створено базову архітектуру проекту (рис. 3.1).

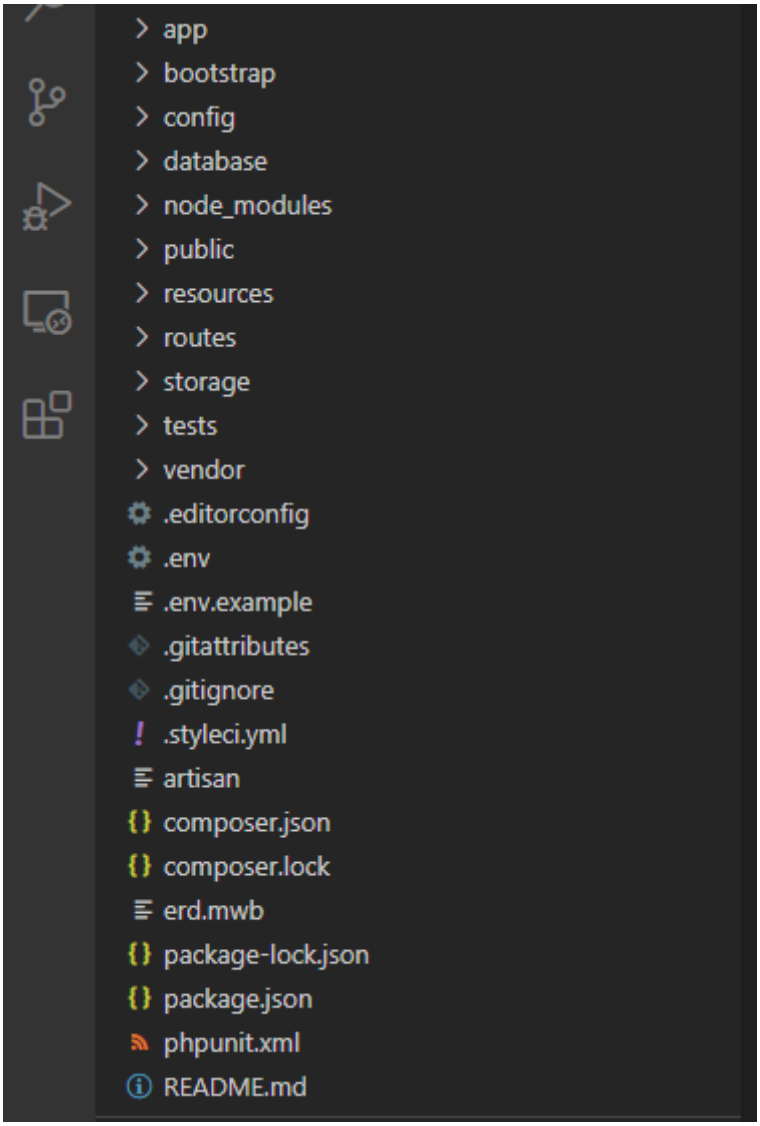

Рисунок 3.1 – Архітектура проекту

Потім починається розробка клієнтської частини сайту, а саме створення всіх необхідних компонентів, налаштування маршрутів доступу і верстка шаблонів та розробка модулів серверної частини сайту, створення всіх необхідних контролерів, моделей та налаштування API роутеру.

3.2.1 Розробка клієнтської частини

Розробка клієнтської частини буде відбуватися за допомогою JavaScript фреймворка Vue.js. Підключення коренового компонента відбувається в файлі site.js.

Спочатку підключемо всі необхідні бібліотеки і компоненти.

```
require('./bootstrap');
import router from "./routes/site";
import Vuetify from "vuetify";

import HeaderComponent from './components/includes/Header';
import MenuComponent from './components/includes/Menu';
import FooterComponent from './components/includes/Footer';
import store from "./store/basket";
window.Vue = require('vue');
window.Vue.use(Vuetify);
```

Потім ініціалізуємо клас Vue, це треба для того, щоб всі компоненти відображались у кореневому блоці.

```
const app = new Vue({
    el: '#site',
    components: {
        HeaderComponent,
        MenuComponent,
        FooterComponent
    },
    vuetify: new Vuetify({
        icons: {
```

```
            iconfont: 'mdi',
        }
    }),
    store,
    router
});
```

Кореневий Vue компонент підключає такі компоненти, які відображають весь контент додатку, який знаходиться в файлі resources/views/ site.blade.php.

```
<!DOCTYPE html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <meta name="csrf-token" content="{{ csrf_token() }}">
        <title>Піца</title>
        <link href="{{ mix('css/site.css') }}" rel="stylesheet">
        <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
    </head>
    <body>
        <v-app id="site" id="inspire">
            <div id="wrapper">
                <header-component></header-component>
                <v-content class="mx-4">
                    <menu-component></menu-component>
                    <router-view></router-view>
                </v-content>
            </div>
            <footer-component></footer-component>
        </v-app>
        <script src="{{ asset('js/site.js') }}"></script>
</html>
```

Всі сторінки сайту відображаються завдяки спеціальним url адресам. Які прописані в файлі resources/js/routes/site.js. Частина коду з адресами сайту наведено нижче в лістингу.

```
let router = new Router({
    mode: 'history',
    routes: [
      {
        path: '/',
        name: 'home',
        component: Home
      },
      {
        path: '/category/:category/',
        name: 'categoryItems',
        component: categoryItems
      }
    ]
})
```

В залежності від адреси відображається необхідний компонент.

3.2.2 Розробка серверної частини

Laravel надає безліч можливостей для розробки серверної частини сайту, в тому числі розробку API для клієнтської частини.

Всі запити що надходять на сервер обробляються головним маршрутизатором, що знаходиться в файлі routes/api.php. Маршрутизація визначає, як додаток відповідає на клієнтський запит до конкретної адреси.

```
Route::get('products', 'ProductsController@getProducts');
Route::get('product/{id}', 'ProductsController@getProductId');
Route::get('products/{category}', 'ProductsController@getProductsCategory');
Route::post('product', 'ProductsController@postProduct');
Route::post('product/{id}', 'ProductsController@editProduct');
Route::delete('product/{id}', 'ProductsController@deleteProduct');
```

Потім в залежності від типу запиту підключається необхідний контролер з необхідним методом який опрацює дані отримані з бази даних. Контролер – з'єднує моделі, види і інші компоненти необхідні для роботи системи. А також відповідає за обробку запитів користувача, а саме отримання і обробку даних та відправку готового результату клієнтській частині системи. Приклад такого контролеру наведено нижче:

```
class CategoriesController extends Controller {
    function getCategories() {
        $data = Categories::get();
        return response()->json($data);
    }
    function postCategory(Request $request) {
        $category = new Categories();
        $data = $request->all();
        $response = $category->create($data);
        return response()->json($response);
    }
    function editCategory(Request $request, $id) {
        $category = Categories::find($id);
        $data = $request->all();
        $category->update($data);
        return response('ok', 200);
    }
    function deleteCategory($id) {
        Categories::find($id)->delete();
    }
}
```

Кожна таблиця має відповідний клас-модель, який використовується для роботи з цією таблицею. Моделі дозволяють запитувати дані з таблиць, а також вставляти в них нові записи. Приклад моделі для таблиці користувачів.

```
<?php
namespace App;
```

```
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Passport\HasApiTokens;
class User extends Authenticatable {
    use HasApiTokens,Notifiable;
    protected $fillable = [
        'name', 'email', 'password', 'role',
    ];
    protected $hidden = [
        'password', 'remember_token',
    ];
}
```

3.3 Використання програмного додатку

Даний дипломний проект, а точніше веб-сайт, що розроблений для замовлення піци, складається з двох частин. А саме клієнтська частина для перегляду інформації на сайті, вона доступна всім користувачам та адміністративної панелі з можливістю додання і редагування інформації яка доступна лише для адміністраторів сайту.

3.3.1 Клієнтська частина сайту

При відкритті сайту в браузері відображається головна сторінка (рис. 3.2), на шапці якої розташована деяка контактна інформація, форма пошуку, а також фірмова назву сайту.

Рисунок 3.2 – Головна сторінка

Нижче шапки розташована навігаційна панель, яка містить посилання на основні сторінки сайту і категорії товару.

Картки с товаром містять в собі зображення, назву, детальний опис, а також декілька видів даного товару з вказаною ціною (рис 3.3). Кожен товар можна додати до кошику натиснувши на відповідну кнопку.

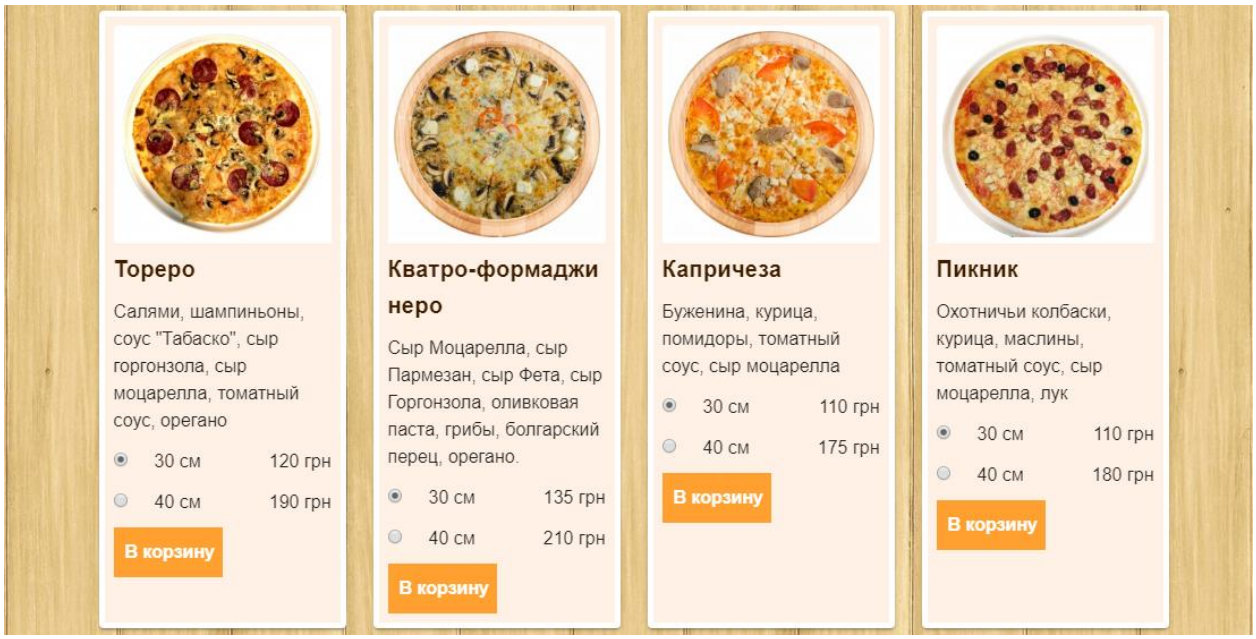

Рисунок 3.3 – Картки з товаром

Кошик являє собою список з обраною продукцією, кнопки для додання або зменшення кількості кожного з товару. До кожного товару можна додати додаткові інгредієнти натиснувши кнопку «Додати інгредієнти+» (рис. 3.4).

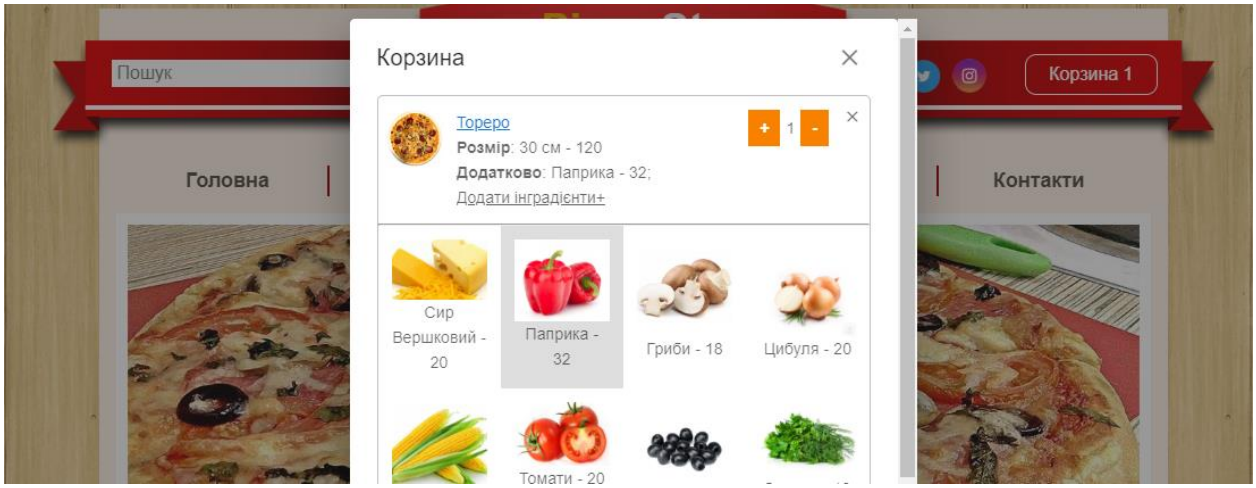

Рисунок 3.4 – Кошик товару

Внизу модального вікна знаходиться загальна сума кошику з товаром і кнопка для оформлення замовлення (рис. 3.5).

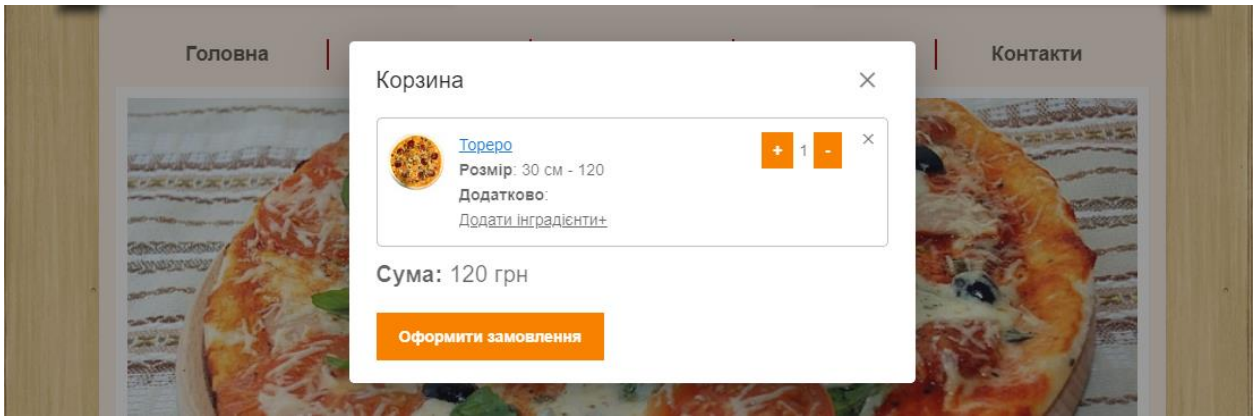

Рисунок 3.5 – Загальна сума товару в кошику

Також на сайті реалізований пошук товару за назвою продукції. Після введення символів в форму пошуку відразу відображається список з результатом пошуку.

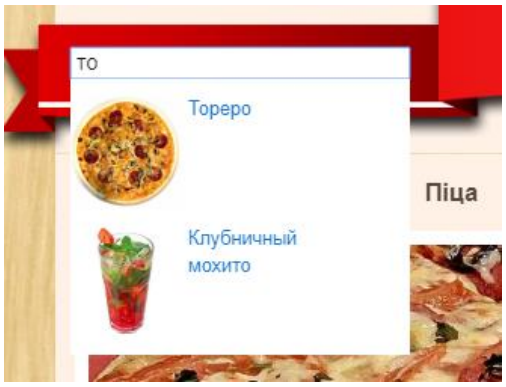

Рисунок 3.6 – Результат пошуку

Кожен товар можна відкрити на новій сторінці для перегляду детальної інформації, змінити розмір і додати до кошику. Також на сторінці можна, додати додаткові інгредієнти натиснувши відповідну кнопку.

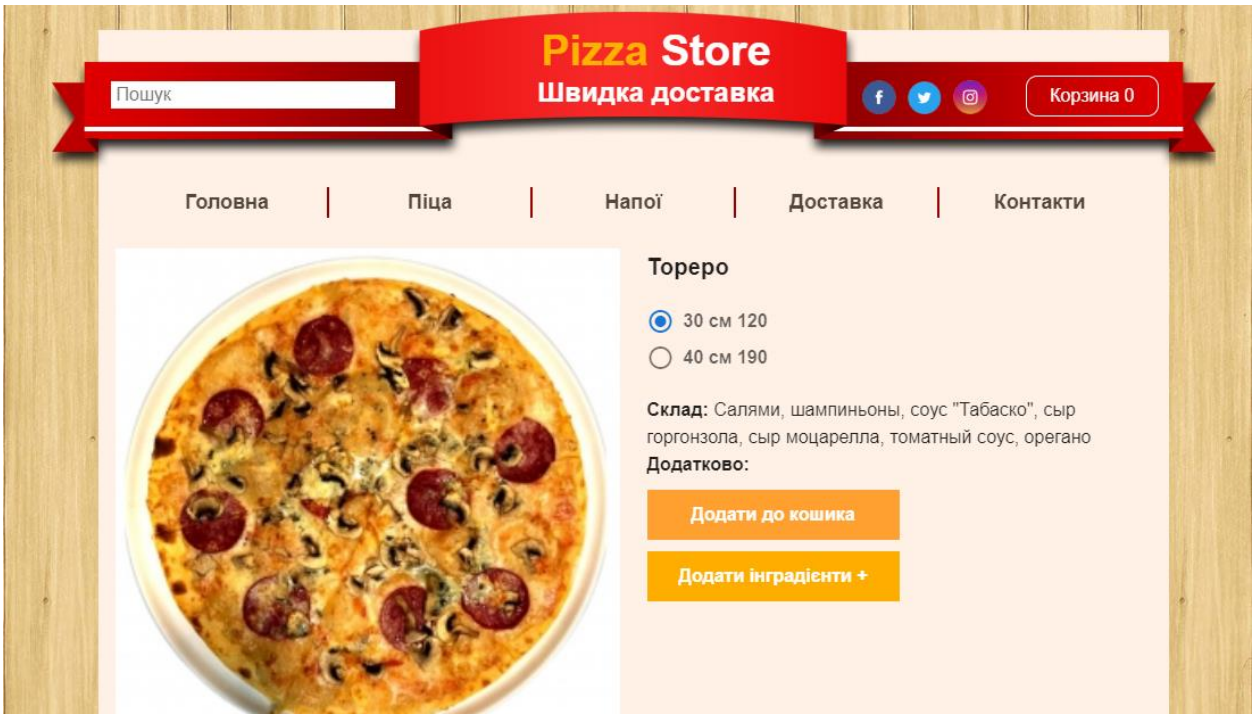

Рисунок 3.7 – Сторінка з детальної інформацією про товар

Сторінка для оформлення замовлення складається з списку товару, що був доданий до кошику, форму для заповнення контактних даних про замовника, а точніше прізвище та ім’я, адресу доставки, спосіб оплати і

можливість додати коментар за бажанням (рис. 3.8). Також є загальна сума замовлення і кількість обраних товарів.

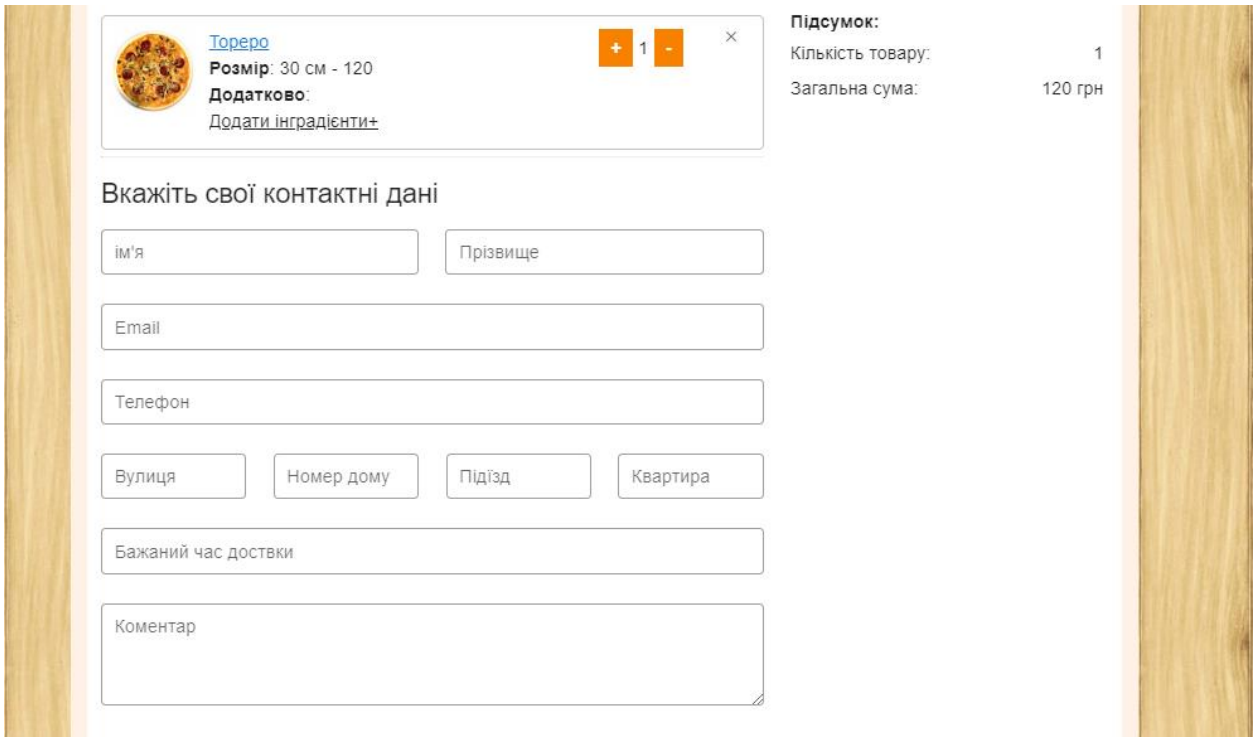

Рисунок 3.8 – Сторінка для оформлення замовлення

Після заповнення всіх необхідних полів у формі замовлення доставки відобразиться повідомлення про успішне оформлення заявки (рис. 3.9).

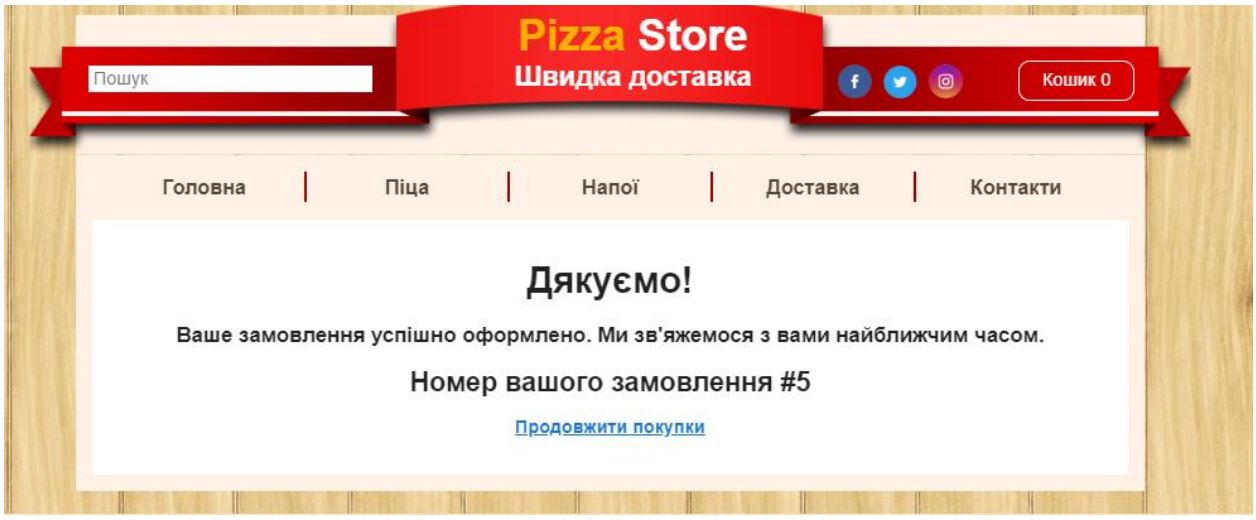

Рисунок 3.9 – Повідомлення про успішне оформлення заявки

3.3.2 Адміністративна частина сайту

Для управління сайтом, додання товару і обробки замовлень сайт має адміністративну панель. Доступ до неї мають лише такі групи користувачів: адміністратори і менеджери. Щоб до неї потрапити необхідно авторизуватись. Для авторизації потрібно ввести пошту і пароль користувача у відповідні поля.

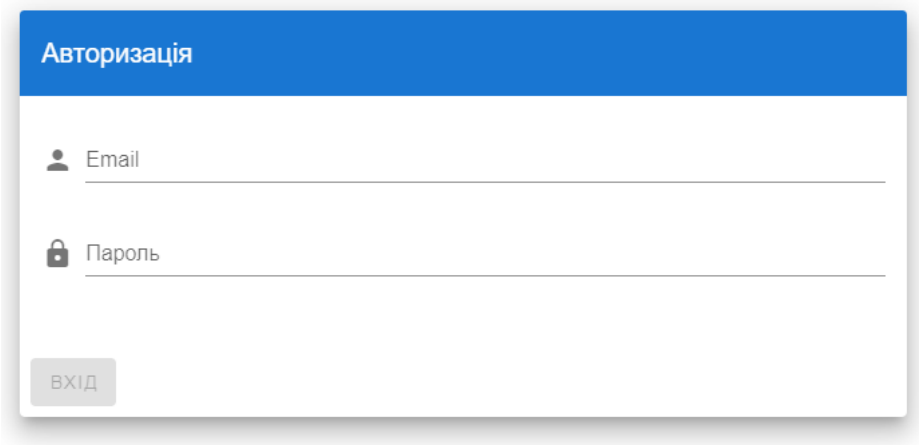

Рисунок 3.10 – Форма авторизації

Після авторизації в якості адміністратора користувач потравляє на сторінку з загальними налаштуваннями. На якій знаходиться загальна інформація про сайт, а саме опис сайту, інформація про доставку, адреса, email, графік роботи. Також є можливість додавати категорії товарів, соціальні мережі, номера телефонів і зображення для слайд шоу на головній сторінці сайту (рис. 3.11, 3.12).

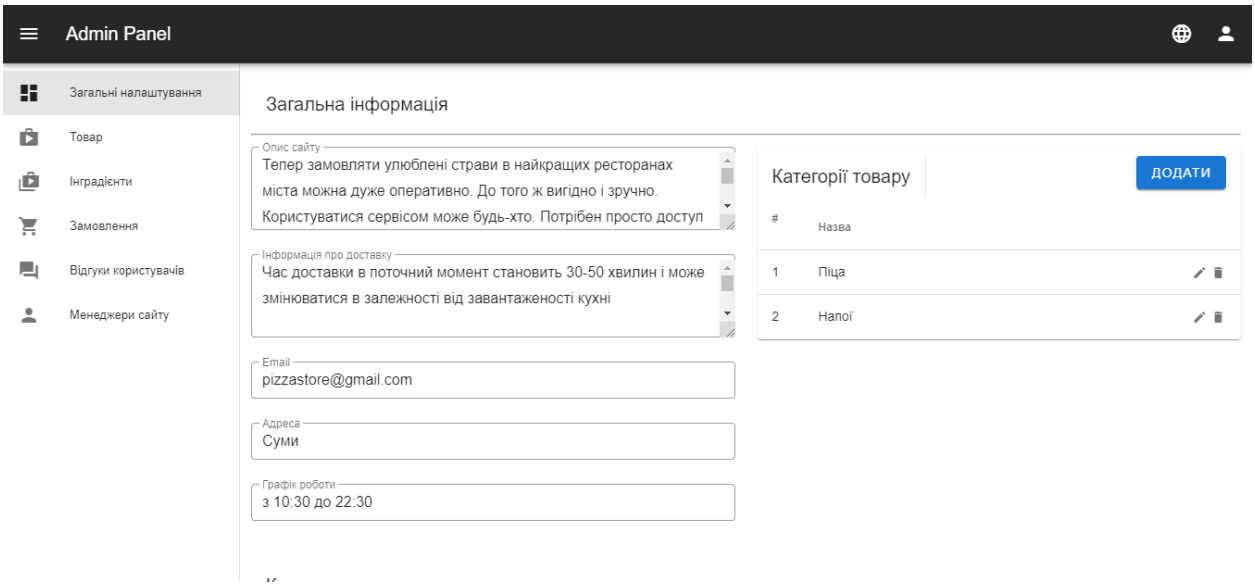

Рисунок 3.11 – Налаштування загальної інформації та категорій

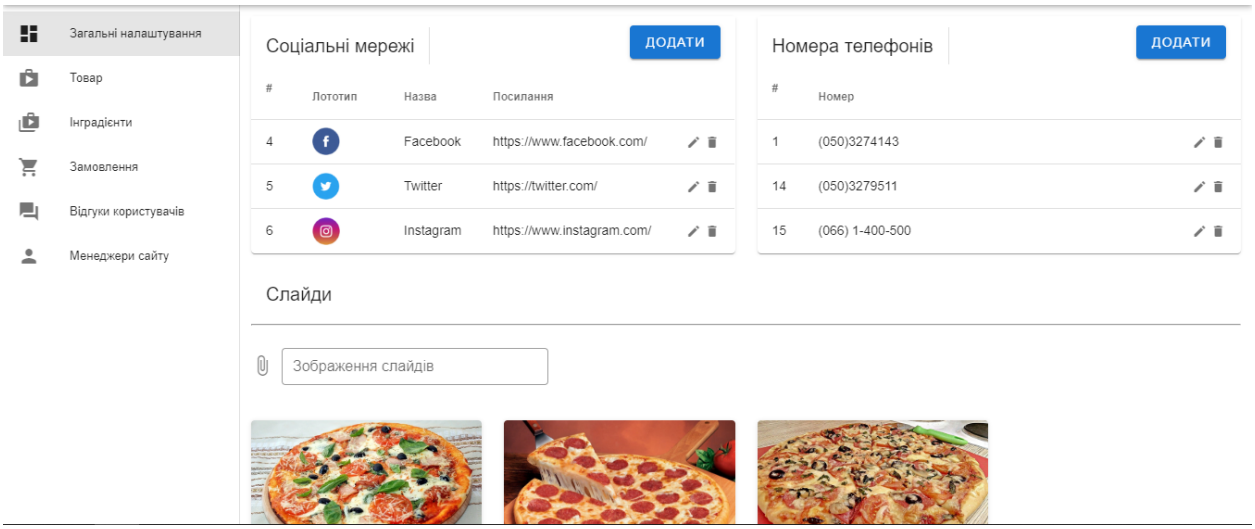

Рисунок 3.12 – Налаштування соціальних мереж, номерів телефонів, зображень слайд-шоу

Після авторизації в якості менеджера користувач потравляє на сторінку «Товар» на якій міститься список товару, що розміщено на сайті (рис. 3.13). За необхідністю продукцію можна видалити, або редагувати. Сторінка «Інгредієнти» має аналогічну структуру.

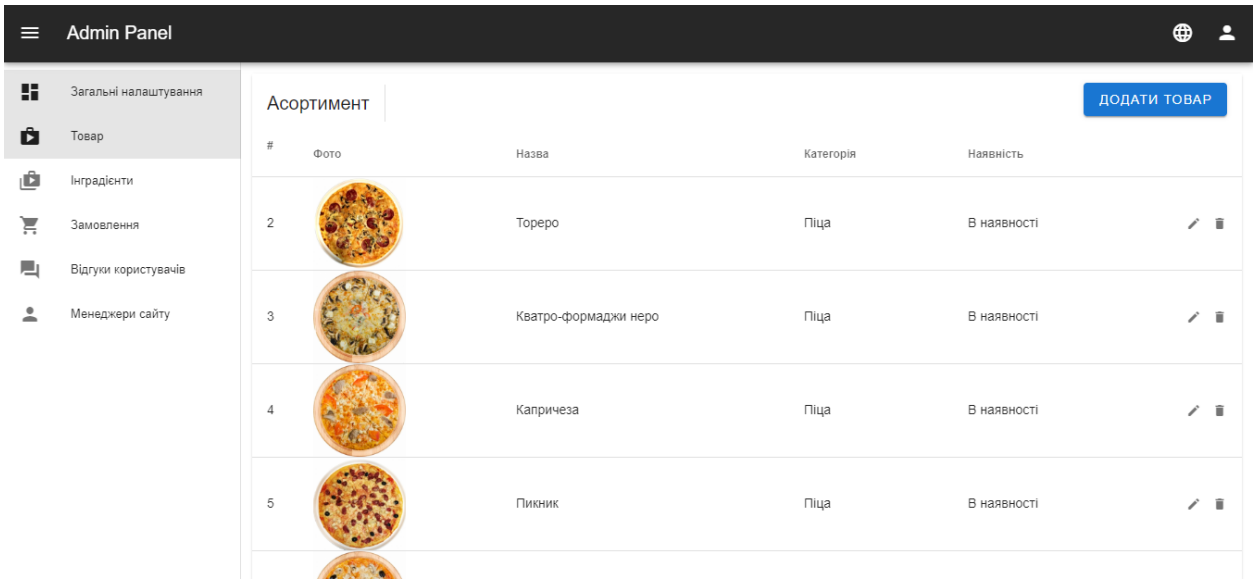

Рисунок 3.13 – Сторінка з товаром

Натиснувши кнопку «Додати товар» відкривається сторінка з формою яка має всі необхідні поля для заповнення інформації про продукцію (рис. 3.14).

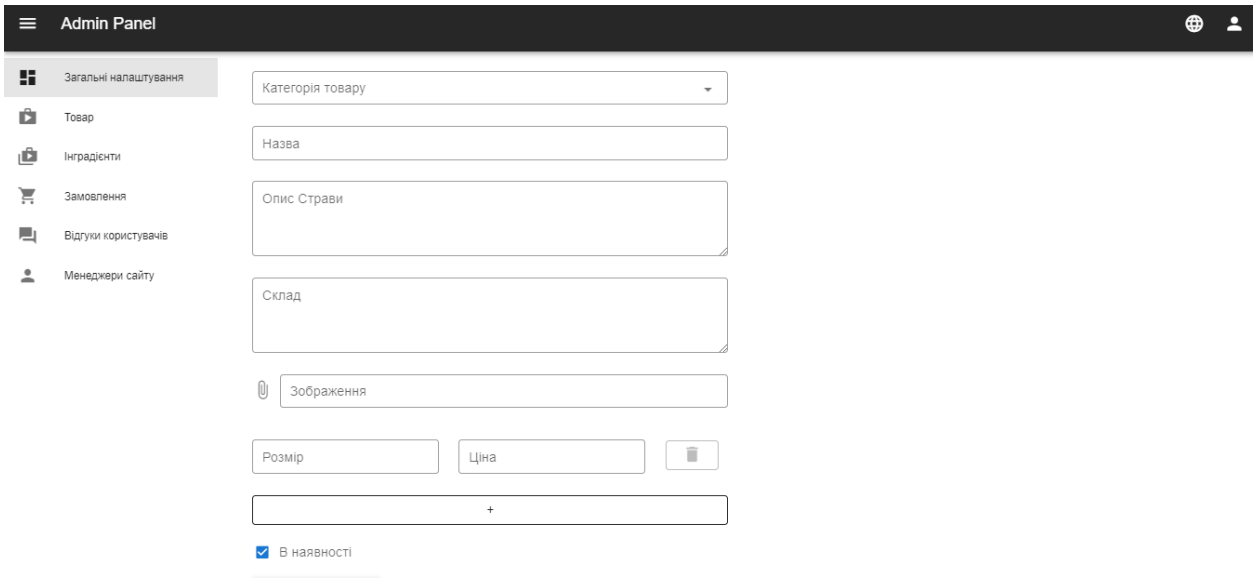

Рисунок 3.14 – Форма для додання товару

На сторінці «Замовлення» заходиться список замовлень користувачів сайту на доставку товару (рис. 3.15). Кожне замовлення за необхідністю можна видалити, а також змінити статус виконання.

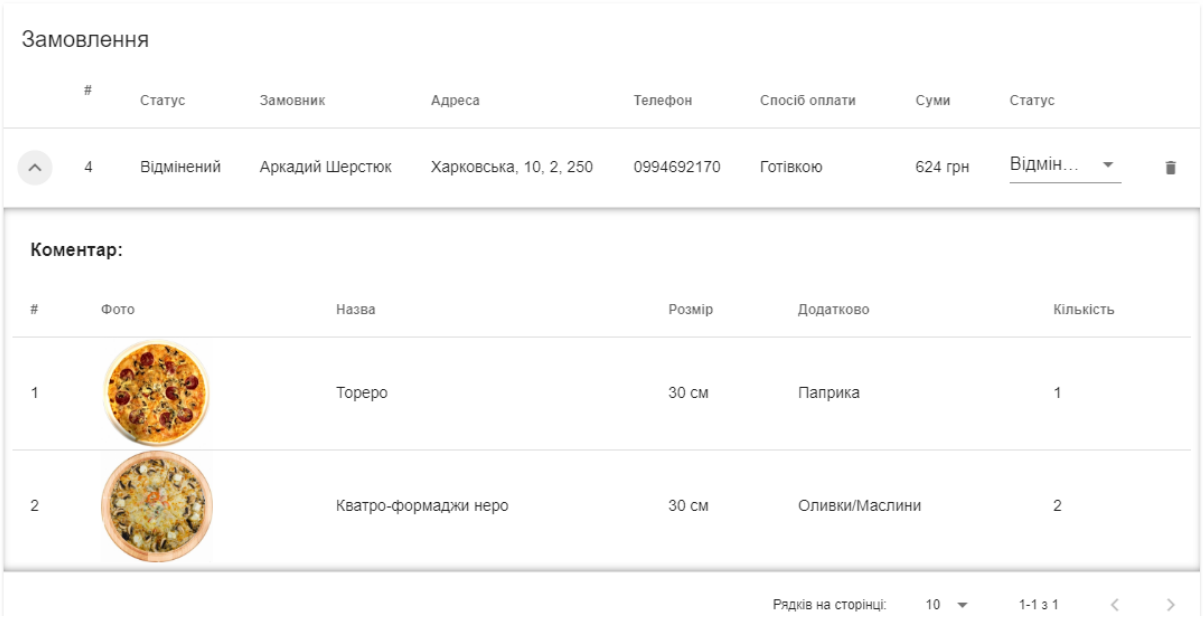

Рисунок 3.15 – Список замовлень

На сайті також є можливість додавати коментарі до кожної продукції. Для перегляду всіх коментарів адміністративній панелі потрібно відкрити сторінку «Відгуки користувачів» на якій знаходиться таблиця зі всіма відгуками включаючи інформацію про автора і назву продукції до якої був залишений коментар (рис. 3.16).

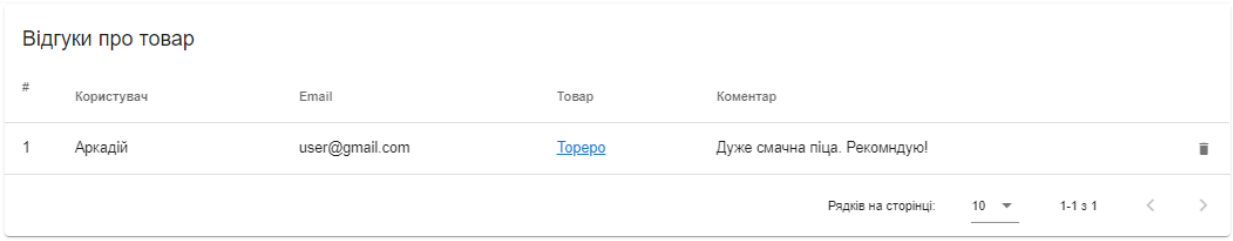

Рисунок 3.16 – Список коментарів

До сторінки «Список менеджерів» мають доступ лише адміністратори сайту. На ній знаходиться список менеджерів сайту які відповідають за додання продукції, обробку замовлень і відгуків користувачів. За необхідністю їх можна видаляти (рис. 3.17).

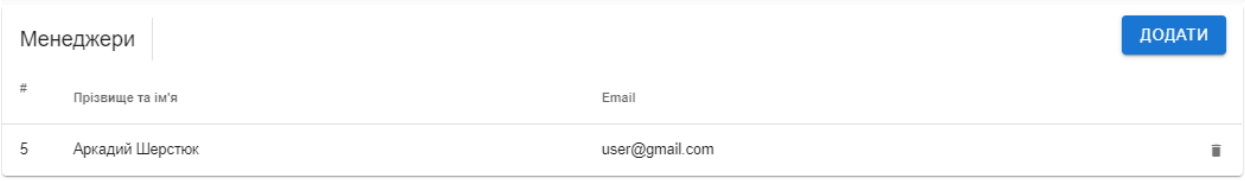

Рисунок 3.17 – Список менеджерів сайту

Для додання нового менеджера необхідно натиснути кнопку «Додати» після чого відкриється модальне вікно з формою яка має поля: прізвища та ім’я, email, паролю.

Рисунок 3.18 – Форма реєстрації менеджера

ВИСНОВОК

Мета дипломного проекту проектування і розробка інформаційного веб-ресурс для замовлення доставки піци. Мета досягнута, поставлені завдання виконані:

1. Досліджена предметна область;
2. Проаналізовано WEB - сайти аналогічної тематики;
3. Обрана стратегію розробки;
4. Спроектована модель сайту;
5. Розроблено web - сайт з використанням фреймворку Laravel.

В процесі роботи досліджена предметна область. Проведено аналіз web-сайтів аналогічної тематики для виявлення недоліків системи, які враховувалися при створенні сайту. Оптимізовано web-інтерфейс і навігація сайту, для того щоб користувачеві було зручніше орієнтуватися в віртуальному просторі.

Проект реалізований за допомогою фреймворку Laravel, основою якого являється мова програмування PHP.

Всі цілі і завдання були виконані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Як працює CRM-система [Електронний ресурс] – Режим доступу до ресурсу: https://salesap.ru/crm_sistemy_chto_eto/. (дата звернення: 02.05.2020).

2. Що таке сайт? Інтернет сайт? Види сайтів [Електронний ресурс] – Режим доступу до ресурсу: http://moolkin.ru/chto-takoe-sayt-internet-sayt-vidy-saytov/. (дата звернення: 02.05.2020).

3. The 2020 Roadmap To Fullstack Web Development [Електронний ресурс] – Режим доступу до ресурсу: https://codingthesmartway.com/the-2020-roadmap-to-fullstack-web-development/. (дата звернення: 22.04.2020).

4. Крокфорд Дуглас. JavaScript. Сильні сторони, 2016.— 176 с.

5. Що таке PHP? [Електронний ресурс] – Режим доступу до ресурсу: https://www.php.net/manual/en/intro-whatis.php. (дата звернення: 22.04.2020).

6. Анді Гутманс, Стіг Баккен, Дерік Ретханс. Програмування живлення PHP5, 2015. — 704 с.

7. Марк Сафронов: Разработка веб-приложений в Yii 2, 2015. - 392 с.

8. Документація Laravel [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/. (дата звернення: 22.04.2020).

9. Створення базового додатку MVC Laravel 5 Види сайтів [Електронний ресурс] – Режим доступу до ресурсу: https://selftaughtcoders.com/from-idea-to-launch/lesson-17/laravel-5-mvc-application-in-10-minutes/. (дата звернення: 21.04.2020).

10. Laravel – Функції безпеки [Електронний ресурс] – Режим доступу до ресурсу: https://webformyself.com/laravel-funkcii-bezopasnosti/. (дата звернення: 24.04.2020).

11. The Pros and Cons of 8 Popular Databases [Електронний ресурс] – Режим доступу до ресурсу: https://www.keycdn.com/blog/popular-databases. (дата звернення: 23.04.2020).

12. MySQL Stored Procedure Advantages [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/What-are-the-advantages-and-disadvantages-of-using-MySQL-stored-procedures. (дата звернення: 23.04.2020).

13. Діаграми UML [Електронний ресурс] – Режим доступу до ресурсу: https://planerka.info/item/diagrammy-kommunikacij-uml/. (дата звернення: 23.04.2020).

14. Теорія та практика UML [Електронний ресурс] – Режим доступу до ресурсу: http://www.it-gost.ru/articles/view_articles/94. (дата звернення: 23.04.2020).

15. Best Personal and Niche Blogs (30+ Real Examples) [Електронний ресурс] – Режим доступу до ресурсу: https://firstsiteguide.com/examples-of-blogs/. (дата звернення: 22.04.2020).

16. Документація Git [Електронний ресурс] – Режим доступу до ресурсу: https://git-scm.com/. (дата звернення: 24.04.2020).

17. Хавербеке Марейн. Виразний JavaScript. 2 видання, 2015. – 745 с.

18. Документація Vue.js [Електронний ресурс] – Режим доступу до ресурсу: https://ru.vuejs.org/v2/guide/. (дата звернення: 23.04.2020).

19. Хеслоп П. HTML самого початку. С.-Пб: Санкт-Петербург, 2014. – 450с.

20. Що таке CSS [Електронний ресурс] – Режим доступу до ресурсу: http://phpist.com.ua/css/5-whatcss/. (дата звернення: 21.04.2020).

ДОДАТОК А

1. Фрагмент коду з файлу Home.vue: відповідає за відображення головної сторінки сайту і підключення додаткових компонентів.

```
<template>
  <div>
    <div id="content">
      <Slider></Slider>
    </div>

    <v-container v-for="category in data" :key="category.id">
      <h1 class="my-2" style="color:#3d1e00">{{ category.title }}</h1>
      <v-row>
        <v-col md="3" v-for="product in category.products" :key="product.id">
          <Item :item="product"></Item>
        </v-col>
      </v-row>
    </v-container>
  </div>
</template>

<script>

import Slider from './includes/Slider';
import Item from './includes/Item';

export default {
  name: 'Home',

  components: {
    Slider,
    Item
  },
```

```
    data: () => ({
        data: []
    }),

    created() {
        this.getProducts();
    },

    methods: {
        getProducts() {
          axios.get('/api/home-product')
            .then((response) => {
                this.data = response.data.map(category => {
                    category.products = category.products.map(item => {
                        item.selectSize = item.size[0]
                        return item
                    })
                    return category
                })
            })
        }
    }
}
</script>
<style lang="css">

</style>
```

2. Код з файлу item.vue: відповідає за відображення блоку з товаром.

```
<template>
    <v-card class="product-item pa-2 ma-0">
        <img v-if="$props.item.photo" :src="$props.item.photo">
```

```
        <img v-else src="/img/default.png">
        <div class="title">
            <router-link :to="'/product/'+$props.item.id">{{ $props.item.title }}</router-link>
        </div>
        <div class="description">
            {{ $props.item.structure }}
        </div>

        <v-row v-for="itemSize in $props.item.size" :key="itemSize.id">
            <v-col md="2" class="pb-0">
                <input
                    type="radio"
                    :name="'size_product_'+itemSize.products_id"
                    :id="'size_'+itemSize.id"
                    :value="itemSize"
                    v-model="item.selectSize"
                >
            </v-col>
            <v-col md="5" class="pb-0">
                <label :for="'size_'+itemSize.id">{{ itemSize.size }}</label>
            </v-col>
            <v-col md="5" class="text-end pb-0">
                {{ itemSize.cost }} грн
            </v-col>
        </v-row>

        <div class="basket">
            <input v-if="$props.item.availability" type="button"
@click="addToBusket($props.item)" value="В корзину">
            <b v-else>Немає в наявності</b>
        </div>
    </v-card>
</template>

<script>
```

```
import { mapMutations } from 'vuex';

export default {
    props: ['item'],

    methods: {
        ...mapMutations(["setItemBasket"]),

        addToBusket(item) {
            this.setItemBasket(item);
        }
    }
}
</script>

<style lang="css">
    .product-item {
        background: #fff1e5 !important;
        border: 5px solid #fff;
        box-sizing: border-box;
        height: 100%;
    }
    .product-item .title {
        margin-bottom: 10px;
    }
    .product-item .title a {
        text-decoration: none;
        font-weight: bold;
        color:#3d1e00;
    }
    .product-item img {
        border: 5px solid #fff;
        box-sizing: border-box;
    }
```

```
    .basket {
        margin-top: 10px;
    }
    .basket input[type="button"] {
        width: 100%;
        padding: 10px;
        background: #ffa031;
    }
</style>
```

ДОДАТОК Б

1. Код з файлу OrdersController.php: відповідає за обробку і збереження замовлень в базу даних.

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Models\Orders;
use App\Models\ProductHasOrders;

class OrdersController extends Controller
{
   function getOrders() {
      $data = Orders::get();
      for ($i=0; $i < count($data); $i++) {
         $data[$i]['products'] = ProductHasOrders::with('product')->where('orders_id', $data[$i]['id'])->get();
      }
      return response()->json($data);
   }

   function postOrder(Request $request) {
      $order = new Orders();
      $data = $request->all();
      $responsOrder = $order->create($data);
      foreach ($request->products as $keyProduct => $product) {
         $productHasOrders = new ProductHasOrders;
         $productHasOrders->products_id = $product['id'];
         $productHasOrders->orders_id = $responsOrder->id;
         $productHasOrders->count = $product['count'];
         $productHasOrders->size = $product['selectSize']['size'];

         $ingredients = [];
```

```
            foreach ($product['ingredients'] as $key => $ingredient) {
                if($ingredient['select']) {
                    array_push($ingredients, $ingredient['title']);
                }
            }

            $productHasOrders->ingradient = implode("; ", $ingredients);
            $productHasOrders->save();
        }
        return response()->json(['orderId' => $responsOrder->id]);
    }

    function changeStatus(Request $request, $id) {
        $order = Orders::find($id);
        $order->status = $request->status;
        $order->save();
        return response('ok', 200);
    }

    function deleteOrder($id) {
        Orders::find($id)->delete();
        return response('ok', 200);
    }

}
```

2. Код з файлу ProductsController.php: відповідає за обробку і збереження продукції в базу даних.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
```

```
use App\Models\Products;
use App\Models\ProductSize;
use App\Models\Categories;

class ProductsController extends Controller
{
    function getProducts() {
        $data = Products::with('size', 'category', 'ingredients')->get();
        return response()->json($data);
    }
    function getProductId($id) {
        $product = Products::with('size', 'ingredients', 'comments')->where('id', $id)->first();
        return response()->json($product);
    }
    function getProductsCategory($category) {
        $product = Products::with('size', 'ingredients')->where('categories_id', $category)->get();
        return response()->json($product);
    }
    function getHomeProducts() {
        $categories = Categories::get();
        for ($i=0; $i < count($categories); $i++) {
            $categories[$i]['products'] = Products::with('size', 'ingredients')->where('categories_id',
$categories[$i]['id'])->limit(8)->get();
        }
        return response()->json($categories);
    }
    function postProduct(Request $request) {
        $products = new Products();
        $data = $request->all();
        if($request['photo']) {
            $name = "/img/products/".uniqid().'.'.$request['photo']->getClientOriginalExtension();
            $request['photo']->move(public_path()."/img/products/", $name);
            $data['photo'] = $name;
        }
        $productsResponse = $products->create($data);
```

```
    $size = json_decode($request['size']);
    for($i = 0; $i < count($size); $i++) {
        $productSize = new ProductSize;
        $productSize->size = $size[$i]->size;
        $productSize->cost = $size[$i]->cost;
        $productSize->products_id = $productsResponse['id'];
        $productSize->save();
    }
    return response('ok', 200);
}
function editProduct(Request $request, $id) {
    $product = Products::find($id);
    $data = $request->all();
    if(gettype($request['photo']) == "object") {
        $name = "/img/products/".uniqid().'.'.$request['photo']->getClientOriginalExtension();
        $request['photo']->move(public_path()."/img/products/", $name);
        $data['photo'] = $name;
    } else {
        $data['photo'] = $product->photo;
    }
    $product->update($data);
    $size = json_decode($request['size']);
    for($i = 0; $i < count($size); $i++) {
        if($size[$i]->id) {
            $productSize = ProductSize::find($size[$i]->id);
        } else {
            $productSize = new ProductSize;
        }
        $productSize->size = $size[$i]->size;
        $productSize->cost = $size[$i]->cost;
        $productSize->products_id = $product->id;
        $productSize->save();
    }
    return response('ok', 200);
}
```

```
    function deleteProduct($id) {
        Products::find($id)->delete();
    }
}
```