КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

Кваліфікаційна робота

на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

РОЗРОБКА ВЕБ-ОРІЄНТОВАНОЇ ПЛАТФОРМИ ДЛЯ ПРОДАЖУ КРАФТОВИХ ПРОДУКТІВ

Виконала студентка 4-го курсу
Потієнко Ірина Юріївна

(підпис)

Науковий керівник:
доцент, кандидат технічних наук
Ткаченко Олексій Миколайович

(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теорії та технології програмування
«01» червня 2022 р.,протокол № 10

Завідувач кафедри

М. С. Нікітченко _____

(підпис)

Київ – 2022

РЕФЕРАТ

Обсяг роботи 76 сторінок, 38 ілюстрацій, 40 джерел посилань.

БАЗИ ДАНИХ, ВЕБ-ПЛАТФОРМА, ІНТЕРНЕТ-МАГАЗИН, РОЗРОБКА, КРАФТОВИЙ ТОВАР, МОВИ ПРОГРАМУВАННЯ, ОПЛАТА, ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА, ШВИДКІСТЬ, ЯКІСТЬ.

Об’єктом роботи є процес просування виробленої крафтової продукції за допомогою веб-системи для продажу крафтових товарів з врахуванням структури різних товарів та замовлення крафтових товарів, що супроводжують виробництво та з’являються при розвитку власних справ в економічному просторі нашої країни.

Основна ідея полягає в тому, щоб допомогти виробнику крафтової продукції просто і доступно створити портфоліо своїх товарів, де споживачі змогли б в зручному веб-інтерфейсі вибрати і придбати крафтову продукцію.

Предметом роботи є веб-орієнтовні системи для підтримки продажів крафтових продуктів, котрі вирішують проблеми швидкого і якісного розміщення та замовлення товарів для малих підприємств, забезпечуючи прозору роботу з різними форматами замовлень користувачів, а також зручну оплату.

Метою роботи є розробка веб-орієнтованої платформи, яка допоможе виробникам крафтової продукції розміщувати для реалізації свою продукцію та послуги з різних областей промислового і харчового виробництва, образотворчого мистецтва та інших креативних послуг. А також споживачам допоможе знайти якісні крафтові товари та послуги і за потреби їх придбати.

Методи розроблення: використання засобів для програмування мови C# та СУБД MySQL, фреймворка ASP.NET Core MVC, відкритої платформи Docker, платформи CircleCI та сервісів платформи AWS.

Результати роботи: виконано загальний огляд засобів на ринку, проаналізовано переваги та недоліки використання певних засобів розробки, вибрані мова програмування та розробки БД, розроблено програмний продукт веб-платформу «Handmade-space» як відкритий та прозорий майданчик для продажу крафтових продуктів, котрий стане неоціненним вкладом у

відновлення та розширення економіки України, для створення повноцінної конкуренції у сфері виготовлення продуктів харчування та різних споживчих товарів загалом.

Програмний продукт веб-система «Handmade-space» для продажу крафтової продукції може застосовуватися як основний канал просування вироблених товарів та послуг.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

База даних — це організована сукупність структурованої інформації або даних, які зазвичай зберігаються у комп’ютерній системі.

Веб-орієнтована платформа – це система, створена з використанням веб-застосунку та призначена для автоматизованого виконання необхідних користувачу дій.

Інтернет-магазин — це засіб для представлення та реалізації товару, роботи або послуги у мережі інтернет.

Система управління базами даних (скорочено СУБД) – це комплекс програм, котрий забезпечує введення, зберігання, пошук, опрацювання даних у базі даних.

Крафтовий виріб – виріб, котрий виготовляється ремісником, або повністю вручну, або за допомогою ручних інструментів чи механічних засобів, доки безпосередній ручний внесок ремісника залишається найважливішою складовою готового продукту.

HTTP (скорочено від Hyper Text Transfer Protocol) – протокол передачі гіпертекстових документів, що використовується в комп'ютерних мережах.

Model-View-Controller (скорочено MVC) – архітектурний шаблон, котрий використовується для проектування та розробки програмного забезпечення.

Open Source – програмне забезпечення з відкритим вихідним кодом.

SQL (скорочено від structured query language) – структурована мова запитів, яка є основною для роботи з реляційними СКБД.

англ. – англійською.

БД – база даних.

МП – мова програмування.

ОС – операційна система.

ПЗ – програмне забезпечення.

СУБД – система управління базами даних.

JIT – just in time.

ВСТУП

Оцінка сучасного стану об’єкта розробки. Разом із бурхливим розвитком інтернет-технологій зросла потреба виробників власних товарів та послуг у вмілих та зручних веб-порталах для презентації, продажу та замовлення крафтових продуктів, оскільки інтернет є вже не лише засобом комунікації та обміну даними, але й широко доступним торговим та рекламним майданчиком.

Потреба у вміннях просування виробленої крафтової продукції може виникнути в різних галузях, таких як маркетинг чи власна підприємницька діяльність. Знайти виробників крафтової продукції, які могли б виконати необхідні завдання, можна:

- звернувшись до знайомих;
- на ярмарках виробів та товарів власного виробництва;
- на різних порталах та майданчиках, які надають різноманітні послуги.

Виробники для пошуку нових напрямків виробництва можуть використовувати різні сервіси пошуку креативних напрямків підприємництва. Проте, для ефективного пошуку необхідні:

- достатній досвід (вже напрацьовані власноруч handmade роботи);
- постійне оновлення власного асортименту виробів;
- оцінка інших товарів та виробів (виробників чи споживацьких запитів), виконаних підприємцями за їх задумами.

Актуальність роботи та підстави для її виконання. Проблема пошуку напрямку просування крафтової продукції актуальна не лише для самих підприємців, а й для замовників, котрі надають подібні послуги, та на пошуки зручного, простого та дієвого засобу для продажів подібного товару витрачають дуже багато сил і часу.

Проблема якісної крафтової продукції існує досить давно, оскільки потрібно враховувати особливості певних видів продукції та прикладне застосування послуг, що супроводжують виробництво та замовлення товарів. В Україні ж всі компанії, котрі можуть надати сервіс для розміщення крафтової продукції, працюють по-старому. Часові діапазони виконання розміщення таких

товарів коливаються до 3 робочих днів. Знайти якісні системи викликає труднощі, а тим паче пересвідчитись у дієвості послуг, які вони надають.

Основною проблемою є уніфікація товарів та послуг та наповнення ними порталу для формування єдиного цінового діапазону крафтової продукції.

У воєнний та особливо післявоєнний час з розвитком України та економічних взаємовідносин значно зросте потреба у якісному представленні крафтової продукції, де норми і правила публікацій тих чи інших напрямків виготовлення власної продукції будуть встановлюватись креативними виробниками. Тому у таких послугах завжди буде потреба.

Мета й завдання роботи. Метою дипломної роботи є створення веб-орієнтованої платформи, яка допоможе виробникам крафтової продукції розміщувати та реалізовувати власну продукцію та послуги з різних областей промислового і харчового виробництва, моди, ілюстрації та інших креативних областей. А споживачам допоможе знайти якісні крафтові товари та послуги. Для досягнення цієї мети поставлено такі завдання:

- Дослідити існуючі засоби для продажу крафтових продуктів.
- Спроектувати та розробити веб-платформу для розміщення та продажу власних крафтових товарів та виробів.

Об'єкт, методи й засоби розроблення. Об’єктом розробки веб-орієнтованої платформи для продажу крафтових продуктів «Handmade-space» є процес швидкого і якісного розміщення, замовлення та продажу крафтових товарів для малих підприємств, що забезпечить прозору роботу з різними форматами замовлень користувачів, а також їх зручну оплату. Предметом роботи є веб-орієнтовні системи для підтримки продажів крафтових продуктів, котрі вирішують проблеми швидкого і якісного розміщення та замовлення товарів для малих підприємств, забезпечуючи прозору роботу з різними форматами замовлень користувачів, а також зручну оплату.

Для розробки програмного застосунку використана еволюційна модель. Спочатку розробляється його початкова версія та передається для оцінки кінцевим користувачам, після чого із врахуванням їх думки та замовника

дорабляється. Аналогічним чином розробляються, передаються й оцінюються проміжні версії програмного продукту, поки він не буде повністю готовим та не відповідатиме усім вимогам замовника.[1]

В якості інструменту створення програмного застосунку обрано JetBrains Rider 2021.3.4 – інтегроване середовище розробки (IDE) мовою програмування C#, яке є безкоштовним за студентською підпискою та вільно поширюваним.

Можливі сфери застосування. Веб-орієнтована платформа для продажу крафтових продуктів «Handmade-space» може застосовуватися як основний канал просування крафтових товарів та послуг.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Для визначення функцій, вимог та бізнес-логіки майбутньої веб-платформи для продажу крафтових товарів є необхідним дослідити предметну область продажів товарів, зокрема крафтових, та проаналізувати аналогічні веб-сервіси для продажу товарів, що існують та функціонують у мережі Інтернет. На основі цього аналізу і буде виділено напрямки розробки майбутньої веб-платформи.

1.1 Предмет сфери розробки

1.1.1 Крафтові вироби

В останні роки використання слова «крафтовий» в українській мові стає усе більш популярним. У перекладі з англійської «craft» означає майстерну роботу або ремесло.

За визначенням, крафтові вироби – вироби, які виготовляються ремісниками або повністю вручну, або за допомогою ручних інструментів чи механічних засобів, доки безпосередній ручний внесок ремісника залишається найважливішою складовою готового продукту. Особливість крафтових виробів випливає з їхніх відмінних рис, які можуть бути утилітарними, естетичними, художніми, творчими, культурно прив’язаними, декоративними, функціональними, традиційними, релігійно та соціально символічними та значущими».[2]

Сам же крафт – це дрібне виробництво (без використання технологій, характерних для масового промислового виробництва), яке виготовляє невеликі партії товару за індивідуальними рецептами на малих потужностях. За рахунок цього є можливість контролювати якість кожної одиниці товару.

Термін «крафт» виник у США у часи появи усе більшої кількості дрібних некомерційних пивоварень. Вони відкривалися поціновувачами справжнього живого пива, котрі не бажали пити однотипний алкогольний напій від відомих брендів, яким були заставлені полиці магазинів. Саме таке пиво, котре виготовлялося незалежними пивоварами у невеликих об’ємах, почали називати крафтовим.[3]

Найбільш вдалий український відповідник терміну крафтовий – словосполучення «власного виробництва». Наприклад, заклади харчування можуть пропонувати своїм клієнтам продукти або напої власного виробництва.

Проте, у цих двох термінах присутня все ж деяка різниця. Наприклад, якщо певний заклад харчування, котрий є частиною великого промислового концерну, пропонує клієнтам випічку, вироблену цим концерном, тоді така випічка є товаром власного виробництва, але при цьому не є крафтовим. Якщо ж заклад харчування пропонує випічку, яку виготовляють на продаж жінки з сусіднього села, то такий товар є товаром власного крафтового виробництва.

Як стверджує велика кількість українських виробників крафтових продуктів, метою справжнього крафтового виробника є не «кустарне» виготовлення продуктів сумнівного походження. Справжнє прагнення цих людей – об’єднати традиційні рецепти, що дійшли до нас від поколінь наших пращурів, з інноваційними підходами до виробництва продукції, котрі забезпечують відмінну якість продукту. Такі продукти мають ставати справжніми «фішками», характерними саме для території України, «ноу-хау» ідеями, у захваті від яких буде людина з будь-якого куточка нашої планети.

1.1.2 Замовлення в Інтернет-магазині

Електронна комерція за допомогою системи інтернет-магазинів є одним із найбільш динамічних сегментів інтернет-економіки. Щорічні колосальні темпи зростання інтернет-продажів є глибокою структурною зміною, яка впливає на роздрібну торгівлю. Тому для кожної компанії стає все більш важливим створення конкурентних переваг. Одним із найбільш стратегічно важливих елементів бізнес-концепції електронної комерції є система інтернет-магазинів.

Система інтернет-магазинів – це програмний комплекс для створення та роботи інтернет-магазину. Існують різні постачальники, які пропонують різні програмні рішення. Вони або безкоштовні (з відкритим кодом), або їх можна знайти у різних цінових сегментах (магазин, індивідуальне програмне забезпечення).[4] Успіх продажів компанії електронної комерції багато в чому залежить від правильного вибору системи інтернет-магазинів. Найважливіші

критерії вибору включають відповідність пошуковій системі, варіанти презентації та дизайну продукту, способи оплати та зручність для користувачів.

Згідно з офіційним визначенням, інтернет-магазин є невід'ємною частиною процесу, відомого як інтернет-покупки. Інтернет-покупки – це процес, у якому відвідувачі можуть придбати певні продукти чи послуги, запропоновані в інтернет-магазині.[5]

У системі онлайн-покупок або інтернет-магазину існують три найпоширеніших способи ведення бізнесу[5]:

- B2C (бізнес до споживача) – передбачає, що процес інтернет-покупок відбувається між покупцем та підприємцем, тобто виробником або постачальником послуг;
- B2B (бізнес до бізнесу) – процес онлайн-торгівлі відбувається між двома підприємцями;
- B2B2C (розроблений як свого роду комбінація попередніх двох методів) – онлайн-торгівля відбувається або між двома підприємцями, або між покупцем і підприємцем за умови, що в цьому способі торгівлі між ними також є посередник;
- C2C (споживач до споживача) – онлайн-торгівля відбувається між двома користувачами у форматі аукціону.

Крім вищезгаданих типів інтернет-магазинів, існують також:

- C2B (споживач до бізнесу) – онлайн-торгівлю веде покупець, котрий запитує про конкретну послугу або товар у продавця, маючи деякі власні запити (наприклад, бюджет, який він має на покупку певного продукту чи послуги);
- B2A (бізнес до адміністрації) – виконання фінансових операцій між підприємцями, тобто компаніями, та численними державними установами;
- C2A (споживач до адміністрації) – стосується всіх необхідних транзакцій, які особа здійснює щодо різних державних органів;
- B2E (бізнес для працівника) – призначений в першу чергу для співробітників в рамках певної компанії.

Хоча більшість видів інтернет-магазинів не мають широкого поширення (принаймні, не в нашій країні), постійно робляться висновки щодо впровадження згаданих моделей інтернет-магазинів та їх суттєвого впливу як на ведення бізнесу, так і на окремих осіб, і навіть на цілі організації та органи державної влади.

Загалом, незважаючи на схожість з роботою звичайних магазинів, сьогодні інтернет-магазини стають одним з найбільш перспективних способів ведення і розвитку великого і малого бізнесу за рахунок зниження витрат та збільшення прибутку.

1.2 Аналіз існуючих на ринку аналогів

1.2.1 Інтернет-магазин у соціальній мережі

Соціальні мережі є важливим компонентом для просування та продажу товарів.

Соціальна мережа – інтернет-співтовариство користувачів, об’єднаних за певною ознакою на базі одного веб-сайту.[6] Вони можуть мати соціальну або ділову мету, а також бути місцем для діяльності маркетологів, котрі прагнуть залучити клієнтів до певного бізнесу.[7]

Найвідомішими соціальними мережами є Facebook, Twitter, LinkedIn та Instagram. Зокрема, Facebook залишається найбільшою та найпопулярнішою з них: станом на 31 грудня 2021 року веб-сайтом користуються 2,91 мільярда людей щомісяця.[7]

Можна виділити деякі особливості та переваги ведення інтернет-магазину у соціальній мережі для підприємця [8]:

- постійне збільшення кількості нових користувачів (наприклад, у Facebook лише в Україні їх налічується близько 15 мільйонів);
- для створення сторінки інтернет-магазину у соціальній мережі достатньо мати власний обліковий запис, котрий можна зробити публічним та наповнити його товарами та новинами про них;

- можливість таргетування реклами на цільову аудиторію, оскільки багато користувачів соціальної мережі добровільно залишає корисну інформацію про себе;
- знайомі та друзі можуть стати першою аудиторією магазину, оскільки саме вони є основою будь-якої соціальної мережі. А згодом, створюючи новий та цікавий контент та проводячи рекламні кампанії, можна розширювати його цільову аудиторію;
- технічна підтримка сторінки у Facebook або Instagram не потребує виділення окремого бюджету. Єдине, на потрібно витратитись, це SMM та реклама.

Також інтернет-магазину у соціальній мережі має ряд недоліків[8]:

- компанія, якою обслуговується соціальна мережа, є власником інформації, що розміщена на її сторінках, а не її користувачі;
- інтернет-магазин у соціальній мережі має суттєво менше можливостей для просування товару аніж власний магазин;
- соціальна мережа є дуже обмеженою у можливостях подачі та оформлення контенту, що майже унеможливлює індивідуальність окремого інтернет-магазину;
- обмеження потенційної клієнтури інтернет-магазину межами соціальної мережі, оскільки немає можливості потрапити на сторінку з пошукового рядка Google або відеороликів YouTube;
- довіра користувачів до інтернет-магазину у соціальній мережі є значно нижчою, ніж до повноцінної торгової платформи;
- інтернет-магазин у соціальній мережі є менш прибутковим, ніж власний повноцінний інтернет-магазин.

Інтернет-магазин у соціальній мережі є гарним варіантом, коли складно визначити його дохідність, зокрема на його старті. Тому на певному етапі краще обрати інший майданчик для ведення власного бізнесу, а сторінку у соціальній мережі за бажанням розвивати як доповнення.

1.2.2 Інтернет-магазин на базі платформи OLX

OLX (англ. OnLine eXchange) — платформа онлайн-оголошень, створена у 2006 році Фабрісом Гріндою та Алеком Оксенфордом у Аргентині і згодом придбана компанією Naspers, є провідною світовою платформою оголошень на ринках Азії, Африки, Близького Сходу, Південної Америки та Європи та доступна у більш ніж 50 країнах світу. OLX об'єднує місцевих жителів для купівлі, продажу та обміну вживаними товарами та послугами, роблячи при цьому процес розміщення оголошення швидким, простим та безкоштовним.[9]

Щомісяця сотні мільйонів людей на місцевих ринках по всьому світу використовують OLX, щоб знайти та продати широкий асортимент товарів, включаючи меблі, музичні інструменти, спортивні товари, автомобілі, дитячі товари, мотоцикли, фотоапарати, мобільні телефони та багато іншого. Працюючи на ринках по всьому світу, OLX допомагає людям покращити своє життя завдяки розумній торгівлі від людини до людини.[9]

Цікавим фактом є те, що OLX не є сполучною ланкою між покупцем і продавцем, а лише виступає як каталізатор для з'єднання людей онлайн через портал оголошень. Спілкування, торги, аванси та транзакції залишаються лише між покупцем і продавцем.[10]

Головними перевагами платформи OLX є:

- сильний та впізнаваний бренд;
- є одним із лідерів ринку у сфері C2C;
- великий досвід роботи із різними гігантами електронної комерції;
- гарна маркетингова стратегія та висока популярність його реклами у мережі інтернет;
- безкоштовне розміщення оголошень, доплачувати потрібно лише за підвищення рейтингу оголошення у загальному списку;
- легкий доступ до будь-яких товарів за невисокою ціною.

Також він має і ряд недоліків:

- дуже велика конкуренція у продажах товарів;
- немає гарантії якості товару;

- досить високий ризик інтернет-шахрайства та розміщення підроблених оголошень.

1.2.3 Інтернет-магазин на базі платформи Prom

Prom.ua —маркетплейс, розроблений українською компанією EVO, де підприємці мають можливість створити інтернет-магазин, а також розмістити власні товари у загальному каталозі.[11]

Основні переваги використання Prom:

- не потрібно створювати та підтримувати інтернет-магазин зусиллями сторонніх розробників, залучати фахівців;
- велике охоплення аудиторії, доступ до великої кількості потенційних покупців;
- просте створення своєї сторінки, налаштування продажів, власний кабінет, безкоштовне Seo-просування, можливість підключення акцій, розпродажів, отримання відгуків клієнтів тощо;
- крім десктопної версії власник отримує адаптивну мобільну версію сайту;
- особливі умови доставки замовлень.

Недоліки:

- Немає можливості налаштувати щось чи доопрацювати додатковий функціонал, як і вивантажити дані платформи з інтернет-магазину. Можливо тільки скористатися можливостями й шаблонами, які є на платформі.
- Немає можливості допрацювати платформу для e-commerce. Можливо лише змінити тарифний план на вигідніший чи вкласти більше коштів для просування в каталогах.
- Кількість товарів залежить від обраного тарифного плану. Максимальна їх кількість — 10 000 товарів. За решту потрібно домовлятися окремо.

- Кількість товарів та фільтрів може впливати на швидкість. Неможливо покращити цей показник чи скористатися іншим хостингом, адже це SaaS-рішення.
- Без додаткових платежів Prom не буде показувати товари покупцям у своєму каталозі. Треба не тільки оплатити пакет на рік, а й постійно поповнювати баланс. З балансу списується комісія за продані з Prom товари. Також при просуванні в каталозі продавець сплачує комісію за переходи на сторінку товару.
- Аналітика доступна за короткий період.

Підбиваючи підсумки, варто порівняти різні платформи для продажу товарів за рядом критеріїв. Це порівняння відображено у табл. 1.1.

Таблиця 1.1 – Порівняння веб-платформ для продажу товарів

Функціонал	Платформа			
	Соціальна мережа	OLX	Prom	**Handmade-space**
Розміщення власної сторінки інтернет-магазину	Є	Немає	Є, платне	**Є, безкоштовне**
Обмеження на кількість товарів магазину	Немає	Немає	Є	**Немає**
Загальний каталог товарів	Немає	Є	Є з розміщенням товарів за додаткову плату	**Є**
Функціонал для узгодження деталей замовлення	У вигляді чату	У вигляді чату	Лише при оформленні замовлення	**У вигляді автоматичної системи**
Аналітика продажів	Немає	Немає	Є для власного магазину та за короткий період	**Є**
Збір та обробка відгуків до замовлень	Немає	Немає	Є	**Є**
Інформація про сервіси доставки	Немає	Є	Є	**Є**

2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ

У попередньому розділі було проаналізовано предметну область продажів товарів онлайн та існуючі аналоги веб-платформи для продажу крафтових продуктів «Handmade-space». Тепер можна перейти до вибору архітектури, методологій та технологій для ефективної реалізації даної веб-системи.

2.1 Вибір архітектури для розробки веб-системи

Веб-застосунок є особливим програмним засобом, котрий реалізує архітектуру «клієнт-сервер» (рис. 2.1).

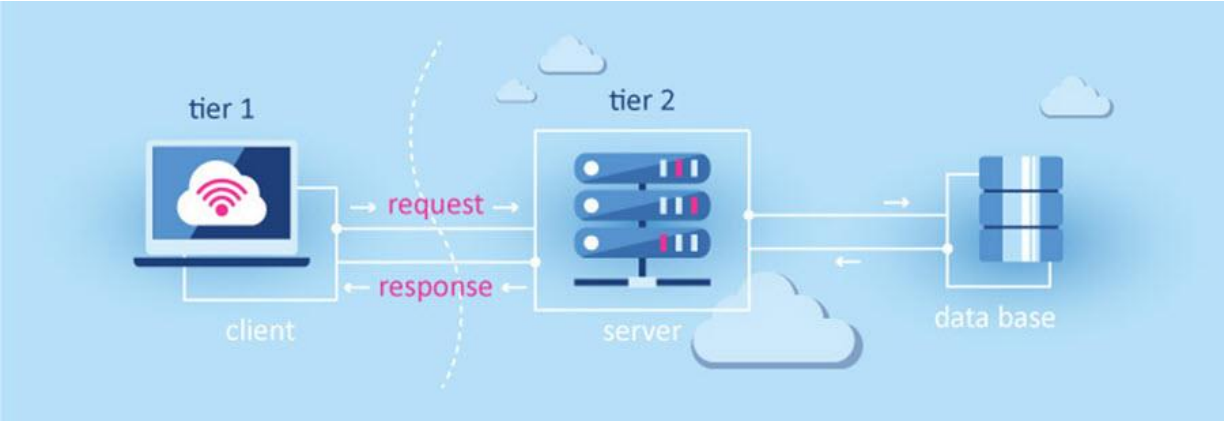

Рисунок 2.1 – Схема архітектури типу «клієнт-сервер» [12]

Архітектура «клієнт-сервер» визначає загальні принципи взаємодії у мережі, у якій наявні сервери, вузли-постачальники деяких специфічних функцій (сервісів) та клієнти (споживачі цих функцій). Клієнт-серверні технології є практичними реалізаціями даної архітектури.[13]

Клієнт-серверна архітектура системи передбачає, що основна частина програми виконується на веб-сервері, котрий обробляє отримані від клієнта запити відповідно до бізнес-логіки продукту і формує відповідь, котру потім надсилає користувачеві. Браузер на стороні клієнта перетворює отриману від сервера відповідь у графічний інтерфейс, зрозумілий користувачеві.[12]

Класична дворівнева архітектура розподіляє базові компоненти веб-системи між двома вузлами (клієнтом та сервером) та використовується у застосунках, де сервер обробляє клієнтські запити безпосередньо та в повному обсязі.[13]

Трирівнева архітектура розподіляє компоненти веб-системи на дві або

більше частин, кожна з яких може виконуватися на окремому комп'ютері та взаємодіяти одна з одною, обмінюючись повідомленнями у заздалегідь узгодженому форматі.[13]

У трирівневій архітектурі компоненти розподіляються таким чином:

- подання даних – на боці клієнта;
- прикладний компонент — на виділеному сервері програм (як варіант, що виконує функції проміжного ПЗ);
- управління ресурсами - на сервері БД, який і представляє дані, що запитуються.

2.2 Вибір мови для програмування веб-систем

Етап вибору мови програмування є вкрай важливим, оскільки в залежності від обраної МП можуть змінюватись такі атрибути веб-системи як:

- швидкість роботи веб-системи;
- складність розробки веб-системи;
- необхідне для роботи веб-системи обладнання веб-серверу.

Також досить велику роль грає досвід, навички та знання програміста для роботи з певною мовою програмування.

2.2.1 Мова програмування Java

Java – строго типізована розроблена американською компанією Sun Microsystems у 1991 році об'єктно-орієнтована мова програмування загального призначення (в подальшому була придбана компанією Oracle). Спочатку названа як Oak Джеймсом Гослінгом, одним із творців мови, вона була створена для програмування користувацьких машин, таких як телевізори, тостери, мікрохвильовки та інша побутова техніка.[14] Також причиною, яка забезпечила розвиток мови, був розвиток всесвітньої мережі інтернет, що збільшувало потребу у портативних та незалежних від платформи системах. Саме тому творці Java на чолі з Гослінгом розробили програми з використанням нової мови, які могли запускатися на всіх типах комп'ютерів. Протягом 1993 року був створений перший браузер HotJava для роботи створених програм. З 1996 Java є

повноцінною об'єктно-орієнтованою мовою програмування, що використовується для написання веб-застосунків. [15]

На противагу багатьом іншим мовам програмування, компілятор мови Java конвертує вихідний код не у машинний код, а у спеціальний проміжний байт-код, котрий у свою чергу може бути інтерпретований у машинний. Байт-код є незалежним від платформи, а тому може запускатися на будь-якому комп'ютері за допомогою віртуальної машини Java (JVM).[15]

Java стала популярною мовою програмування для програм, які мали виконуватися на різних платформах. Її головною інновацією стали апплети – дрібні програми, створені таким чином, що можуть пересилатися через інтернет, а також завантажуватися та виконуватися автоматично веб-браузером.[14] Вони використовуються для керування вхідними даними від користувача, даними, що підтримуються сервером та простими функціями, що виконуються локально на машині користувача.

Java зв’язується з веб-сторінкою через спеціальний тег <APPLET>. На рис. 2.2 проілюстровано цей процес через такі кроки [15]:

1. Користувач надсилає запит до віддаленого сервера. Веб-сервер – це програма, що приймає запити, обробляє їх та надсилає затребуваний у запиті документ.
2. HTML-документ повертається у відповідь до браузера користувача. Він містить тег <APPLET>, котрий визначає апплет.
3. Відповідний код апплету, що попередньо був створений компілятором Java через його вихідний код, передається до користувацького комп'ютера.
4. Браузер на комп'ютері користувача інтерпретує надісланий байткод та надає потрібні вихідні дані.
5. Користувач може мати подальшу взаємодію з апплетом без додаткових завантажень з веб-серверу, оскільки отриманий байткод містить всю інформацію, необхідну для визначення апплету.

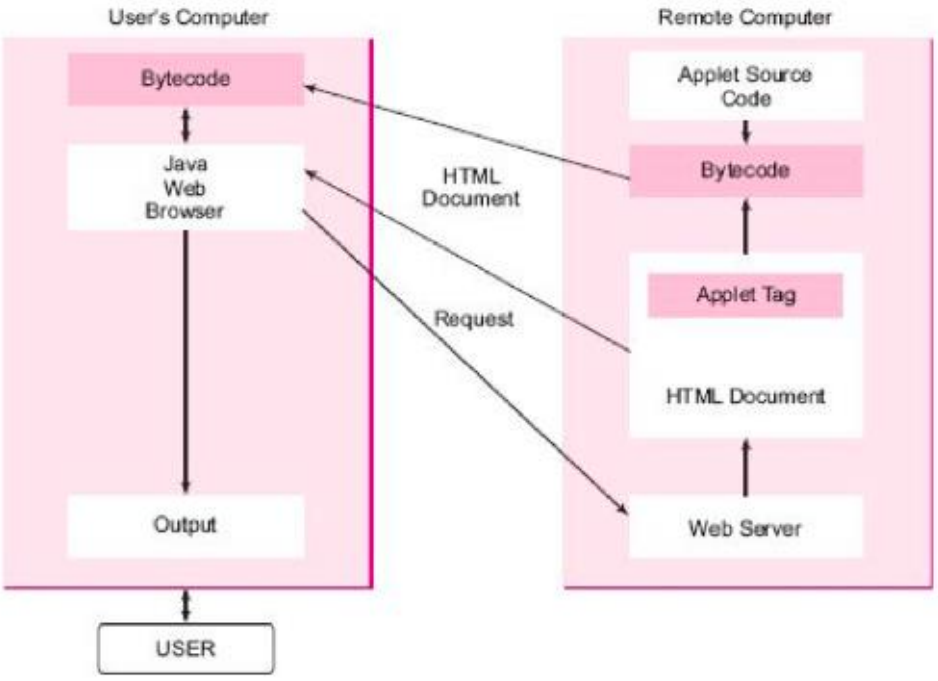

Рисунок 2.2 – Схема взаємодії Java з веб-сторінкою

Багато веб-платформ Java (наприклад, JavaServer Faces) надають бібліотеки для створення готових шаблонів сторінок та управління сеансами, що забезпечує повторне використання коду.

Підбиваючи підсумки, можна стверджувати, що мова програмування Java для створення веб-застосунків надає достатній інструментарій, гарну архітектуру та досить високу швидкість роботи створених веб-систем.

Проте дана мова програмування має також деякі недоліки, основними з яких є громіздкість написаних на Java застосунків та потреба у досить великих системних потужностях для їх швидкої роботи.

Окрім знань правильної архітектури та технологій розробки веб-застосунків, не менш важливу роль грають знання та досвід програміста. Без необхідних знань та навичок у написанні коду мовою Java створений веб-застосунок значно поступатиметься за характеристиками застосункам, розробленим з використанням іншої мови програмування.

2.2.2 Мова програмування Python

Python – потужна скриптова мова програмування загального призначення, котра має ефективні високорівневі структури даних і простий, але при цьому ефективний підхід до об’єктно-орієнтованого програмування.[16] Елегантний

синтаксис та динамічна типізація разом з інтерпретованою природою роблять його ідеальною мовою для написання сценаріїв та швидкої розробки застосунків у багатьох областях та на більшості платформ.[16]

Стандартна бібліотека Python є вільно доступною та включає у себе великий обсяг корисних функцій, при тому що мова має мінімалістичний синтаксис ядра.[17] Веб-застосунок, написаний мовою Python, є повноцінним застосунком з технічної точки зору. Він завантажується у пам'ять, керує своїм внутрішнім станом, котрий вправно зберігає від запиту до запиту.[17]

Для розробки веб-застосунків використовуються різні фреймворки Python, котрі дещо відрізняються своїм функціоналом та вбудованими модулями, проте загалом є дуже схожими. Найпопулярнішими з них є Django, Bottle, Flask та Pyramid.

Дана мова програмування дозволяє досить зручно створювати інтернет-застосунки, проте має ряд недоліків, основними з яких є:

- низька швидкість роботи (через те, що Python використовує у своїй роботі інтерпретатор а не JIT-компілятор);
- вбудовані класи та функції не є можливим модифікувати;
- сторонні бібліотеки функцій існують у малій кількості.

Зазначені недоліки, а до того ж досить важке розуміння розробленого коду через недостатню кількість необхідної документації при обмеженому виборі технологій створення веб-застосунків робить непривабливою мову програмування Python для розробки веб-системи для продажу крафтових продуктів «Handmade-space».

2.2.3 Мова програмування PHP

PHP – проста і при цьому потужна скриптова мова програмування з відкритим сирцевим кодом, котра використовується для генерації сторінок HTML.[18] Вона була створена 1994 року Расмусом Лердорфом для відслідковування переглядів власного резюме. На початку свого існування абревіатура PHP розшифровувалася як «Personal Home Page» (персональна домашня сторінка), проте згодом із розширенням можливостей мови та її

використанням усе більшою кількістю розробників стала означати «Hypertext Preprocessor» (гіпертекстовий препроцесор).[19]

Мова програмування PHP може бути використана у 3 різних способи [18]:

1. Для написання скриптів на стороні сервера.

   Мова оригінально була створена для розробки динамічного веб-контенту, і наразі продовжує використовуватись для цього завдання. Для генерування HTML потрібні лише інтерпретатор PHP та веб-сервер для надсилання документів. Також можливим є генерування XML-документів, графіки, анімації Flash, файлів формату pdf тощо.

2. Для написання скриптів командного рядка.

   PHP може виконувати скрипти з командного рядка, так само як Perl або Unix shell. Більш того, є можливість використовувати ці скрипти для завдань системного адміністрування, таких як парсинг логів або збереження бекапу.

3. Для написання скриптів на стороні клієнта

   PHP може виконуватись на більшості операційних систем, від різних UNIX-подібних систем, таких як Solaris та FreeBSD, до різних версій Windows та MacOS. Також він може працювати з більшістю відомих веб-серверів, включаючи Apache, Microsoft IIS та сервери Netscape/iPlanet.

Можливості використання мови програмування PHP є досить різноманітними, тому можна виділити ряд її переваг:

- наявність готових бібліотек для роботи з найпопулярнішими СУБД;
- відсутність потреби завантажувати потрібні бібліотеки та вказувати спеціальні параметри компіляції;
- можливість вбудовувати скрипти прямо в код html-сторінки;
- можливість пересилати код довільним браузерам та пристроям, включаючи портативні комп’ютери та стільникові телефони;
- безкоштовність та доступність використання усіх технологій.

Окрім переваг, мова програмування PHP має і ряд недоліків:

- засоби мови для роботи з винятками є досить слабкими;
- налаштування сервера та розгортання застосунків ускладнюється залежністю синтаксису мови від глобальних параметрів конфігурації;
- у застосунків, створених мовою програмування PHP, нижча захищеність, ніж у застосунках із використанням інших мов;
- об'єкти передаються у методи за значенням а не за посиланням, що є незвичним для багатьох програмістів, котрі використовують інші мови програмування
- непослідовний синтаксис — щодо мови PHP, особливо старої частини, що заснована на функціях, можна побачити, що частина з них має префікси array_, str_, частина немає. Параметри функцій можуть бути розташовані не зовсім логічно і не так, як в іншій функції цієї групи.
- низька швидкодія (через те, що в роботі не застосовується JIT-компілятор);

Недостатня захищеність додатків, низька швидкодія та недоліки синтаксису роблять мову програмування PHP не привабливою для використання при розробці веб-системи для продажу крафтових продуктів «Handmade-space».

2.2.4 Мова програмування C#

Мова програмування C# – об'єктно-орієнтована строго типізована мова програмування, створена для платформи .NET. Розроблена командою Андерса Гейлсберга, Скота Вілтамута та Пітера Гольде під керівництвом Microsoft Research (належить Microsoft).[20]

C# має сильний взаємозв'язок з середовищем виконання .NET Framework, що пояснюється двома причинами. По-перше, C# спочатку призначався для створення коду, який має виконуватись у середовищі .NET Framework. І по-друге, використовувані в C# бібліотеки визначені середовищем .NET Framework. Насправді це означає, що C# і .NET Framework тісно пов'язані один з одним, хоча теоретично їх цілком можна відокремити.

Середовище .NET Framework визначає також загальномовне середовище виконання (Common Language Runtime — CLR) – систему, яка управляє виконанням програм. Як складова частина середовища .NET Framework вона підтримує багатомовне програмування та забезпечує безпечне виконання програм.[20]

Однією із найпопулярніших технологій для створення веб-застосунків мовою C# є ASP.NET Entity Framework Core.

Entity Framework – фреймворк, який представляє спеціальну об'єктно-орієнтовану технологію на базі .NET для роботи з даними. Дана технологія є більш високим рівнем абстракції ніж традиційні засоби технології ADO.NET. Вона дозволяє абстрагуватися від типу сховища та самої бази даних та працювати однаково із даними незалежно від типу СУБД.[19 20]

ASP – це технологія попередньої обробки від Microsoft, котра надає можливість під час процесу формування веб-сторінки під’єднувати її програмні модулі, зокрема зовнішні COM-компоненти. Вона працює на веб-сервері IIS та на платформі операційних систем Windows NT.[21]

ASP.NET Core - це кросплатформений фреймворк з відкритим вихідним кодом, що використовується для створення сучасних хмарних застосунків, котрі працюють на безкоштовній відкритій міжплатформеній програмі .NET Core. Фреймворк був розроблений таким чином, аби забезпечити оптимізовану структуру розробки для систем, котрі запускаються локально або розгортаються у хмарі. Він складається з модульних компонентів, тому зберігається гнучкість під час створення рішень.[21]

У мови C# виділяють багато переваг:

- продовжує багато вдосконалюватися та доопрацьовуватися компанією Майкрософт, оскільки був створений пізніше, ніж Java та інші мови;
- безкоштовність для невеликих компаній, стартапів, студентів та деяких індивідуальних розробників ряду додаткових інструментів: Visual Studio, хмара Azure, Windows Server, Parallels Desktop для Mac Pro та інші;
- типи даних мають фіксований розмір, що підвищує «мобільність» мови

та спрощує програмування, тому що завжди точно відомо, з якими даними доведеться мати справу;

- автоматичний «збір сміття» та очищення пам’яті середовищем CLR;
- велика кількість зручного та зрозумілого синтаксичного «цукру», що полегшує перехід програмістів з інших мов програмування.

Проте C# має і деякі недоліки:

- пріоритетна орієнтованість на ОС Windows;
- ліцензійна версія мови для комерційного використання є досить дорогою.

Підсумовуючи написане вище, мову програмування C# можна обрати в якості основної для розробки веб-платформи для продажу крафтових продуктів «Handmade-space», оскільки підведені підсумки показують, що недоліки не є значними і на досягнення мети (створення веб-сервісу для продажу крафтових продуктів «Handmade-space») не впливають.

2.3 Вибір СКБД для управління контентом веб-системи

При написанні програм у будь-якій сфері, неможливо обійтись без використання бази даних, оскільки навіть невеликі системи потребують зберігання та відтворення даних.

База даних — це організована сукупність структурованої інформації або даних, які зазвичай зберігаються у комп’ютерній системі. База даних зазвичай контролюється системою управління базами даних (СУБД). Дані та СУБД разом із додатками, що з ними пов’язані, називають системою баз даних, або скорочено базою даних.[22]

Існують різні типи структур баз даних[23]:

- ієрархічна: відповідає порядку ранжування або зв’язку «батько-син» для структурування даних;
- мережева: схожа на ієрархічну базу даних, проте надає можливість дочірньому запису зв'язуватися з різними батьківськими записами, дозволяючи двосторонні зв'язки;

- об'єктно-орієнтована: у такій базі даних інформація зберігається у вигляді об'єктів;
- реляційна: орієнтована на таблицю, де кожен біт даних має зв'язок з кожним іншим бітом даних;
- нереляційна (NoSQL): використовує різноманітні формати, такі як документи, графіки, широкі стовпці тощо, що забезпечує велику гнучкість та масштабованість для дизайну бази даних.

Дані в найпоширеніших типах баз даних, які діють сьогодні, зазвичай моделюються у рядках і стовпцях таблиць, що робить обробку та запити даних ефективними. Дані можна легко отримувати, керувати ними, змінювати, оновлювати, контролювати та організовувати. Більшість баз даних використовує мову структурованих запитів (SQL) для запису та запиту даних.[22]

Дані веб-платформи для продажу крафтових продуктів мають визначену та стабільну структуру, а тому для оперування ними найкращим вибором буде реляційна база даних.

Відповідно і вибрана СКБД повинна використовувати реляційну модель для роботи з даними, а також тип зв'язку у вигляді реляцій між сутностями з різних таблиць. Кожен запис складатиметься з певної кількості атрибутів (стовпців) різного типу і матиме унікальний ключ-ідентифікатор, що зберігатиметься у таблиці у вигляді окремого поля[24].

2.3.1 СКБД PostgreSQL

PostgreSQL — це розширена реляційна система керування базами даних корпоративного класу з відкритим вихідним кодом, котра підтримує як реляційні, так і нереляційні типи запитів. Дана високостабільна СУБД підтримується більш ніж 20-річним розвитком спільноти, що сприяє її високому рівню стійкості, цілісності та коректності. [25]

PostgreSQL використовується як основне сховище даних для багатьох веб, мобільних, геопросторових та аналітичних застосунків.

Переваги використання PostgreSQL [25]:

- Багаті функції та розширення

  PostgreSQL надає надійні набори функцій, що включають у себе керування кількома версіями одночасного виконання (MVCC), відновлення в момент часу, детальний контроль доступу, табличні простори, асинхронну реплікацію та вкладені транзакції, резервне копіювання, удосконалений та оптимізований планувальник запитів із записом наперед. Дана СКБД підтримує міжнародні набори символів, багатобайтове кодування, Unicode, а також сортування, чутливість до регістру та форматування. PostgreSQL має високу масштабованість як за кількістю даних, якими він може керувати, так і за кількістю одночасних користувачів, яких він може вмістити.

- Надійність та відповідність стандартам

  Попередній запис PostgreSQL робить його дуже стійкою до відмов базою даних. Також вона є сумісною із ACID, має повну підтримку зовнішніх ключів, об’єднань, переглядів, тригерів і збережених процедур багатьма різними мовами та включає більшість типів даних стандарту SQL:2008, зокрема INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL і TIMESTAMP. Більш того, надається підтримка зберігання великих двійкових об’єктів, таких як зображення, аудіо- та відеофайли.

- Ліцензія з відкритим кодом

  Вихідний код PostgreSQL доступний за ліцензією з відкритим вихідним кодом, що дає свободу вільно та безкоштовно використовувати, модифікувати та впроваджувати його. PostgreSQL не вимагає ліцензування, що усуває ризик надмірного розгортання. Спеціальна спільнота авторів та ентузіастів PostgreSQL регулярно знаходить помилки та виправляє їх, сприяючи загальній безпеці системи баз даних.

Недоліки СКБД PostgreSQL:

- дотримання строгої структури даних та таблиць є необхідним;

- невисока швидкість роботи із запитами на читання та вставлення;
- робота із безструктурними даними не є зручною через обов’язкове використання відношень

PostgreSQL є сенс використовувати у випадку, якщо[25]:

- цілісність та надійність даних є основними вимогами до системи
- за необхідності використовувати процедури, призначені для користувача
- структура даних є досить складною
- за майбутньої інтеграції системи, зокрема, PostgreSQL може суттєво полегшити процес переходу на іншу оплачувану СКБД

Не є доцільним використовувати PostgreSQL у випадках якщо:

- швидкість обробки запитів на читання є в основних вимогах до системи
- немає потреби у складних структурах, цілісності даних та відповідності ACID

2.3.2 СКБД Microsoft SQL Server

Microsoft SQL Server – це набір програмного забезпечення для баз даних, створений корпорацією Microsoft, котрий включає в себе: механізм реляційної бази даних, який зберігає дані в таблицях, стовпцях і рядках; сервіси інтеграції (SSIS), які є інструментом переміщення даних для імпорту, експорту та перетворення даних; сервіси звітування (SSRS), які використовуються для створення звітів та їх надання кінцевим користувачам; сервіси аналізу (SSAS), які є багатовимірною базою даних, що використовується для запиту даних з основного механізму бази даних. [26]

Microsoft SQL Server (MSSQL) широко використовується у різних проєктах, як персональних так і корпоративних. MSSQL — це масштабована СУБД, яка включає в себе кілька інструментів ETL (Extract, Transform and Load) та служби звітності, де дані можливо додавати, змінювати та читати за допомогою стандартизованої мови структурованих запитів (SQL). [26]

Transact-SQL – діалект мови SQL з розширеннями, що використовується у MSSQL як реалізація стандарту ANSI / ISO, розроблений Microsoft спільно із Sybase. [26]

Дана СКБД має ряд таких переваг [27]:

- включає у себе професійне програмне забезпечення для керування базами даних корпоративного рівня, а також має гарні можливості для збільшення масштабованості
- має інтеграцію із платформою .NET
- має гарну підтримку відновлення даних завдяки використанню файлів журналів, кешування та резервного копіювання
- вирізняється високою швидкодією та продуктивністю роботи
- проста у використанні.

Проте, вона також має і ряд недоліків:

1) Microsoft SQL Server є моноплатформенним та призначений лише для роботи на серверах на базі операційної системи Windows.
2) Ліцензування для комерційного використання даної СКБД та сервіси для хостингу є досить дорогими.

2.3.3 СКБД MySQL

MySQL – реляційна система керування базами даних із відкритим вихідним кодом, котра була розроблена шведською компанією MySQL AB у 1994 році та названа ім’ям дочки одного із засновників. MySQL є невід’ємною частиною LAMP стеку, котрий включає у себе такі компоненти як Linux, Apache, MySQL, Perl/Python/PHP.[28] Більш того, саме MySQL використовується у таких відомих системах, як Facebook, Twitter, YouTube та навіть NASA.[29] На зараз розробка та адміністрування даної СКБД підтримується корпорацією Oracle.[28]

MySQL є клієнт-серверною системою, котра відповідно складається з таких компонентів:

- багатопоточного SQL-сервера, котрий працює з даними;
- клієнта з його бібліотеками та програмами, котрий дає вказівки що потрібно зробити з даними;
- широкого діапазону API-інтерфейсів, котрі дозволяють додаткам, розробленим мовами Python, PHP, C / C ++, Java, Perl, .NET тощо

підключатися до бази даних MySQL.

СКБД MySQL є однією з найпопулярніших СКБД у світі за версією її розробників, тому що має ряд переваг [30]:

- є швидкою у виконання команд, а тому може працювати із таблицями, кількість рядків яких досягає 50 млн. (обмеження розміру файлу за замовчуванням для таблиці становить 4 ГБ, але можливо збільшити його до теоретичного обмеження у 8 мільйонів терабайт);
- потужна програма, яка може обробляти великий набір функціональних можливостей найдорожчих і найпотужніших пакетів баз даних;
- використовує стандартний діалект мови SQL;
- є базою даних з відкритим вихідним кодом, а тому є безкоштовною для використання;
- має ліцензію GPL з відкритим кодом, що дозволяє програмістам змінювати програмне забезпечення SQL відповідно до їх власного специфічного середовища;
- підтримує багато операційних систем та мов програмування;
- має просту і ефективну систему безпеки.

Також MySQL не може обійтися без ряду недоліків:

- не повністю реалізований SQL-стандарт;
- не підтримуються тригери та збережені процедури;
- деякі операції реалізовані менш надійно, ніж у інших СКБД.

Підсумовуючи написане вище, СУБД MySQL можна обрати в якості основної для розробки веб-сервісу для продажу крафтових продуктів «Handmade-space», оскільки підведені підсумки по недолікам показують, що вони не є значними і на досягнення цілі (створення та роботу з базою даних веб-сервісу для продажу крафтових продуктів «Handmade-space») не впливають.

2.4 Фреймворк ASP.NET Core MVC

Фреймворк ASP.NET Core MVC — це багатий фреймворк з відкритим вихідним кодом для створення веб-програм та API за допомогою шаблону проектування Model-View-Controller.

ASP.NET Core MVC забезпечує спосіб створення динамічних веб-сайтів на основі шаблонів, що дозволяє чітко розділяти проблеми. Він дає повний контроль над розміткою, підтримує TDD-дружню розробку та використовує новітні веб-стандарти.

2.4.1 Архітектурний шаблон MVC

Архітектурний шаблон Model-View-Controller (MVC) розділяє код веб-системи на три частини, пов’язані між собою: модель даних, вигляд та контролер. Основна ідея даного шаблону полягає у тому, щоб відокремити інтерфейс користувача (представлення) і дані (модель) таким чином, щоб робота з даними системи та вигляд її інтерфейсу мінімально залежали одне від одного.[31]

Цей шаблон допомагає досягти розділення проблем. Використовуючи цей шаблон, запити користувачів направляються до контролера, який відповідає за роботу з моделлю для виконання дій користувача та/або отримання результатів запитів. Контролер вибирає подання для відображення користувачеві та надає йому дані моделі, які йому потрібні.[32]

На рисунку 2.3 показано три основні компоненти та їх взаємодія між собою. Фактично, модель у шаблоні не залежить ні від вигляду, ні від контролера, а ті у свою чергу мають залежність від моделі даних. Такий поділ дозволяє розробляти та тестувати роботу з моделлю незалежно від її візуального представлення.

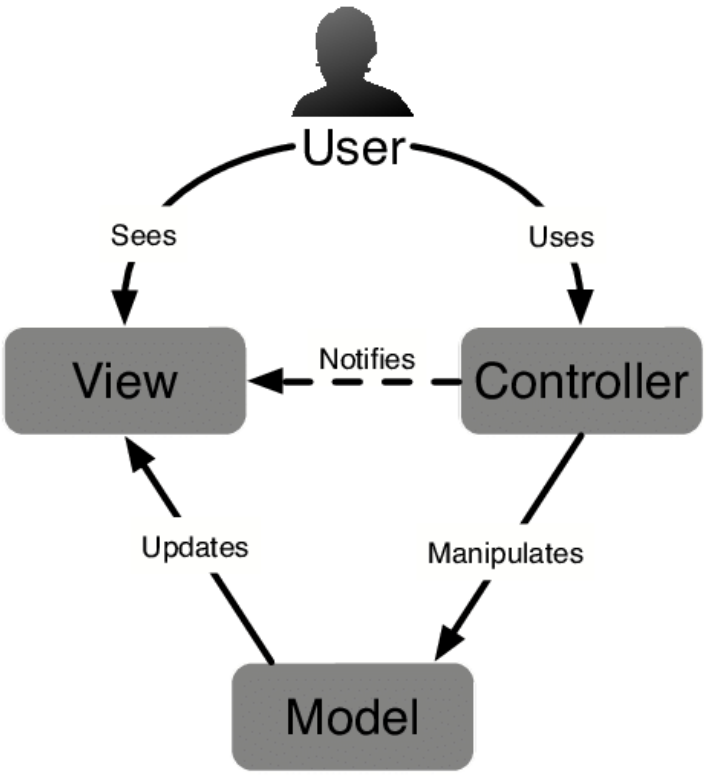

Рисунок 2.3 - Взаємодія компонентів

Призначення моделі [31]

Модель у MVC є представленням стану програми, її бізнес-логіки або операцій, що мають нею виконуватися. Бізнес-логіка має бути інкапсульованою у моделі разом з будь-якою логікою реалізації збереження стану програми. Строго типізовані подання часом використовують типи ViewModel, котрі призначені для відображення вмісту даних в цьому поданні.

Призначення представлення (інтерфейсу користувача) [31]

Views відповідають за представлення вмісту через інтерфейс користувача. Вони використовують механізм перегляду Razor для вбудовування коду .NET у розмітку HTML. У представленнях має бути закладено мінімум логіки, і будь-яка логіка в них має стосуватися відображення вмісту. Якщо модель є досить складною і потребує для відображення даних виконання великої кількісті додаткової логіки у файлах перегляду, є можливість скористатися View Component, ViewModel або шаблонами представлення, щоб спростити перегляд.

Призначення контролера [31]

Контролери — це компоненти, які обробляють взаємодію користувача, працюють з моделлю і, зрештою, обирають подання для візуалізації. У програмі

MVC подання лише відображає інформацію, а контролер у свою чергу обробляє та реагує на введення даних користувачем та взаємодію з ним. У шаблоні MVC контролер є початковою точкою входу і відповідає за вибір типів моделей для роботи та відображення (звідси його назва — він контролює, як програма реагує на заданий запит).

Таке розмежування обов’язків допомагає масштабувати програму з точки зору складності, оскільки легше кодувати, налагоджувати та тестувати щось (модель, перегляд чи контролер), що виконує якусь одну дію. Складніше оновлювати, тестувати та налагоджувати код, який має залежності, поширені на дві чи більше з цих трьох областей. [32]

Метою даного шаблону є розробка гнучкого дизайну програмного забезпечення, котрий полегшує подальше масштабування та зміни у програмі, а також уможливлює повторне використання коду у ній. У великих системах, крім того, використання MVC сприяє кращій впорядкованості структури та більшому її розумінню, оскільки зменшує загальну складність системи.

2.5 Платформа Docker

Docker – платформа із відкритим вихідним кодом, призначенням якої є автоматизація розробки застосунків у контейнерах. Розроблена компанією Docker Inc., що на початку існування мала назву dotCloud Inc., та реалізована під ліцензією Apache 2.0.[33]

Особливістю даної платформи є те, вона надає механізм розгортання застосунків поверх середовища виконання віртуалізованого контейнера. Docker забезпечує легке та швидке середовище, у якому код розробленої програми працюватиме однаково як на комп'ютері розробника, так і у тестовому середовищі, а згодом у продакшені. [34] Простими словами, Docker є програмним забезпеченням, яке надає можливість у певній ділянці пам'яті (контейнері) ізольовано встановити необхідну ОС (операційну систему), версію використовуваної мови програмування, налаштувати змінні оточення, встановити різні залежності і надавати доступ тільки за певних умов.

Docker складається з таких компонентів[34]:

- клієнт та сервер Docker;
- образи Docker (англ. Docker image);
- контейнери Docker;
- реєстри.

Docker є клієнт-серверним застосунком (рис. 2.4). Docker-клієнт надсилає запити докер-серверу або демону, котрий у свою чергу виконує всю потрібну роботу. Docker постачається з клієнтським двійковим файлом командного рядка та повним API, необхідним для комунікації клієнта та демона. Демон та клієнт Docker є можливість запускати в одній системі або під'єднувати локальний клієнт до віддаленого демона. Клієнтом Docker також може виступати Docker Compose, що дозволяє працювати з додатками, котрі складаються з більше ніж одного контейнера.

Образ Docker - це файл з набором команд, на підставі якого в подальшому буде запущений контейнер. [34] Його структура складається з шарів, кожен з яких додає окремий ряд потрібних інструкцій для роботи майбутніх контейнерів. Образи фактично є будівельними блоками, з яких складається життєвий цикл Docker, тому вони легкі, портативні, легко збережувані та оновлювані.

Реєстр – це місце, де зберігаються створені образи.[34] Вони можуть бути публічними, яким є наприклад Docker Hub – етакий гітхаб для докер-зображень, або приватними.

Контейнер – це екземпляр образу, в якому може працювати створена система ізольовано від інших процесів відповідно до вказаних у файлі образу інструкцій. Фактично він є запущеним образом і може містити один або більше виконуваних процесів.[34]

Використання Docker дозволяє швидше і ефективніше доставляти або переміщувати код, стандартизує операції, що виконуються додатками, і в цілому економить засоби, оптимізуючи використання ресурсів. Завдяки Docker користувачі одержують об'єкт, який з високою надійністю можна запускати на будь-якій платформі. Простий і зрозумілий синтаксис Docker забезпечує повний контроль над операціями, що виконуються.

Основні переваги використання Docker:

- доставка ізольованих послуг з необхідною періодичністю;
- спрощення процесу розгортання, виявлення проблем та відкату для їх усунення до попередніх версій;
- можливість ефективно переносити розроблені програми з локальних машин, на яких ведеться розробка, до робочого середовища;
- підвищення ефективності використання ресурсів.

Також варто згадати і про його недоліки:

- при збільшенні масштабованості та навантаження необхідне дуже чітке та якісне налаштування систем;
- обмежена зворотна сумісність за деякими напрямками;
- процес видалення контейнерів потребує велику кількість часу та зусиль;
- контейнеризація є надбудовою над ОС, котра ускладнює реалізацію системи та збільшує навантаження на неї.

Docker спрощує складання та запуск розподілених мікросервісних архітектур, розгортання коду за допомогою стандартизованих конвеєрів безперервної інтеграції та доставки, створення високомасштабованих систем обробки даних та повністю керованих платформ для розробників.

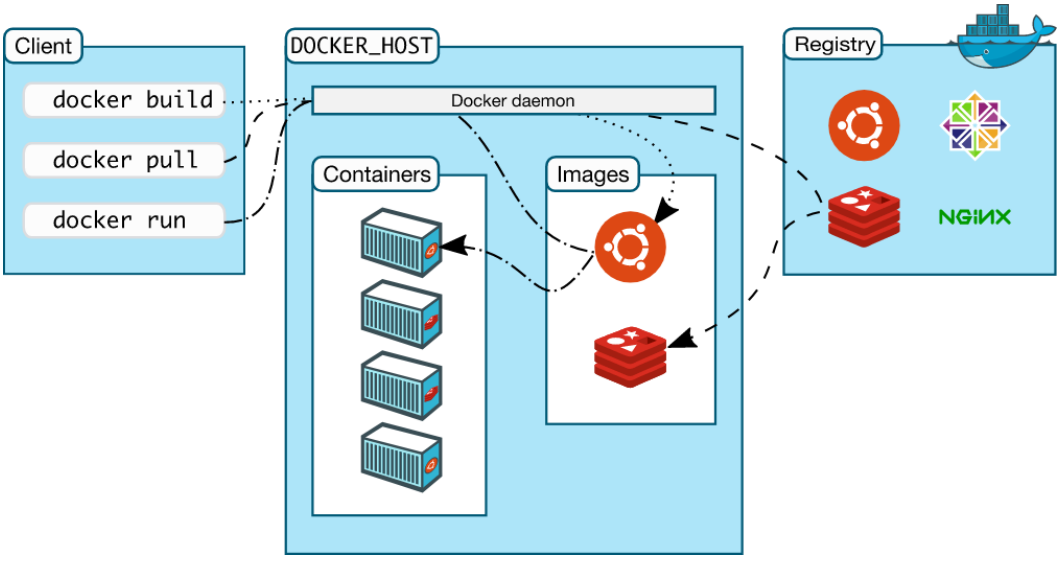

Рисунок 2.4 – Схема архітектури платформи Docker [33]

Контейнер стає блоком для розповсюдження та тестування користувацької програми, і після завершення розробки застосунку, система розгорнута у виробничому середовищі, як контейнер і організована служба.

2.6 Платформи для інтеграції та доставки веб-системи до користувача

2.6.1 Практика CI/CD

CI/CD - це одна з DevOps практик, яка дозволяє розробникам частіше і надійніше розгортати зміни ПЗ, звести до мінімуму помилки, підвищити темпи збірки та якість продукту, що розробляється. Є комбінацією continuous integration і continuous delivery. Основний зміст даної практики можна побачити на рисунку 2.5.

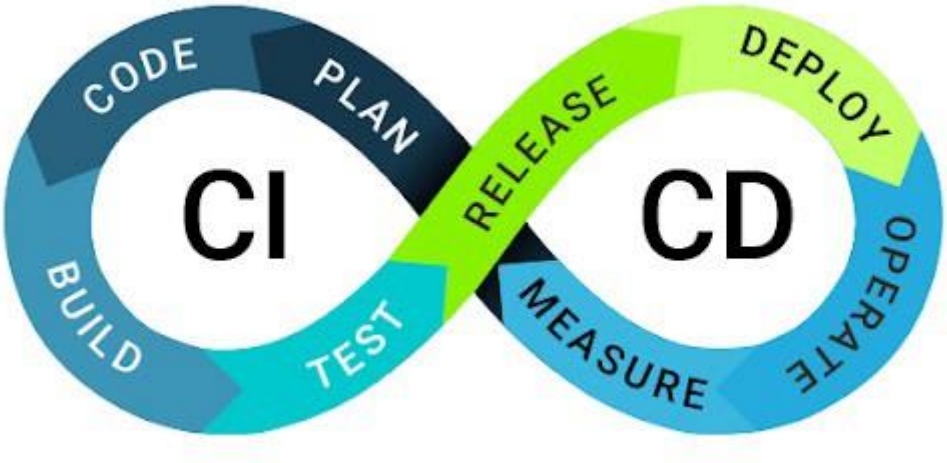

Рисунок 2.5 – Відображення змісту поняття CI/CD [35]

Безперервна інтеграція (CI) – процес постійної розробки програмного забезпечення з інтеграцією в основну гілку. Автоматично збирає софт, тестує його та сповіщає, якщо щось йде не так. [35]

Безперервна доставка (CD) – процес постійної доставки програмного забезпечення до споживача. Забезпечує розробку проекту невеликими частинами та гарантує, що його можна віддати в реліз у будь-який час без додаткових ручних перевірок. [35]

Цикл CI/CD виглядає таким чином [35]:

1) Розробник пише код, проводить початкове тестування, щоб не було помилок, і фіксує зміни у своїй робочій гілці. Потім з'єднує

модифікований код із робочим кодом з основної гілки.

2) Система, обрана інструментом CI, помічає, що у коді відбулися зміни та запускає автоматичне складання та автоматичне тестування програми.
3) Якщо автоматичне тестування пройшло успішно, програмне забезпечення віддається для ручного тестування команді тестувальників.
4) Після виправлення недоліків, виявлених під час ручного тестування, запускається автоматизована установка програмного забезпечення на серверах компанії.
5) Здійснюється підтримка нової версії програми та її моніторинг.
6) Збираються запити виправлення недоліків і багів, розробник вносить зміни у код, і процес повторюється.

Основна перевага CI/CD у тому, що розробнику потрібно тільки написати код, а решта процесів з тестування, збирання та доставки проходять автоматично.

2.6.2 Платформа CircleCI

CircleCI — це платформа безперервної інтеграції та доставки, яка допомагає командам розробників швидко випускати код і автоматизувати збірку, тестування та розгортання. CircleCI можна налаштувати для ефективного запуску дуже складних конвеєрів із кешуванням рівня докерів, класами ресурсів тощо. Після того, як репозиторії на GitHub або Bitbucket авторизовані та додані як проект до circleci.com, кожен код ініціює виконання завдань CircleCI. [36] CircleCI також надсилає повідомлення електронною поштою про успіх або невдачу після завершення тестів. Роботу даної платформи можна побачити на рис. 2.6.

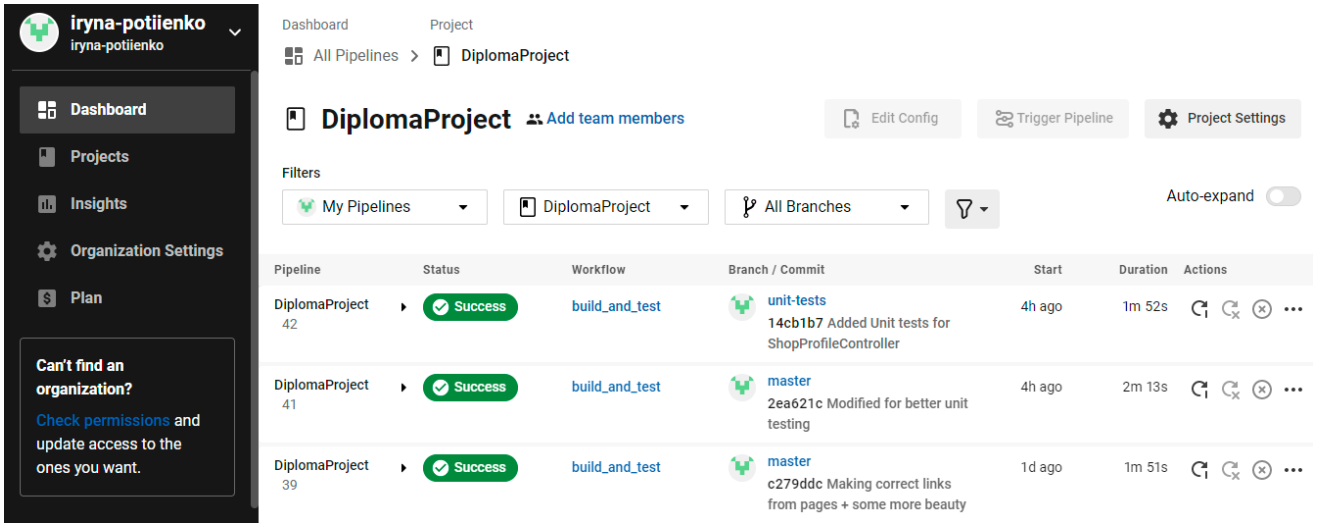

Рисунок 2.6 – Платформа CircleCI

2.6.3 Сервіси платформи Amazon

Amazon Web Services (AWS) — це комплексна платформа хмарних обчислень, яка включає пропозиції інфраструктури як послуги (IaaS) та платформи як послуги (PaaS). Служби AWS пропонують масштабовані рішення для обчислень, зберігання даних, баз даних, аналітики тощо.[37]

Amazon Elastic Container Service (Amazon ECS) — це масштабований та швидкий сервіс керування контейнерами у кластерах. У Amazon ECS контейнери визначаються у завданні, яке використовується для виконання окремого завдання або завдання у сервісі. У цьому контексті сервіс — це конфігурація, яку можна використовувати для одночасного виконання та підтримки певної кількості завдань у кластері. Є можливим запускати свої завдання та послуги, використовуючи безсерверну інфраструктуру, якою керує AWS Fargate. Крім того, для більшого контролю над інфраструктурою можна запускати свої завдання та служби на кластері екземплярів Amazon EC2.[37]

Amazon Elastic Compute Cloud (Amazon EC2) забезпечує масштабовані обчислювальні потужності у хмарі AWS. Використання Amazon EC2 надає можливість швидше розробляти та розгортати програми, оскільки немає необхідності інвестувати в апаратне забезпечення. Даний сервіс може використовуватися, щоб запустити стільки віртуальних серверів, скільки потрібно, налаштувати безпеку та мережу, а також керувати сховищем. Amazon EC2 дає змогу збільшувати або зменшувати масштаб, аби впоратися зі змінами

вимог або збільшенням кількості відвідувань системи, зменшуючи потребу в прогнозуванні трафіку.[37]

Служба реляційної бази даних Amazon (Amazon RDS) — це веб-сервіс, який полегшує налаштування, роботу та масштабування реляційної бази даних у хмарі AWS. Він забезпечує економну потужність із можливістю зміни розміру для стандартної реляційної бази даних і керує загальними задачами адміністрування бази даних. [37]

Amazon Elastic Container Registry (ECR) — це повністю керований реєстр контейнерів, який дозволяє легко зберігати, керувати, ділитися та розгортати зображення та артефакти контейнерів у будь-якому місці.[37]

3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Загальні вимоги до інформаційної системи

Для зручної та стабільної роботи користувачів до веб-орієнованої платформи для продажу крафтових продуктів «Handmade-space» висувається ряд вимог.

Дана веб-платформа має забезпечувати:

- можливість продавцям створювати та ефективно працювати із сторінками профілів власних магазинів;
- можливість власнику додавати нові товари до профілю свого магазину та ефективно їх просувати у загальних каталогах системи;
- зручну систему замовлень користувачами товарів у магазинах та збору відгуків до них;
- можливість знаходити найближчі за розташуванням магазини на карті;
- зручну історію замовлень користувача та магазину;
- можливість користувачу зберігати бажаний для майбутньої покупки товар у його списку бажань;
- можливість залишати коментарі до магазинів та товарів авторизованим користувачам.

Перш ніж почати власне проектування бази даних даної веб-платформи, необхідно визначити діапазон її основних завдань та їх реалізацію у вигляді запропонованих користувачам можливостей у залежності від їх прав доступу.

3.2 Функціональні можливості користувачів системи «Handmade-space»

У веб-орієнтованій платформі для продажу крафтових продуктів «Handmade-space» для забезпечення роботи основного функціоналу системи та виконання поставлених вимог передбачається наявність двох основних ролей для користувачів – покупець та продавець. Розмежування прав доступу користувачів у системі здійснюється шляхом обмежень на перегляд та редагування певної інформації для певної групи користувачів в залежності від їх ролей.

Загалом у веб-орієнтованій системі для продажу крафтових продуктів «Handmade-space» функціонують такі групи за ролями користувачів:

1) «Користувач-гість»;
2) «Користувач-покупець»;
3) «Користувач-продавець»;
4) «Користувач-адміністратор».

Користувач із роллю «гість» – користувач-гість – має права користуватися лише загальними можливостями роботи з веб-платформою, а саме можливостями перегляду магазинів та товарів у них. Даний користувач отримує такі права відразу при вході на веб-сайт системи.

Користувач-покупець має більш широкі права порівняно з попередньою роллю. Він може оформити замовлення обраного товару та домовитись із користувачем-продавцем про його отримання, використовуючи внутрішню систему платформи. Коли замовлення успішно виконано та клієнт отримав покупку, йому рекомендується залишити відгук на платформі про дане замовлення, що відповідно вплине на рейтинг цього магазину на платформі. Також у профілі користувачів буде зберігатися інформація про замовлення, що виконані та що знаходяться в процесі виконання. Для отримання переваг даного типу користувача потрібно пройти реєстрацію, або, якщо аккаунт у користувача вже існує – процедуру авторизації.

Користувач-продавець має ще більш розширені можливості відносно користувача-покупця. В його основні можливості входить ведення сторінки профілю свого магазину, куди він може публікувати інформацію про себе та свій товар, яка включає в себе фотографії, ціну товару, терміни, у які це замовлення може бути виконане, склад товару тощо. Також він отримує замовлення від покупців та за потреби домовляється про терміни виготовлення, отримання товару тощо, використовуючи внутрішню систему платформи. Інформація про виконані замовлення та замовлення в процесі виконання доступні у вигляді списків у профілі кожного магазину. На додаток, для кожного магазину доступною є аналітика успішності виконання замовлень, продажів та загалом

його роботи на рівні із загальною аналітикою щодо всіх магазинів платформи. А також розміщення магазину можна дізнатися, знайшовши його на карті.

Більш того, на кожній такій сторінці доступна інформація про загальну кількість виконаних продавцем замовлень, скільки з них були успішними (вдалими, від покупця не було скарг) тощо. На основі цих даних формуватиметься певний рейтинг задля полегшення пошуку покупцями товарів від різних постачальників.

Наступна і остання роль користувача, котра надає адміністраторські права та розширює можливості користувача-покупця – користувач-адміністратор. Дана роль необхідна для того, аби підтримувати коректну роботу платформи. В обов'язки користувача-адміністратора також входить верифікація профілю магазину. Даний тип користувацького аккаунту не можна отримати, пройшовши процедуру реєстрації, він призначається тільки власниками веб-сервісу.

Якщо говорити трохи детальніше про замовлення та його оплату, то воно виглядає так:

Клієнт замовляє товар у продавця, оформлюючи потрібну анкету на сайті. Продавець у свою чергу переглядає цю анкету та в залежності від того, чи прийнятними на даний момент є умови замовлення, має можливість це замовлення підтвердити, скасувати або узгодити з користувачем його деталі. У разі узгодження та підтвердження продавцем замовлення покупцю надходить сформований чек на оплату товару, де описані усі домовленості щодо деталей самого товару, його доставки, ціни та інші коментарі. Після цього замовлення відправляється в роботу, а користувач отримує можливість слідкувати за ходом його виконання. Узагальнений процес замовлення товару можна побачити на діаграмі активності (рис. 3.1).

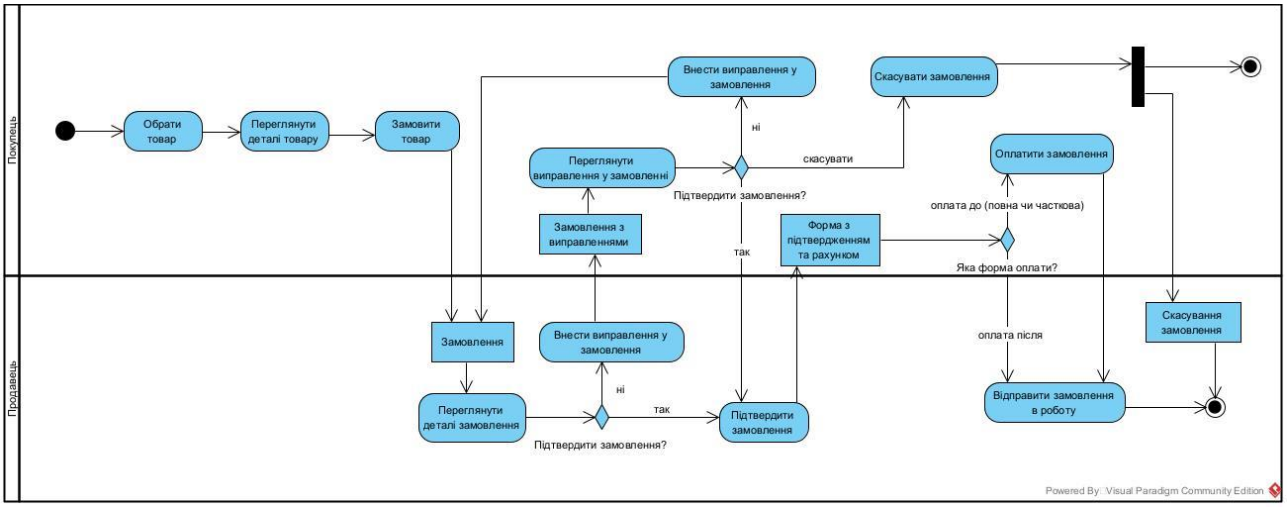

Рисунок 3.1 – Діаграма активності процесу замовлення товару

Також на сайті є можливість зробити замовлення одразу з доставкою через нову пошту або інші аналогічні сервіси, якщо вона передбачена та ними не забороняється (нова пошта, наприклад, не доставляє товари, що швидко псуються).

У разі невдоволення споживача замовленням йому надаються детальні контакти для зворотного зв’язку. Якщо виконавець неякісно виконав замовлення чи не у встановлений час, кошти повернуться споживачу у повному обсязі через банківський переказ.

Функціональні можливості усіх користувачів в залежності від їх ролі у веб-орієнтованій системі для продажу крафтових продуктів «Handmade-space» зображені на рис. А.1 у додатку А у вигляді діаграми прецедентів.

Після визначення функціональних можливостей користувачів в залежності від їх ролей у веб-орієнтованій системі «Handmade-space», можна перейти до проектування бази даних даної системи.

3.3 Проектування бази даних веб-орієнтованої платформи «Handmade-space»

Веб-орієнтована платформа, що проектується, реалізує клієнт-серверну архітектуру та передбачає, що БД буде розміщена на віддаленому (remote) сервері. З браузера користувача система надсилатиме запити до серверної частини програми, що у свою чергу буде розміщена на віддаленому сервері AWS.

Після визначення у пункті 3.2 функціональних вимог до веб-платформи «Handmade-space» спроектована база даних, що реалізує реляційну модель даних. При розробці системи за допомогою СУБД MySQL створена база даних та визначено основні сутності проектованої системи.

3.3.1 Опис таблиць бази даних веб-платформи «Handmade-space»

Для бази даних даної веб-платформи визначені такі 15 сутностей:

1) користувач;
2) роль користувача;
3) профіль магазину;
4) товар магазину;
5) категорія товару;
6) підкатегорія товару;
7) замовлення;
8) товар у замовленні;
9) кошик;
10) відгук користувача про замовлення;
11) коментар до товару;
12) коментар до магазину;
13) бажаний для майбутньої покупки користувачем товар;
14) спосіб доставки замовлення;
15) стадія готовності замовлення.

Кожна сутність містить первинний ключ, призначений для ідентифікації унікального екземпляра сутності. Він існує для кожного типу сутності у вигляді додаткового атрибута id. Даний атрибут зберігає унікальний номер екземпляра сутності у системі, котрий автоматично генерується СУБД при його створенні та додаванні в БД. Застосування таких унікальних ключів полегшує процес індексації та пошуку потрібних даних у БД.

Загальний список сутностей бази даних веб-орієнтованої платформи для

продажу крафтових продуктів наведений у табл. 3.1.

Таблиця 3.1 – Список сутностей БД веб-платформи «Handmade-space»

User	Користувачі системи
Role	Ролі користувачів системи
ShopProfile	Профілі магазинів
Product	Товари у магазині
Category	Категорії товарів у магазинах
Subcategory	Підкатегорії товарів у магазинах
Order	Замовлення користувача у магазині
ProductInOrder	Товари у замовленні користувача
Cart	Кошик товарів магазину
OrderFeedback	Відгук користувача про замовлення
ProductComment	Коментарі до товару магазину
ShopComment	Коментарі до магазину
LikedProductsByUsers	Бажані користувачем товари для майбутніх покупок (список бажань);
DeliveryType	Способи доставки замовлення
ReadyStage	Стадії готовності замовлення

Неключові стовпці в усіх таблицях залежать тільки від первинного ключа з унікальним номером екземпляру сутності, сама наявність якого гарантує виконання умов першої та другої нормальної форми. Тому можна стверджувати, що відношення у даній базі даних знаходяться у третій нормальній формі.

Схема бази даних веб-орієнтованої системи для продажу крафтових продуктів «Handmade-space» в цілому зображена на рис. Б.1 у додатку Б у вигляді діаграми бази даних.

3.4 Схема роботи веб-орієнтованої платформи «Handmade-space»

Дана веб-платформа для продажу крафтових товарів «Handmade-space» реалізує архітектуру «клієнт-сервер» через шаблон Model-View-Controller

(MVC) для розробки програмного коду системи. Він розподіляє роботу між клієнтською та серверною частиною системи через такі компоненти, як модель даних, представлення інтерфейсу користувача та модель керування.

ASP.NET Core MVC забезпечує спосіб створення динамічних веб-сайтів на основі шаблонів, що дозволяє чітко розділяти проблеми. Він дає повний контроль над розміткою, підтримує TDD-дружню розробку та використовує новітні веб-стандарти.

На рисунку 3.2 можна побачити реалізацію шаблона MVC у вигляді діаграми класів на прикладі роботи з сутністю ShopProfile. Робота з іншими сутностями системи проводиться аналогічно.

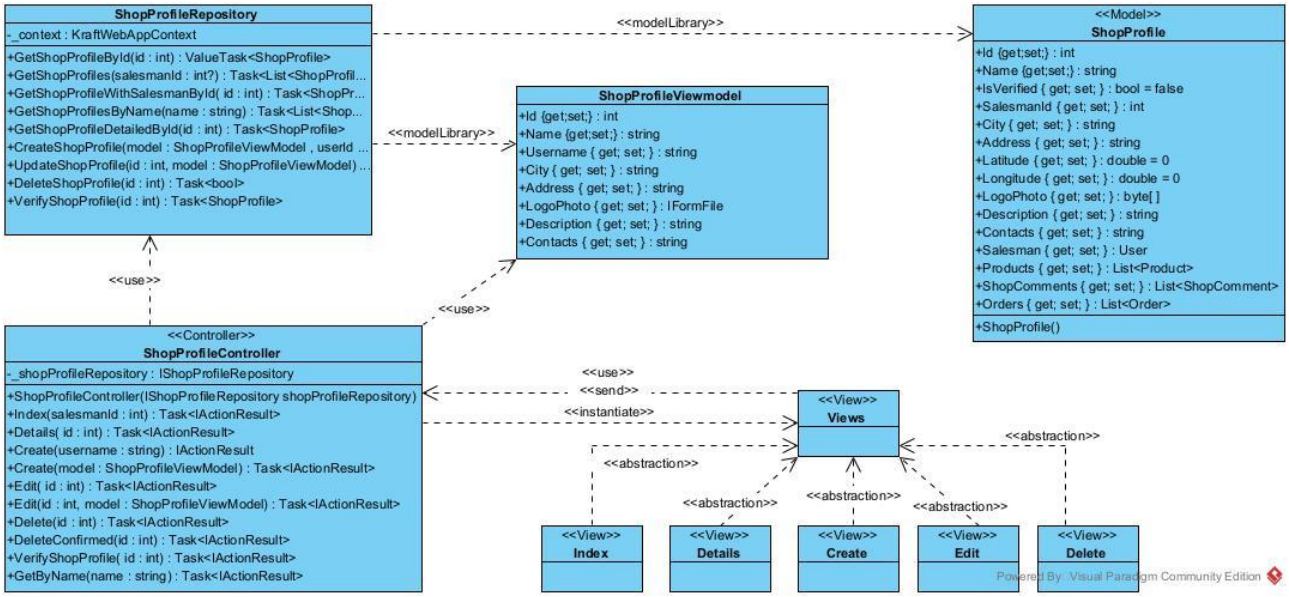

Рисунок 3.2 – Реалізація шаблона MVC у вигляді діаграми класів

3.4.1 Моделі даних веб-платформи

База даних даної веб-платформи є реляційною та складається з таблиць. Ці таблиці відображаються в коді сутностями (enteties), які є моделями даних, котрі зберігаються та використовуються. Код сутності наведено на рис. 3.3.

```
public class ShopProfile
{
    4 usages  Iryna Potiienko
    public ShopProfile()
    {
        Products = new List<Product>();
        ShopComments = new List<ShopComment>();
        Orders = new List<Order>();
    }
    29 usages
    public int Id { get; set; }
    18 usages
    public string Name { get; set; }
    4 usages
    public int SalesmanId { get; set; }
    9 usages
    public bool IsVerified { get; set; }

    9 usages
    public string City { get; set; }
    9 usages
    public string Address { get; set; }
    5 usages
    public int Latitude { get; set; }
    5 usages
    public int Longitude { get; set; }
```

Рисунок 3.3 – Код сутності

Усі сутності зв’язуються разом через контекст даних, зображений на рис. 3.4.

```
public class KraftWebAppContext: DbContext
{
    14 usages
    public DbSet<User> Users { get; set; }
    14 usages
    public DbSet<ShopProfile> ShopProfiles { get; set; }
    11 usages
    public DbSet<Product> Products { get; set; }
    19 usages
    public DbSet<Order> Orders { get; set; }

    7 usages
    public DbSet<OrderFeedback> OrderFeedbacks { get; set; }
    10 usages
    public DbSet<ProductInOrder> ProductsInOrder { get; set; }
    public DbSet<LikedProductsByUsers> LikedProductsByUsers { get; set; }
    public DbSet<ShopComment> ShopComments { get; set; }
    public DbSet<ProductComment> ProductComments { get; set; }
    4 usages
    public DbSet<Cart> Carts { get; set; }
    public DbSet<Category> Categories { get; set; }
    public DbSet<Subcategory> Subcategories { get; set; }
```

Рисунок 3.4 – Контекст даних

3.4.2 Контролери веб-системи

ClassController – це контролер, на який приходять запити. Він є основним і ключовим елементом серверної частини системи. В залежності від типу запиту, який надійшов, GET чи POST, будуть виконуватись різні методи контролера.

Метод контролера повертає код сторінки, на яку буде переадресовано клієнта, або частину сторінки при використанні технології AJAX у конкретній частині веб-платформи.

Код контролера зображений на рис. 3.5.

```
public class ShopProfileController : Controller
{
    private readonly KraftWebAppContext _context;

    Iryna Potiienko
    public ShopProfileController(KraftWebAppContext context)
    {
        _context = context;
    }

    6 usages  Iryna Potiienko
    public async Task<IActionResult> Index(int? salesmanId){...}
    2 usages  Iryna Potiienko
    public async Task<IActionResult> Details(int? id){...}
    Iryna Potiienko
    public IActionResult Create(){...}

    [HttpPost]
    [ValidateAntiForgeryToken]
    Iryna Potiienko
    public async Task<IActionResult> Create([Bind( params include: "Id,Name,
```

Рисунок 3.5 – Реалізація контролера

Контролер використовує методи класу репозиторію, котрий у свою чергу виконує запити до бази даних. Інтерфейс репозиторію зображений на рис. 3.6.

```
8 usages  1 inheritor  Iryna Potiienko
public interface IShopProfileRepository
{
    2 usages  1 implementation  Iryna Potiienko
    public Task<List<ShopProfile>> GetShopProfiles(int? salesmanId);
    1 usage  1 implementation  Iryna Potiienko
    public ValueTask<ShopProfile?> GetShopProfileById(int id);
    1 usage  1 implementation  Iryna Potiienko
    public Task<ShopProfile> GetShopProfileWithSalesmanById(int id);
    1 usage  1 implementation  Iryna Potiienko
    public Task<List<ShopProfile>> GetShopProfilesByName(string name);
    3 usages  1 implementation  Iryna Potiienko
    public Task<ShopProfile> GetShopProfileDetailedById(int id);
    3 usages  1 implementation  Iryna Potiienko
    public Task<bool> CreateShopProfile(ShopProfileViewModel model, int userId);
    3 usages  1 implementation  Iryna Potiienko
    public Task<ShopProfile?> UpdateShopProfile(int id, ShopProfileViewModel model);
    3 usages  1 implementation  Iryna Potiienko
    public Task<bool> DeleteShopProfile(int id);
    1 usage  1 implementation  Iryna Potiienko
    public Task<ShopProfile> VerifyShopProfile(int id);
}
```

Рисунок 3.6 – Реалізація репозиторію

3.4.3 Представлення сторінок веб-системи

За вигляд веб-сторінок на боці клієнта відповідають представлення, файли

яких розташовані у теці Views, котра має такі структурні частини:

- окремі теки з певним набором представлень сторінок для кожної сутності моделі;
- тека Shared, де розташовані компоненти представлень (View Component) та інші допоміжні елементи HTML-сторінок.

Структура клієнтської частини системи зображена на рис. 3.7.

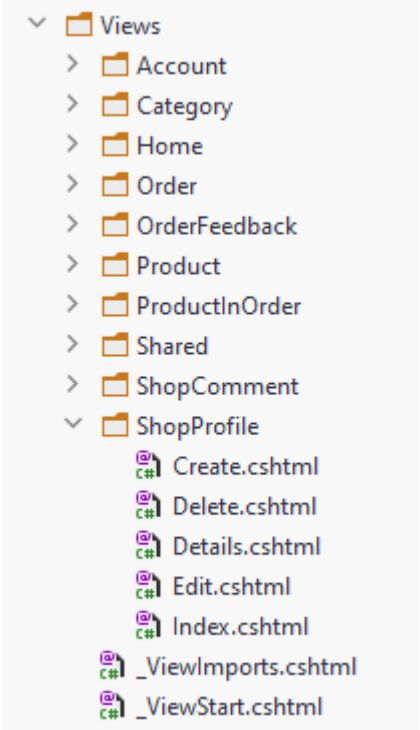

Рисунок 3.7 – Організація теки клієнтської частини

3.5 Реалізація структури інтерфейсу користувача веб-платформи

Для розробки дизайну та організації розміщення елементів на сторінці використані принципи, визначені в [38], а саме: проста інтуїтивно зрозуміла навігація, центральне вирівнювання та читабельний текст, приваблива кольорова гама з виділенням яскравими кольорами потрібних областей, які вирізняють сторінку з поміж інших в браузері.

Загалом, сторінки відрізняються одна від одної, але створені за одним шаблоном (рис. 3.8), котрий складається з таких елементів: верхнє меню навігації веб-сайту, основний контент сторінки та нижній колонтитул (footer). Заголовок системи винесений у ліву частину панелі головного меню веб-сторінки.

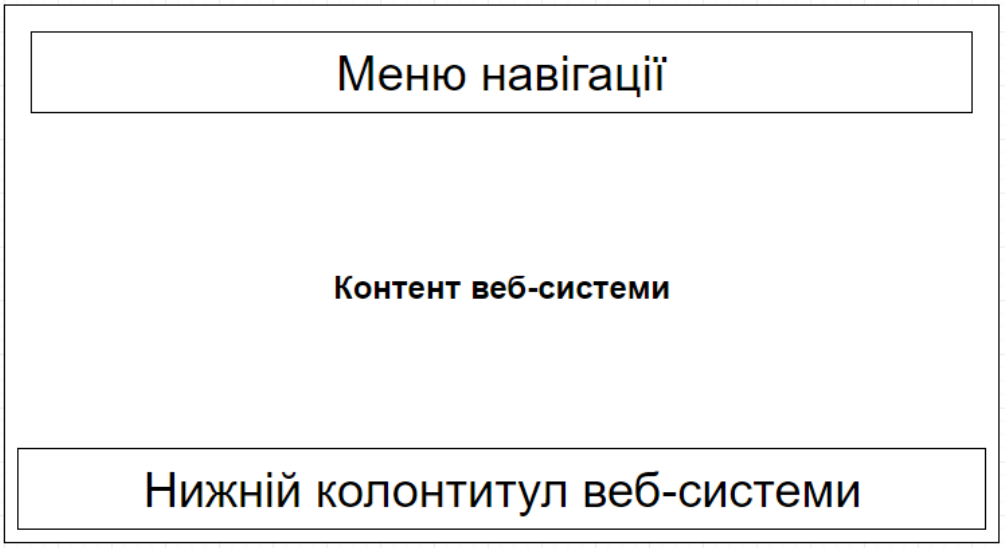

Рисунок 3.8 – Шаблон сторінок веб-платформи «Handmade-space»

Для зручної роботи користувача у даній веб-системі необхідною є наявність стандартизованого меню навігації, яке для зручності розміщене у верхній частині екрану та дозволяє перейти до таких розділів веб-системи:

1) головна сторінка;
2) каталог товарів;
3) каталог магазинів;
4) власний магазин (якщо він є);
5) власний кошик;
6) список замовлень користувача;
7) карта із найближчими магазинами.

Головна сторінка веб-платформи зображена на рис. 3.9. Її контентом служать товари магазинів, які були останніми додані до системи (новинки), та товари, які є найбільш популярними на платформі.

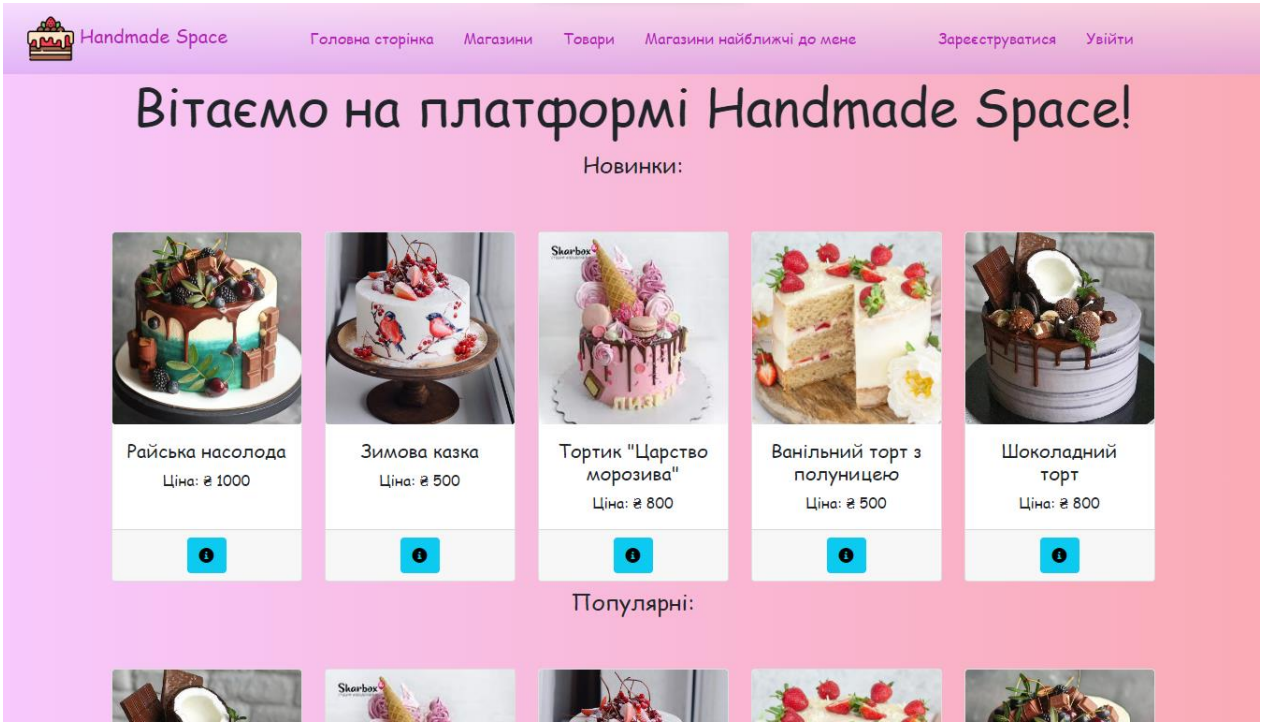

Рисунок 3.9 – Головна сторінка веб-орієнтованої системи «Handmade-space»

Для отримання можливостей користувача-покупця користувачу потрібно увійти до системи, що можливо зробити, натиснувши «Увійти» та перейшовши на сторінку авторизації, зображену на рис. 3.10. Якщо користувач ще не має власного аккаунту у системі, він може створити його, перейшовши на сторінку реєстрації, зображену на рис. 3.11.

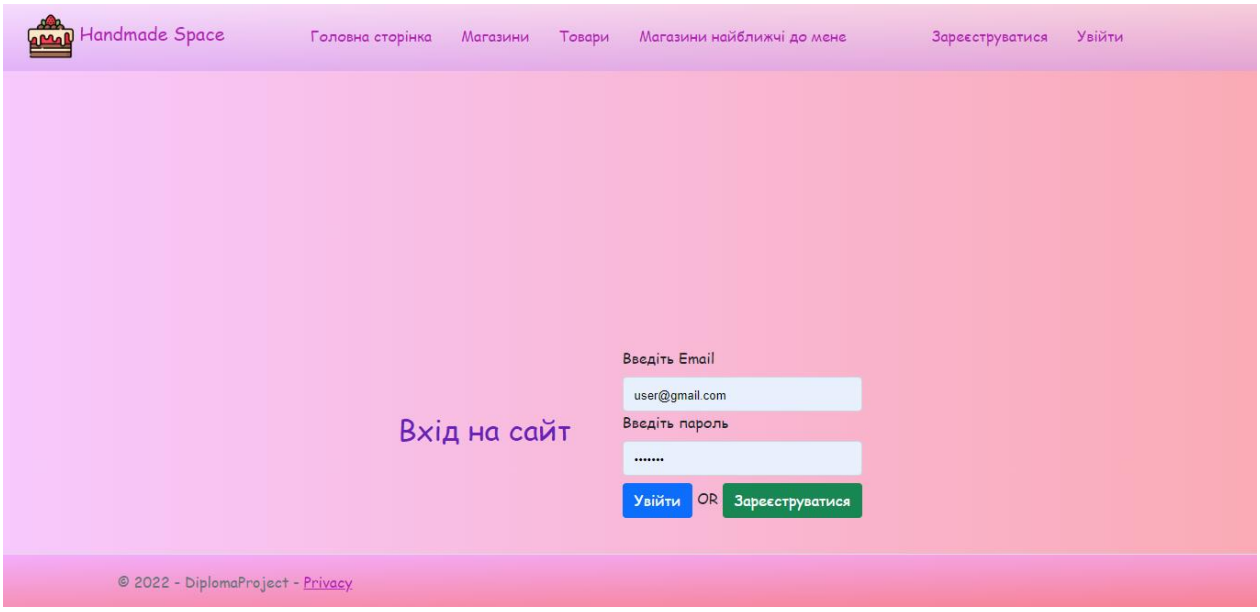

Рисунок 3.10 – Сторінка авторизації

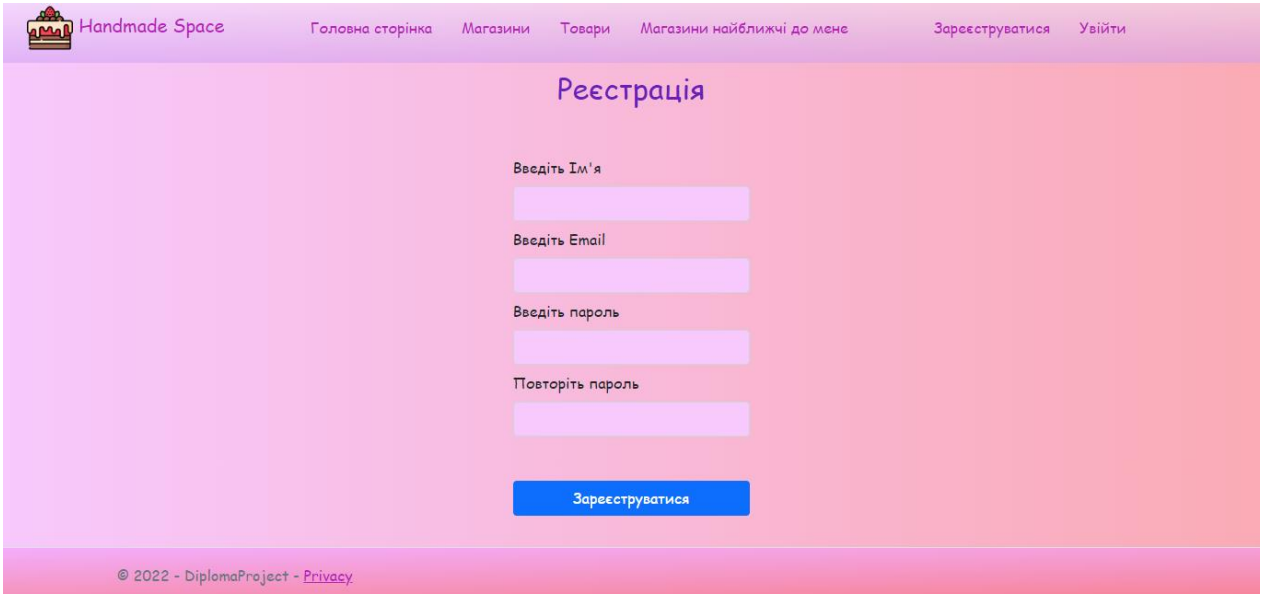

Рисунок 3.11 – Сторінка реєстрації

Після виконання реєстрації та авторизації у системі користувач бачить головну сторінку у вигляді, зображеному на рис 3.12.

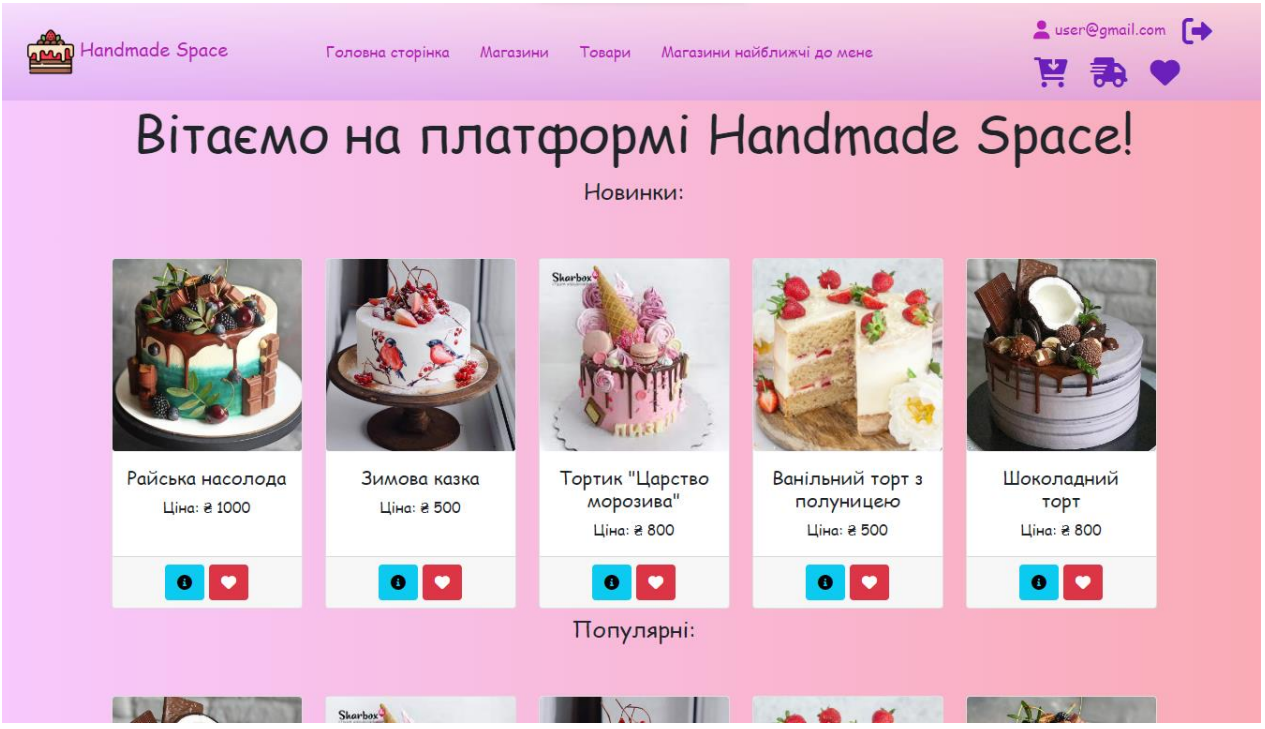

Рисунок 3.12 – Головна сторінка авторизованого користувача

Натиснувши «Магазини» на панелі головного меню користувач може побачити список зареєстрованих у системі магазинів. Ця сторінка зображена на рис. 3.13.

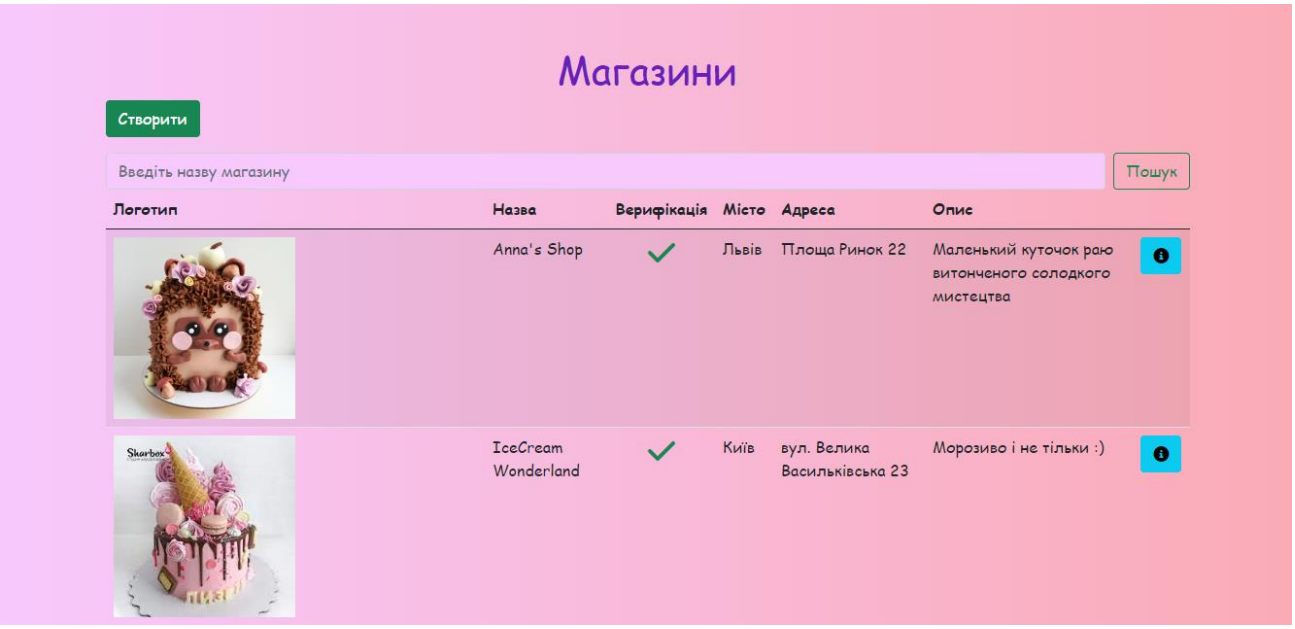

Рисунок 3.13 – Сторінка списку магазинів

Натиснувши на блакитну клавішу інформації про деталі магазину та перейшовши на сторінку, зображену на рис. 3.14, можна побачити детальну інформацію про нього, включаючи коментарі до магазину (рис. 3.15) та аналітику замовлень, що зображена на рис. 3.16. Побачити дані діаграми можливо лише у випадку, якщо даний користувач є власником цього магазину або адміністратором платформи.

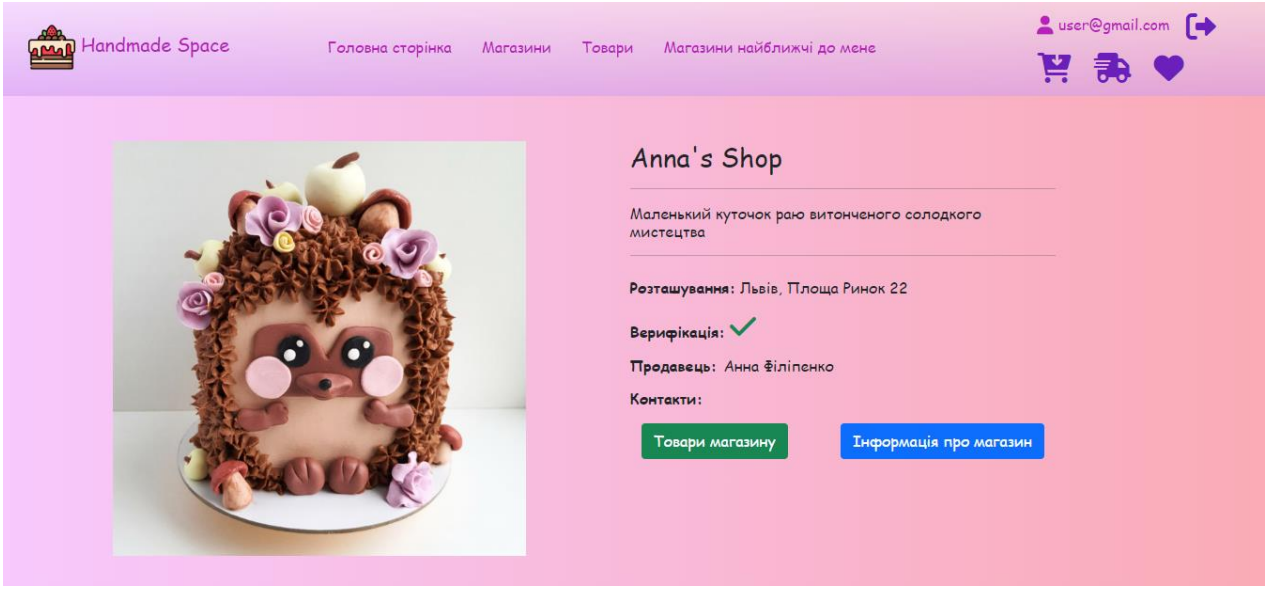

Рисунок 3.14 – Сторінка профілю магазину

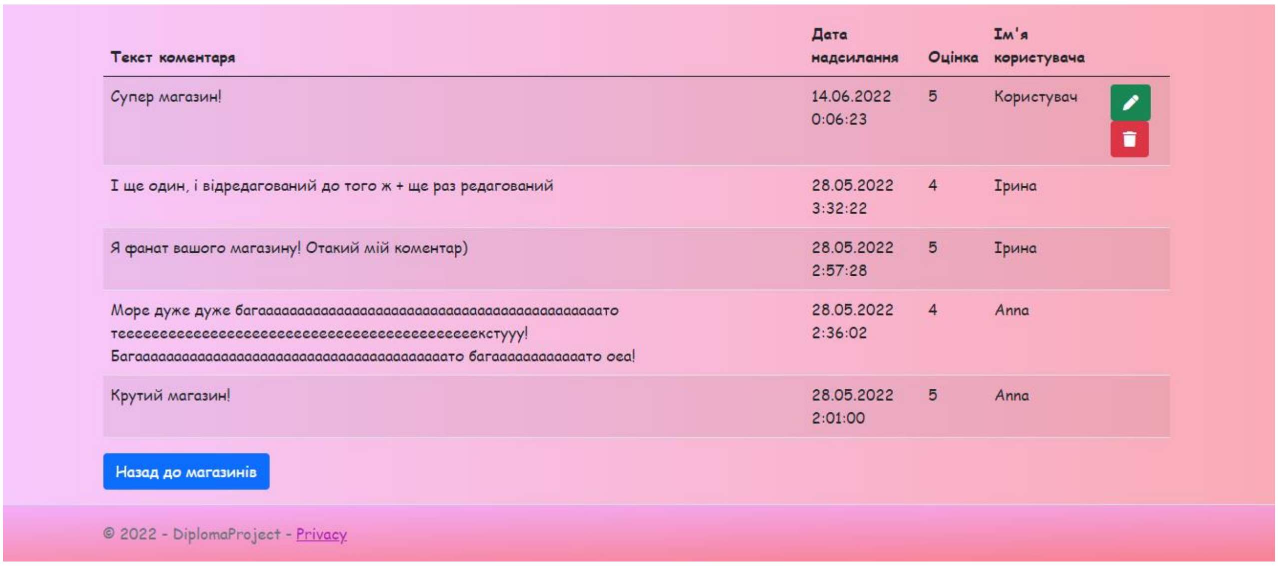

Рисунок 3.15 – Коментарі користувачів на сторінці магазину

Рисунок 3.16 – Аналітика замовлень магазину

Натиснувши на клавішу «Товари магазину», користувач може перейти на сторінку зі списком товарів цього магазину, що зображена на рис. 3.17. Також, можна побачити стрічку пошуку магазину за назвою та меню пошуку за категоріями та підкатегоріями товарів зліва.

Власник магазину має змогу додати новий товар, натиснувши кнопку «Додати товар» на сторінці магазину та перейшовши на відповідну сторінку додавання товару, зображену на рис. 3.18, а також за бажанням відредагувати товар, натиснувши на відповідну клавішу.

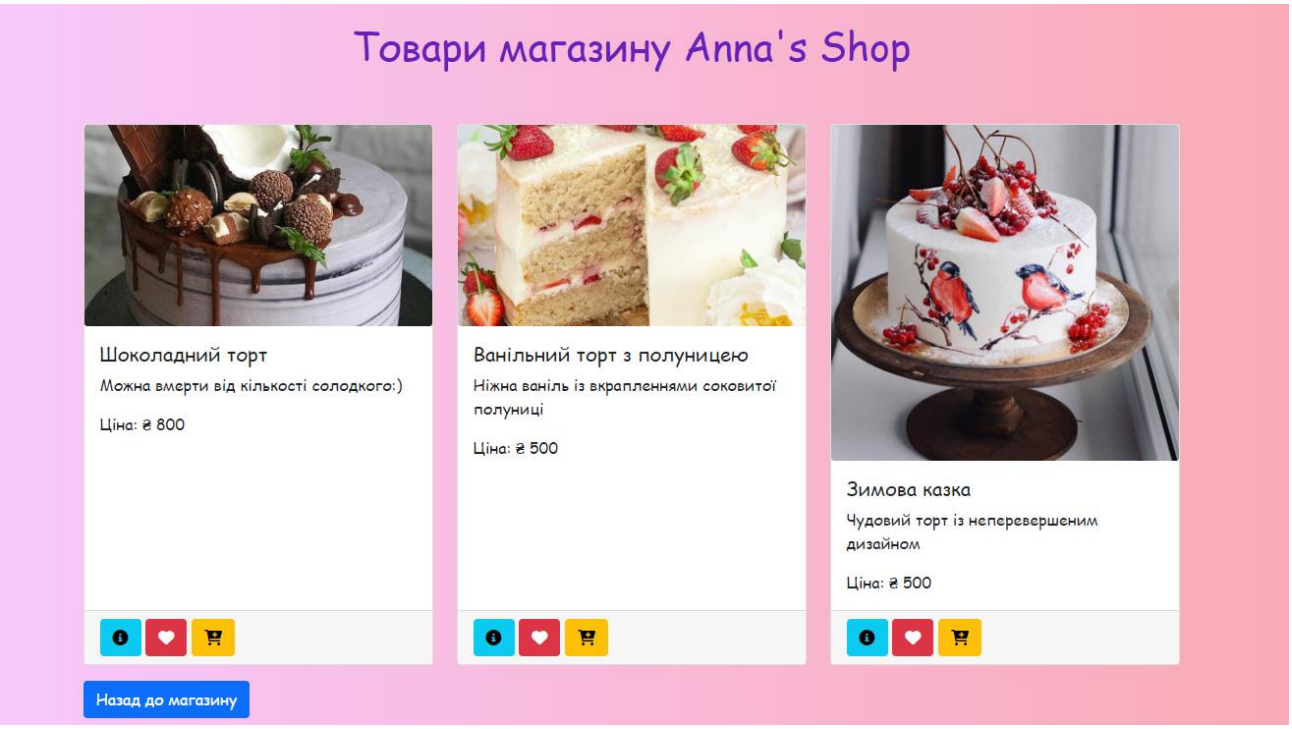

Рисунок 3.17 – Сторінка списку товарів магазину

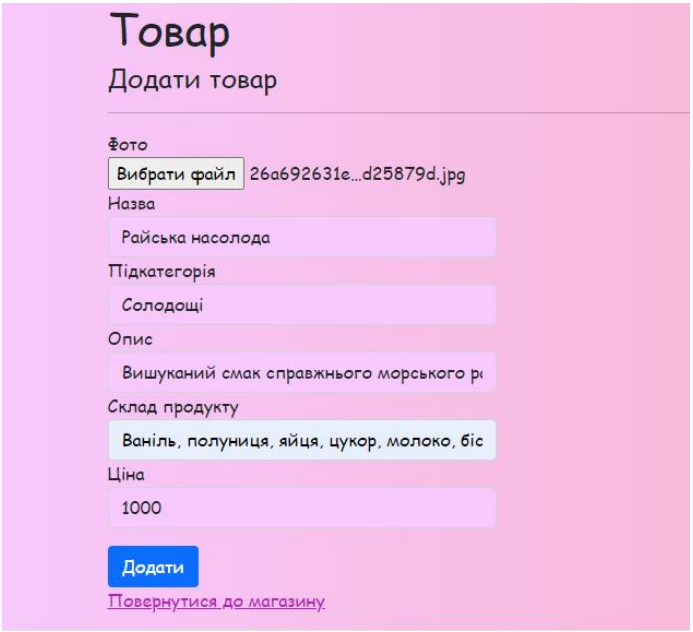

Рисунок 3.18 – Сторінка додавання товару у список товарів магазину

Натиснувши клавішу інформації про товар користувач може побачити сторінку з детальною інформацією про товар, що зображена на рис. 3.19, або додати товар у список бажань, натиснувши відповідну кнопку. Також користувач може замовити товар, натиснувши відповідну кнопку та увівши потрібну інформацію у форму, зображену на рис. 3.20.

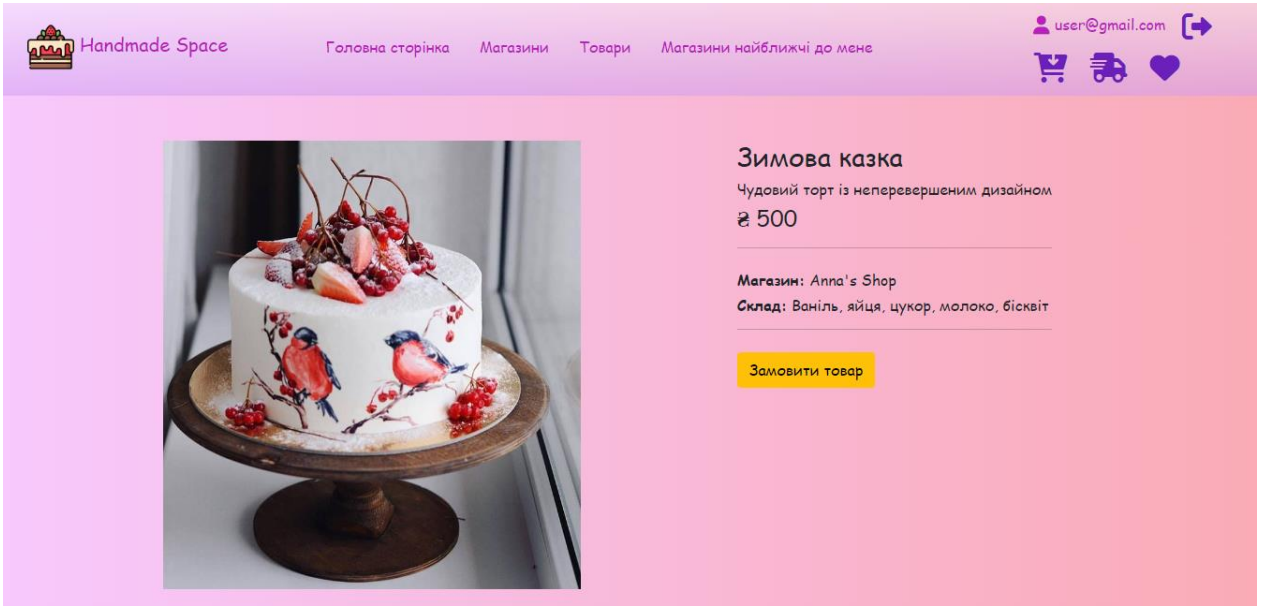

Рисунок 3.19 – Сторінка деталей товару

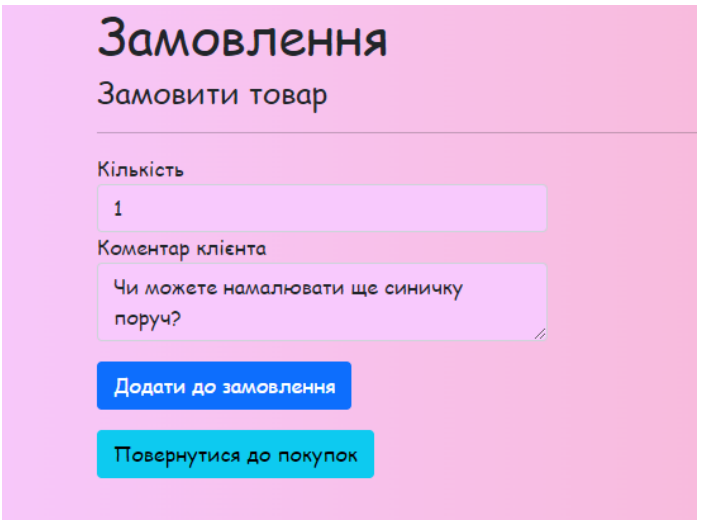

Рисунок 3.20 – Форма замовлення товару

Доданий у замовлення товар з’явиться у кошику, котрий користувач може побачити, натиснувши на зображення кошика на панелі головного меню та перейшовши на сторінку кошика, зображену на рис. 3.21. Користувач може продовжити покупки у цьому магазині, натиснувши клавішу «Повернутися до покупок» або одразу оформити замовлення, натиснувши відповідну кнопку та перейшовши на сторінку оформлення замовлення, зображену на рис. 3.22.

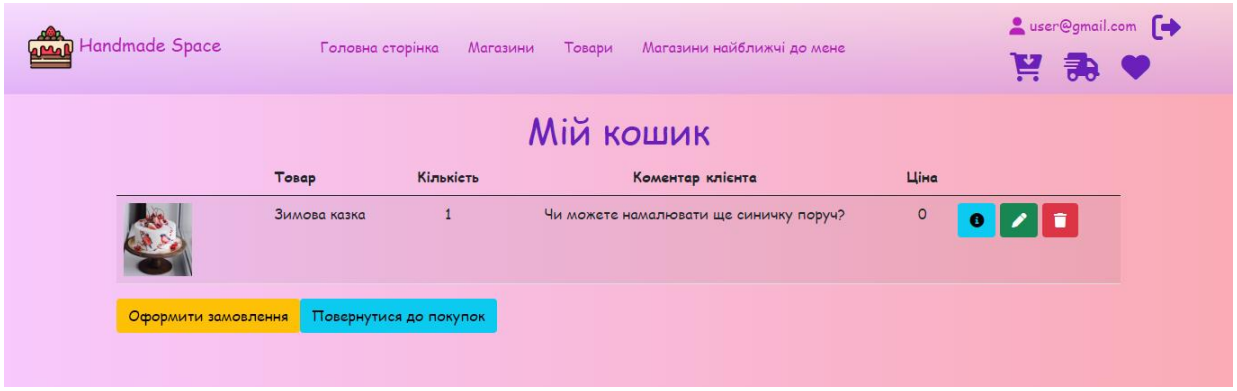

Рисунок 3.21 – Сторінка кошику із доданими у замовлення товарами

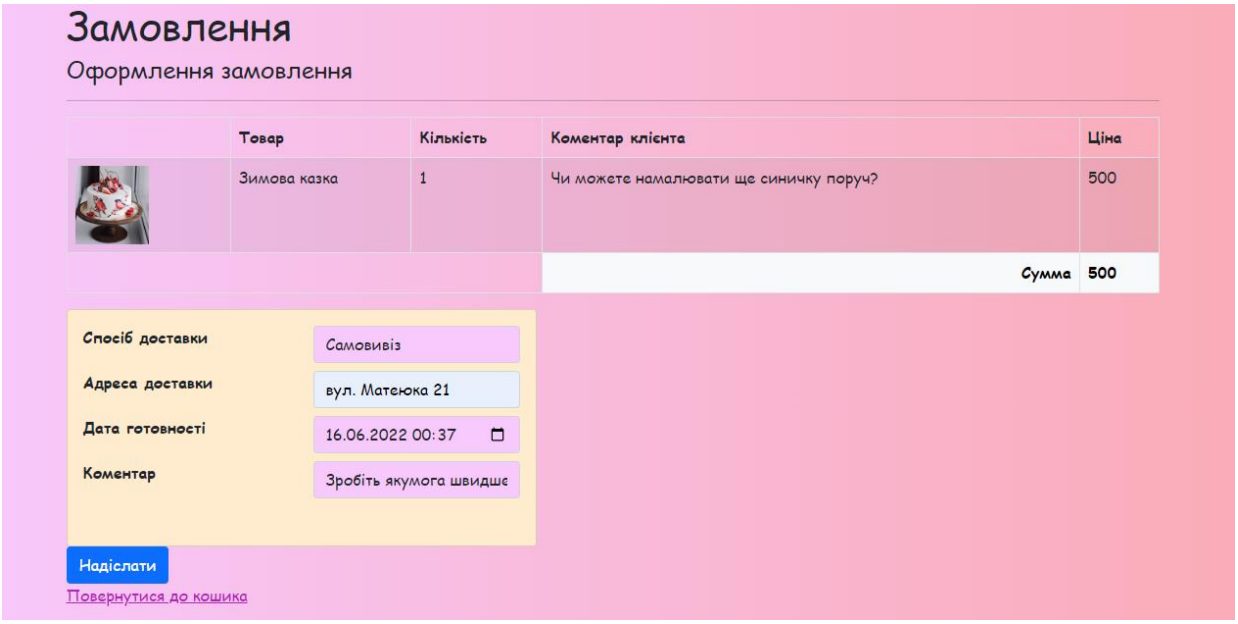

Рисунок 3.22 – Сторінка оформлення замовлення

Список власних замовлень користувач може побачити, натиснувши клавішу «Мої замовлення» на панелі головного меню та перейшовши на відповідну сторінку, зображену на рис. 3.23.

На сторінці деталей магазину для власника даного профілю доступною також є клавіша «Замовлення магазину», натиснувши на яку він може перейти на сторінку замовлень магазину, зображену на рис. 3.24.

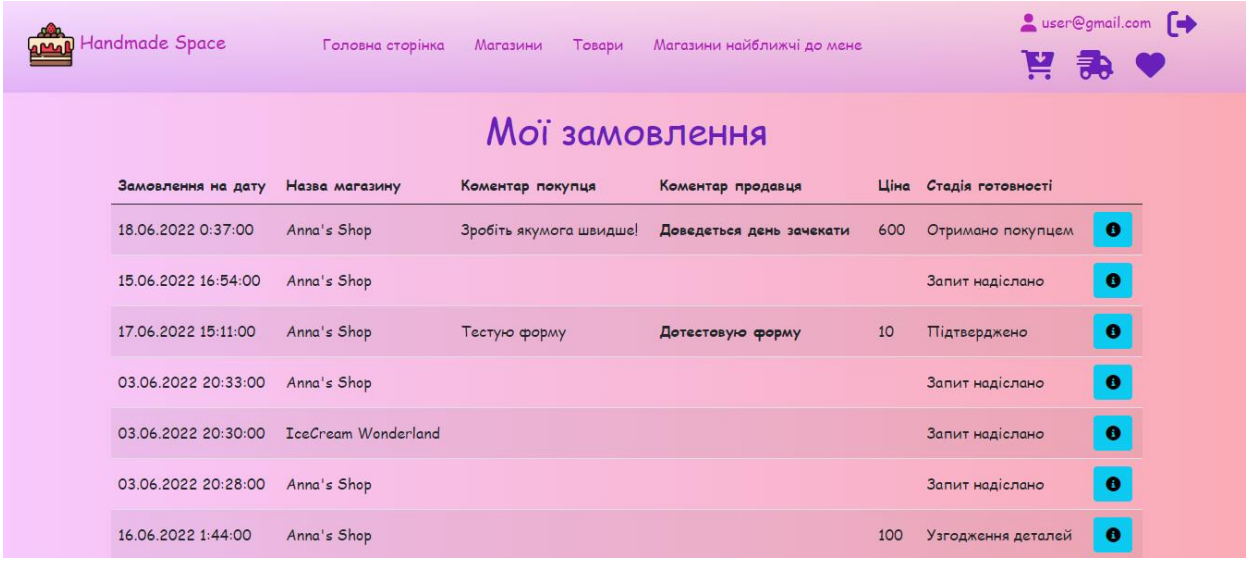

Рисунок 3.23 – Сторінка списку замовлень користувача

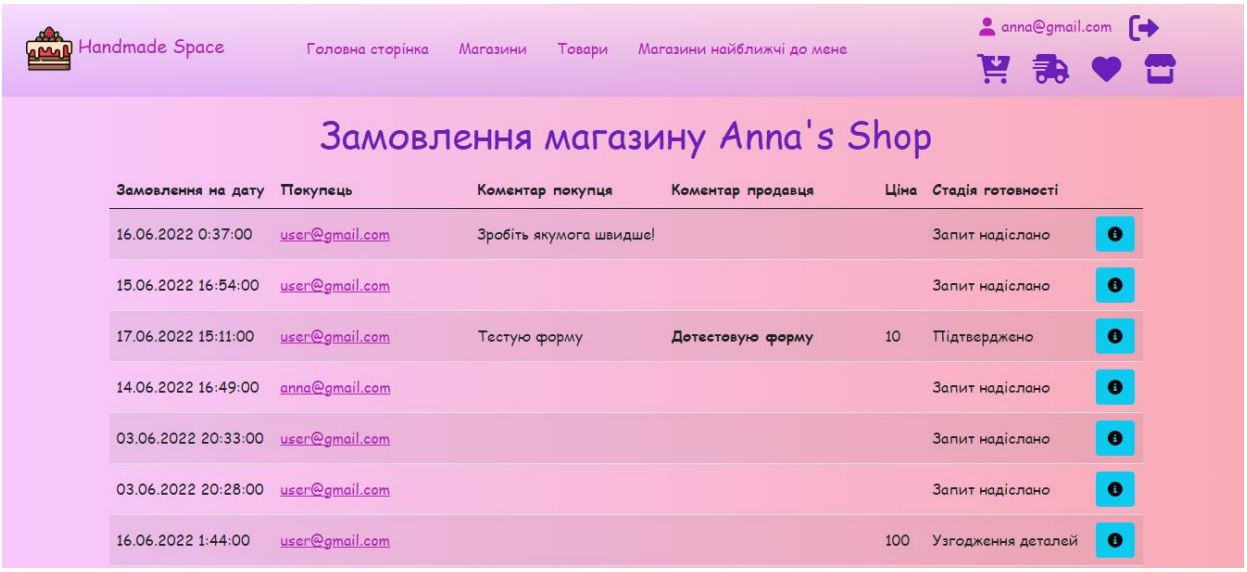

Рисунок 3.24 – Сторінка замовлень магазину

Натиснувши на «Деталі замовлення», користувач може побачити детальну інформацію про нього, перейшовши на зображену на рис. 3.25 сторінку. Також тут користувач має змогу підтвердити замовлення або за бажанням додатково узгодити деталі замовлення, натиснувши відповідну клавішу та перейшовши на сторінку узгодження деталей замовлення, зображену на рис. 3.26.

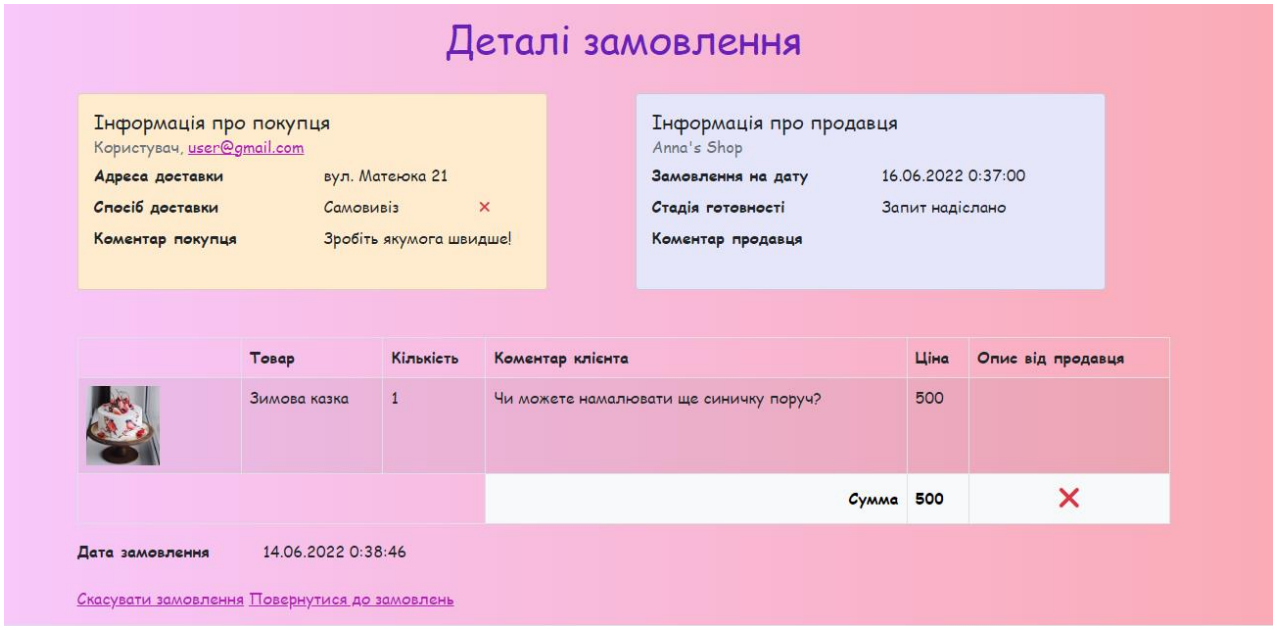

Рисунок 3.25 – Сторінка деталей замовлення

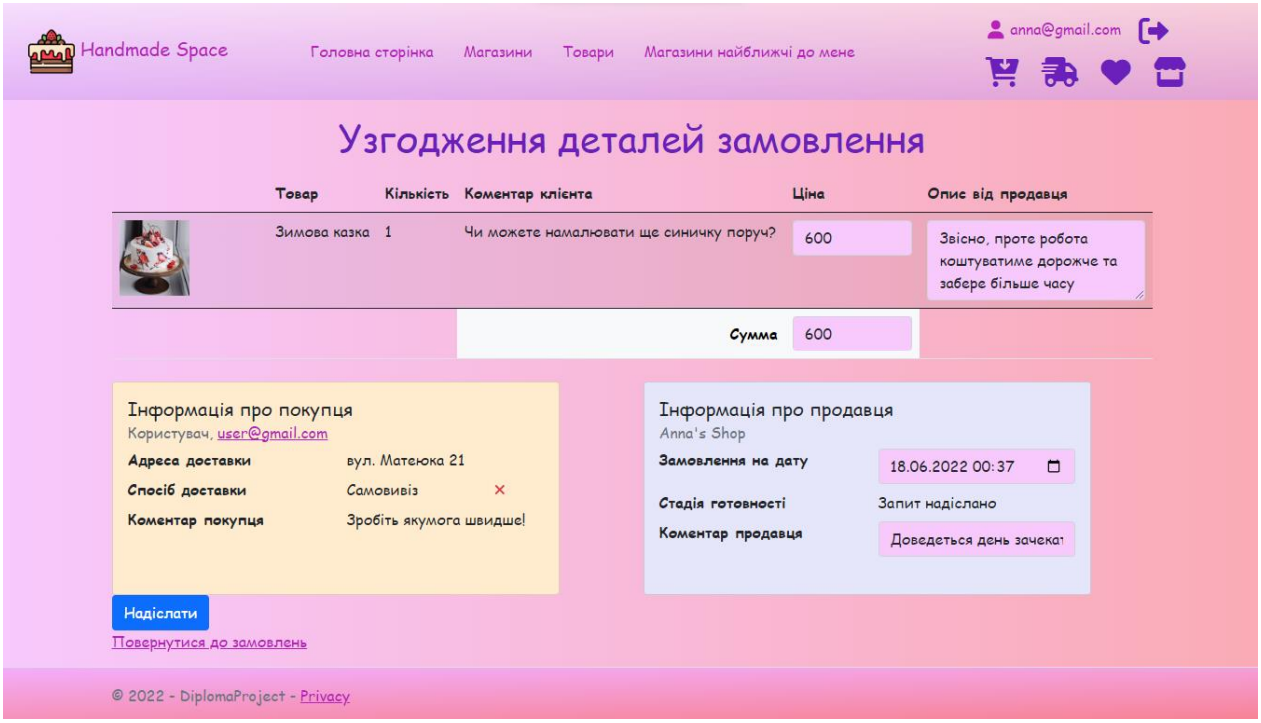

Рисунок 3.26 – Сторінка узгодження деталей замовлення

При отриманні свого замовлення користувач може залишити відгук, натиснувши відповідну клавішу і перейшовши на сторінку, зображену на рис. 3.27.

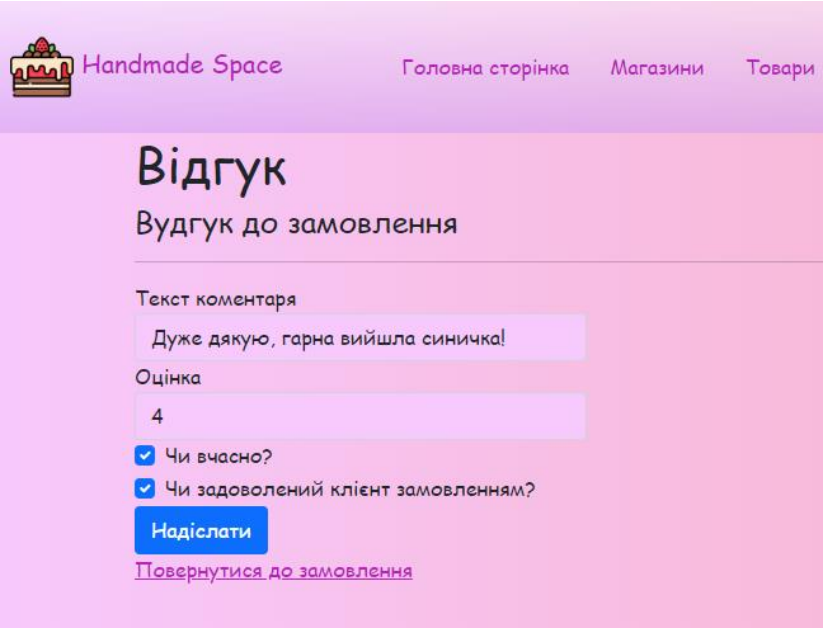

Рисунок 3.27 – Сторінка відгуку користувача щодо виконання замовлення

Додатково, користувач може побачити на карті магазини (рис. 3.28), котрі знаходяться поблизу його поточного місця розташування, натиснувши на клавішу «Магазини найближчі до мене» на панелі головного меню. Натиснувши на червоний маркер, зображений на карті, він побачить вікно з інформацією про магазин та матиме змогу одразу перейти на його сторінку, натиснувши відповідну клавішу в інформаційному вікні.

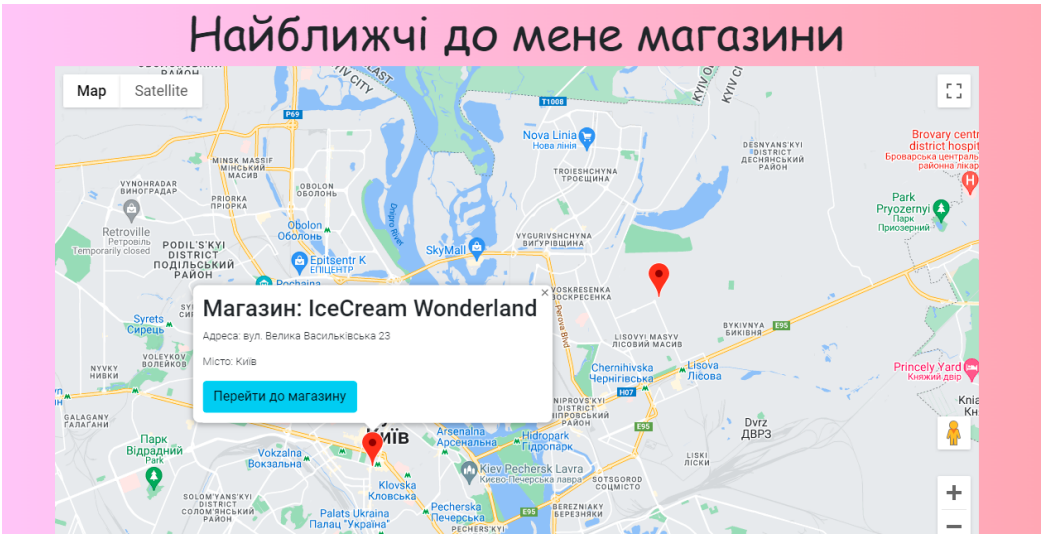

Рисунок 3.28 – Сторінка найближчих магазинів

Розроблена веб-орієнтована платформа для продажу крафтових товарів «Handmade-space» відповідає сучасним стандартам розробки веб-застосунків, реалізує клієнт-серверну архітектуру у вигляді шаблону MVC, має дизайн,

визначений у стандартах [38], надає гарні та зручні можливості роботи користувачів та реалізує увесь необхідний функціонал, визначений у підрозділі 3.2.

3.6 Тестування системи

Для того, щоб упевнитись у коректній роботі веб-орієнтованої платформи, зокрема у відповідності вимогам архітектури та очікуваній поведінці системи, повинно бути проведене модульне тестування.

Модульне тестування (англ. Unit testing) — метод тестування ПЗ, у якому тестується окремо кожен модуль коду системи. Модуль – найменша частина програми, яка може бути протестована, та яка має зазвичай один або кілька входів та один вихід. У процедурному програмуванні модулем є окрема функція або процедура, в об'єктно-орієнтованому програмуванні це метод. [39]

При модульному тестуванні тестується кожна атомарна функціональність застосунку окремо у штучно створеному середовищі, яке створюється шляхом заповнення відсутніх частин програми фіктивними макетними об’єктами.[39] Наприклад, є функція, для роботи якої потрібні змінні або об’єкти, яких ще не створено. Під час модульного тестування вони враховуються у вигляді макетних об’єктів, створених виключно з метою модульного тестування, проведеного в цьому розділі коду.

xUnit.net — це безкоштовний інструмент модульного тестування для .NET Framework з відкритим вихідним кодом, орієнтований на спільноту. Написаний оригінальним винахідником NUnit v2, xUnit.net — це новітня технологія модульного тестування C#, F#, VB.NET та інших мов .NET. xUnit.net працює з ReSharper, CodeRush, TestDriven.NET і Xamarin. Він є частиною .NET Foundation і діє відповідно до їх кодексу поведінки. Він ліцензований під Apache 2 (ліцензія, схвалена OSI).[40]

Функціонал контролерів даної веб-платформи для продажу крафтових продуктів «Handmade-space» покрито юніт-тестами на 95% з використанням пакетів модульного тестування xUnit та Moq для створення макетних об’єктів. Результати успішного проходження усіх тестів можна побачити на рис. 3.29.

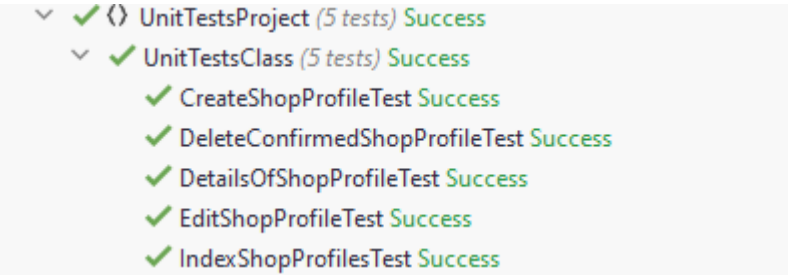

Рисунок 3.29 – Результати виконання юніт-тестів

3.7 Впровадження та інтеграція системи

Після виконання усіх попередніх кроків можна перейти до впровадження веб-орієнтованої платформи «Handmade-space». На основі підходу CI/CD від надісланого на гітхаб у вигляді комміту до вигляду інтерфейса користувача, що описаний в пункті 3.7, код веб-застосунку проходить деякі основні ітерації, що відображені на рис. 3.30.

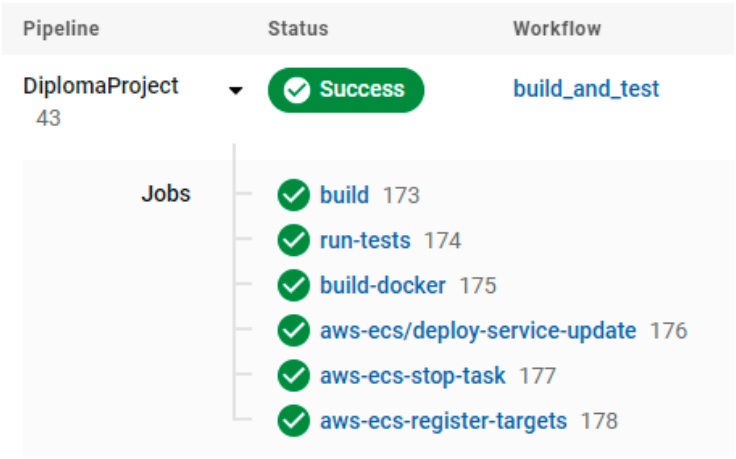

Рисунок 3.30 – Результат виконання кроків CircleCI

Під час першого кроку компілюється код веб-платформи. На другому кроці в середовищі CircleCI виконуються юніт-тести, описані у п. 3.8. На третьому кроці створюється докер-зображення веб-застосунку та надсилається у сервіс ECR Amazon. На подальших кроках щойно створене зображення надсилається у сервіс ECS Amazon, аби згодом розгорнути застосунок на віртуальній машині AWS та використати можливості даної платформи.

У результаті усіх проведених ітерацій маємо веб-платформу, запущену на

домені *www.handmade-space.top*, до якої мають доступ користувачі із будь-якої точки земної кулі.

3.8 Медіа маркетинг та розвиток системи

Під час входу на ринок тестується бізнес – модель при 0% комісії. Потім упродовж розвитку платформи та залучення більшої кількості користувачів та магазинів планується розділення послуг за різними типами підписок. За оплачуваної підписки у системі продавцю буде доступна більш широка аналітика продажів власного магазину, а також загальна аналітика по усіх магазинах на платформі.

При додаванні нового товару він з'являтиметься та перебуватиме деякий час на головній сторінці у списку новинок, що є гарною рекламою для цього товару та магазину загалом. Через деякий час товар зникатиме із списку новинок та з головної сторінки, і подальша його реклама на головній сторінці платформи буде проводитись на оплачуваній основі.

Проте найкращим вкладом у майбутній розвиток магазину всеодно залишатиметься його верифікація адміністратором. Верифікований магазин – це перевірена юридично оформлена інстанція, за якість товарів якої платформа зможе нести відповідальність перед своїми користувачами на основі поданих його власником потрібних документів. На основі цього до товарів верифікованого магазину покупці матимуть набагато більше довіри, а отже і продажі цього магазину зростатимуть. І відповідно за додавання нових товарів з верифікованого магазину вже стягуватиметься деяка комісія.

Також оплачуваним може стати створення додатково нових магазинів одним користувачем. При створенні першого магазину користувач матиме безкоштовний пробний період, поки його магазин тільки починає збирати своїх прихильників та потенційних покупців на платформі.

За допомогою медіа порталів таких як instagram.com, facebook.com, twiter.com планується створити групи для того, щоб користувачі та майбутні замовники платформи могли слідкувати за її розвитком. Завдяки такому медіа маркетингу залучатимуться потенційні замовники та клієнти у майбутньому.

ВИСНОВКИ

Проведено аналіз існуючих веб-платформ для продажу товарів.

Вивчено досвід побудови веб-платформ з точки зору пошуку комерційної ефективності.

Створено безкоштовну веб-платформу для розміщення інформації про власні крафтові товари та вироби на сторінках власних магазинів з можливостями: користувачам-покупцям та користувачам-продавцям ефективно контактувати один з одним; користувачам зберігати та переглядати у списку бажань вподобані товари та вироби; користувачам-продавцям переглядати аналітику профілю власного магазину (кількість прийнятих в роботу та успішно виконаних замовлень за певний період тощо) та загальну аналітику магазинів платформи.

Запропонована веб-платформа реалізує можливість розміщення інформації про крафтові товари, виготовлені талановитими та працьовитими людьми прямо на місці, на сторінці профілю магазину. Це є особливо актуальним для нашої країни в умовах воєнного та повоєнного часів в силу наявності проблем у логістиці поставок як готових продуктів, так і необхідної для їх виготовлення сировини.

Дана веб-система у майбутньому може стати гарантом уніфікованої ціни та стандартів якості крафтових продуктів, котрі будуть розміщені на її сторінках. Наявність подібного відкритого та прозорого майданчику для продажу крафтових продуктів стане неоціненним вкладом у відновлення та розширення економіки України, для створення повноцінної конкуренції у сфері виготовлення продуктів харчування та різних споживчих товарів загалом.

Заплановано продовжувати розвиток даної веб-платформи, зокрема, шляхом додавання корисних сервісів Google для збільшення продажів товарів та популярності платформи в цілому. Наступними напрямками для вдосконалення системи є, в першу чергу, розширення можливостей оформлення сторінок профілю магазину та вдосконалення механізмів узгодження та виконання замовлень. Кінцевою метою подальших вдосконалень є створення повноцінної відкритої платформи, що дозволить просто та зручно замовляти будь-який натуральний та якісний крафтовий продукт кожному жителю України, а згодом, можливо, і світу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Соммервилл И. Инженерия программного обеспечения, 6-е издание: Пер. с англ. – М.: Издательский дом "Вильямс", 2002. – 624 с.
2. Definition of craft or artisanal products [Електронний ресурс] – Режим доступу: http://uis.unesco.org/en/glossary-term/ (18.05.2022)
3. Minecraft для фермера: абетка розвитку дрібнооптового виробництва харчів [Електронний ресурс] – Режим доступу: https://agravery.com/uk/posts/author/show?slug=minesraft-dla-fermera-abetka-rozvitku-dribnooptovogo-virobnictva-harciv (18.05.2022)
4. Criteria for the Internet Shop System [Електронний ресурс] – Режим доступу: https://ecommerceinstitut.de/criteria-internet-shop-system/#_ftn1 (28.05.2022)
5. What is online store? [Електронний ресурс] – Режим доступу: https://www.oxfordwebstudio.com/en/did-you-know/what-is-online-store (30.05.2022)
6. Соціальна мережа [Електронний ресурс] – Режим доступу: https://igroup.com.ua/seo-articles/sotsialna-merezha/ (30.05.2022)
7. Social Networking [Електронний ресурс] – Режим доступу: https://www.investopedia.com/terms/s/social-networking.asp (30.05.2022)
8. Інтернет-магазин в соціальних мережах: плюси та мінуси [Електронний ресурс] – Режим доступу: https://ag.marketing/blog/internet-mahazyn-v-sotsial-nykh-merezhakh/ (02.06.2022)
9. Платформа онлайн-оголошень «OLX» [Електронний ресурс] – Режим доступу: https://olx-om.zendesk.com/hc/en-us/articles/208704098-What-Is-OLX- (19.05.2022)
10. OLX Business Model | How Does OLX Make Money? [Електронний ресурс] – Режим доступу: https://www.feedough.com/how-does-olx-make-money/ (19.05.2022)
11. Маркетплейс «Prom» [Електронний ресурс] – Режим доступу: https://prom.ua/ (19.05.2022)
12. Клієнт-серверна архітектура [Електронний ресурс] – Режим доступу:

https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (04.06.2022)

13. Компоненты сетевого приложения. Клиент-серверное взаимодействие и роли серверов. [Електронний ресурс] – Режим доступу: https://www.4stud.info/networking/lecture5.html (22.05.2022)
14. Balagurusamy E. Programming with Java / Balagurusamy E. – India: McGraw-Hill Education, 2019. – 600 с.
15. Rohit K. Programming with Java / Khurana Rohit. – New Delhi: Vikas Publishing, 2014. – 436 с.
16. The Python Tutorial [Електронний ресурс] – Режим доступу: https://docs.python.org/3/tutorial/index.html (22.05.2022)
17. Web-разработка на Python глазами PHP-программиста [Електронний ресурс] – Режим доступу: https://habrahabr.ru/post/243961/ (22.05.2022)
18. Lerdorf R. Programming PHP / R. Lerdorf, K. Tatroe, P. MacIntyre. – United States: O'Reilly Media, 2009. – 542 с.
19. Ullman L. PHP and MySQL for Dynamic Web Sites / Larry Edward Ullman. – United Kingdom: Peachpit Press, 2003. – 572 с.
20. Шилдт Г. C# 4.0: полное руководство.: Пер. с англ. — М.: ООО "И.Д. Вильямс", 2011. – 1056 с.
21. Офіційна документація .NET [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/dotnet/ (26.02.2022)
22. What Is a Database? [Електронний ресурс] – Режим доступу: https://www.oracle.com/database/what-is-database/ (28.02.2022)
23. A Quick Overview of Different Types of Databases [Електронний ресурс] – Режим доступу: https://www.astera.com/type/blog/a-quick-overview-of-different-types-of-databases/ (28.02.2022)
24. Дейт К. Дж. Введение в системы баз данных. – СПб.: Вильямс, 2005. – 1316 с.: ил.
25. What is PostgreSQL? [Електронний ресурс] – Режим доступу: https://aws.amazon.com/rds/postgresql/what-is-postgresql/ (29.02.2022)

26. What is MSSQL (Microsoft SQL)? [Електронний ресурс] – Режим доступу: https://www.atlantic.net/vps-hosting/what-is-mssql/ (29.02.2022)
27. Advantages & Disadvantages of Microsoft SQL [Електронний ресурс] – Режим доступу: https://www.techwalla.com/articles/advantages-disadvantages-of-microsoft-sql (29.02.2022)
28. Ashwin P. Learn SQL with MySQL / Pajankar Ashwin. – India: BPB Publications, 2020. – 132 с.
29. MySQL Customers by Industry [Електронний ресурс] – Режим доступу: https://www.mysql.com/fr/customers/industry/ (29.02.2022)
30. MySQL Tutorial [Електронний ресурс] – Режим доступу: https://www.javatpoint.com/mysql-tutorial (29.02.2022)
31. Overview of ASP.NET Core MVC [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-6.0 (28.02.2022)
32. MVC [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Glossary/MVC (28.02.2022)
33. Офіційна документація платформи Docker [Електронний ресурс] – Режим доступу: https://docs.docker.com/ (22.05.2022)
34. Waud E. Docker Quick Start Guide: Learn Docker Like a Boss, and Finally Own Your Applications / Earl Waud. – United Kingdom: Packt Publishing, 2018. – 230 с.
35. Коли та навіщо потрібен CI/CD [Електронний ресурс] – Режим доступу: https://habr.com/ru/company/southbridge/blog/649027/ (22.05.2022)
36. Офіційна документація платформи CircleCI [Електронний ресурс] – Режим доступу: https://circleci.com/docs/ (22.05.2022)
37. Офіційна документація сервісів Amazon [Електронний ресурс] – Режим доступу: https://docs.aws.amazon.com/ (22.05.2022)
38. Уэйншенк Сьюзан. 100 главных принципов дизайна. Как удержать внимание. – СПб.: Питер, 2012. – 272 с.: ил.

39. Unit Testing Techniques and Best Practices [Електронний ресурс] – Режим доступу: https://www.xenonstack.com/insights/what-is-unit-testing (26.05.2022)
40. Офіційний сайт xUnit [Електронний ресурс] – Режим доступу: https://xunit.net/ (26.05.2022)

ДОДАТКИ

ДОДАТОК А ДІАГРАМА ПРЕЦЕДЕНТІВ ВЕБ-СИСТЕМИ

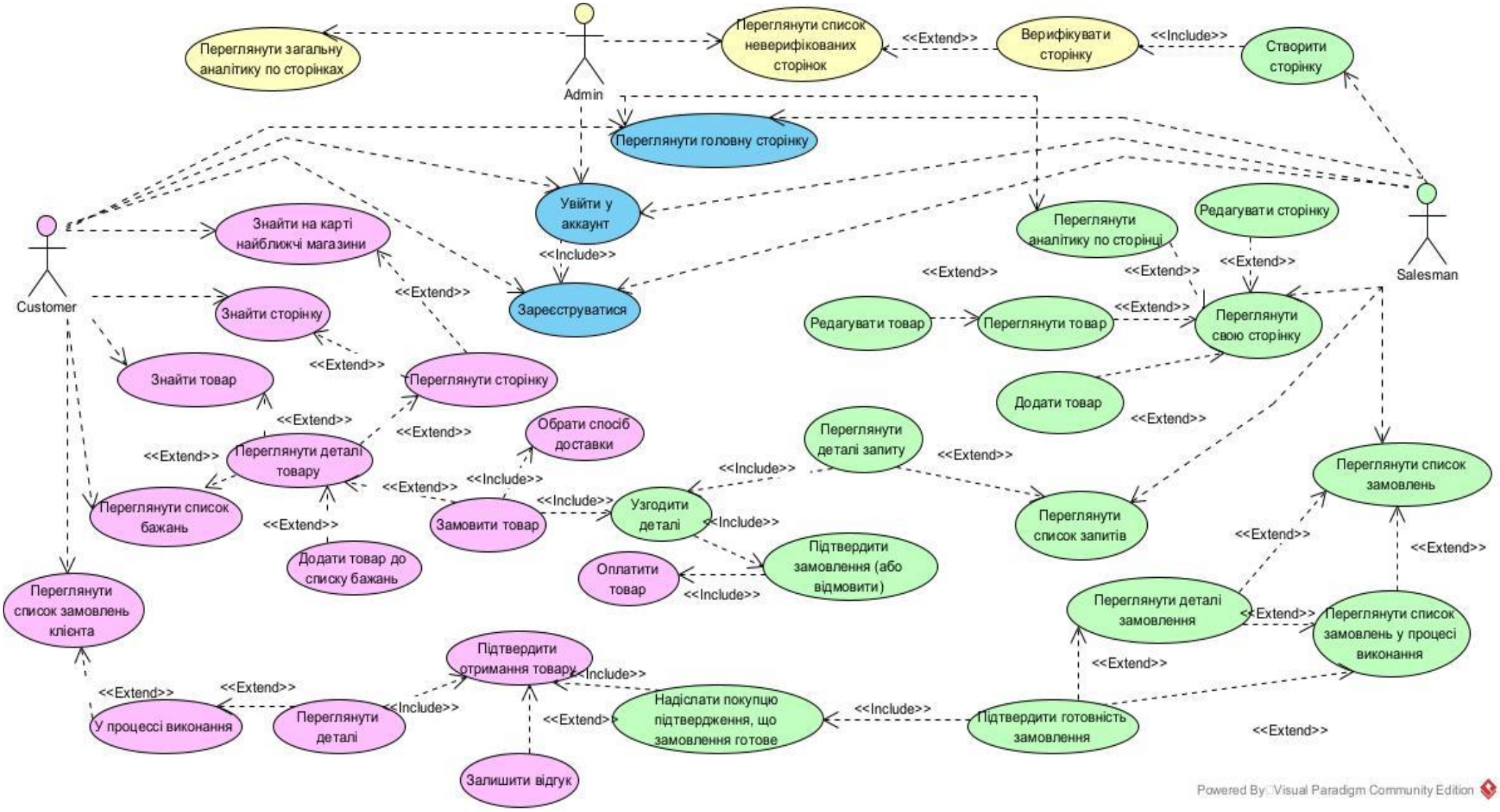

Рисунок А.1 – Діаграма прецедентів веб-орієнтованої системи «Handmade-space»

ДОДАТОК Б СХЕМА БАЗИ ДАНИХ ВЕБ-СИСТЕМИ

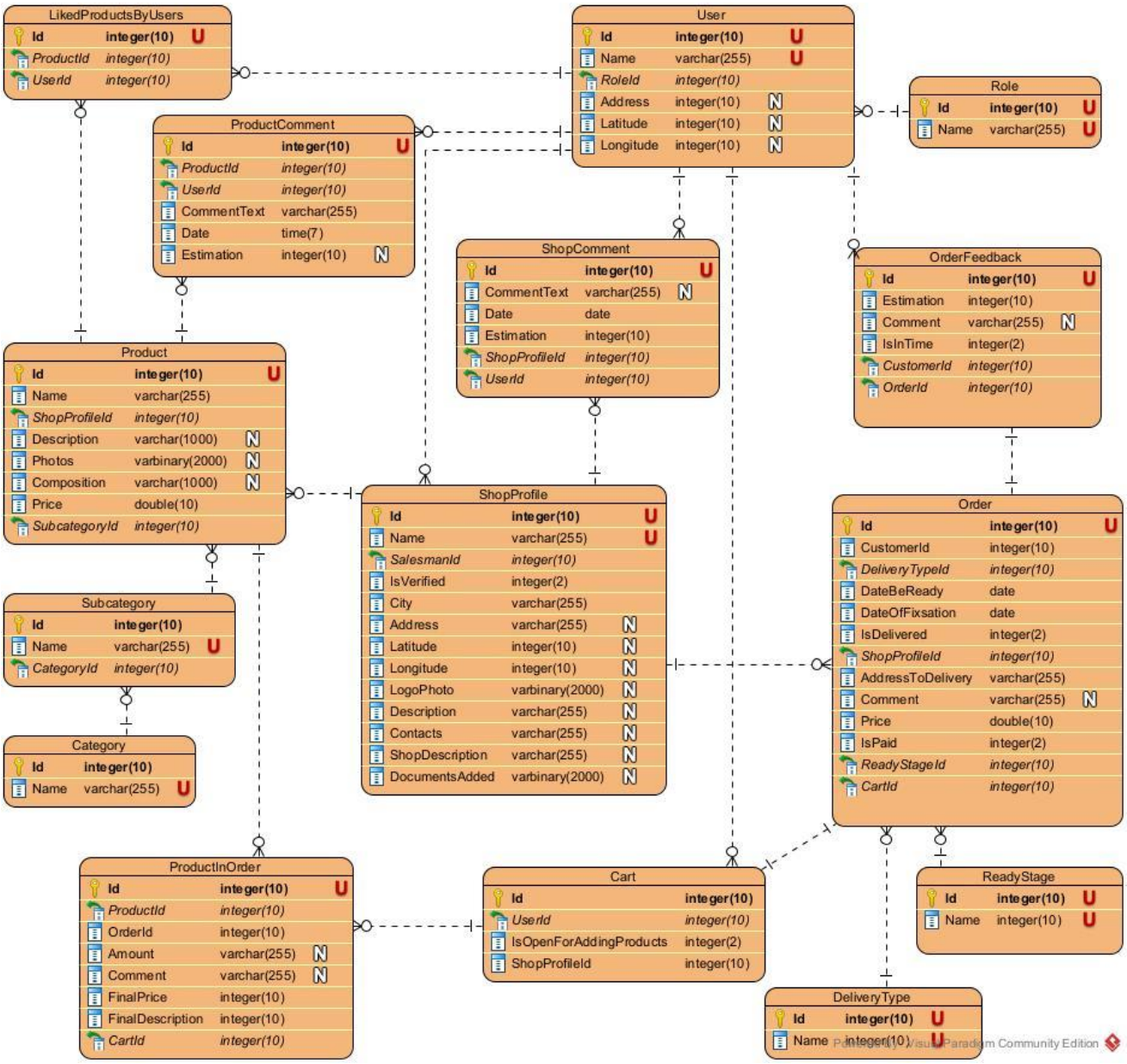

Рисунок Б.1 – Схема бази даних веб-орієнтованої системи «Handmade-space»

ДОДАТОК В ОСНОВНІ КОДИ ПРОГРАМНИХ МОДУЛІВ СИСТЕМИ

Файл Program.cs

```
namespace DiplomaProject
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder => { webBuilder.UseStartup<Startup>(); });
    }
}
```

Файл Startup.cs

```
namespace DiplomaProject
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllersWithViews();
            services.AddControllers();

            var environmentName = Environment.GetEnvironmentVariable("Env");
```

```
    var configuration = new ConfigurationBuilder()
            .AddJsonFile("appsettings.json")
            .AddJsonFile($"appsettings.{environmentName}.json", true)
            .AddEnvironmentVariables()
            .Build();

    // other service configurations go here
    var connString = configuration.GetConnectionString("DefaultConnection");

    services.AddDbContext<KraftWebAppContext>(options => options.UseMySql(connString,
                ServerVersion.AutoDetect(connString)));

    services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
        .AddCookie(options => //CookieAuthenticationOptions
        {
            options.LoginPath = new Microsoft.AspNetCore.Http.PathString("/Account/Login");
            options.AccessDeniedPath = new Microsoft.AspNetCore.Http.PathString("/Account/Login");
        });

    services.AddScoped<IAccountRepository, AccountRepository>();
    services.AddScoped<OrderRepository>();

    services.AddDistributedMemoryCache();

    services.AddSession(options =>
    {
        options.Cookie.Name = ".DiplomaProject.MyCart.Session";
        //options.IdleTimeout = TimeSpan.FromSeconds(10);
        options.Cookie.HttpOnly = true;
        options.Cookie.IsEssential = true;
    });
}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
```

```
            // The default HSTS value is 30 days. You may want to change this for production scenarios, see
https://aka.ms/aspnetcore-hsts.
            app.UseHsts();
        }

        app.UseHttpsRedirection();
        app.UseStaticFiles();

        app.UseRouting();

        app.UseAuthentication();    // автентификація
        app.UseAuthorization();     // авторизація

        app.UseSession();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllerRoute(
                name: "default",
                pattern: "{controller=Home}/{action=Index}/{id?}");
        });
    }
  }
}
```