МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»,
освітньо-професійної програми «Інформаційні технології проектування»
на тему: Інформаційна система обслуговування клієнтів інтернет-магазину одягу та аксесуарів з принтами. Інтерфейс ведення замовлення.

Здобувача групи Іт.м-24 Поливяного Андрія Володимировича
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Андрій ПОЛИВЯНИЙ
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доцент, к.т.н., доцент Анна НЕНЯ _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ІТ

Світлана ВАЩЕНКО
«_____» _____ 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Поливяному Андрію Володимировичу
(прізвище, ім'я, по батькові)

1 Тема кваліфікаційної роботи Інформаційна система обслуговування клієнтів інтернет-магазину одягу та аксесуарів з принтами. Інтерфейс ведення замовлення.

затверджена наказом по університету від «08» листопада 2023 р. № 1249-VI

2 Термін здачі студентом кваліфікаційної роботи «14» грудня 2023 р.
3 Вхідні дані до кваліфікаційної роботи результати обговорення теми з керівником

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі, методи дослідження, проектування інформаційної системи підтримки обслуговування клієнтів інтернет-магазину одягу та аксесуарів з принтами, програмна реалізація

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації) актуальність проблеми, мета проекту, постановка задачі, функціональні вимоги до проекту, контекстна діаграма IDEF0, декомпозиція IDEF0, діаграма варіантів використання, фізична модель даних, засоби реалізації, практична реалізація, висновки

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання 20 вересня 2023 р.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Підготовка	До 30.09.2023	
2	Планування	До 07.11.2023	
3	Розробка	До 24.12.2023	
4	Завершення	До 04.12.2023	
5	Оформлення документації	До 10.12.2023	

Магістрант _____ Поливяний А.В.

Керівник роботи _____ к.т.н., доц. Неня А.В.

АНОТАЦІЯ

Тема кваліфікаційної роботи магістра «Інформаційна система підтримки обслуговування клієнтів інтернет-магазину одягу та аксесуарів з принтами. Інтерфейс ведення замовлення».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 52 найменувань, додатків. Загальний обсяг роботи − 118 сторінок, у тому числі 77 сторінок основного тексту, 6 сторінки списку використаних джерел, 35 сторінок додатків.

Актуальність роботи визначається необхідністю поліпшення обслуговування клієнтів в умовах зростаючого ринку електронної комерції. Особливий акцент робиться на розробку ефективного інтерфейсу для ведення замовлень в інтернет-магазинах одягу та аксесуарів, що сприяє підвищенню задоволення клієнтів та покращує конкурентоспроможність бізнесу.

Мета роботи: створення інтерфейсу ведення замовлень в інформаційній системі підтримки клієнтів інтернет-магазину, що зробить процес обробки замовлень більш простим та швидким, тим самим підвищуючи задоволеність клієнтів і сприяючи зростанню продажів. В даній роботі було застосовано наступні методи дослідження: аналіз вимог, аналіз конкурентів, технічний аналіз, мануальне та функціональне тестування.

Результатом виконання роботи є розроблена інформаційна система підтримки обслуговування клієнтів інтернет-магазину одягу та аксесуарів з принтами.

Ключові слова: Інформаційна система, інтерфейс замовлень, електронна комерція, обслуговування клієнтів, MySQL, Laravel, Vue.js, Inertia.js.

ЗМІСТ

ВСТУП

У сучасному світі, де цифрові технології стрімко розвиваються та переформатовують традиційні бізнес-процеси, створення ефективних інформаційних систем стає невід'ємною частиною успішної комерційної діяльності. Особливо це стосується сфери електронної комерції, де динамічне управління клієнтським обслуговуванням та ефективність обробки замовлень визначають конкурентоспроможність бізнесу. Ця конкурентоспроможність ставить максимізацію корисності і прибутку як головну мету на ринках.[1]

Тематичне дослідження спрямоване на розробку та аналіз інформаційної системи, яка має на меті оптимізувати процеси обслуговування клієнтів в інтернет-магазині одягу та аксесуарів. Головним завданням інформаційної системи є створення зручного та інтуїтивно зрозумілого інтерфейсу для ведення замовлень, використання якого дозволить покращити взаємодію з клієнтами та сприятиме підвищенню загальної ефективності бізнес-процесів.

У рамках підготовки до написання кваліфікаційної роботи, я пройшов переддипломну практику в компанії MindK, яка спеціалізується на створенні веб-сайтів, веб-додатків, веб-сервісів, хмарних систем та мобільних застосунків. Метою моєї практики було здобуття практичних навичок у розробці та підтримці інформаційних систем, що мають важливе значення для сучасних онлайн-платформ. Індивідуальне завдання практики полягало у дослідженні підходів до вдосконалення системи обслуговування клієнтів для інтернет-магазину. Компанія MindK, маючи великий досвід у створенні високотехнологічних рішень, стала ідеальною базою для моєї практики, де я зміг застосувати теоретичні знання на практиці та розвинути свої професійні навички.

Основним фокусом кваліфікаційної роботи магістра є інтеграція сучасних технологій та методів розробки, які сприяють автоматизації та поліпшенню процесів обслуговування. Це включає аналіз потреб користувачів, проектування бази даних,

розробку фронтенду та бекенду системи, адаптивного інтерфейсу, а також впровадження ефективних алгоритмів для обробки та відстеження замовлень.

Основна мета цієї роботи полягає у розробці інтерфейсу ведення замовлень в інформаційній системі підтримки клієнтів інтернет-магазину "printcrafters.com.ua". Для досягнення цієї мети, визначено низку конкретних завдань: 1) проведення детального аналізу предметної області, що включає вивчення потреб користувачів та вимог до системи; 2) проектування бази даних інформаційної системи одягу та аксесуарів, забезпечуючи ефективне управління даними; 3) розробка архітектури та програмного коду інформаційної системи, зосереджуючись на надійності та користувацькому досвіді; 4) реалізація функціоналу та інтерфейсу системи ведення замовлення, забезпечуючи інтуїтивне та ефективне управління замовленнями; 5) розгортання та тестування розробленої інформаційної системи та інтерфейсу ведення замовлення для забезпечення їх надійності та відповідності вимогам користувачів.

Таким чином, дана тематичне дослідження не тільки сприяє розвитку технічних навичок і знань в області розробки інформаційних систем, але й вносить вагомий вклад у розвиток електронної комерції, зокрема у сфері оптимізації процесів обслуговування клієнтів.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБАСТІ

1.1 Актуальність електронної комерції в сучасному ринку

В електронній комерції інновації та якість електронних послуг відіграють надзвичайно важливу роль у забезпеченні задоволення та лояльності клієнтів [2] Це особливо актуально для інтернет-магазинів одягу та аксесуарів, де конкуренція є високою, а вибір клієнтів часто заснований на досвіді користування сайтом або додатком. Дослідження, проведене Rahman та іншими (2022), підкреслює значення інновацій у електронній комерції, зосереджуючись на впливі цих інновацій на задоволення та лояльність клієнтів, а також на якість електронних послуг у контексті інтернет-магазинів в Індонезії [2]

Якість та продуктивність вебсайту або додатку в електронній комерції є ключовими факторами, що впливають на сприйняття та вибір споживачів. Це підкреслює потребу в активних інноваціях та безперервному удосконаленні цифрових платформ, щоб забезпечити високий рівень задоволення клієнтів та їх лояльності. Інновації в електронній комерції та якість електронних послуг визначаються як ключові антецеденти задоволення клієнтів, де особливу увагу приділяють таким аспектам, як зручність користування, швидкість відгуку та наявність потрібної інформації[2]

Для підвищення задоволення та лояльності клієнтів, інтернет-магазини повинні постійно вводити нові послуги та функції, такі як опції оплати при доставці, просте повернення товарів, що сприяє зручності транзакцій та поліпшенню загального досвіду користувача. Ці зміни в інтерфейсі ведення замовлень та системі обслуговування клієнтів мають важливе значення для забезпечення високого рівня задоволення та утримання клієнтів [2].

Розглянемо концептуальну основу, яка відіграє вирішальну роль у сучасному світі бізнесу. Згідно з джерелом [3], термін "Електронна комерція" визначається як процес купівлі та продажу товарів і послуг через інтернет. Цей процес включає в себе

обмін грошима та передачу даних, необхідних для завершення транзакцій. Ці бізнес-транзакції відбуваються між підприємствами (у форматі Business-to-Business"), між компаніями та індивідуальними покупцями (у форматі Business-to-Consumer), від особи до особи (Consumer-to-Consumer), а також коли кінцеві споживачі пропонують товари чи послуги компаніям (у форматі Consumer-to-Business).[4] Він представляє собою сучасний підхід до торгівлі, заснований на використанні цифрових технологій, з метою забезпечення більшої зручності, швидкості та ефективності в комерційних взаємовідносинах між продавцями та покупцями. Відповідно, все більше та більше компаній інтегрують електронний ринок у свої стратегії з метою збільшення видимості та доступу до глобального ринку, що призводить до зростання електронної комерції [5].

Сучасний ринок електронної комерції вирізняється активною конкуренцією та високими вимогами споживачів, особливо помітно це у сегменті інтернет-магазинів одягу, який демонструє вражаючі темпи зростання. За даними CNMC (Comisión Nacional de los Mercados y la Competencia) (2020), електронна комерція в Іспанії показала щорічне зростання понад 20% у період з 2014 по 2019 рік. Високий потенціал зростання спостерігається, зокрема, у секторах туризму та одягу, як підкреслює Galeano (2019). Ця динаміка свідчить про значну активізацію споживачів у цифровому просторі, яка є ключовим фактором для розвитку електронної комерції. [1] Цей тренд зростання онлайн-торгівлі отримав новий імпульс із появою пандемії COVID-19, що стала каталізатором глобальних змін у споживчій поведінці та бізнес-моделях. Відповідно до інформації [6] Бюро перепису населення США, стале зростання онлайн-продажів у країні, що спостерігалося протягом останніх десятиліть, помітно прискорилося із початком пандемії. У другому кварталі 2020 року онлайн-продажі зросли на 31,80% порівняно з першим кварталом і на 44,50% порівняно з попереднім роком[6]. Особливо значним зростанням відзначені сектори продуктів харчування та ресторанних послуг. Пандемія внесла свої корективи у поведінку споживачів, відчутно вплинувши на сегменти, такі як супермаркети, спортивні товари та косметика. В цьому контексті електронна комерція перетворилася на потужний інструмент адаптації до змін, відкриваючи нові можливості для розвитку бізнесу та

виходу малих підприємств на міжнародний рівень. Зростання обсягів продажів у цій сфері (рис.1.1), яке досягло 4206 мільярдів доларів у 2020 році з 1336 мільярдів у 2014 році, підтверджує важливість електронної комерції у світовому економічному просторі [6] Спостерігаючи за динамікою розвитку сфери електронної комерції в Україні можна помітити позитивну тенденцію її росту. Однак, порівняно з аналогічними процесами у інших державах, темпи прогресу є відносно помірними (рис.1.2).

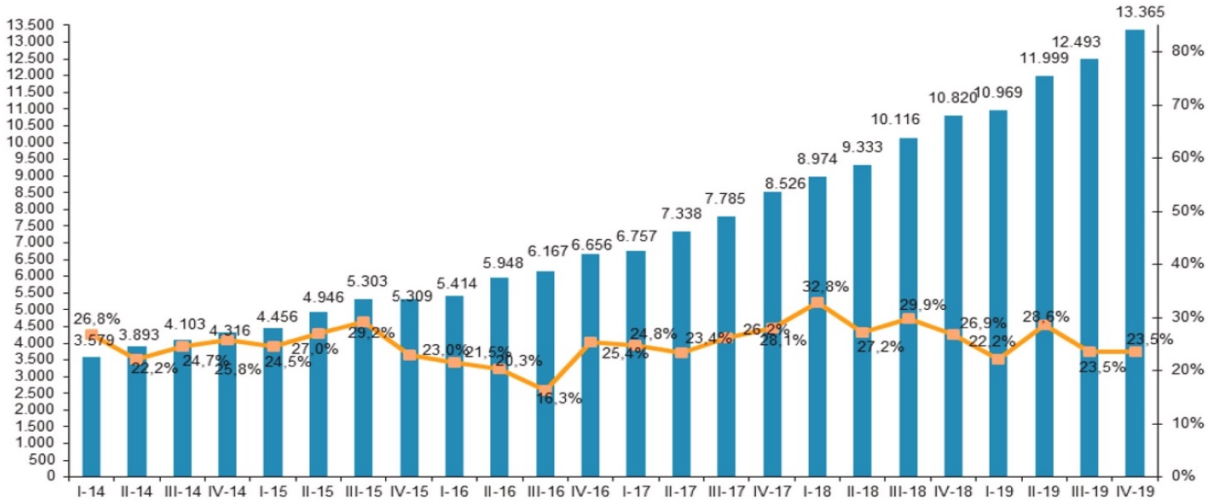


Рисунок 1.1 – Квартальна динаміка обороту електронної комерції та річний приріст в Іспанії (мільйони євро та відсоток)

Джерело: [1]

Протягом останніх років споживачі отримали кращий досвід онлайн-покупок, і цей тренд, безсумнівно, продовжиться і надалі. За останніми дослідженнями [2], зростання асортименту продукції та брендів змусило деяких аналітиків припускати, що онлайн-покупки можуть витіснити традиційні магазини. Хоча в деяких областях це дійсно відбулося, попит на звичайні магазини залишається у тих ринкових сегментах, де споживачі віддають перевагу безпосередньо бачити та торкатися товару перед покупкою. Однак наявність онлайн-покупок створила більш освічених споживачів, які можуть зручно порівнювати ціни і товари, не витрачаючи багато часу. В результаті, онлайн-покупки відкрили двері для багатьох невеликих роздрібних продавців, які не змогли б працювати, якби їм довелося нести високі витрати на утримання звичайних магазинів. Врешті-решт це стало виграшною ситуацією як для

споживачів, так і для продавців. Майбутнє онлайн-магазинів одягу залежить від обізнаних фахівців, які можуть створювати персоналізовані та привабливі віртуальні торгові досвіди для споживачів. Ті, хто розпочинає кар'єру в сфері бізнес-адміністрування, повинні використовувати найсучасніші цифрові технології та інструменти, такі як аналітика даних, машинне навчання та мобільні тенденції, щоб відповідати вимогам сучасних онлайн-покупців.

Споживачі сучасних інтернет-магазинів прагнуть не лише до широкого асортименту та якісних товарів, але й до високого рівня обслуговування та задоволення. Цей фактор є необхідними для того, щоб завоювати лояльність клієнтів, яка, в свою чергу забезпечує виживання організації в довгостроковій перспективі [7]. Лояльність - це зобов'язання купувати та підтримувати улюблений продукт чи послугу в майбутньому, навіть якщо зміна обставин або маркетингові зусилля можуть спонукати клієнтів звернутися до іншої компанії [8]Така тенденція зумовлює необхідність глибокого аналізу ринку та розробки ефективних інформаційних систем для підтримки обслуговування клієнтів. Недавні європейські дослідження, опубліковані Євростатом (2021 рік), підтверджують, що 70% загального населення брали участь у електронній комерції. Близько 71% населення стверджують, що їхні покупки були задовільними. Ця участь у електронній комерції в Європі і її ріст показані на рис. 1.3 [1]

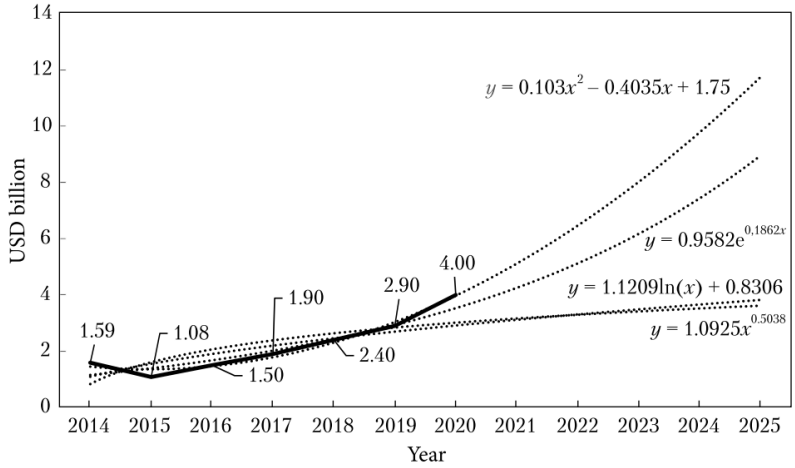


Рисунок 1.2 – Прогноз зростання ринку електронної комерції України до 2025 року, мільярди доларів США

Джерело: [9]

Онлайн-шопінг, який економить час та зусилля, дозволяє здійснювати покупки з будь-якого місця в будь-який час, стає все більш вигідним через пропозиції кращих угод та продуктів за розумними цінами без додаткових витрат на транспорт або харчування [10] Сучасні системи електронної комерції значно покращили користувацький досвід, зменшуючи вартість і надаючи надійні та безпечні рішення у порівнянні з традиційними фізичними магазинами. Це підкреслює важливість інтеграції технологій у роздрібну торгівлю, оскільки це не тільки підвищує зручність покупців, але й сприяє стійкості бізнесу. Технологічний прогрес дозволив споживачам легко порівнювати товари, сприяючи розвитку малого роздрібного бізнесу, який раніше був обмежений через високі витрати на фізичні магазини. Це також відкриває нові можливості для інновацій та креативності в способах представлення продуктів та взаємодії з клієнтами. Перспективи інтернет-магазинів залежать від здатності створювати персоналізовані та залучення віртуальних досвідів покупок для споживачів.

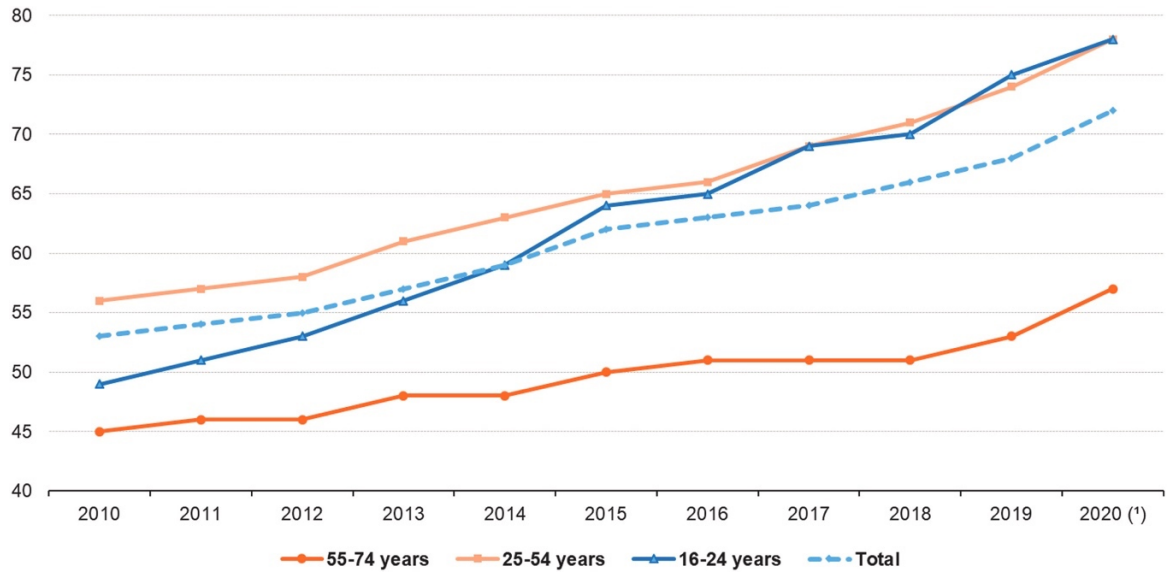


Рисунок 1.3 – Інтернет-користувачі, які купували або замовляли товари чи послуги для особистого використання за віком

Джерело: [1]

На сьогоднішній день, коли цифрові технології стрімко розвиваються, якість електронних сервісів (e-service quality) набуває особливого значення, оскільки має безпосередній вплив на рівень задоволення клієнтів та, відповідно, на їх лояльність. Висока конкуренція та різноманіття онлайн-сервісів змушують компанії неустанно покращувати якість своїх електронних послуг. У контексті все більш обізнаних та вимогливих клієнтів, якість і швидкість надання електронних послуг безпосередньо впливають на їх рішення щодо покупок та повторного використання сервісу [11]. Ця тенденція підтверджується дослідженням, опублікованим у Gadjah Mada International Journal of Business [12], де аналізується значення інновацій у сфері електронної комерції та якості електронних послуг для задоволеності та лояльності клієнтів. Зокрема, у центрі уваги дослідження опинилися онлайн-платформи в Індонезії. Виявлено, що нововведення в роботі цих платформ, такі як оновлення функціоналу веб-сайтів або додатків, поліпшення дизайну та введення нових послуг, мають суттєвий вплив на рівень задоволеності споживачів. Ключові аспекти якості електронних послуг, такі як ефективність, надійність, виконання обіцянок, конфіденційність, швидкість реагування, компенсації та зв’язок з клієнтами, впливають на рішення споживачів про лояльність до сервісу. І хоча задоволені клієнти, як правило, є більш лояльними, дослідження також вказує на те, що це відношення може бути складним та не завжди прямолінійним. Спостерігається, що у контексті індонезійського онлайн-шопінгу споживачі часто більш задоволені послугами, ніж виявляють лояльність до них. Це вказує на швидкозмінний характер ринку онлайн-шопінгу, де переваги та лояльність клієнтів можуть швидко змінюватися.

Виходячи з цих висновків, рекомендується, щоб постачальники електронних послуг зосереджувалися на інноваціях та покращенні якості своїх послуг, адже це може збільшити задоволеність клієнтів та сприяти зростанню їхньої лояльності. У швидкозмінному світі онлайн-комерції важливо адаптуватися до змінних умов ринку та задовольняти поточні потреби клієнтів [12]

1.2 Огляд існуючих платформ та систем обслуговування клієнтів в інтернет-магазинах

На початку XXI століття, із розвитком інтернет-технологій, спостерігається значне зростання сектора електронної комерції. Особливо вражаючі темпи зростання відзначаються у сфері інтернет-магазинів одягу [13] Сучасні покупці все частіше віддають перевагу онлайн-шопінгу, оскільки це зручно, швидко і дає можливість доступу до широкого асортименту товарів з усього світу. В цьому контексті системи обслуговування клієнтів грають ключову роль у забезпеченні високої задоволеності клієнтів та їхньої лояльності до бренду.

Розглядаючи технологічні аспекти, слід відмітити, що сучасні інформаційні системи в електронній комерції включають інтеграцію з CRM-системами (Customer Relationship Management), автоматизацію процесів замовлень та функції аналітики. Деякі популярні CRM-системи, такі як Bitrix24, Salesforce, HubSpot та Zoho CRM, використовуються бізнесами для оптимізації взаємодії з клієнтами та ефективності продажів.

Система Bitrix24 має переваги, такі як зручність у використанні, великий набір інструментів для співпраці та управління проектами, а також безкоштовний план для малого бізнесу. Проте, система може бути складною для розуміння новими користувачами через велику кількість функцій, і інколи користувачі відзначають недостатню швидкість роботи при великому навантаженні [14] Інша CRM система Salesforce відома своєю гнучкістю та можливістю глибокої настройки, а також широким спектром функціоналу для великих компаній. Однак, висока вартість та складність налаштувань можуть бути бар’єром для малого та середнього бізнесу. В свою чергу HubSpot пропонує простий у використанні інтерфейс і хороші інструменти для інбаунд-маркетингу [15] але його основні плюси можуть швидко перетворитися на мінуси, якщо потрібно додатковий функціонал, який зазвичай доступний лише у платних планах. А Zoho CRM пропонує доступні ціни і широкий спектр функцій для малих та середніх підприємств, але іноді користувачі відзначають

менш інтуїтивно зрозумілий інтерфейс та обмеження на інтеграцію без додаткових витрат [16]

Багато онлайн магазинів, включно з платформами на кшталт Shopify, WooCommerce, Magento, та OpenCart, інтегровані з цими CRM-системами, що дозволяє автоматизувати ведення клієнтських баз, процеси продажів та маркетингові кампанії. Кожна з цих платформ має свої особливості, переваги та недоліки.

Shopify – це всесвітньо відома платформа електронної комерції, яка надає користувачам повний набір інструментів для створення та управління онлайн-магазином. Завдяки своїй зручності та легкості в налаштуванні, Shopify стала однією з найпопулярніших платформ для стартапів та вже існуючих роздрібних брендів. Ця платформа вирізняється своєю простотою в налаштуванні та використанні, потребує низького рівня технічних знань якщо мова йде про створення та запуск базового онлайн-магазину, завдяки своїм готовим до використання темам та інтуїтивному інтерфейсу користувача. Однак, для розробки власної, унікальної теми, вимагаються глибші знання у веб-розробці. Shopify це SaaS (Software as a Service) рішення [17], тому користувачі платять абонплату щомісяця, що включає хостинг та підтримку. Мінусом може бути менша гнучкість у порівнянні з відкритими платформами та вищі витрати на транзакції. Shopify використовується такими брендами як Gymshark, Kylie Cosmetics, Fashion Nova та інші.

Інша платформа WooCommerce є стороннім плагіном для Wordpress, завдяки якому можна перетворити звичайний сайт на інтернет магазин. Він гнучкий та має можливість глибокої індивідуалізації, але вимагає більше технічних знань для налаштування та оптимізації. Проаналізувавши платформу, стає зрозуміло, що хоча основний функціонал WooCommerce безкоштовний, багато корисних додатків та тем є платними, що може збільшити загальну вартість ведення магазину. Однак, з огляду на широку підтримку спільноти та велику кількість готових рішень, WooCommerce залишається популярним вибором для бізнесів різного розміру. Приклади брендів які використовують WooCommerce: Blue Star Coffee Roasters, Porter and York, UPS та інші.

Magento – це одна з найбільш потужних платформ електронної комерції з відкритим кодом, яка пропонує велику гнучкість та контроль над функціоналом, виглядом та змістом інтернет-магазину. Це ідеальний вибір для середніх та великих підприємств зі складними бізнес-процесами та великою кількістю продуктів. Magento відомий своєю масштабованістю, розширеною системою управління продуктами, вдосконаленими можливостями пошуку та каталогізації, а також вбудованими функціями для міжнародної торгівлі. Однак, складність платформи може вимагати високий рівень технічних знань для налаштування та обслуговування, що призводить до додаткових витрат на розробників та консультантів. Також, Magento вимагає сильного хостингу через великі вимоги до ресурсів, що також може збільшити витрати. Великі бренди, такі як Canon, Ford і Nike, використовують цю платформу.

OpenCart – це багатофункціональна, легка у використанні, відкрита платформа електронної комерції, яка пропонує ідеальне рішення для малих та середніх підприємств. Її простий інтерфейс керування і легкість встановлення модулів роблять OpenCart популярним вибором серед підприємців, які шукають ефективне та вартісно доступне рішення для запуску онлайн-бізнесу.Ця платформа має відкритий код що дає змогу легко налаштувати і адаптувати магазин до специфічних потреб бізнесу. Однак, це також може збільшити залежність від розробників для впровадження складніших налаштувань або індивідуальних функцій.

Після детального аналізу існуючих платформ, було зроблено їх порівняння, результати якого представлено у Таблиці 1.1.

Проаналізувавши інформацію з таблиці 1, можна зробити висновок, що кожна з розглянутих платформ має свої сильні та слабкі сторони. OpenCart та WooCommerce вирізняються своєю безкоштовністю та легкістю в освоєнні, тоді як Magento, хоч і складний, пропонує високий рівень підтримки рольового контролю доступу та інтеграції з іншими сервісами. Shopify, зі свого боку, показує збалансованість між легкістю освоєння та функціональністю, хоча і не є безкоштовним.

Таблиця 1.1 – Порівняльна таблиця існуючих систем обслуговування клієнтів

	Open cart	**Woocommerce**	**Magento**	**Shopify**
Складний та незрозумілий інтерфейс	-	+	-	-
Безкоштовний	+	+	-	-
Легко в освоєнні	+	+	-	+
Вимоги до технічних знань	Середні	Середні	Високі	Середні
Підтримка рольового контролю доступу	Обмежена	Обмежена	Висока	Висока
Адаптивний дизайн	+	+	+	+
Інтеграція з іншими сервісами	Висока	Середня	Висока	Висока

Джерело: розроблено автором

Щоб покращити обслуговування клієнтів інтернет-магазину, важливо врахувати, що зроблене порівняння платформ надає цінний огляд функціональних і нефункціональних характеристик, що є ключовими для задоволення потреб користувачів. При розробці нового рішення слід зосередитися на усуненні недоліків існуючих систем, таких як: складний інтерфейс або високі вимоги до технічних знань, одночасно підтримуючи та покращуючи їхні сильні сторони, як-от адаптивний дизайн та гнучка інтеграція з іншими сервісами. Визначення ясних та докладних функціональних вимог, які включають оптимізацію процесу оформлення замовлень і післяпродажного обслуговування, а також нефункціональних вимог, які забезпечують швидкість, безпеку та надійність системи, допоможе створити інтерфейс, що відповідає очікуванням сучасних споживачів. При створенні такого інтерфейсу для обробки замовлень, основними критеріями, що відрізнятимуть його від існуючих рішень, будуть простота управління, мінімальні технічні вимоги для

користувачів та гнучкість інтеграції з різноманітними платіжними шлюзами та бізнес-інструментами. Такий підхід сприятиме здобуттю конкурентних переваг на ринку та підвищенню вірності клієнтів завдяки поліпшеному користувацькому досвіду.

1.3 Виявлення проблем і викликів в інтерфейсі ведення замовлень

У контексті соціальної електронної комерції інтерфейси управління замовленнями стикаються з унікальними викликами та проблемами, які потребують ретельного аналізу та інноваційних рішень. Відсутність доступних у відкритому доступі інтерфейсів ускладнює об’єктивний аналіз та оцінку існуючих рішень у цій області. Тому було використано методику порівняльного аналізу існуючих систем, з метою виявлення та детального дослідження типових проблем і викликів, які впливають на ефективність інтерфейсів ведення замовлень. Ці виклики випливають з потреби збалансувати соціальні аспекти та комерційну ефективність, забезпечуючи зручний та інтерактивний інтерфейс для користувачів.

Ключовим елементом успіху є здатність інтерфейсів адаптуватися до швидко змінних трендів у поведінці споживачів та технологіях, щоб створити досвід покупки, який відповідає як сучасним потребам, так і очікуванням користувачів. Покращення інтерфейсу ведення замовлень не лише підвищує якість обслуговування, але й безпосередньо впливає на задоволеність та лояльність клієнтів, сприяючи створенню міцних та тривалих відносин між брендом та його споживачами.

Однією з основних проблем є інтеграція функціональності замовлення з різноманітними соціальними платформами. Це вимагає забезпечення сумісності та безперервності досвіду користувача, водночас підтримуючи безпеку та конфіденційність даних. На додачу до цього, є потреба в оптимізації інтерфейсів, щоб вони були привабливими та зручними для користувачів, незалежно від їхнього досвіду в інтернет-шопінгу. Це передбачає розробку інтуїтивно зрозумілих меню, зручних фільтрів та пошукових систем, які допоможуть користувачам легко

знаходити та обирати продукти. За даними [18] можна зробити висновок про значення інтуїтивно зрозумілого інтерфейсу в контексті задоволення користувачів. Це дослідження фокусується на розробці та оцінці інструменту для вимірювання задоволення користувачів інтерфейсом людина-комп'ютер (Questionnaire for User Interface Satisfaction, QUIS). Основна увага приділяється визначенню того, як різні елементи інтерфейсу, включаючи організацію екрану, термінологію та інформаційну підтримку, впливають на сприйняття та задоволеність користувачів. Дослідження підкреслює важливість розробки інтуїтивно зрозумілих інтерфейсів, які полегшують взаємодію користувача з системою та підвищують загальне задоволення від використання програмного продукту. Наприклад, високо оцінювалися інтерфейси, що мають чітку організацію екрану, легкість навігації, зрозумілість термінології, а також зручність вивчення та користування системою. Ці аспекти безпосередньо впливають на здатність користувачів легко знаходити та обирати продукти, забезпечуючи позитивний досвід користування.

Підтримка високого рівня індивідуалізації та персоналізації користувацького досвіду стає ще одним важливим аспектом. Це включає в себе забезпечення рекомендацій, заснованих на попередніх покупках та перевагах користувача, а також можливість налаштування сповіщень та спеціальних пропозицій. Згідно із [19], задоволення та довіра клієнтів щодо придбаного продукту чи послуги ведуть до підвищення їх лояльності. Це демонструє, що індивідуалізація та персоналізація покупок можуть формувати позитивний досвід, який, у свою чергу, призводить до створення та підтримки тривалих відносин з клієнтами. Такі функції не тільки підвищують задоволеність клієнтів, але й сприяють збільшенню продажів.

Іншим викликом є забезпечення високої швидкості обробки замовлень та ефективного управління запасами. [20] Системи мають бути достатньо гнучкими, щоб впоратися з великими обсягами замовлень, особливо під час пікових періодів, наприклад, під час святкових розпродажів або спеціальних акцій. Це також включає в себе інтеграцію з постачальниками та логістичними партнерами для забезпечення своєчасної та точної доставки. Також безпека платежів та обробка фінансових транзакцій є критичною складовою. Платформам необхідно впроваджувати передові

технології захисту даних та шифрування, щоб забезпечити безпечну оплату і захист особистої інформації клієнтів. Це має велике значення не тільки для дотримання нормативних вимог, але й для підтримки довіри та відданості клієнтів. З огляду на зростаючу популярність мобільних покупок, інтерфейси мають бути оптимізовані для забезпечення зручності та ефективності на всіх типах пристроїв. Це включає в себе не тільки адаптацію розміру екрану, але й забезпечення, щоб всі функції були доступні та зручні для використання на мобільних пристроях, тому можна підкреслити що розвиток мобільних інтерфейсів та адаптація до різних екранів та пристроїв стають все більш важливими для підвищення лояльності клієнтів.

Останнім, але не менш важливим, є аналіз та використання зворотного зв’язку від користувачів. Інтерфейси управління замовленнями мають включати механізми для збору та аналізу відгуків клієнтів, що дозволяє постійно покращувати функціональність та користувацький досвід. У статті [8] зазначено, що задоволення та довіра клієнтів щодо Tokopedia впливають на їх лояльність до платформи. Аналіз включає декілька змінних, таких як інтерфейс користувача, якість інформації про послуги, сприйнята безпека та зворотний зв’язок від клієнтів. Результати дослідження вказують на те, що зворотний зв’язок від клієнтів має значний вплив на задоволеність та довіру клієнтів, які є критичними факторами для підтримки лояльності клієнтів. Активне впровадження відгуків клієнтів у процес розробки та оптимізації інтерфейсів допомагає не тільки вирішувати існуючі проблеми, але й антиципувати майбутні потреби та вимоги користувачів.

В інтерфейсі ведення замовлень в сфері соціальної електронної комерції виявлено низку проблем і викликів, які вимагають уважного підходу та інноваційних рішень. Інтеграція функціональності замовлення з різними соціальними платформами, необхідність забезпечення інтуїтивно зрозумілих та зручних інтерфейсів, а також забезпечення безперервності та безпеки досвіду користувача є ключовими аспектами, що потребують уваги а також підтримка високого рівня індивідуалізації та персоналізації користувацького досвіду, включаючи рекомендації, засновані на попередніх покупках та перевагах користувача, є важливою для залучення та задоволення клієнтів.

2. ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ.

2.1 Формулювання мети та постановка задач.

Чітке встановлення мети під час розробки інформаційних систем є критично важливим для успіху проекту. Процес визначення конкретних стратегічних завдань та мети є вирішальним на початкових етапах створення системи. [21] Мета має точно відповідати основним завданням та призначенню системи. Основною метою є створення інтерфейсу ведення замовлень в інформаційній системі підтримки клієнтів інтернет-магазину, що зробить процес обробки замовлень більш простим та швидким, тим самим підвищуючи задоволеність клієнтів і сприяючи зростанню продажів. [18]

Для успішної розробки інформаційної системи було визначено мету за допомогою SMART- методу.

Детальний опис задач, визначених для досягнення мети:

1. Оцінка існуючих систем обслуговування клієнтів в інтернет-магазинах одягу: аналіз поточних стратегій та підходів до обслуговування клієнтів, оцінка їх ефективності.

2. Виявлення ключових викликів та проблем: вивчення типових проблем, з якими зіштовхуються клієнти при онлайн-шопінгу, зокрема питання, пов’язані з доставкою, поверненням товару, візуалізацією продукції.

3. Розробка критеріїв для оцінки ефективності обслуговування клієнтів: встановлення параметрів для оцінки якості обслуговування, включаючи швидкість відповіді на запити клієнтів, доступність інформації про продукти, легкість навігації по сайту.

4. Розробка пропозицій щодо поліпшення системи обслуговування: формування практичних рекомендацій з удосконалення процесів обслуговування, включаючи впровадження нових технологій, оптимізацію логістики, покращення інтерактивності веб-сайту.

5. Обгрунтування та валідація рекомендацій: проведення експертної оцінки розроблених пропозицій, включаючи моделювання та аналіз ризиків.

6. Проектування бази даних інформаційної системи підтримки обслуговування клієнтів інтернет магазину.

7. Розробка архітектури та написання програмного коду інформаційної системи підтримки обслуговування клієнтів інтернет-магазину одягу та аксесуарів з принтами.

8. Реалізація функціоналу та інтерфейс системи ведення замовлення

9. Розгортання та тестування розробленої інформаційної системи тау інтерфейсу ведення замовлення.

10. Висновки та рекомендації: підсумок проведеного аналізу з акцентом на якісні та кількісні показники отриманих результатів та викладенням рекомендацій щодо їх використання.

2.2 Детальний опис вимог до програмного продукту

Проекти можна оцінювати за двома параметрами: наскільки чітко визначені цілі і наскільки чітко визначені методи їх досягнення [22] Це включає в себе детальне розуміння кінцевих результатів, яких очікує замовник, а також конкретних кроків, які необхідно вжити для їх досягнення. Ясність у цих аспектах сприяє ефективному плануванню, розподілу ресурсів та управлінню проектом, зменшуючи ймовірність непередбачених проблем та збільшуючи шанси на успішне завершення проекту.

Функціональні вимоги:

1. Система Управління Замовленнями: Можливість відстеження статусу замовлення клієнтами, зручний функціонал керування та обробки замовлень менеджерами інформаційної системи підтримки клієнтів інтернет магазину.
2. Інтеграція з сервісом логістики: Автоматизоване формування ТТН (товарно-транспортна накладна) після підтвердження замовлення, трекінг статусу відправлень.
3. Сповіщення менеджерів: Розробити модуль сповіщення, що автоматично надсилатиме повідомлення менеджерам у Telegram у разі нових замовлень, змін у статусі замовлень чи інших критичних подій, що вимагають їх уваги або втручання.
4. Мобільна Адаптація: Повноцінна функціональність веб-сайту в мобільній версії та додатках.

Нефункціональні вимоги:

1. Зручність Користування: Інтуїтивний інтерфейс, легка навігація.
2. Безпека Даних: Захист особистих та фінансових даних клієнтів.
3. Швидкодія: Висока швидкість управління замовленнями.
4. Надійність: Висока доступність сервісу, мінімізація часу простою.
5. Масштабованість: Система має бути здатною до масштабування для підтримки зростаючого числа замовлень та користувачів.
6. Мінімальні технічні знання: Система має бути простою для користувачів без глибоких технічних знань.

Саме виходячи з описаних вище вимог, був складений детальний опис задач, який допоміг ретельно прорахувати кожен крок, необхідний для розробки інтерфейсу користувача та оптимізації процесу замовлення.

2.3 Інструменти реалізації

Для реалізації інформаційної системи підтримки обслуговування клієнтів інтернет-магазину одягу та аксесуарів з принтами була обрана монолітна архітектура. **Монолітна архітектура** передбачає створення додатку як єдиного, неподільного блоку з усіма тісно пов'язаними і залежними один від одного компонентами [23], [24]. Вона характеризується простотою в розгортанні та експлуатації, але в цьому є свої недоліки. Але спочатку розглянемо переваги монолітної архітектури:

1. **Просте розгортання**: розгортання включає лише один каталог, уникаючи складностей багаторазового розгортання [24]
2. **Спрощений процес розробки**: монолітний підхід може бути легшим для розробників, оскільки він передбачає роботу з однією великою кодовою базою.
3. **Економічно вигідне**: вартість розробки менша, адже ресурсів та часу на розробку буде витрачено менше, вартість хостингу для одного веб додатка нижча порівняно з мікросервісною архітектурою, де кожен мікросервіс може вимагати окремого середовища.
4. **Консолідація Ресурсів**: у монолітній архітектурі, оскільки все зосереджено в одному місці, легше керувати ресурсами, такими як бази даних, кеш-пам'ять, та залежності між компонентами.
5. **Тестування**: можливість досягти високого показнику покриття коду, оскільки всі компоненти і тести зібрані разом, це спрощує інтеграційне тестування

   До недоліків монолітної архітектури слід віднести:

1. **Високий рівень зв'язування**: монолітні архітектури демонструють вищий рівень зв'язування між об'єктами, що може ускладнити модифікацію та масштабування додатку [23]
2. **Складність для розуміння**: Висока цикломатична та когнітивна складність, що може зробити додаток складнішим для розуміння та підтримки з ростом його розмірів [23]

3. **Складність у масштабуванні**: монолітні архітектури можуть виявитися менш гнучкими при масштабуванні, оскільки це вимагає масштабування всього додатку, а не окремих його частин.
4. **Складнощі у модифікації**: у монолітних додатках зміни в одній частині можуть вплинути на інші, що вимагає додаткової координації та тестування.
5. **Не надійний**: помилка в одній частині монолітного додатку може спричинити збої в інших частинах, що підвищує ризик для загальної стабільності системи.

Незважаючи на недоліки монолітної архітектури вона все ще залишається на ринку адже її використовують для розробки малих та середніх проектів, що мають обмежений бюджет та час на розробку.

Планування розробки інформаційної системи передбачає використання сучасного набору технологій, які спрямовані на забезпечення ефективності та швидкості розробки. Основні технологій, які плануються застосувати в процесі розробки: PHP 8 [25], Laravel[26], MySQL[27], HTML&CSS[28], JavaScript[29], TypeScript [30], Inertia.js [31] Vue.js[32], Docker[33], PHPUnit[34], CI/CD за допомогою Github Actions[35], Amazon SES[36], Git [37] та інтегроване середовище розробки PhpStorm.

Laravel – це фреймворк на базі PHP, який дозволяє створювати високоякісні захищені веб-додаток [38] Він полегшує роботу розробників, економлячи час на розробку. Він використовує популярний шаблон проектування MVC (рис. 2.4). Шаблон MVC розділяє бізнес-логіку (Model), координацію керування (Controller) і представлення даних (View) на 3 різні рівні, що сприяє чіткому розділенню логіки, UI та управління даними. Це дозволяє розробникам працювати над окремими аспектами додатку незалежно від інших.

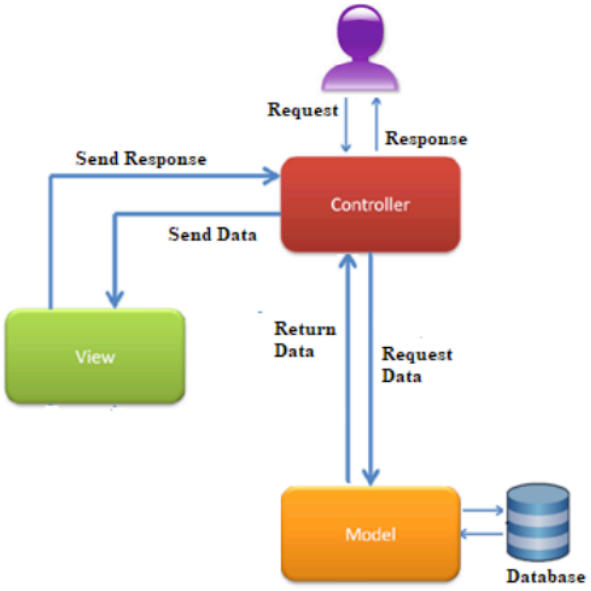


Рисунок 2.4 – Model-View-Controller Архітектура

Джерело: [39]

Для створення маркетплейсу інформаційної системи підтримки клієнтів інтернет-магазину одягу та аксесуарів з принтами було вирішено використати Vue.js фреймворк (рис. 2.5). Vue.js є легковісним фронтенд-фреймворком, який базується на моделі MVVM (Model-View-ViewModel) в веб-додатках [40] MVVM використовується як основа для створення інтерактивних та реактивних веб-інтерфейсів. Vue.js спрощує зв'язування даних між HTML (Видом) та JavaScript (ViewModel), автоматизуючи оновлення Виду при зміні даних без необхідності вручну маніпулювати DOM або прослуховувати події. Його компонентна архітектура дозволяє ефективно управляти та повторно використовувати код, що робить процес розробки більш ефективним і зменшує час на впровадження змін або додавання нового коду.

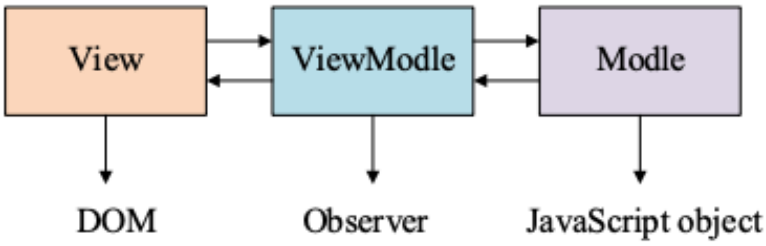


Рисунок 2.5 – Діаграма MVVM Архітектури

Джерело: [40]

Для реалізації зручності користування, було прийнято рішення використати підхід побудови SPA додатку без необхідності писати API, що можливо реалізувати за допомогою Inertia.js. Сторінки оновлюються без перезавантаження, що забезпечує швидку відповідь і покращує загальний користувацький досвід.

MySQL – це популярна система управління реляційними базами даних. Вона широко застосовується завдяки своїй надійності, гнучкості та простоті використання. Ця СУБД традиційно вважається більш простою у встановленні та управлінні. На етапі планування було обрано РСУБД MySQL за її надійність у зберіганні великих об’ємів інформації та високу продуктивність. Реляційні СУБД використовують строгу структуру таблиць і зв’язків між ними, що дозволяє виконувати складні запити з високою точністю і ефективністю.

Даний стек забезпечує швидку розробку, забезпечуючи ефективну інтеграцію між бекендом і фронтендом, що дозволяє розробникам максимально сконцентруватися на створенні бізнес-логіки, користувацького інтерфейсу та виконати поставлені задачі до проекту.

3. ПРОЕКТУВАННЯ ІНТЕРФЕЙСУ ВЕДЕННЯ ЗАМОВЛЕННЯ ДЛЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДТРИМКИ ОБСЛУГОВУВАННЯ КЛІЄНТІВ

3.1 Розробка стратегії підвищення ефективності обслуговування клієнтів

Згідно з дослідження[41] користувачі онлайн-магазинів одягу зіштовхуються з низкою проблем. Однією з ключових труднощів є відсутність можливості безпосередньо оцінити якість товару перед покупкою, що збільшує ризики при виборі продукції через інтернет. Оцінка якості товару в онлайн-магазинах одягу дійсно становить суттєвий виклик для споживачів. У цьому контексті, важливість детального опису та високоякісних зображень продуктів стає ключовою. Інтернет-магазинам необхідно зосереджуватися на наданні достатньої інформації про матеріали, розміри та фактури, щоб компенсувати відсутність фізичного досвіду. Також, інтеграція додаткових інструментів, таких як віртуальні примірювальні та інтерактивні 3D-моделі, може стати ефективним рішенням для зменшення невизначеності покупців та підвищення їхньої довіри до онлайн-покупок.

Цікавим є той факт, що, незважаючи на високий рівень сприйняття ризику, споживачі поступово адаптуються до особливостей онлайн-шопінгу [41] Один з прикладів такої адаптації - використання політики повернення товарів, які не відповідають очікуванням. Також важливою є роль брендів та їх вплив на лояльність та переключення брендів серед споживачів. Економічна значущість сектору моди в електронній комерції підкреслює потребу в розробці ефективних стратегій залучення та утримання клієнтів.

Враховуючи ці аспекти, важливо розробляти системи обслуговування клієнтів, які мінімізують ризики та підвищують задоволеність споживачів. Це може включати детальні описи товарів, якісні фотографії, поліпшену політику повернення та ефективну комунікацію з клієнтами.

3.2 Структурно-функціональне моделювання

Розроблення інформаційної системи підтримки обслуговування клієнтів для інтернет-магазину одягу та аксесуарів з принтами включає зосередження на структурно-функціональному моделюванні. Було прийнято рішення використовувати методологію IDEFO [42] Цей метод служить важливим засобом для структурного та ефективного представлення бізнес-процесів. У рамках цього процесу була побудована IDEF0 діаграма, яка зображена на рис.3.1. Ця діаграма візуалізує та аналізує ключові аспекти процесу обслуговування клієнтів - від приймання замовлень до їх виконання та доставки. Ця методологія дозволяє детально описати вхідні дані, процеси, вихідні дані, механізми та контрольні параметри, сприяючи виявленню ключових точок для оптимізації та підвищення ефективності системи.

Моделювання за допомогою IDEFO передбачає створення блок-схем, які демонструють, як вхідні дані трансформуються в вихідні через різноманітні процеси. Ці моделі є корисними для ідентифікації потенційних проблем та сприяють покращенню загальної діяльності системи. У процесі розробки інтерфейсу ведення замовлень було побудовано IDEF0 діаграму, яка враховує всі аспекти бізнес-процесів, від збору замовлень до їх виконання та доставки, забезпечуючи ясність та ефективність у роботі інтернет-магазину.

На рис. 3.2 представлена декомпозиція контекстної діаграми в нотації IDEF0 рівня A0 з рисунку 3.1.

Рисунок 3.1 – Контекстна діаграма в нотації IDEF0 для інформаційної системи підтримки клієнтів інтернет магазину одягу та аксесуарів з принтами, рівень A-0. Джерело: розроблено автором

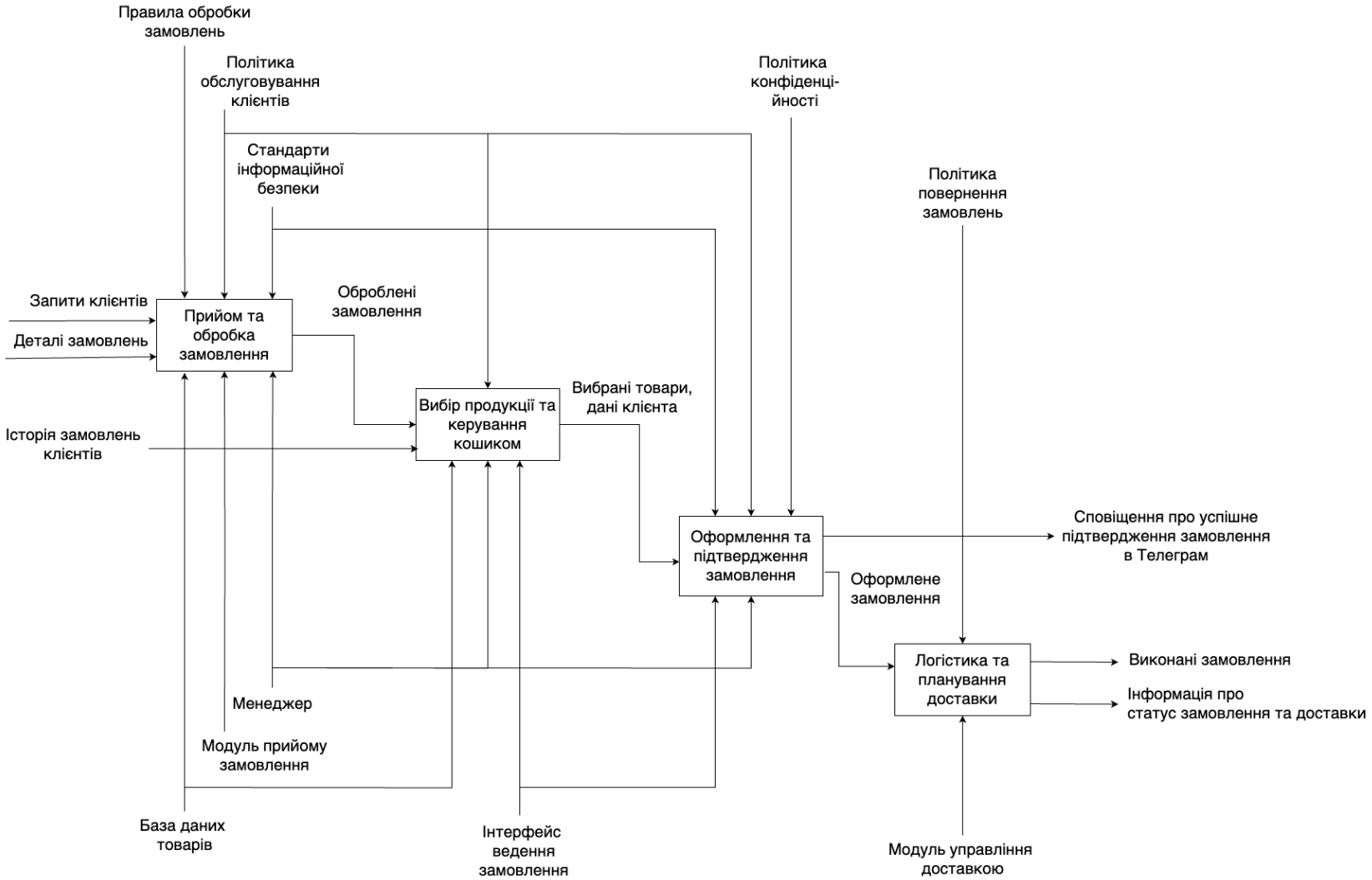


Рисунок 3.2 – Декомпозиція контекстної діаграми в нотації IDEF0 для інформаційної системи підтримки клієнтів інтернет магазину одягу та аксесуарів з принтами, рівень A0

Джерело: розроблено автором

3.3 Розробка архітектури баз даних

Архітектура бази даних є ключовим компонентом системи, що забезпечує зберігання, управління та обробку даних, які використовуються у всіх аспектах функціонування магазину. Для розробки інформаційної системи для інтернет-магазину, потрібна висока точність у зберіганні та обробці транзакцій клієнтів та структуровано працювати з великою кількістю записів.

Реляційна база даних – це тип бази даних, який використовує реляційну модель для зберігання та управління даними [43] Вони структуровані та зберігають дані у таблицях, які пов’язані між собою. Основною перевагою реляційних баз даних є забезпечення консистентності даних, легкість управління та огляду бази даних. Вони також сприяють безпеці, оскільки дозволяють шифрувати певні таблиці та контролювати доступ до даних. Крім того, реляційні бази даних забезпечують легкість резервного копіювання та відновлення даних.

В свою чергу нереляційні бази даних відрізняються від традиційних реляційних баз даних своєю структурою та підходом до зберігання даних. Вони не використовують стандартну схему таблиць, рядків та стовпців, а натомість пропонують більш гнучкі способи зберігання даних, такі як key-value pairs, документи, графи чи широкі стовпці. Ця гнучкість робить нереляційні бази даних підходящими для нових додатків, які потребують додаткової адаптивності та можливості швидкої зміни структури даних без необхідності перепроектування цілої бази даних [44]

При аналізі вибору бази даних було вирішено зупинитись на MySQL, яка має ряд переваг, важливих для реалізації проекту:

1. Структурованість даних: ідеально підходять для управління структурованими даними, такими як каталоги продуктів, інформація про клієнтів, замовлення та інші.

2. Масштабованість та продуктивність: має хорошу продуктивність при обробці великих обсягів даних та транзакцій.

3. Надійність та безпека: підтримує шифрування та управління доступом, захист від SQL-ін’єкцій і т.д.

4. Легкість управління: простота у налаштуванні та управлінні.

В рамках процесу проектування архітектури бази даних для інтернет-магазину одягу та аксесуарів з принтами було побудовано ER діаграму, яка зображена на рисунку 3.3. ER діаграма була створена з метою візуалізації структури бази даних, яка представляє сутності, їх атрибути та зв’язки між ними. Це дозволяє краще зрозуміти модель даних, на якій базується система, що важливо для забезпечення ефективної роботи системи.

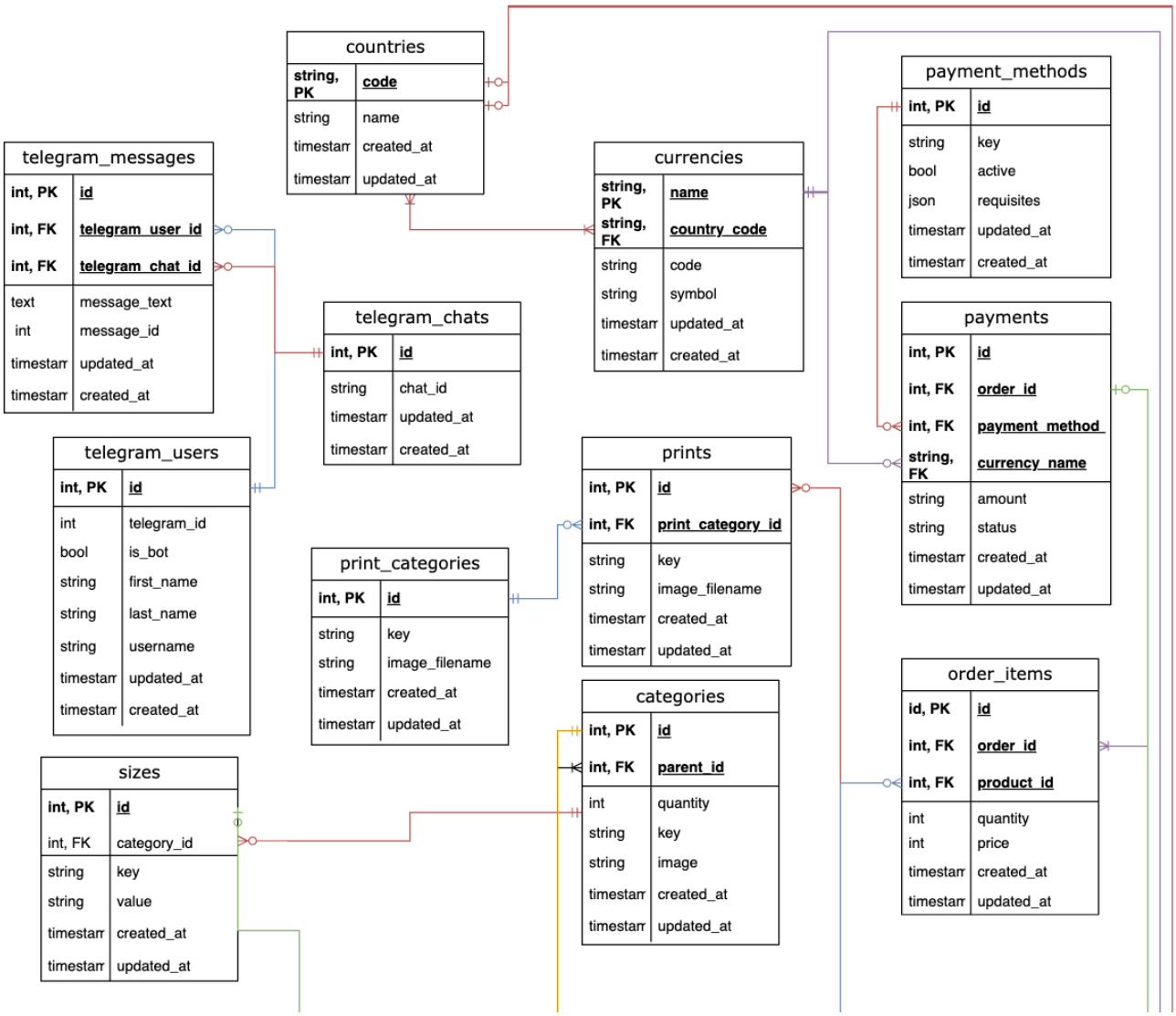


Рисунок 3.3 – ER діаграма бази даних для інформаційної системи підтримки клієнтів інтернет магазину одягу та аксесуарів з принтами, частина 1

Джерело: розроблено автором

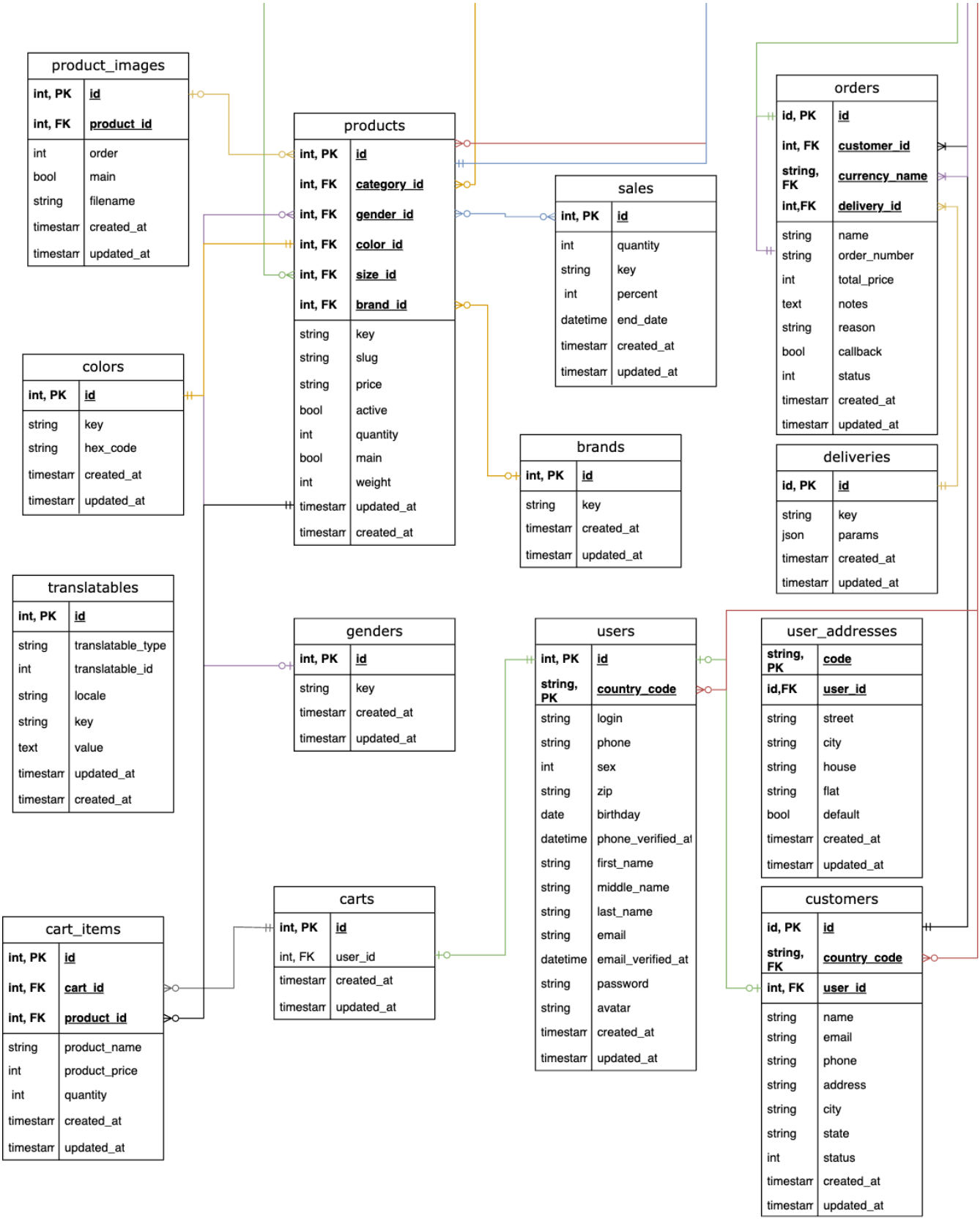


Рисунок 3.4 – ER діаграма бази даних для інформаційної системи підтримки клієнтів інтернет магазину одягу та аксесуарів з принтами, частина 2

Джерело: розроблено автором

Короткий опис кожної таблиці представлений у списку нижче:

1. carts: Відображає кошики покупок, створені користувачами на сайті.
2. categories: Категорії товарів.
3. colors: Включає доступні кольори для кожного товару.
4. countries: Містить список країн.
5. currencies: Зберігає інформацію про валюти.
6. customers: Відповідає за зберігання даних про клієнтів магазину.
7. deliveries: Записи про доставку, включаючи методи, терміни та статуси.
8. genders: Класифікація продуктів за статевим призначенням.
9. orders: Таблиця замовлень, що містить всю інформацію про покупки клієнтів.
10. order_items: Деталізація замовлень, яка відображає кожен товар у замовленні.
11. payment_methods: Доступні методи оплати в магазині.
12. payments: Записи про здійснені платежі та їх статус.
13. print_categories: Категорії принтів.
14. prints: Зображення принтів, які можуть бути використані на товарах.
15. product_images: Фотографії товарів.
16. products: Основна таблиця продуктів, що містить інформацію про всі товари.
17. telegram_chats: Інформація про чати в Telegram.
18. telegram_messages: Записи про повідомлення, надіслані через Telegram.
19. telegram_users: Дані користувачів, які взаємодіють з магазином через Telegram.
20. translatables: Містить переклади для інтерфейсу магазину.
21. user_addresses: Адреси користувачів для доставки.
22. users: Інформація про користувачів, які зареєстровані на сайті.

3.4 Моделювання варіантів використання

Для візуалізації взаємодії користувачів з системою та між системними компонентами була створена use case diagram, яка дозволила нам чітко представити

всі можливі сценарії використання системи. Use case diagram, або діаграма випадків використання, застосовують для графічного та вербального опису планованої функціональності програмного проекту. Вона є частиною мови об’єктного моделювання (UML) і дозволяє візуалізувати взаємодії між акторами (користувачами або зовнішніми системами) та програмною системою, яка розробляється.

Використання діаграм випадків дозволяє поліпшити аналіз та деталізацію функціональних вимог проекту, що зібрані на початковій фазі, та сприяє кращому розумінню та визначенню меж програмної системи, акторів, основних випадків використання та взаємодії між ними [45] Функціонал та очікувані результати взаємодії з кінцевим користувачем представлені на рис. 3.5

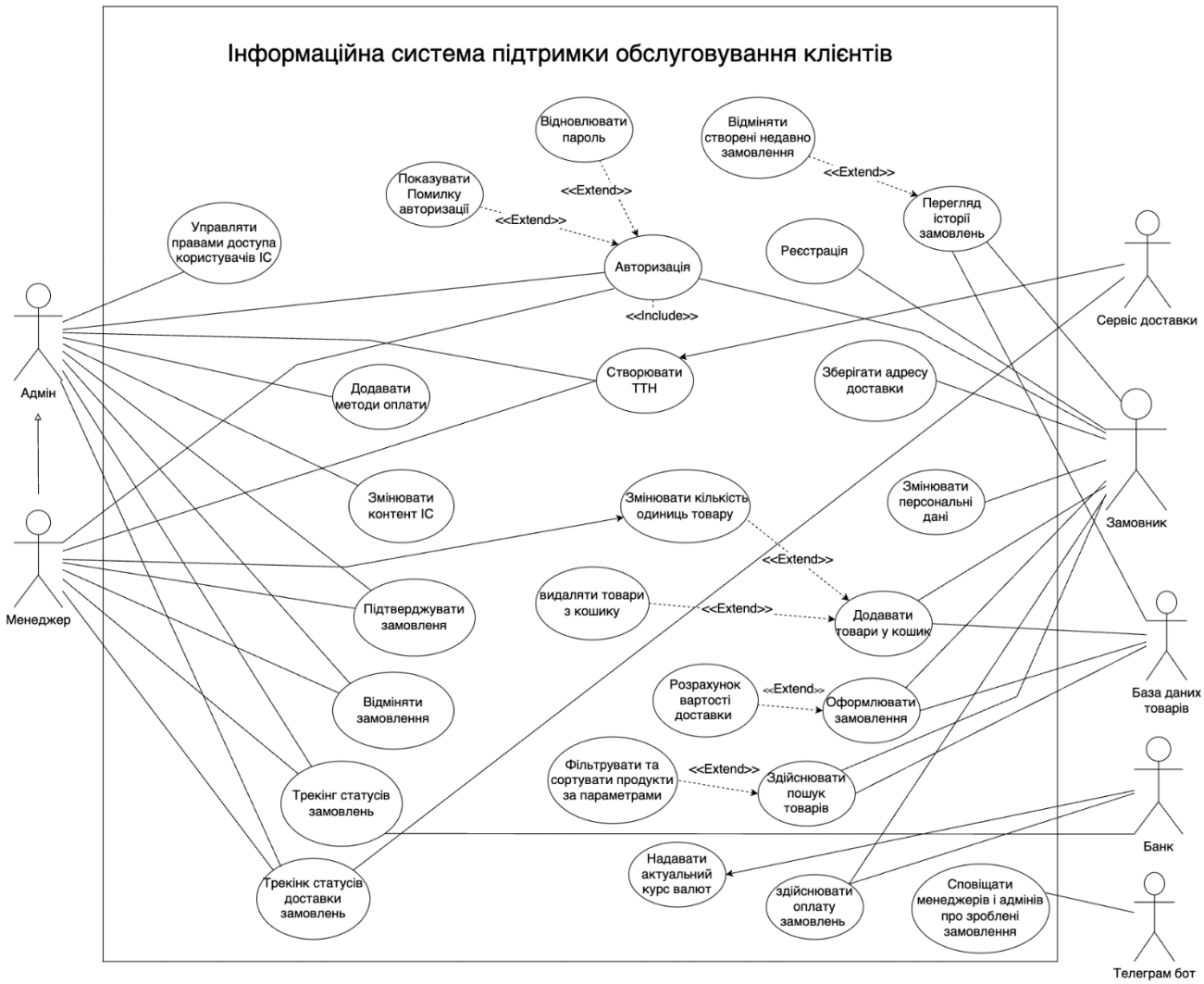


Рисунок 3.5 – Діаграма варіантів використання програмного продукту

Джерело: розроблено автором

4. ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Архітектура інформаційної системи

Центральними аспектами архітектури є модульність, масштабованість, безпека та легкість у підтримці. Модульність забезпечується за допомогою визначених інтерфейсів та відокремлених компонентів, які дозволяють легко додавати, оновлювати чи замінювати функціональні частини системи без впливу на інші модулі. Масштабованість гарантує, що система може адаптуватись до зростаючого навантаження, збільшуючи або зменшуючи ресурси. Безпека є пріоритетом, оскільки інформаційна система зберігає та обробляє конфіденційні дані клієнтів, інформацію про фінансові транзакції, тому система застосовує сучасні методи шифрування та протоколи доступу, такі як SSL/TLS (Secure Sockets Layer / Transport Layer Security)[46]. До методів шифрування які застосовуються у системі для зберігання паролів та іншої конфіденційної інформації відноситься bcrypt [47]. Легкість у підтримці досягається завдяки модульності, стандартизованим процесам розгортання, таким як CI/CD за допомогою GitHub Actions, та автоматизованим інструментам моніторингу, наприклад, Laravel Forge, які дозволяють швидко виявляти та усувати потенційні проблеми. Docker сприяє цьому процесу, надаючи засоби для створення ізольованого та уніфікованого середовища для ІС, що полегшує розгортання та тестування [33], [35].

Архітектура інформаційної системи має наступний стек технологій:

- Laravel Framework – це основа бекенду системи, яка використовується як об’єднувальна платформа для створення та управління всіма серверними аспектами інформаційної системи. Він слідує патерну MVC (Model-View-Controller) і пропонує великий набір функцій, таких як аутентифікація, роутинг, сесії, кешування та ORM (Object-Relational Mapping). Це забезпечує швидку та ефективну розробку, а також легкість у підтримці коду.

- Vue.js та Inertia.js – це інструменти, які дозволяють створювати інтерактивні користувацькі інтерфейси. Vue.js використовується для побудови динамічних компонентів на сторінці, а Inertia.js дозволяє створювати односторінкові додатки (SPA), використовуючи класичні серверні маршрути Laravel для навігації без необхідності побудови API.

- Redis та MySQL – це системи управління базами даних, де Redis використовується для кешування та як брокер черг, а MySQL – для зберігання постійних даних.

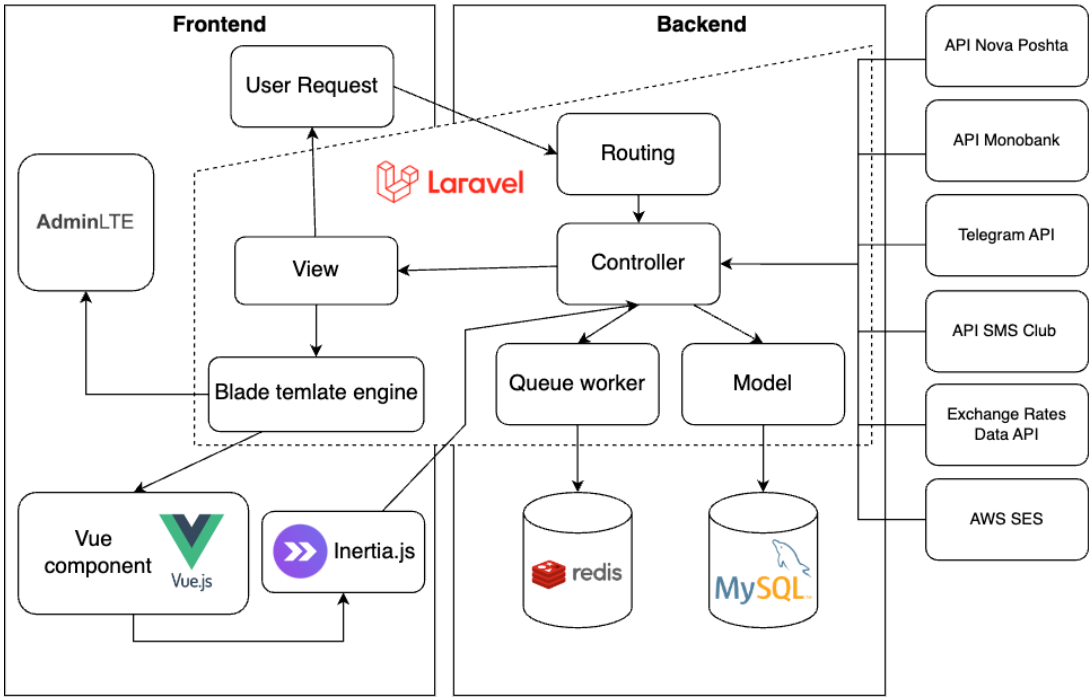


Рисунок 4.1 – Схема архітектури інформаційної системи підтримки обслуговування клієнтів інтернет-магазину одягу та аксесуарів з принтами.

Джерело: розроблено автором

Схема, представлена на рисунку 4.1, відображає архітектуру інформаційної системи (ІС) підтримки обслуговування клієнтів інтернет-магазину. Вона включає наступні компоненти:

- Routing – це процес визначення того, як веб-додаток відповідає на кожен вхідний запит користувача до сервера, включаючи URI (Uniform Resource Identifier) та специфічний HTTP (HyperText Transfer Protocol) метод (GET, POST і т.д.).
- Controller – це компонент, що керує взаємодією між моделями та видами. Він обробляє вхідні запити, виконує необхідну бізнес-логіку та повертає відповіді користувачу.
- View - відповідає за відображення інформації користувачу. На даній схемі View є мостом між користувачем та системою, який дозволяє їм інтерактивно взаємодіяти з веб-сайтом.
- Model - відповідає за взаємодію з базою даних, отримання даних, їхнє зберігання та обробку.
- Queue worker – це система, що дозволяє відкласти виконання завдань, які не вимагають негайної обробки, таких як відправлення електронних листів або сповіщення через Телеграм чи СМС шлюз.
- AdminLTE – це зручний інтерфейс, що забезпечує адміністраторам інтернет-магазину доступ до функціоналу управління інформаційної системи. Вона використовується для управління замовленнями, товарами та іншими адміністративними задачами.
- Blade template engine – це система шаблонів Laravel, яка забезпечує інструментами для створення динамічного HTML контенту. Шаблони Blade використовуються для генерації відповідей на запити користувачів.

Інформаційна система інтегрована зі сторонніми сервісами такими як Нова Пошта, Монобанк, ApiLayer, SMS Club, Telegram за допомогою REST API (Representational State Transfer Application Programming Interface). А інтеграція з AWS SES сервісом відбувається за допомогою вбудованого драйверу в фреймворк Laravel. Завдяки сторонній інтеграції зменшується час на обробку замовлень та підвищується задоволення клієнтів через швидке та зручне обслуговування. Кожен сервіс відповідає за свою задачу:

- API Nova Poshta – дозволяє системі інтегрувати функціонал логістичного сервісу, такий як створення накладних, відстеження статусів відправлень та отримання інформації про відправлення для клієнтів та менеджерів.
- API Monobank – забезпечує автоматизацію процесів підтвердження платежів та замовлень, що спрощує фінансову взаємодію між клієнтами та магазином.
- Telegram API – використовується для оперативного сповіщення користувачів, менеджерів, та адміністрацію магазину, дозволяючи відправляти повідомлення безпосередньо в Telegram.
- API SMS Club – дозволяє відправляти SMS-повідомлення клієнтам, що включає в себе одноразові паролі та інформацію про замовлення та відправлення.
- Exchange Rates Data API (ApiLayer) – це сервіс, який забезпечує актуальною інформацією про курси валют, необхідними для показу цін у різних валютах на сайті.
- AWS SES – сервіс електронної пошти від Amazon, який дозволяє системі відправляти листи з підтвердженнями замовлень та сповіщеннями про статуси.

У майбутньому ІС може бути розширена для впровадження додаткових функцій, таких як інтеграція штучного інтелекту для управління замовленнями, створення віртуальних помічників для клієнтів, автоматизація процесів управління ІС та інші інноваційні рішення, що спрямовані на покращення досвіду користувача та ефективності бізнес-процесів.

На етапі ініціювання проекту було використано інструмент Docker Compose разом з Laravel Sail для спрощення процесу розгортання багатоконтейнерної архітектури [48] Laravel Sail надає зручний інтерфейс командного рядка для взаємодії з Docker Compose, що дозволяє запустити всі сервіси, описані у файлі конфігурації docker-compose.yml, виконавши команду “sail up”. Цей інтегрований підхід є особливо корисним у розробці, тестуванні та етапах CI/CD, оскільки забезпечує консистентне середовище для всіх сервісів, які взаємодіють між собою, сприяючи ефективній розробці.

На рис.4.2 представлено налаштування контейнера для сервісу “web” – це основний контейнер додатку, що відповідає за обслуговування інформаційної

системи. Він будується на основі Dockerfile що встановлює всі необхідні залежності та конфігурації для розгортання та запуску Laravel додатку в контейнері.

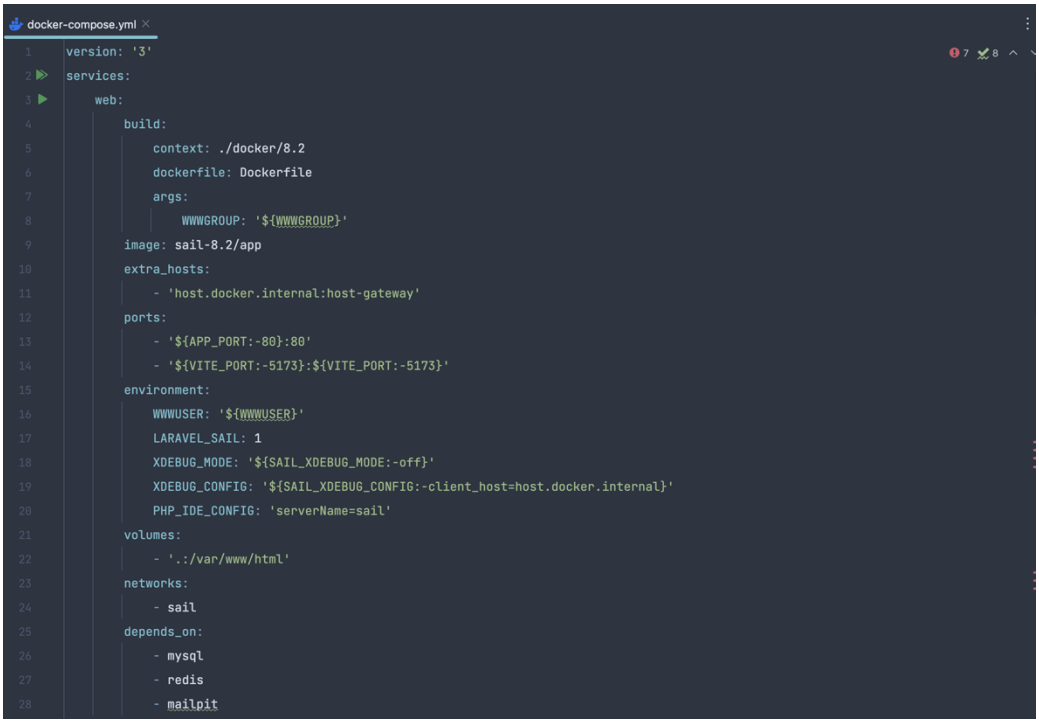
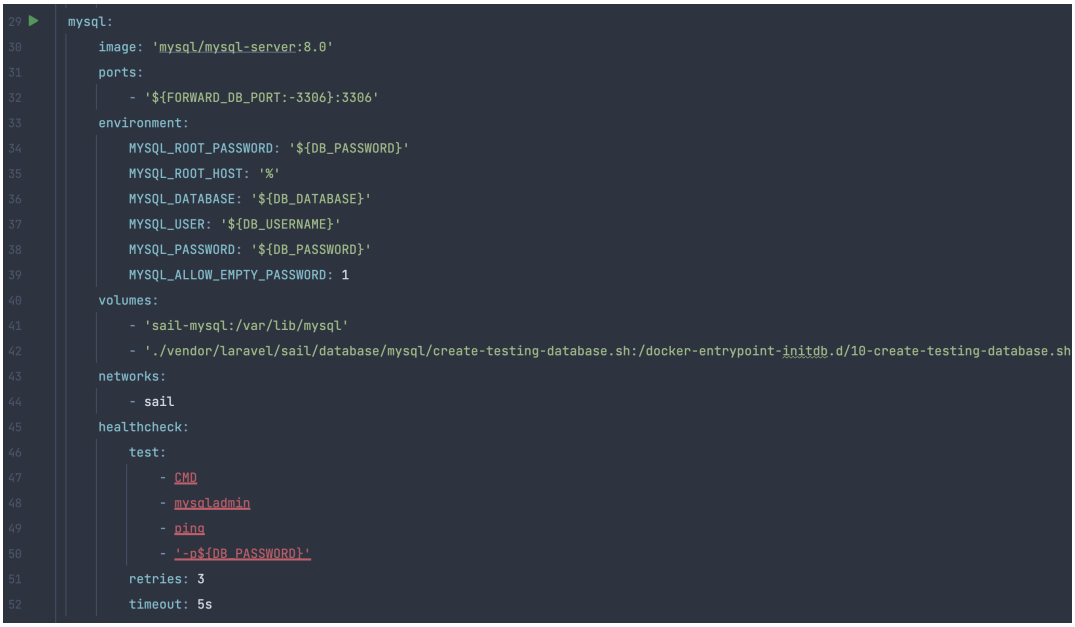


Рисунок 4.2 – Файл конфігурації Docker Compose. Налаштування сервісу web

Джерело: розроблено автором

На рис.4.3 зображено налаштування сервісу бази даних “mysql” де використовуються томи для збереження даних бази даних за межами контейнера для забезпечення постійності даних. До служби прив’язана мережа sail, яка забезпечує зв’язок з іншими контейнерами, що використовують цю ж мережу.

Рисунок 4.4 показує як налаштований сервіс бази даних “redis”, який виконує роль брокера черг та системи кешування в інформаційній системі.



```
mysql:
    image: 'mysql/mysql-server:8.0'
    ports:
        - '${FORWARD_DB_PORT:-3306}:3306'
    environment:
        MYSQL_ROOT_PASSWORD: '${DB_PASSWORD}'
        MYSQL_ROOT_HOST: '%'
        MYSQL_DATABASE: '${DB_DATABASE}'
        MYSQL_USER: '${DB_USERNAME}'
        MYSQL_PASSWORD: '${DB_PASSWORD}'
        MYSQL_ALLOW_EMPTY_PASSWORD: 1
    volumes:
        - 'sail-mysql:/var/lib/mysql'
        - './vendor/laravel/sail/database/mysql/create-testing-database.sh:/docker-entrypoint-initdb.d/10-create-testing-database.sh
    networks:
        - sail
    healthcheck:
        test:
            - CMD
            - mysqladmin
            - ping
            - '-p${DB_PASSWORD}'
        retries: 3
        timeout: 5s
```

Рисунок 4.3 – Файл конфігурації Docker Compose. Налаштування сервісу mysql

Джерело: розроблено автором

```
redis:
    image: 'redis:alpine'
    ports:
        - '${FORWARD_REDIS_PORT:-6379}:6379'
    volumes:
        - 'sail-redis:/data'
    networks:
        - sail
    healthcheck:
        test:
            - CMD
            - redis-cli
            - ping
        retries: 3
        timeout: 5s
```

Рисунок 4.4 – Файл конфігурації Docker Compose. Налаштування сервісу redis.

Джерело: розроблено автором

4.2 Реалізація бази даних для інформаційної системи

У процесі реаліазації бази даних (БД) для інформаційної системи було використано механізм міграцій, який є вбудованою функцією Laravel. Файли міграцій дозволяють визначити схему бази даних у вигляді коду, що сприяє контролю версій та спрощує управління змінами структури БД. Кожен файл міграції містить методи up() та down(), які відповідно використовуються для застосування та відкату змін.

На рисунках 4.5 – 4.6 представлено приклади файлів міграцій, які описують структуру таблиць для продуктів та категорій в базі даних інформаційної системи. У файлі міграції для таблиці товарів визначено поля, такі як ідентифікатор категорії (category_id), назву, ціну та інші характеристики товару. У файлі міграції для таблиці категорій описані поля, що включають назву категорії, зображення та опис. Зв’язок між таблицями встановлено через зовнішній ключ category_id, який реалізує зв’язок "один до багатьох": одна категорія може містити багато продуктів, але кожен продукт належить лише одній категорії. Ці міграції є лише частиною повного опису структури бази даних, де інша частина детально представлена у додатку Б. Варто зазначити, що це не кінцеві версії таблиць, оскільки в ході розробки інформаційної системи були виконані додаткові міграції, які доповнюють таблиці новими полями та відносинами, що відображає ітеративний процес розробки та постійне вдосконалення бази даних.
Для взаємодії з базою даних у розробленій системі використовуються Eloquent ORM моделі Laravel, які надають зручний інтерфейс для запитів та роботи з даними. Це забезпечує високий рівень абстракції від конкретної СУБД, дозволяючи зосередитися на бізнес-логіці застосунку замість нюансів SQL-синтаксису).

```php
<?php


use App\Enums\CurrencyEnum;
use App\Enums\GenderEnum;
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {
    /**
     * Run the migrations.
     */
    Andrii Polyvianyi *
    public function up(): void
    {
        Schema::create( table: 'products', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger( column: 'category_id');
            $table->foreignId( column: 'brand_id')->nullable()
                ->constrained()
                ->onDelete( action: 'set null');
            $table->string( column: 'name');
            $table->text( column: 'description')->nullable();
            $table->unsignedInteger( column: 'price');
            $table->string( column: 'currency_name')->default(CurrencyEnum::UAH->value)->nullable();
            $table->unsignedInteger( column: 'stock_quantity');
            $table->enum( column: 'gender', [GenderEnum::getValues()])->nullable();
            $table->integer( column: 'views')->default( value: 0);
            $table->dateTime( column: 'new_until')->default(DB::raw('(NOW() + INTERVAL 7 DAY)'));
            $table->timestamps();
            $table->foreign( columns: 'category_id')
                ->references( columns: 'id')
                ->on( table: 'categories')
                ->onUpdate( action: 'cascade')
                ->onDelete( action: 'cascade');
            $table->foreign( columns: 'currency_name')->references( columns: 'name')
                ->on( table: 'currencies')->onDelete( action: 'set null');
        });
    }

    /**
     * Reverse the migrations.
     */
    Andrii Polyvianyi
    public function down(): void
    {
        Schema::dropIfExists( table: 'products');
    }
};
```

Рисунок 4.5 – Файл міграції таблиці товарів

Джерело: розроблено автором

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {
    /**
     * Run the migrations.
     */
    Andrii Polyvianyi *
    public function up(): void
    {
        Schema::create( table: 'categories', function (Blueprint $table) {
            $table->id();
            $table->string( column: 'name');
            $table->string( column: 'img')->default( value: 'category_default.png');
            $table->text( column: 'description')->nullable();
            $table->unsignedBigInteger( column: 'parent_id')->nullable();
            $table->foreign( columns: 'parent_id')->references( columns: 'id')
                ->on( table: 'categories')->onDelete( action: 'cascade');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    Andrii Polyvianyi
    public function down(): void
    {
        Schema::dropIfExists( table: 'categories');
    }
};
```

Рисунок 4.6 – Файл міграції таблиці категорій

Джерело: розроблено автором

Eloquent є ORM (Object-Relational Mapping) системою фреймворку Laravel, що надає об’єктно-орієнтований інтерфейс для роботи з базами даних[49] Вона дозволяє

визначати моделі, що відповідають таблицям у базі даних, та пропонує зручні механізми для управління записами, включно з автоматичними часовими мітками, м’яким видаленням, масовим призначенням атрибутів, та захистом від SQL-ін’єкцій. У процесі розробки інформаційної системи були створені численні моделі Eloquent, кожна з яких відповідає за певну функціональність та сегмент даних у системі. Наприклад, модель Product використовується для представлення продуктів у системі, включаючи інформацію про назву, опис, ціну та інші характеристики. Модель Order управляє даними замовлень, тримаючи записи про статуси замовлень, інформацію про покупців та зв’язані з ними товари. Customer представляє клієнтів магазину, їхні контактні дані та історію покупок. PaymentMethod описує доступні способи оплати у системі. Модель User стандартна для більшості систем і керує реєстрацією, авторизацією та профілями користувачів.

Кінцевий вигляд структури частини таблиць бази даних зображений на рис 4.7, де зображені взаємозв’язок між різними елементами даних. Схема ілюструє реляційні зв’язки з ключами, атрибутами та типами даних, вказуючи на складність та взаємозалежність моделей в системі. Повна структура бази даних із всіма таблицями та відносинами між ними включена до додатку Б.

Для підключення БД MySQL до інформаційної системи було використано вбудований драйвер Laravel. Конфігурація з’єднання з базою даних визначена у файлі “config/database.php”.

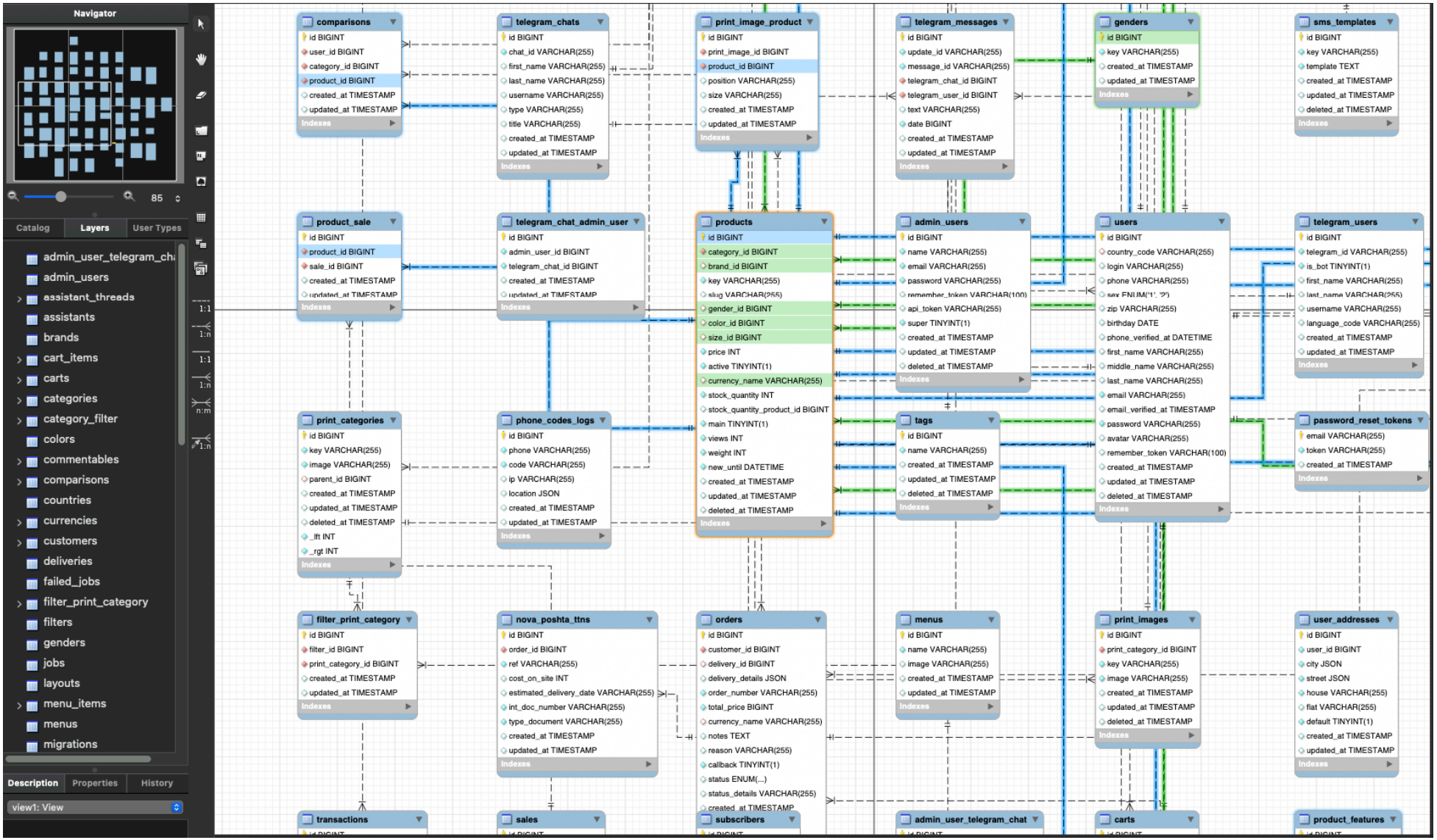


Рисунок 4.7 – Схема реалізованої бази даних

Джерело: розроблено автором

4.3 Реалізація клієнтської частини

Розробка клієнтської частини інформаційної системи підтримки обслуговування клієнтів поділена на дві частини, це розробка інтерфейсу для клієнтів, та розробка інтерфейсу для аміністрації інтернет магазину.

Розглянемо першу частину – розробку інтерфейсу для клієнтів. Цей інтерфейс є вітриною інтернет-магазину, де користувачі можуть переглядати продукти, управляти своїми замовленнями та особистими даними. Для забезпечення зручності та інтуїтивного дизайну використовуються сучасні технології фронтенду, такі як Vue.js, зі спрощенням процесів навігації та оформлення замовлення (рис. 4.8). Окрім того, інтерфейс оптимізований для різних пристроїв, від мобільних телефонів до настільних комп'ютерів, що забезпечує відмінний користувацький досвід незалежно від платформи.

app.blade.php

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <title inertia>{{config('app.name')}}</title>
    @include('helpers.head')
    <!-- Scripts -->
    @routes
    @vite(['resources/sass/base.scss',
            'resources/js/app.js',
            "resources/js/Pages/{$page['component']}.vue"])
    @inertiaHead
    <x-translations/>

</head>
<body>
@include('includes.base.preloader')
@inertia
<!-- scripts -->
@include('helpers.body')
<!-- /scripts -->
</body>
</html>
```

Рисунок 4.8 – Шаблон основного документа Blade інформаційної системи з використанням Inertia.js

Джерело: розроблено автором

Точкою входу для клієнтської частини інформаційної системи для клієнтів слугує корінний файл blade, який використовує Inertia.js, як зображено на рисунку 4.8. Inertia.js дозволяє рендерити JavaScript-компоненти на стороні клієнта, зберігаючи стан сторінки без повної її перезавантаження, що сприяє підвищенню швидкості та зручності користувацького досвіду. В коді за допомогою blade синтаксису описані ключові директиви за допомогою який відбувається побудова кілєнтської частини:

- Директива @routes генерує JavaScript-об’єкт з усіма маршрутами Laravel, що дозволяє використовувати імена маршрутів у клієнтських Vue компонентах.
- @inertia служить як вихідний пункт для Inertia.js, вказуючи, де має бути ініціалізований кореневий Vue компонент. Це означає, що будь-який інший JavaScript-компонент або логіка будуть впроваджені та управлятися через цей кореневий компонент.
- @inertiaHead використовується для вставки метаданих та інших елементів HTML <head>, що динамічно генеруються з JavaScript-компонентів. Це дозволяє, наприклад, змінювати заголовки сторінок або метатеги залежно від поточного стану SPA (Single-Page Application), що є корисним для SEO та користувацького досвіду.
- Директива @vite в Laravel є частиною інтеграції з Vite, інструментом для сучасної розробки фронтенду, який забезпечує швидке перезавантаження модулів та оптимізовану побудову ресурсів. Коли ця директива використовується в файлі blade, вона автоматично генерує необхідні HTML теги для підключення JavaScript та CSS ресурсів, які були скомпільовані за допомогою Vite. Інструмент Vite займається компіляцією та заміною модулів. Це підвищує продуктивність розробки та дозволяє миттєво бачити зміни у браузері без необхідності ручного оновлення.

Після того як корінний Blade шаблон з директивою @inertia завантажується в браузері, відбувається ініціація процесу рендерингу SPA (Single Page Application). Inertia.js, який інтегрований з Laravel на стороні серверу, відправляє ініціалізаційні дані, включаючи стан та Vue компоненти, які повинні бути відрендерені на стороні клієнта. Ці дані передаються у формі JSON відповіді, яка містить необхідні дані для генерації потрібного юзер-інтерфейсу. На клієнтській стороні, Vite виконує роль

менеджера збірки, компілює Vue компоненти, та завантажує основний JavaScript файл app.js, що створює інстанс Inertia додатку.

На рис. 4.9 представлено основний JavaScript файл app.js, який є ключовим для ініціалізації фронтенду ІС. Цей файл включає в себе виклики функцій та бібліотек, необхідних для запуску та координації роботи інтерфейсу користувача, зокрема, визначення кореневих компонентів, конфігурації роутингу, та інтеграції з Vue.js через Inertia.js.

```
const app = createInertiaApp({
    title: (title) : string  => `${title} - ${appName}`,
    resolve: (name) =>
        resolvePageComponent(
            `./Pages/${name}.vue`,
            import.meta.glob("./Pages/**/*.vue")
        ),
    setup({ el, App, props, plugin }) {
        const app = createSSRApp({ render: () => h(App, props) })
            .use(plugin)
            .mixin(translations)
            .mixin(common)
            .mixin(notifier)
            .use(ZiggyVue, Ziggy)
            .use(Notifications)
            .use(FloatingVue)
            .use(store);

        app.config.globalProperties.$request = $request;
        app.config.globalProperties.$search = $search;
        app.mount(el);
        return app;
    },
```

Рисунок 4.9– Ініціалізація фронтенду в app.js

Джерело: розроблено автором

В таблиці 4.1 наведений перелік компонентів сторінок із стислим описом їх задач.

Таблиця 4.1– Створені компоненти сторінок та їх задача.

№	Назва компоненту	Задача
1	Main.vue	Відображення головної сторінки інформаційної системи.
2	Shop/Products.vue	Відображення сторінки товарів, із внутрішніми компонентами TopBar, ProductSidebar, ProductList та інші, що формують кінцевий вигляд сторінки товарів .
3	Shop/Product.vue	Відображення сторінки товару.
4	Auth/Login.vue	Відображення сторінки авторизації клієнтів у власний кабінет.
5	Auth/Register.vue	Відображення сторінки реєстрація клієнтів в інформаційній системі.
6	Auth/ForgotPassword.vue	Відображення сторінки відновлення паролю до кабінету користувача.
7	Comparisons/Comparisons.vue	Відображення сторінки порівнянь товарів.
8	Wishlist.vue	Відображення сторінки бажаних товарів.
9	Cart/Cart.vue	Відображення сторінки кошика.
10	Profile/Order.vue	Відображення сторінки історії замовлень клієнта.
11	Payment/Checkout.vue	Відображення сторінки оформлення замовлень.

Джерело: розроблено автором

На рис. 4.10 Зображено результат побудови сторінки бажаних товарів, де користувач може переглядати товари, які він додав до цього списку. Інші компоненти додані до додатку Б.

```vue
<template>
  <div>
    <Head :title="__('products.head')" />
    <Breadcrumbs v-bind="{breadcrumbs}" />
    <section class="section">
      <div class="container small-size">
        <template v-if="wishlist.data.length > 0">
          <ProductList
            :products="wishlist.data"
            :meta="wishlist.meta"
            :links="wishlist.links"
          />
        </template>
        <div
          v-else
          class="text-center"
        >
          {{ __('wishlist.empty_wishlist') }}
        </div>
      </div>
    </section>
  </div>
</template>
```

Рисунок 4.10 – Wishlist.vue - Vue компонент сторінки бажаних товарів

Джерело: розроблено автором

Щодо інтерфейсу для адміністрації, то він розроблений з акцентом на функціональність та ефективність керування ІС. Інструменти управління контентом, замовленнями, клієнтською базою, відстеження відправлень, управління товарами є інтегрованими в одному місці для зручності менеджерів та адміністраторів. В даному випадку використовувався готовий шаблон AdminLTE для побудови функціонального інтерфейсу. Для зручної та швидкої розробки використовувася сторонній пакет “jeroennoten/laravel-adminlte”. AdminLTE інтегрується з Laravel через Blade шаблони, забезпечуючи масив компонентів для відображення різноманітних UI елементів, таких як форми, кнопки, сповіщення, іконки та інші елементи управління,. Також пакет дозволяє настроїти візуальний стиль інтерфейсу ІС для адміністраторів відповідно до вимог. Конфігурацію представлено у файлі “config/adminlte.php”.

На рисунку 4.11 зображений шаблон Blade, який представляє головний інтерфейс користувацького виду сторінки замовлень ІС, призначений для адміністраторів та менеджерів. Blade види включають в себе використання компонентів Blade, що значно спрощують процес розробки та підтримки коду шляхом інкапсуляції повторюваних елементів інтерфейсу користувача. Компоненти Blade використовуються для створення модульного та перевикористовуваного коду, дозволяючи легко модифікувати та оновлювати окремі частини інтерфейсу без необхідності вносити зміни в кожен вид. Це забезпечує більш чисту та організовану структуру коду, що підвищує читабельність та підтримку коду та зменшує ймовірність помилок.

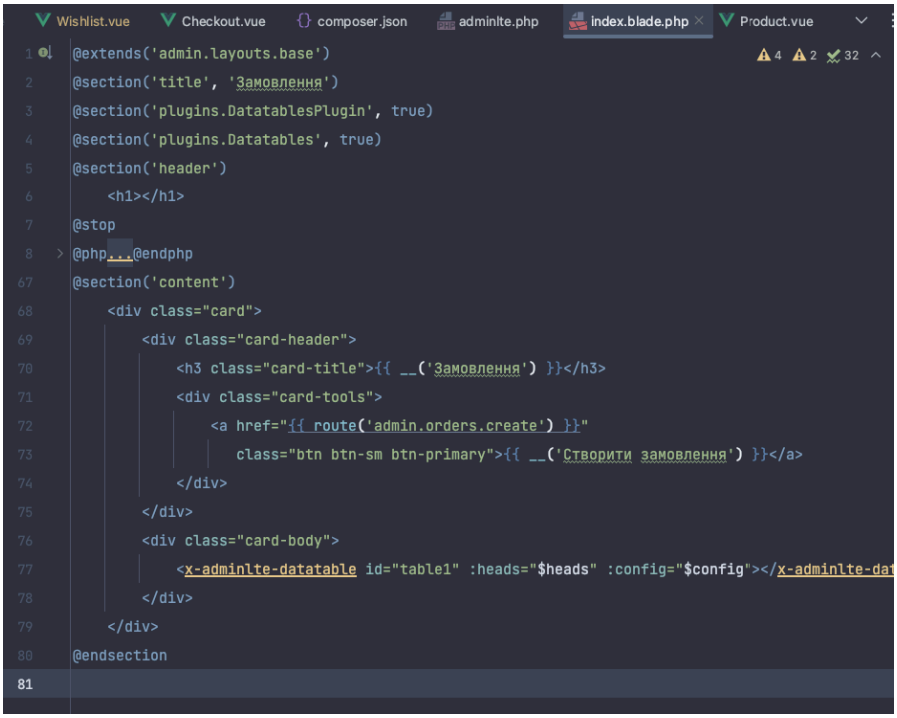


Рисунок 4.11 – Шаблон Blade виду, який відображає сторінку замовлень ІС для адміністраторів і менеджерів

Джерело: розроблено автором

4.4 Реалізація серверної частини

Серверна частина поділена на дві основні підсистеми: одна обробляє запити від клієнтів, а інша – від адміністраторів. Клієнтська підсистема включає функціональність створення та керування замовленнями, взаємодію з кошиком покупок, пошук та фільтрацію товарів, реєстрацію, аутентифікацію та підтвердження користувачів за допомогою OTP (One-Time Password) тобто "одноразового паролю", маніпуляції з валютами та локалізацію. Інтеграція з зовнішніми сервісами, такими як системи доставки та курсів валют, розширює функціональні можливості системи, дозволяючи автоматизувати процеси, пов’язані з логістико.

Підсистема для адміністраторів займається управлінням товарним асортиментом, обробкою замовлень, керуванням правами доступу користувачів за допомогою системи RBAC, сповіщенням менеджерів та адміністрації через месенджер Телеграм, створенням ТТН та підтвердженням оплат.

Розглядаючи першу підсистему, можна виділити функціонал оформлення замолення, що представлений на рис. 4.13. Код представлений в контроллері CheckoutController містить в собі метод submitCheckoutForm, що виконується при обробці POST запиту (рис. 4.12), здійснює валідацію форми замовлення та ініціює створення запису замовлення в базі даних. Він використовує сервіси, такі як CustomerService, OrderService та інші, для виконання бізнес-логіки, пов’язаної з обробкою замовлень. Наприклад, для перевірки та підтвердження номеру телефону клієнта через OTP, використовується PhoneConfirmationService. Це забезпечує додатковий рівень безпеки і знижує ризик шахрайства. Крім обробки даних форми замовлення, та створення нового замовлення, здійснюється прив’язка замовлення до клієнта та ініціювання процесу оплати. Також цей метод використовує PaymentService для обробки платіжних даних та CartService для очищення кошика після успішного оформлення замовлення. Також за допомогою подій та слухачів, таких як event(new OrderCreated($order));, система також може автоматично

надсилати сповіщення користувачам, адміністраторам та менеджерам про нові або змінені замовлення, що забезпечує їх швидке оброблення та виконання.

Рисунок 4.12 – Конфігурація маршрутів Laravel для чекауту

Джерело: розроблено автором

Розглядаючи другу підсистему, можна виділити функціонал підтвердження, скасування замовлень. На рис. 4.14 представлено фрагмент коду, що ілюструє методи в контролері \Admin\OrderController, які відповідають за логіку підтвердження та скасування замовлень.

```
    public function submitCheckoutForm(
        SubmitCheckoutRequest $request,
        CustomerServiceInterface $customerService,
        PhoneConfirmationService $phoneConfirmationService,
        OrderServiceInterface $orderService,
        CartServiceInterface $cartService,
        PaymentServiceInterface $paymentService,
        PaymentMethodRepositoryInterface $paymentMethodRepository
    ) {
        $data = $request->validated();

        if (!$phoneConfirmationService->checkPhoneVerification($data['phone'], $data['confirm_phone_c
            return back()->withInput()->withErrors(['phone' => __( key: 'Phone is not verified.')]);
        }
        $customer = $customerService->createCustomerIfNotExist([
            "user_id" => Auth::id(),
            'name' => $data['lastName'] . " " . $data['firstName'],
            'phone' => $data['phone'],
            'email' => $data['email'],
            'address' => $customerService->generateAddress(),
            'city' => $customerService->getCity(),
            'state' => $customerService->getState(),
            'country' => $customerService->getCountry(),
            'postal_code' => $customerService->getPostalCode(),
            'ip' => $customerService->getIpAddress(),
            'currency' => $customerService->getCurrency(),
            'coordinates' => $customerService->getCoordinates(),
        ]);
        $order = $orderService->createOrder($customer, $data);
        if (!$order) {
            return back()->with('error', __( key: 'The order could not be created, please contact sup.
        }

        $paymentMethod = $paymentMethodRepository->getPaymentMethodByKey($data['paymentMethod']);
        $paymentService->createPayment($order, $paymentMethod);

        $cartService->clearCart();

        event(new OrderCreated($order));

        if ($data['call_me_back']) {
            return redirect(route( name: 'products.index'))->with('success', __( key: 'Your order has b
        }

        return redirect(route( name: 'products.index'))->with(
            'success',
            __( key: 'Order placed, thank you for your purchase, stay wi...')
        );
    }
}
```

Рисунок 4.13 – метод обробки форми оформлення замовлення для клієнтів.

Джерело: розроблено автором

```
1 usage   Andrii Polyvianyi
public function confirmOrder(Order $order, OrderServiceInterface $orderService)
{
    try {
        if ($orderService->confirmOrder($order)) {
            return back()->with('success', 'Замовлення успішно підтверджено.');
        }
    } catch (\Exception $exception) {
        return back()->with('error', $exception->getMessage());
    }

    return back()->with('error', 'Не можливо підтвердити замовлення.');
}

1 usage   Andrii Polyvianyi *
public function cancelOrder(Request $request, Order $order, OrderServiceInterface $orderService)
{
    $request->validate([
        'reason' => 'in:' . join( separator: ',', ReasonsCancelOrderEnum::getValues()),
        'other_reason' => 'nullable|string',
        'notify_customer' => 'nullable'
    ]);
    if ($request->reason === ReasonsCancelOrderEnum::OTHER->value) {
        $reason = ReasonsCancelOrderEnum::from($request->reason)->getLabel() . " "
            . $request->other_reason;
    } else {
        $reason = ReasonsCancelOrderEnum::from($request->reason)->getLabel();
    }

    $orderService->setOrderCanceled($order, $reason,  notify_customer: $request->notify_customer == 'on');

    return back()->with('success', 'Замовлення успішно скасовано.');
}
```

Рисунок 4.14 – Методи підтвердження та скасування замовлень для адміністраторів

Джерело: розроблено автором

Метод confirmOrder в OrderService перевіряє статус платежу та статус замовлення (рис.4.15). Якщо платіж виконано готівкою або успішно оброблено іншим способом, він оновлює статус замовлення відповідно до результату платежу. При успішному платежі, якщо замовлення ще не було оплачене, воно встановлює статус "очікування платежу". Якщо замовлення вже було оплачене, встановлюється статус "в обробці". Метод cancelOrder в OrderService встановлює статус замовлення “Скасовано” та причину скасування, якщо вона була передана в запиті. Після цього замовлення зберігається з новим статусом у БД та тригерить подію OrderCanceled, яка може сповістити клієнта про скасування замовлення, якщо передано параметр notify_customer.

```php
public function confirmOrder(Order $order): bool
{
    $payment = $order->payment;
    if (!$payment->paymentMethod->isCash() && !$payment->isSuccess()) {
        $order->status = OrderStatusEnum::PENDING_PAYMENT->value;
        if ($saved = $order->save()) {
            event(new PendingPaymentForOrder($order));
            return $saved;
        }
    } else {
        if ($payment->paymentMethod->isCash() || $payment->isSuccess()) {
            $order->status = OrderStatusEnum::PROCESSING->value;
            if ($saved = $order->save()) {
                event(new OrderConfirmed($order));
                return $saved;
            }
        }
    }
    return false;
}
```

Рисунок 4.15 – бізнес-логіка підтвердження замовлення в OrderService класі.

Джерело: розроблено автором

В таблиці 4.2 перелічені можливі варіанти статусів замовлень

Таблиця 4.2 Можливі статуси замовлень.

№	**Статус замовлення**
1	Очікує підтвердження
2	Очікує оплати
3	Готується до відправки
4	Відправлено
5	Статуси з Нової Пошти: Прибув на відділення, Відправлення прямує до міста YYYY…
7	Виконано
8	Скасовано

Джерело: розроблено автором

В таблиці наведені статуси замовлень, які можуть бути, що допомагають і як внутрішнім менеджерам, так і клієнтам зрозуміти, на якому етапі знаходиться замовлення, і включають етапи, як-от очікування на підтвердження, очікування на оплату, підготовка до відправки, статуси доставки та інші. Це підвищує задоволеність клієнтів завдяки прозорості процесу.

4.5 Тестування та розгортання системи

У рамках розробки проводилось ретельне тестування системи, було розроблено 121 автоматизованих тести, які були виконані та успішно пройдені. (рис. 4.16) Це охоплює широкий спектр функціональності, включаючи перевірки частини функціоналу ведення замовлення, інтерфейсу ведення замовлення, доступу для адміністрації та користувачів, а також основні веб-сторінки та їх відгуки на запити.

Окрім автоматизованого тестування, було також здійснено ретельне ручне тестування, що додатково підтвердило надійність та стабільність роботи системи.

Інформаційна система була успішно розгорнута на віртуальному сервері Digital Ocean (рис.4.17), використовуючи Laravel Forge для спрощеного управління сервером та автоматизації задач (рис. 418). Для підвищення безпеки та прискорення завантаження, система була підключена до сервісу CloudFlare (рис.4.19), що забезпечує додатковий шар захисту та кешування контенту. Було налаштовано зашифрований https протокол для забезпечення конфіденційності даних користувачів. Також впроваджений механізм автоматичного розгортання через CI/CD за допомогою Github Actions, що забезпечує неперервну інтеграцію та доставку змін у продакшн без зайвих затримок (рис.4.20).

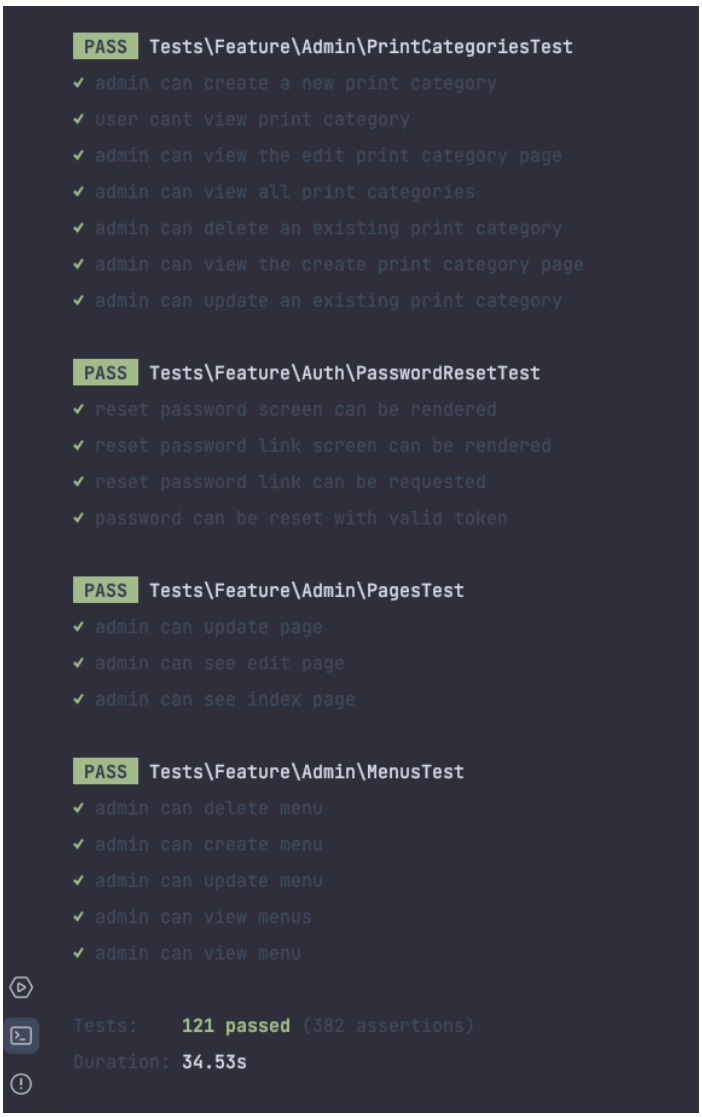


Рисунок 4.16 – Автоматизоване тестування інформаційної системи підтримки обслуговування клієнтів

Джерело: розроблено автором

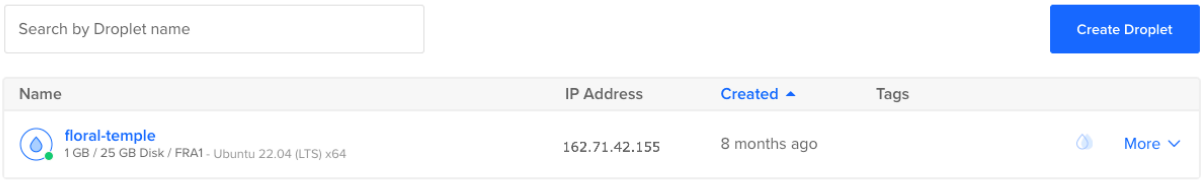


Рисунок 4.17 – Дроплет з розгорнутою ІС в провайдері хмарних інфраструктур Digital Ocean

Джерело: розроблено автором

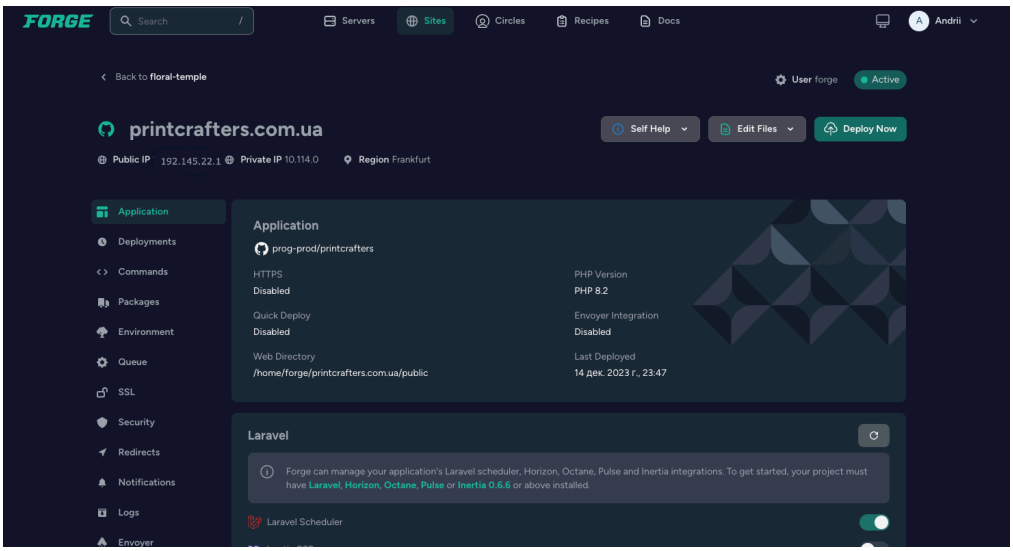


Рисунок 4.18 – Інтерфейс управління Laravel Forge для сайту printcrafters.com.ua

Джерело: розроблено автором

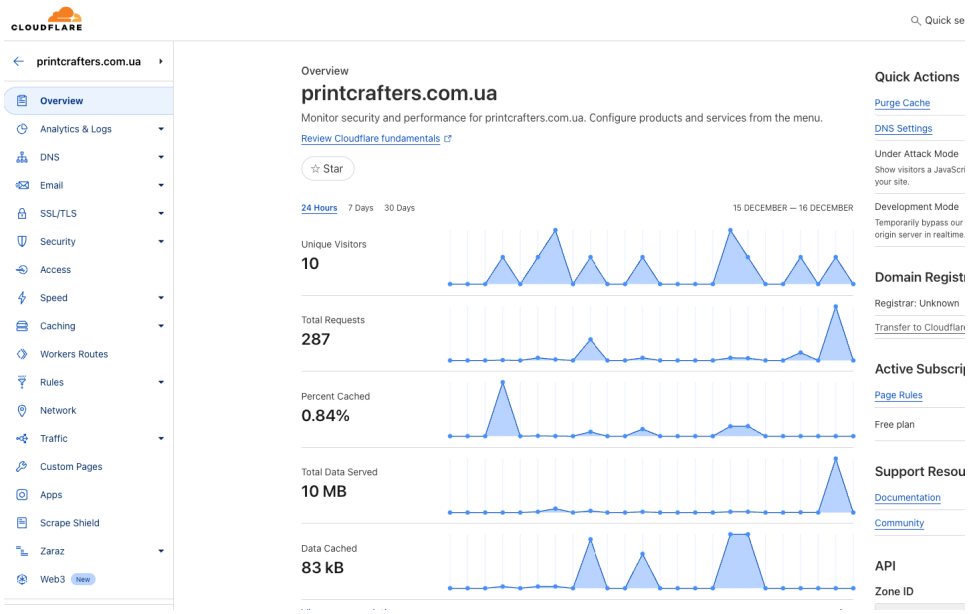


Рисунок 4.19 – Сервіс CloudFlare для сайту printcrafters.com.ua

Джерело: розроблено автором

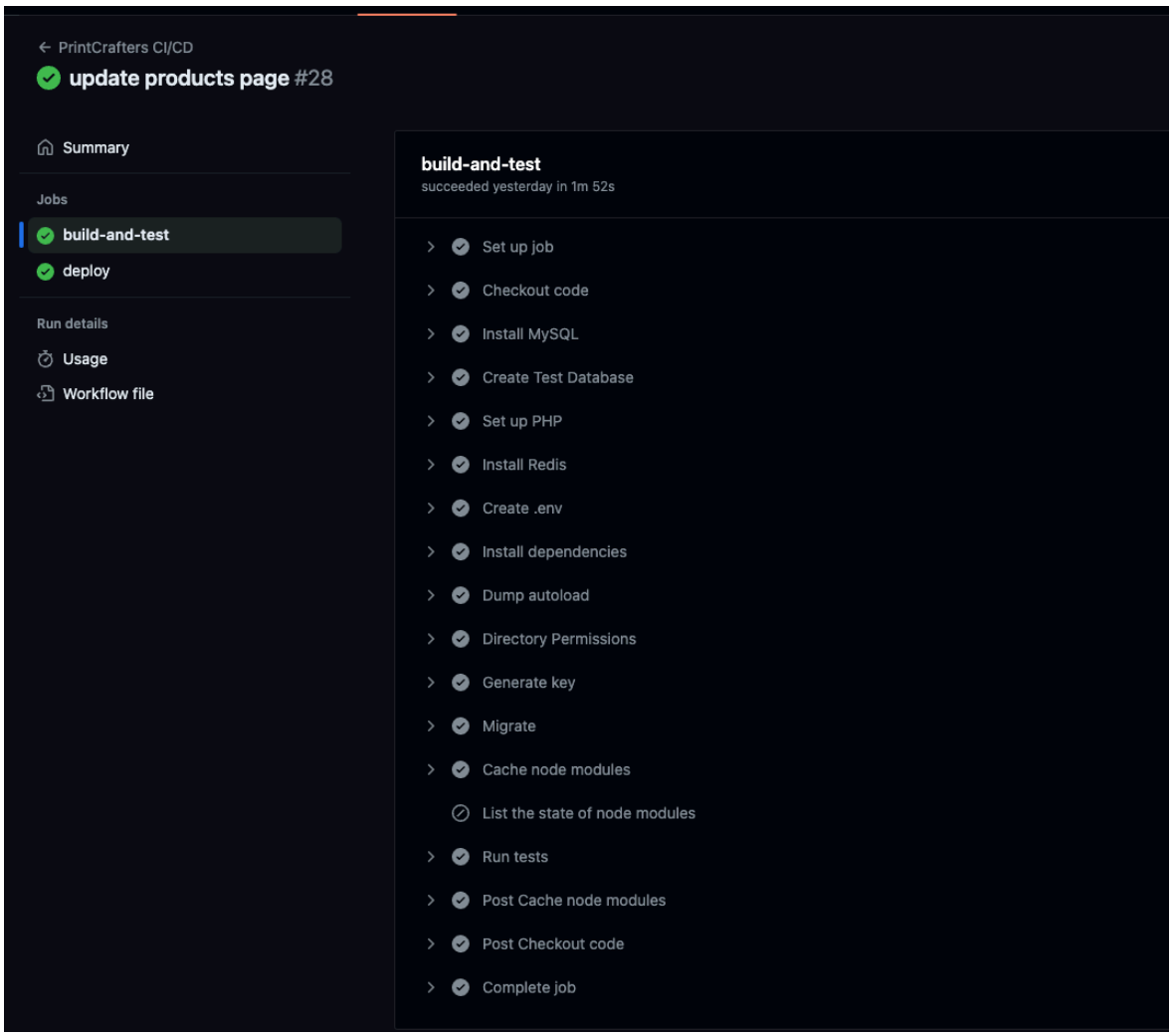


Рисунок 4.20 – Механізм автодеплоя через CI/CD GitHub Actions

Джерело: розроблено автором

4.6 Настанови з використання інформаційної системи клієнтам

На початку використання користувача зустрічає головна сторінка, де показані товари з кращих колекцій, акційні пропозиції, список категорій. В верхній частині інформаційної системи знаходиться головне навігаційне меню, де знаходяться списки посилань на сторінку категорій товарів. При натисканні на посилання в головному меню користувач потрапляє на сторінку товарів з вибраної категорії (рис.4.21).

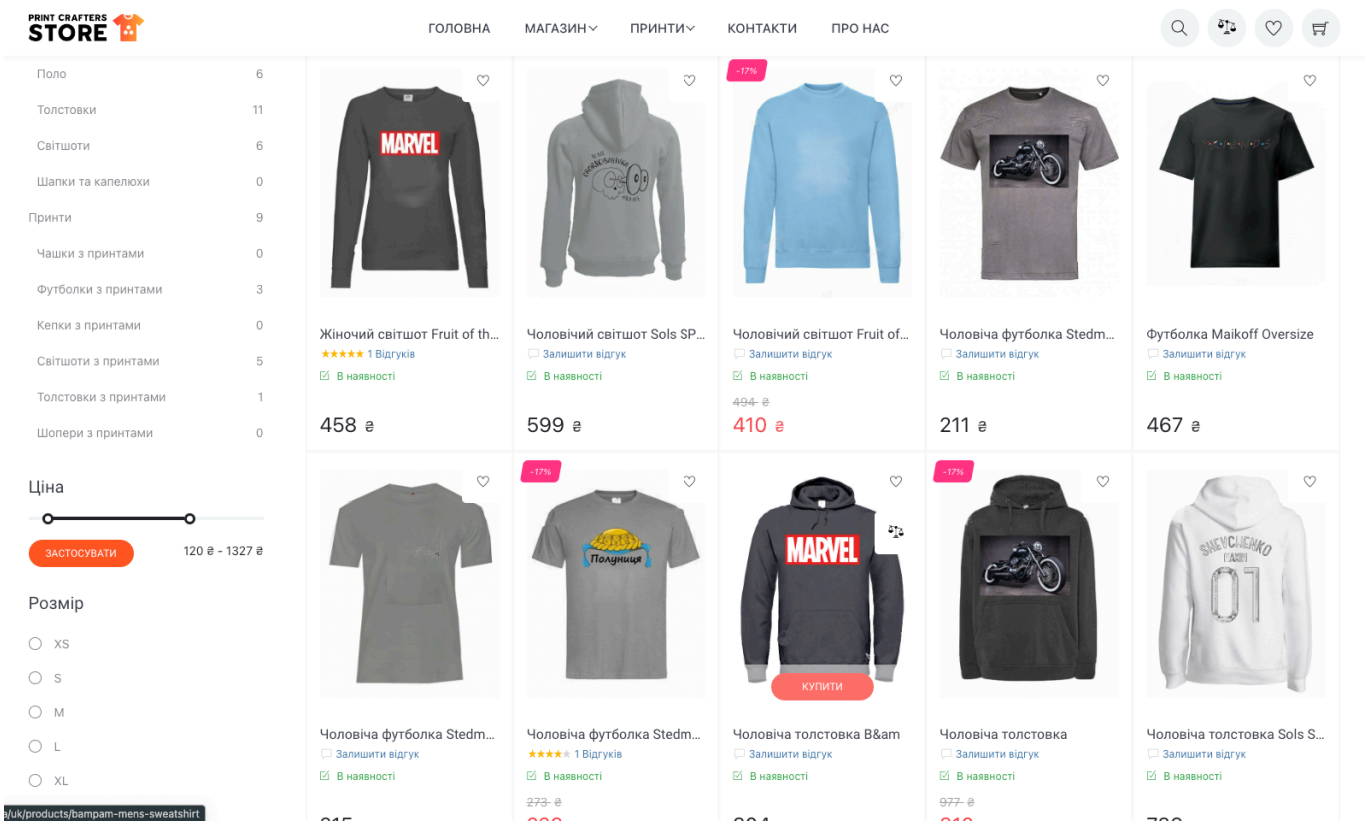


Рисунок 4.21 – Сторінка товарів для клієнтів в ІС підтримки обслуговування клієнтів інтернет-магазину одягу та аксесуарів з принтами

Джерело: розроблено автором

Після вибору потрібного товару клієнт натискає кнопку купити та переходить на сторінку вибраного товару (рис. 4.22). Після цього він може вибрати кількість одиниць товару відповідно до кількості товарів на складі, вибрати потрібні параметри товару, розмір, колір, додати товар до списку бажаних товарів, додати товар до списку порівнянь та додати його до кошика. Крім того на цій сторінці клієнт може переглянути схожі товари з цим принтом, якщо у товара є принт, та схожі товари.

Якщо користувач захоче додати більше одиниць товару, він може це зробити через сторінку кошику клієнта (рис.4.23). Звідти може перейти на сторінку оформлення замовлення (рис.4.24), де потрібно заповнити форму замовлення: Ім’я, Прізвище, Мобільний телефон, Електронна пошта, Спосіб доставки, Спосіб оплати.

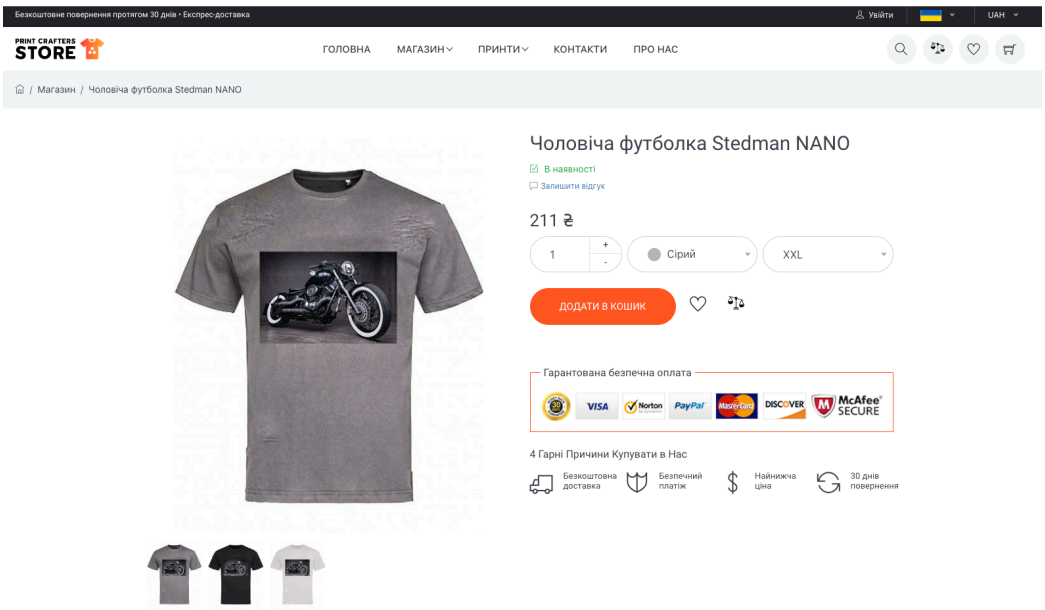


Рисунок 4.22 – Сторінка товару для клієнтів

Джерело: розроблено автором

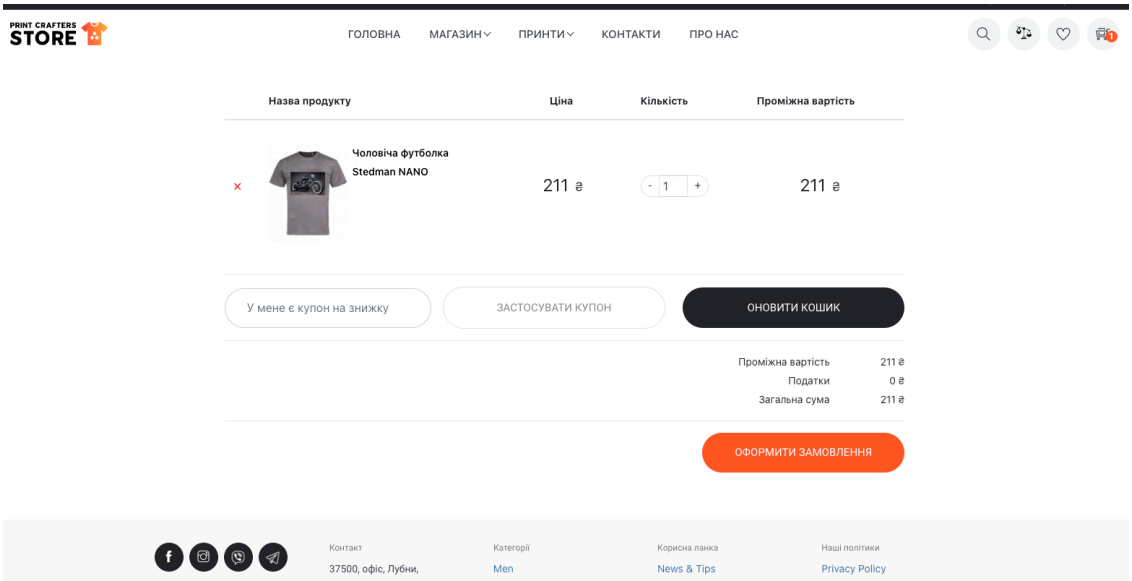


Рисунок 4.23 – Сторінка кошику клієнта

Джерело: розроблено автором

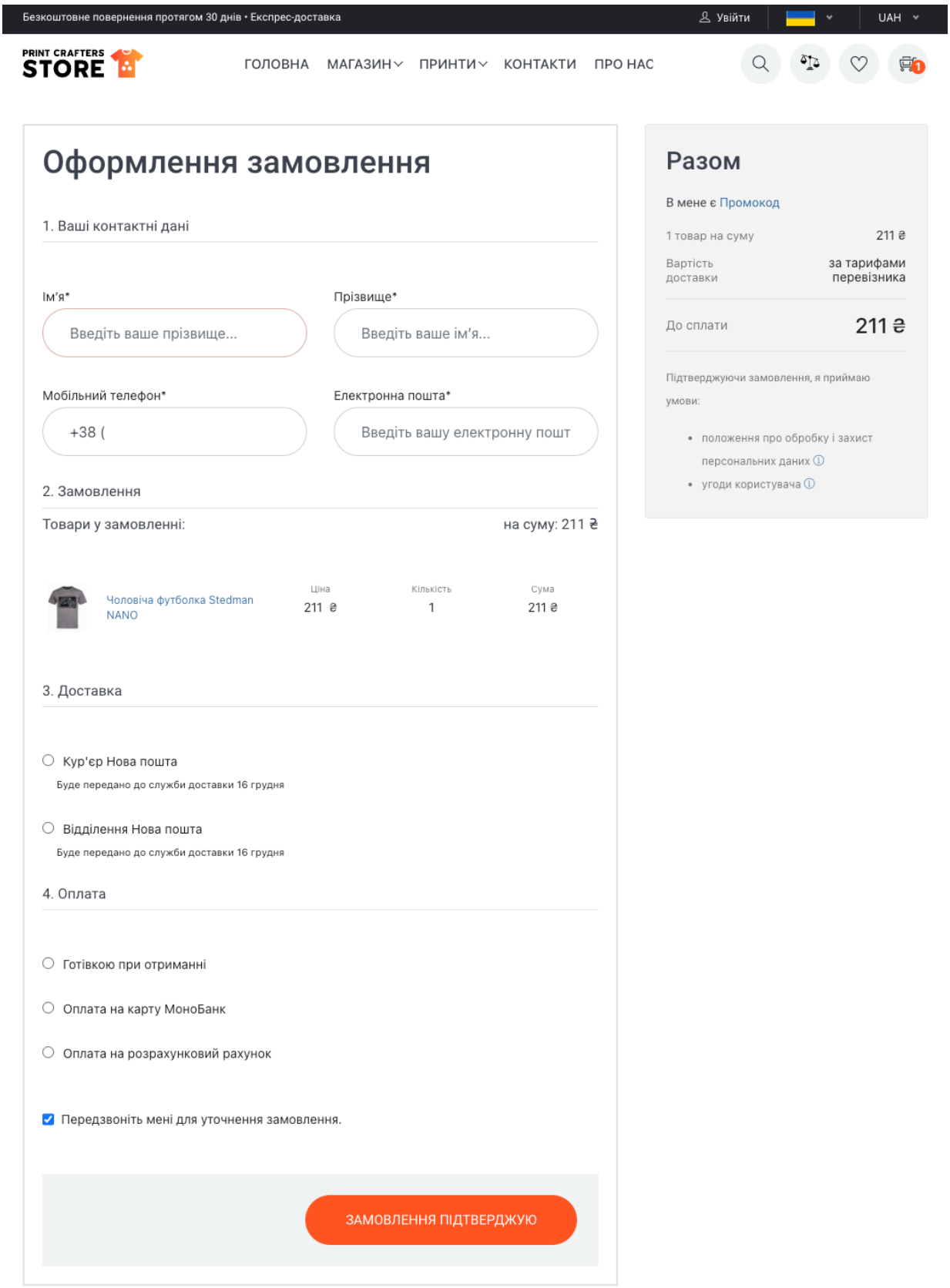


Рисунок 4.24 – Сторінка оформлення замовлення

Джерело: розроблено автором

Після заповнення форми замовлення клієнт повинен підтвердити номер телефону за допомогою OTP (рис.4.25).

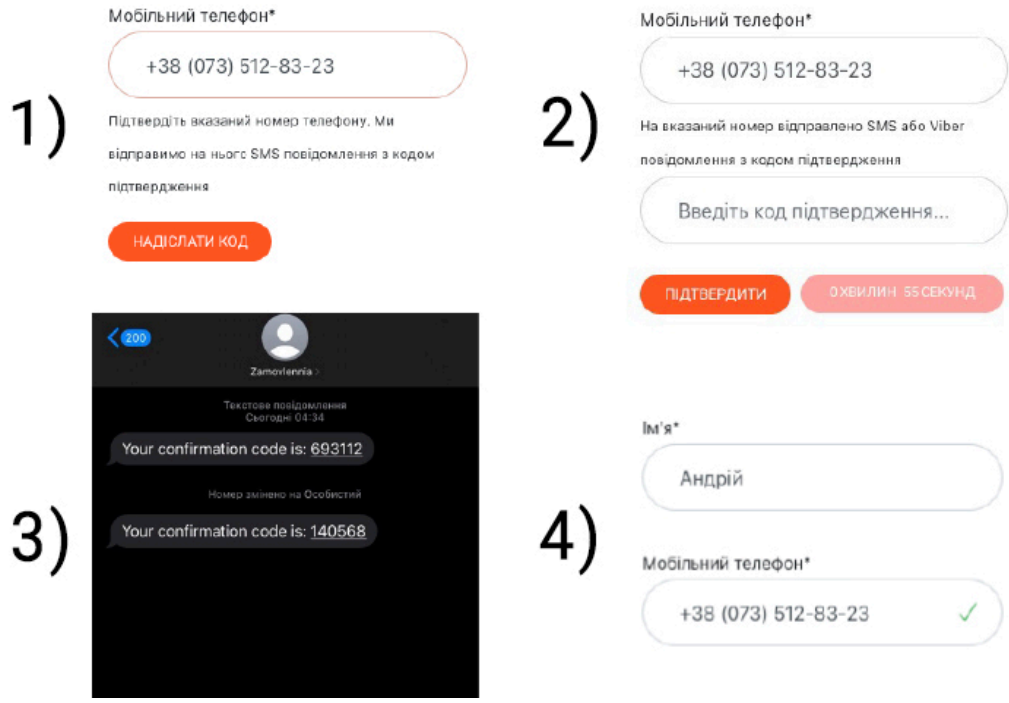


Рисунок 4.25 – Верифікація клієнта за допомогою OTP

Джерело: розроблено автором

Етапи підтвердження номеру телефону:

1. Клієнт вводить номер телефону отримувача замовлення.
2. Відправляється OTP код-пароль із 6 символів на вказаний номер телефону.
3. Клієнт отримує СМС повідомлення з вказаним одноразовим паролем.
4. Клієнт підтверджує номер телефону вводячи код пароль в форму, після чого ІС повідомляє клієнта про успішне підтвердження.

Цей функціонал використовується для підтвердження особи покупця під час замовлення, що допомагає запобігти шахрайству, оскільки тільки особа, яка має доступ до номера телефону, може ввести OTP та завершити покупку, що значно ускладнює життя зловмисникам, які можуть намагатися використати чужі дані.

Коли клієнт обирає спосіб доставки, вказує місто та номер відділення Нової Пошти, він одразу ж отримує інформацію про вартість доставки. Ця сума додається до загальної вартості замовлення, і клієнт може бачити повну суму до сплати в відповідному полі на сторінці оформлення замовлення. (рис.26)

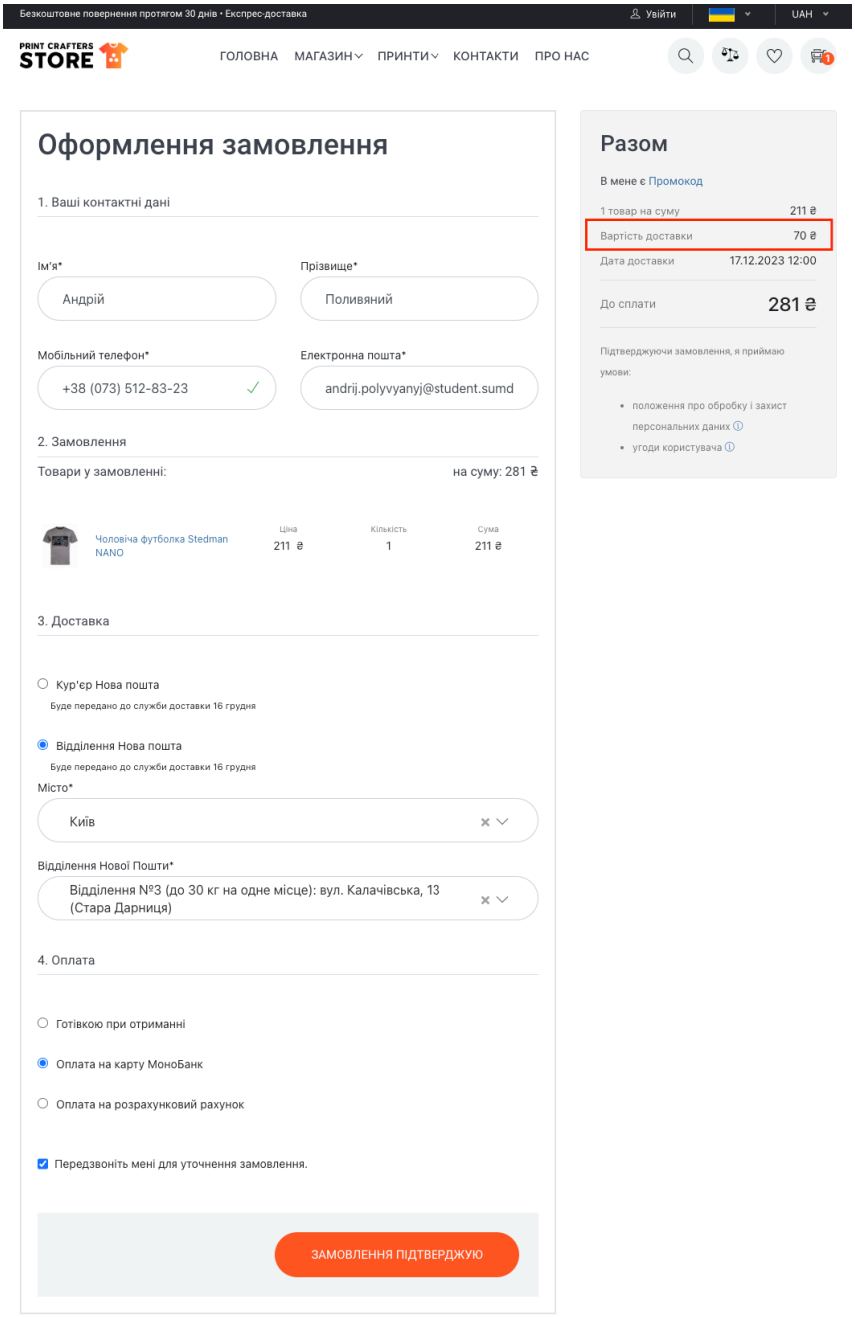


Рисунок 4.26 – Сторінка оформлення замовлення. Розрахунок вартості доставки в сумі замовлення

Джерело: розроблено автором

Після успішного оформлення замовлення клієнт бачить лист з деталями замовлення в себе на пошті (рис.4.27).

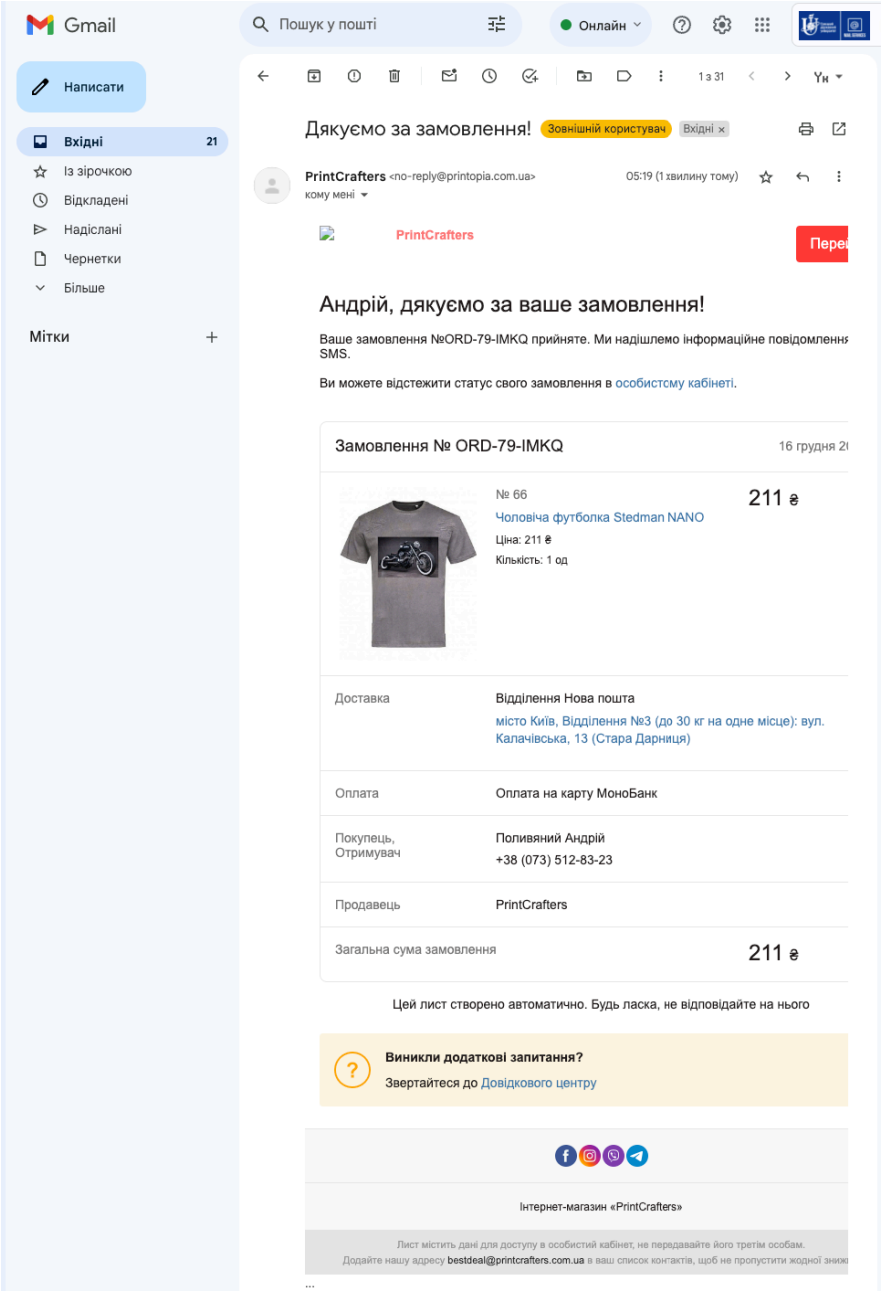


Рисунок 4.27 – Електронний лист клієнта із підтвердженням замовлення

Джерело: розроблено автором

Для того щоб адміністратору або менеджеру відправилось повідомлення про зроблене замовлення потрібно прив’язати Телеграм чат до аккаунту адміністратора, це можна зробити за допомогою команди /bind в телеграм боті (рис.4.28).

Клієнти можуть переглядати свої оформлені замовлення в особистому кабінеті під розділом "Мої замовлення", як це зображено на рисунку 4.29.

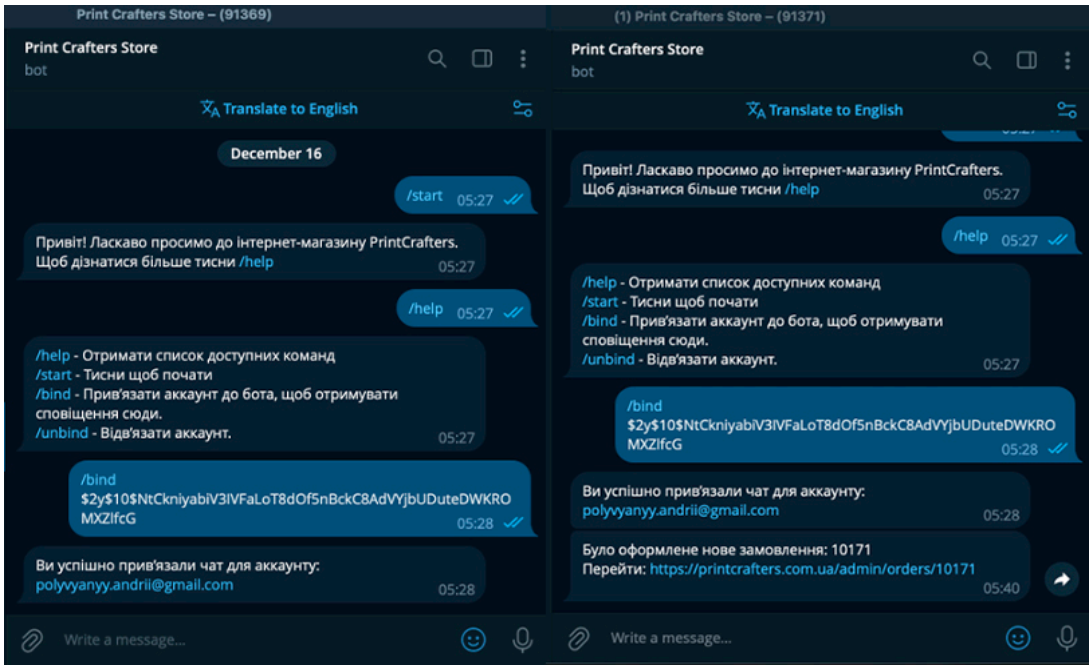


Рисунок 4.28 – Прив'язка телеграм чату до аккаунту адміністратора ІС підтримки обслуговування клієнтів

Джерело: розроблено автором

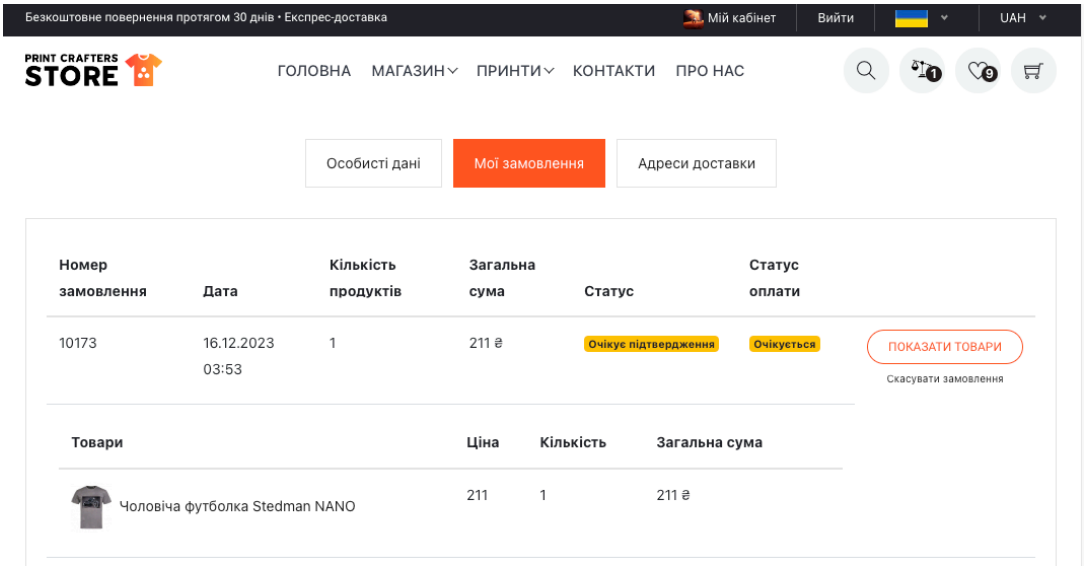


Рисунок 4.29 – Особистий кабінет клієнта. Історія замовлень

Джерело: розроблено автором

4.7 Настанови з використання інформаційної системи менеджерам та адміністраторам

На початку використання інтерфейсу ведення замовлення менеджерів зустрічає форма авторизації (рис.4.30). Якщо в менеджера немає доступу до аккаунта його потрібно запросити у адміністратора.

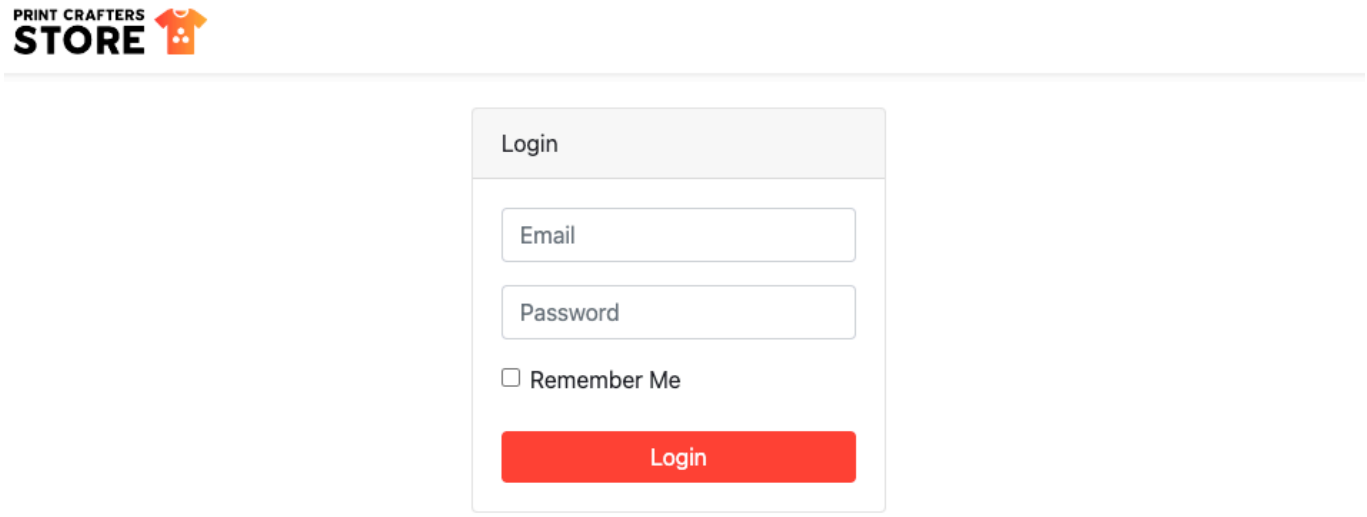


Рисунок 4.30 – Вхід до інтерфейсу ведення замовлень

Джерело: розроблено автором

Після успішної авторизації менеджер має доступ до інтерфейсу ведення замовлень, управління товарами, категоріями товарів, контентом ІС, та інші можливості.

На рис.4.31 представлена сторінка управління товарами, де можна додавати нові товари, редагувати існуючі товари або навіть видаляти товари або архівувати. Зручний інтерфейс дозволяє легко шукати, сортувати, навігувати між товарами, ефективно управляти запасами та швидко оновлювати інформацію про товари, сприяючи оптимізації робочого процесу.

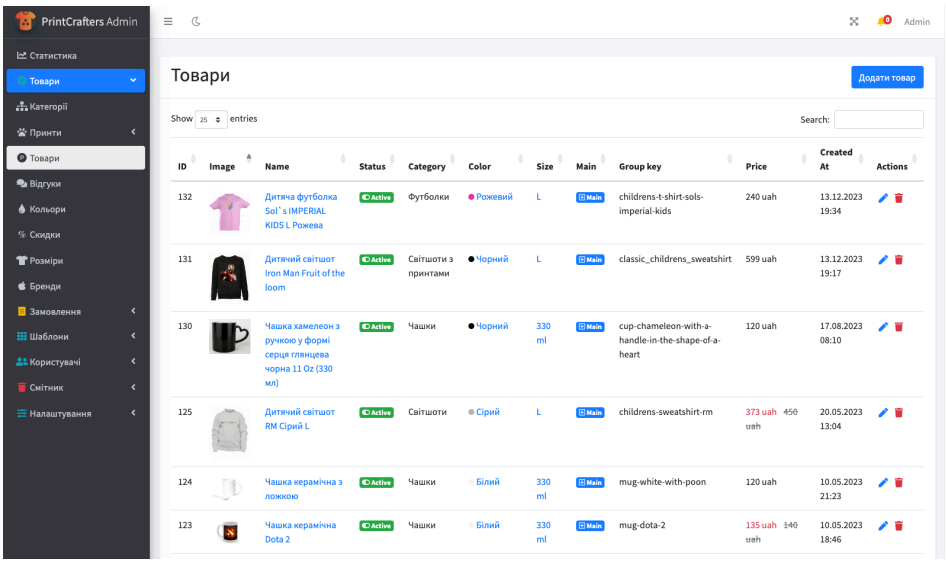


Рисунок 4.31 – Сторінка товарів

Джерело: розроблено автором

Після того як клієнт зробив замовлення менеджер чи адміністратор може бачити його на сторінці «Замовлення» (рис.4.32). Після чого зможе перейти на сторінку замовлення (рис.4.33) та прийняти рішення підтвердити (рис.4.34) чи скасувати його (рис.4.36).

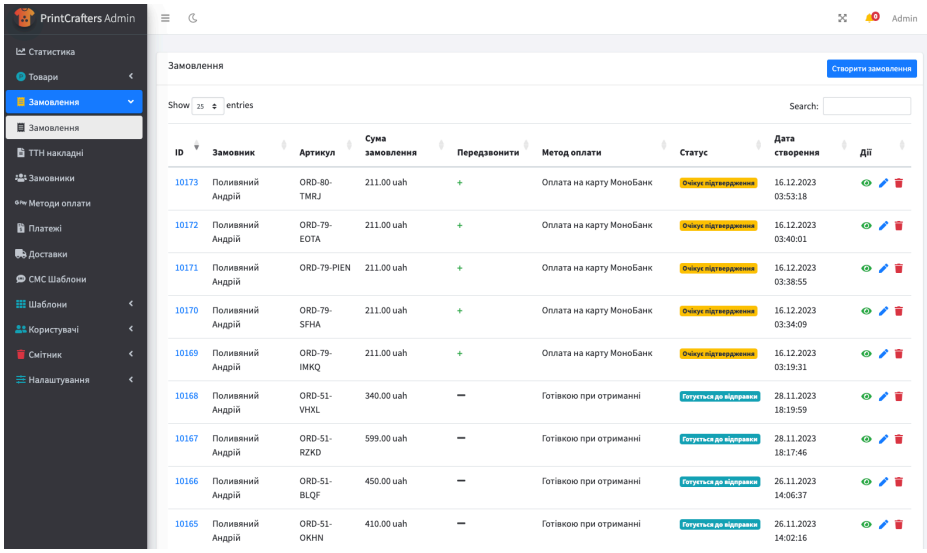


Рисунок 4.32 – Інтерфейс ведення замовлень. Сторінка замовлень

Джерело: розроблено автором

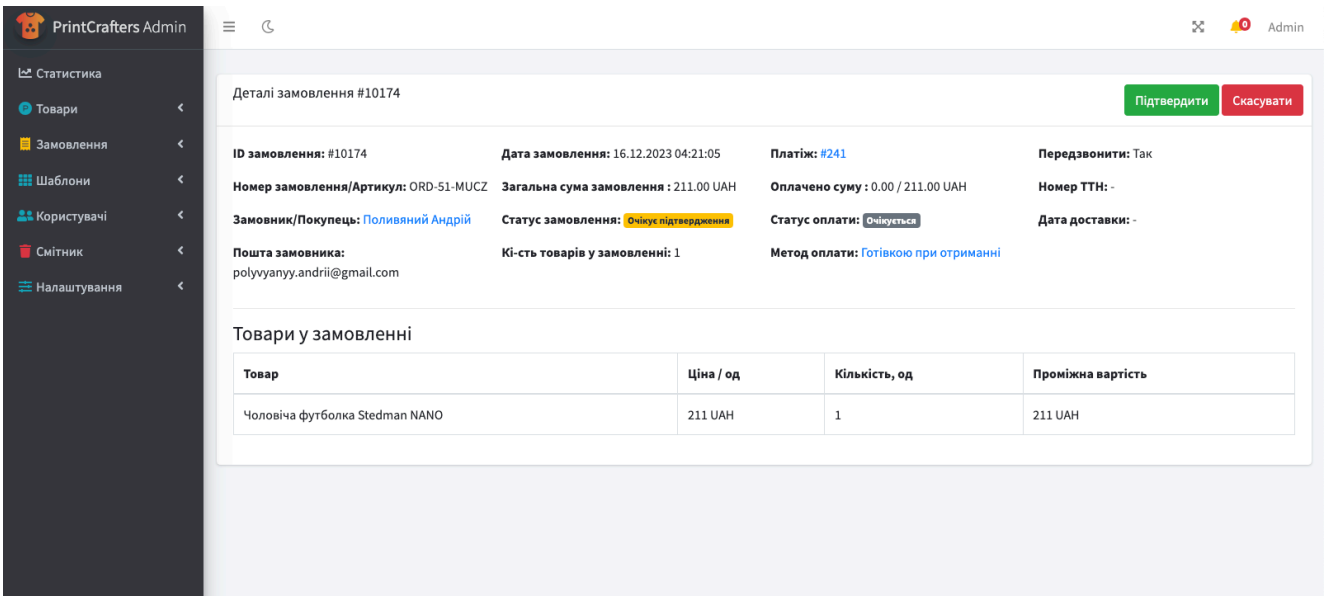


Рисунок 4.33 – Інтерфейс ведення замовлень. Сторінка замовлення

Джерело: розроблено автором

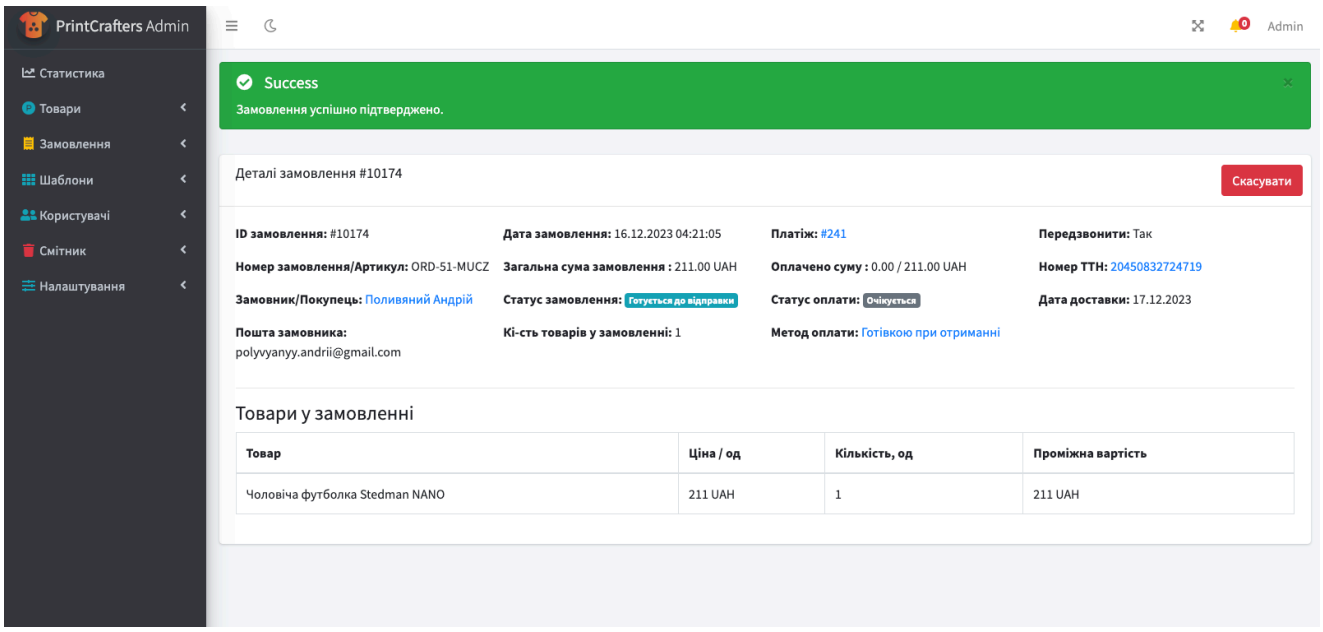


Рисунок 4.34 – Інтерфейс ведення замовлень. Сторінка замовлення. Підтвердження замовлення

Джерело: розроблено автором

Якщо менеджер підтверджує замовлення, в такому разі створюється нова ТТН за допомогою інтеграції з сервісом Нова Пошта та відправляється повідомлення клієнту про зміну статусу замовлення та інформацію про деталі доставки замовлення, номер ТТН, ціна, дата доставки (рис.4.35).

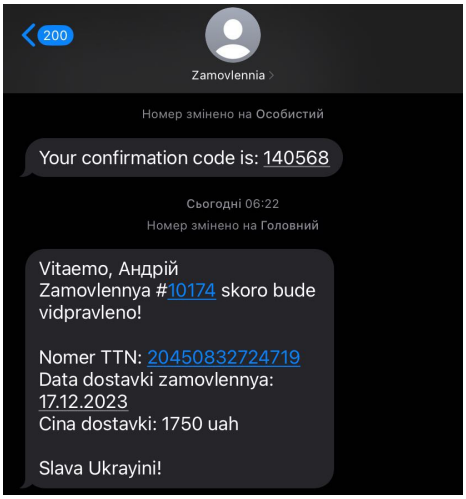


Рисунок 4.35 – СМС інформування клієнта про зміну статусу замовлення та інформація про доставку

Джерело: розроблено автором

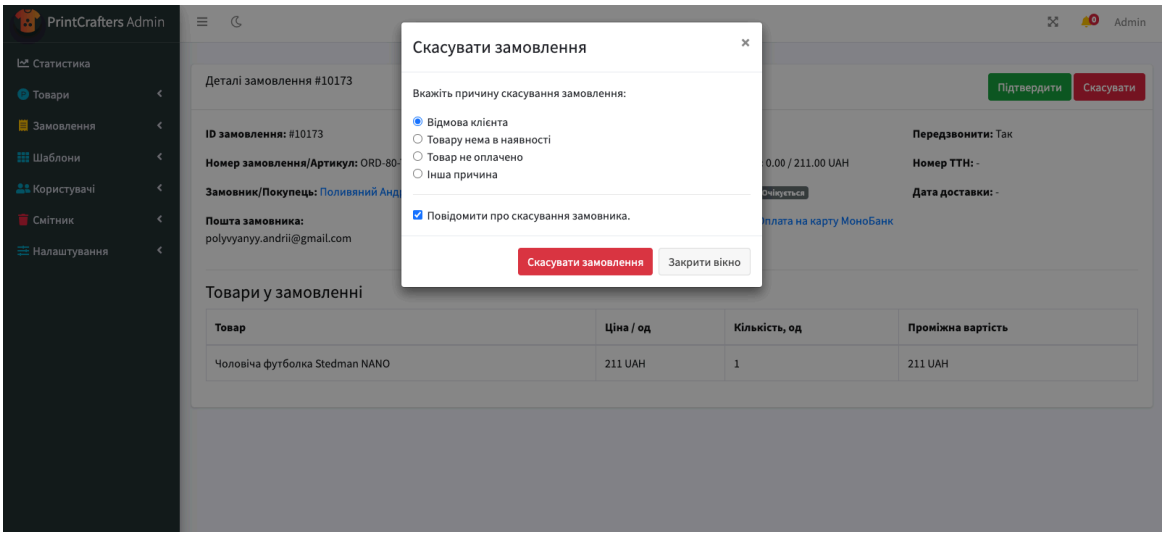


Рисунок 4.36 – Інтерфейс ведення замовлень. Скасування замовлення

Джерело: розроблено автором

Якщо хтось із сторін скасовує замовлення відправляється повідомлення менеджерам в Телеграм (рис.4.37).

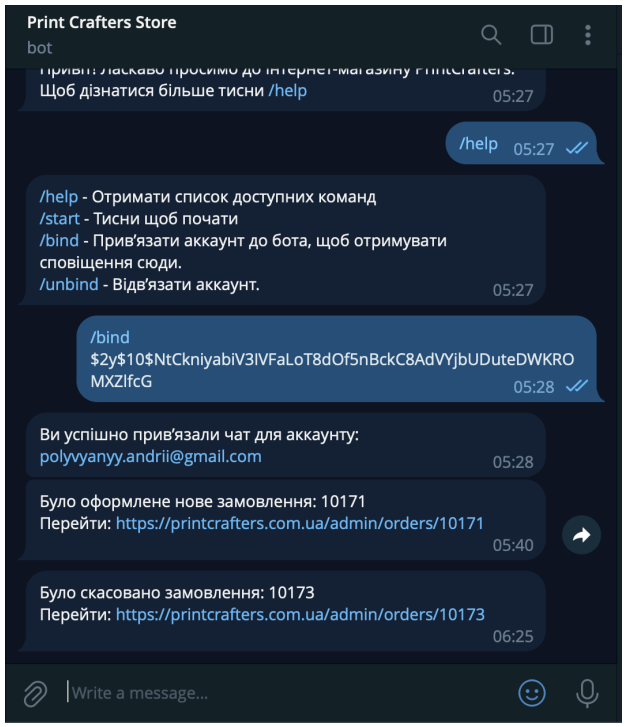


Рисунок 4.37 – Нотифікація про зміну статусу замовлення в Телеграм

Джерело: розроблено автором

ВИСНОВКИ

Метою кваліфікаційної роботи магістра було створення інтерфейсу ведення замовлень в інформаційній системі підтримки клієнтів інтернет-магазину "printcrafters.com.ua". В рамках виконання кваліфікаційної роботи магістра було проведено аналіз, планування, проектування та розробку інформаційної системи підтримки клієнтів інтернет-магазину одягу та аксесуарів з принтами та інтерфейсу ведення замовлення, що допомагає покращувати обслуговування клієнтів та сприяє зручності та швидкості оформлення покупок.

Проведено дослідження предметної області, яке показало, що інформаційна система повинна включати наступні функції: трекінг статусів замовлення, сповіщення про зроблені замовлення, сповіщення про зміну статусу замовлень, автоматизація оплати замовлення через платіжні системи в інтерфейсі ведення замовлення та розрахунок вартості доставки в інтерфейсі оформлення замовлення, що забезпечує зручність і прозорість для клієнтів, сприяє швидкому прийняттю рішень про покупку та підвищує загальну задоволеність користувачів. Були поставлені чіткі задачі, побудований календарний план проекту за допомогою діаграми Ганта, що забезпечує послідовне виконання робіт відповідно до встановлених термінів. Також було визначено функціональні та нефункціональні вимоги, сформовано головну мету проекту методом SMART, що дозволяє чітко відстежувати прогрес і забезпечує досягнення конкретних, вимірюваних результатів.

Результати проектування ІС представлені контекстною діаграмою бізнес-процесів в нотації IDEF0 та її декомпозицією, які дають чітке уявлення про взаємодію системи з зовнішнім середовищем та внутрішню структуру процесів. Спроектовану архітектуру бази даних представлено в нотації ER-діаграм. Крім того, були вирішені поставлені задачі а саме:

- Оцінено існуючі системи обслуговування клієнтів, визначено аспекти для оптимізації.

- Виявлено типові проблеми користувачів та запропоновано шляхи їх вирішення.
- Розроблено критерії оцінки обслуговування, що підвищують якість сервісу.
- Сформовано рекомендації для поліпшення сервісу, включаючи технологічні нововведення.
- Проведено валідацію розроблених пропозицій із застосуванням експертного оцінювання.
- Спроектовано базу даних з ефективною структурою управління даними.
- Розроблено архітектуру та програмний код, що забезпечує функціональність системи.
- Реалізовано інтерфейс для зручного ведення замовлень.
- Розгорнуто та протестовано систему, забезпечивши її готовність до впровадження.
- Підготовлено висновки та рекомендації, які відображають реалізацію проекту та перспективи його розвитку.

В рамках виконання кваліфікаційної роботи було реалізовано інформаційну систему підтримки обслуговування клієнтів, яка демонструє високу продуктивність та відповідає сучасним вимогам ефективності та функціональності. Система успішно інтегрує ключові функції, необхідні для забезпечення зручності та швидкості оформлення покупок, включаючи трекінг статусів замовлення, швидке та зручне сповіщення менеджерів, та автоматизацію оплати. Тестування системи підтвердило її надійність і здатність до масштабування, а також забезпечило підстави для впевненості у її довгостроковій стабільності та ефективному використанні в умовах реальної експлуатації. Завдяки гнучкій архітектурі, система може бути легко адаптована під змінні потреби бізнесу та зростаючий обсяг клієнтської бази, що робить її важливим інструментом для підвищення рівня задоволеності клієнтів і, як наслідок, для зміцнення лояльності до бренду.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ:

[1] F. Fernández-Bonilla, C. Gijón, і B. De la Vega, «E-commerce in Spain: Determining factors and the importance of the e-trust», *Telecomm Policy*, вип. 46, вип. 1, 2022, doi: 10.1016/j.telpol.2021.102280.

[2] S. Rahman, F. Fadrul, Y. Yusrizal, R. Marlyna, і M. M. Momin, «Improving the Satisfaction and Loyalty of Online Shopping Customers Based on E-Commerce Innovation and E-Service Quality», *Gadjah Mada International Journal of Business*, вип. 24, вип. 1, 2022, doi: 10.22146/gamaijb.58783.

[3] A. Rosário і R. Raimundo, «Consumer marketing strategy and e-commerce in the last decade: A literature review», *Journal of Theoretical and Applied Electronic Commerce Research*, вип. 16, вип. 7. 2021. doi: 10.3390/jtaer16070164.

[4] J. A. Cano, A. Londoño-Pineda, M. F. Castro, H. B. Paz, C. Rodas, і T. Arias, «A Bibliometric Analysis and Systematic Review on E-Marketplaces, Open Innovation, and Sustainability», *Sustainability (Switzerland)*, вип. 14, вип. 9. 2022. doi: 10.3390/su14095456.

[5] M. Tsagkias, T. H. King, S. Kallumadi, V. Murdock, і M. de Rijke, «Challenges and research opportunities in eCommerce search and recommendations», *ACM SIGIR Forum*, вип. 54, вип. 1, 2020, doi: 10.1145/3451964.3451966.

[6] N. V. Florea *et al.*, «Trends and Perspectives of Romanian E-Commerce Sector Based on Mathematical Simulation», *Electronics (Switzerland)*, вип. 11, вип. 15, 2022, doi: 10.3390/electronics11152295.

[7] A. Supriyanto, B. B. Wiyono, і B. Burhanuddin, «Effects of service quality and customer satisfaction on loyalty of bank customers», *Cogent Business and Management*, вип. 8, вип. 1, 2021, doi: 10.1080/23311975.2021.1937847.

[8] W. N. Hasna і I. Rachmawati, «Factors Affecting the Customer Loyalty of Tokopedia E-Commerce», *International Journal of Science and Management Studies (IJSMS)*, 2021, doi: 10.51386/25815946/ijsms-v4i3p118.

[9] M. M. Ilchuk, A. V. Kyrychenko, і M. V. Vodnitskyi, «DEVELOPMENT OF E-COMMERCE IN UKRAINE IN THE WAR AND POST-WAR CONDITIONS», *Science and Innovation*, вип. 19, вип. 3, 2023, doi: 10.15407/scine19.03.003.

[10] Anjitha R U і Dr. Sajeev J, «Online Cloth Store», *International Journal of Advanced Research in Science, Communication and Technology*, 2023, doi: 10.48175/ijarsct-12965.

[11] A. S. Bhaktha, A. P. Raj, D. N, і G. Paul, «Online Order Management System», *Int J Res Appl Sci Eng Technol*, вип. 11, вип. 8, 2023, doi: 10.22214/ijraset.2023.54265.

[12] S. Dhingra, S. Gupta, і R. Bhatt, «A study of relationship among service quality of E-Commerce websites, customer satisfaction, and purchase intention», *International Journal of e-Business Research*, вип. 16, вип. 3, с. 42–59, Лип 2020, doi: 10.4018/IJEBR.2020070103.

[13] E. Pujastuti, W. W. Winarno, і S. Sudarmawan, «Pengaruh E-Commerce Toko Online Fashion Terhadap Kepercayaan Konsumen», *Creative Information Technology Journal*, вип. 1, вип. 2, 2015, doi: 10.24076/citec.2014v1i2.17.

[14] C. K. Li, «Evaluation of the effectiveness of the implementation of bitrix24 for e-commerce», *Bulletin of «Turan» University*, вип. 1, 2021, doi: 10.46914/1562-2959-2021-1-1-116-121.

[15] HubSpot, «What Is Inbound Marketing?», Hubspot.

[16] M. Janakova, P. Suchanek, P. Sauman, і J. Botlik, «The optimal volume of CRM services for small business», *International Journal of Advanced Science and Technology*, вип. 29, вип. 3, 2020.

[17] O. Sohaib, M. Naderpour, і W. Hussain, «SaaS e-commerce platforms web accessibility evaluation», в *IEEE International Conference on Fuzzy Systems*, 2018. doi: 10.1109/FUZZ-IEEE.2018.8491621.

[18] J. P. Chin, V. A. Diehl, і K. L. Norman, «Development of an instrument measuring user satisfaction of the human-computer interface», в *Conference on*

Human Factors in Computing Systems - Proceedings, Association for Computing Machinery, Трав 1988, с. 213–218. doi: 10.1145/57167.57203.

[19] Kurniawan Yunus Ariyono, Sukma Irdiana, і Kusnanto Darmawan, «EFFECT OF ONLINE BUYING EXPERIENCE ON CUSTOMER LOYALTY», *International Journal of Social Science*, вип. 1, вип. 5, 2022, doi: 10.53625/ijss.v1i5.1293.

[20] J. Tang *et al.*, «A Visualization Approach for Monitoring Order Processing in E-Commerce Warehouse», *IEEE Trans Vis Comput Graph*, вип. 28, вип. 1, 2022, doi: 10.1109/TVCG.2021.3114878.

[21] I. V. Koskosas, «Communicating information systems goals: A case in internet banking security», *Computer Science and Information Systems*, вип. 6, вип. 1, 2009, doi: 10.2298/CSIS0901071K.

[22] J. R. Turner і R. A. Cochrane, «Goals-and-methods matrix: coping with projects with ill defined goals and/or methods of achieving them», *International Journal of Project Management*, вип. 11, вип. 2, 1993, doi: 10.1016/0263-7863(93)90017-H.

[23] M. Milić і D. Makajić-Nikolić, «Development of a Quality-Based Model for Software Architecture Optimization: A Case Study of Monolith and Microservice Architectures», *Symmetry (Basel)*, вип. 14, вип. 9, 2022, doi: 10.3390/sym14091824.

[24] N. Salaheddin ELGHERIANI і N. D. Ali Salem AHME, «MICROSERVICES VS. MONOLITHIC ARCHITECTURES [THE DIFFERENTIAL STRUCTURE BETWEEN TWO ARCHITECTURES]», *MINAR International Journal of Applied Sciences and Technology*, вип. 4, вип. 3, 2022, doi: 10.47832/2717-8234.12.47.

[25] S. Prettyman, *Learn PHP 8*. 2020. doi: 10.1007/978-1-4842-6240-5.

[26] Laravel LLC, «Documentation - Laravel - The PHP Framework For Web Artisans.», Laravel. Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://laravel.com/docs/10.x

[27] Oracle, «MySQL :: About MySQL», Oracle Corporation. Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://www.oracle.com/mysql/what-is-mysql/

[28] MDN, «Web technology for developers». Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://developer.mozilla.org/en-US/docs/Web

[29] MDN, «What is JavaScript? - Learn web development», MDN Web Docs. Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[30] Microsoft Corporation, «TypeScript - JavaScript that scales - Documentation», TypeScript. Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://typescript-site-76.ortam.vercel.app/docs/home.html

[31] J. Reinink, «Inertia JS: The Modern Monolith», Inertia JS. Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://inertiajs.com/

[32] Vue.js, «Introduction | Vue.js», Vue.js. Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://vuejs.org/guide/introduction.html

[33] Docker Inc, «Overview | Docker Documentation», Docker documentacion. Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://docs.docker.com/get-started/overview/

[34] Laravel, «Testing: Getting started», Laravel 8.x Documentation. Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://laravel.com/docs/10.x/testing

[35] Microsoft, «GitHub Actions», www.github.com. Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://docs.github.com/en/actions

[36] Amazon, «AWS Documentation», Welcome to AWS Documentation. Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://docs.aws.amazon.com/

[37] Git, «Git - About Version Control», Online. Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

[38] K. I. Bagwan і S. D. Ghule, «A Modern Review on Laravel-PHP Framework», *IRE Journals*, вип. 2, вип. 12, 2019.

[39] L. A. T. Nguyen, T. S. Huynh, D. T. Tran, і Q. H. Vu, «Design and Implementation of Web Application Based on MVC Laravel Architecture», *European Journal of Electrical Engineering and Computer Science*, вип. 6, вип. 4, 2022, doi: 10.24018/ejece.2022.6.4.448.

[40] J. Song, M. Zhang, і H. Xie, «Design and implementation of a Vue.js-based college teaching system», *International Journal of Emerging Technologies in Learning*, вип. 14, вип. 13, 2019, doi: 10.3991/ijet.v14i13.10709.

[41] U. M. Sutinen, H. Saarijärvi, і M. Yrjölä, «Shop at your own risk? Consumer activities in fashion e-commerce», *Int J Consum Stud*, вип. 46, вип. 4, 2022, doi: 10.1111/ijcs.12759.

[42] Q. Li і Y.-L. Chen, «IDEF0 Function Modeling», в *Modeling and Analysis of Enterprise and Information Systems*, 2009. doi: 10.1007/978-3-540-89556-5_5.

[43] IBM, «What is a Relational Database?», Website. Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://www.ibm.com/topics/relational-databases

[44] T. Sliusarenko і V. Filatov, «RELATIONAL VS NON-RELATIONAL DATABASES», *Grail of Science*, вип. 23, 2023, doi: 10.36074/grail-of-science.23.12.2022.41.

[45] R. Silhavy, P. Silhavy, і Z. Prokopova, «Using actors and use cases for software size estimation», *Electronics (Switzerland)*, вип. 10, вип. 5, 2021, doi: 10.3390/electronics10050592.

[46] S. Thomas, *SSL and TLS essentials*. 2000.

[47] C. Hopkins, «Why You Should Use Bcrypt to Hash Stored Passwords Related Topics», 09.09.2011. Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://www.sitepoint.com/why-you-should-use-bcrypt-to-hash-stored-passwords/

[48] Docker Inc., «Overview of Docker Compose | Docker Documentation», Docker Inc. Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://docs.docker.com/compose/

[49] Laravel, «Eloquent: Getting started», Laravel 8.x Documentation. Дата звернення: 16, Грудень 2023. [Online]. Доступний у: https://laravel.com/docs/10.x/eloquent

[50] K. Boogaard, «How to write SMART goals (with examples)», *Work Life by Atlassian*. 2021.

[51] C. Organ і C. Bottorff, «Work Breakdown Structure (WBS) in Project Management.», *Review of International Comparitive Management*, вип. 17, вип. 1, 2022.

[52] E. L. Prentis, «MASTER PROJECT PLANNING: Scope, Time and Cost», *Project Management Journal*, вип. 20, вип. 1, 1989.

ДОДАТОК А

Деталізація мети методом SMART

Для визначення основної мети використовувався метод SMART (Specific, Measurable, Achievable, Relevant, Time-bound)[50]. Цей підхід дозволяє нам структурувати ціль таким чином, щоб вона була чіткою, вимірюваною, досяжною, відповідною та обмеженою в часі. Процес визначення мети було розроблено на основі глибокого аналізу, що відображено у таблиці А.1.

Таблиця А.1 – Деталізація мети методом SMART

Категорія	**Деталізація Мети**
Specific (конкретна)	Розробка інтерфейсу ведення замовлень для ІС (інформаційної системи) підтримки клієнтів інтернет магазину "printcrafters.com.ua", забезпечуючи користувачам простий у використанні і ефективний процес покупки, а менеджерам зручний та ефективний процес обробки замовлень.
Measurable (вимірювана)	Запуск функціонального інтерфейсу з управління замовленнями, інтеграції з системою управління доставкою та зручним сповіщенням через месерджер Telegram.
Achievable (досяжна, узгоджена)	Є затверджене технічне завдання, знання мов програмування та наступних інструментів розробки: HTML&CSS, SCSS, PHP, JS, TypeScript, Laravel, Mysql, Redis , Docker, Vue.js.
Relevant (реалістична)	Підвищення задоволеності клієнтів і зростання продажів через поліпшення користувацького

	досвіду, зниження часу обробки замовлень та особистісної комунікації.
Time-framed (обмежена в часі)	Завершення розробки та тестування інтерфейсу до 4го грудня 2023, з подальшим запуском в експлуатацію.

Джерело: розроблено автором

Розподіл робіт за структурою (WBS).

Для ефективної розробки інформаційної системи, яка б забезпечувала підтримку обслуговування клієнтів інтернет-магазину, важливим є використання інструментів WBS (Work Breakdown Structure).

Структура розподілу робіт (WBS) - це поділ проекту на рівні ієрархії, де кожен наступний рівень надає більш детальне розуміння завдань, які має виконати команда для досягнення цілей проекту та створення потрібних результатів. [51]WBS структурує загальний обсяг робіт у проекті, визначаючи взаємозв’язки між окремими завданнями та цілями проекту, і служить як інструмент для планування, контролю та моніторингу прогресу. Ця структура допомагає детально організувати та систематизувати всі етапи роботи, забезпечуючи глибоку обробку кожного аспекту проекту за допомогою чітко встановлених стандартів якості та ефективності. Візуальне представлення цієї структури наведено на рисунку А.1.

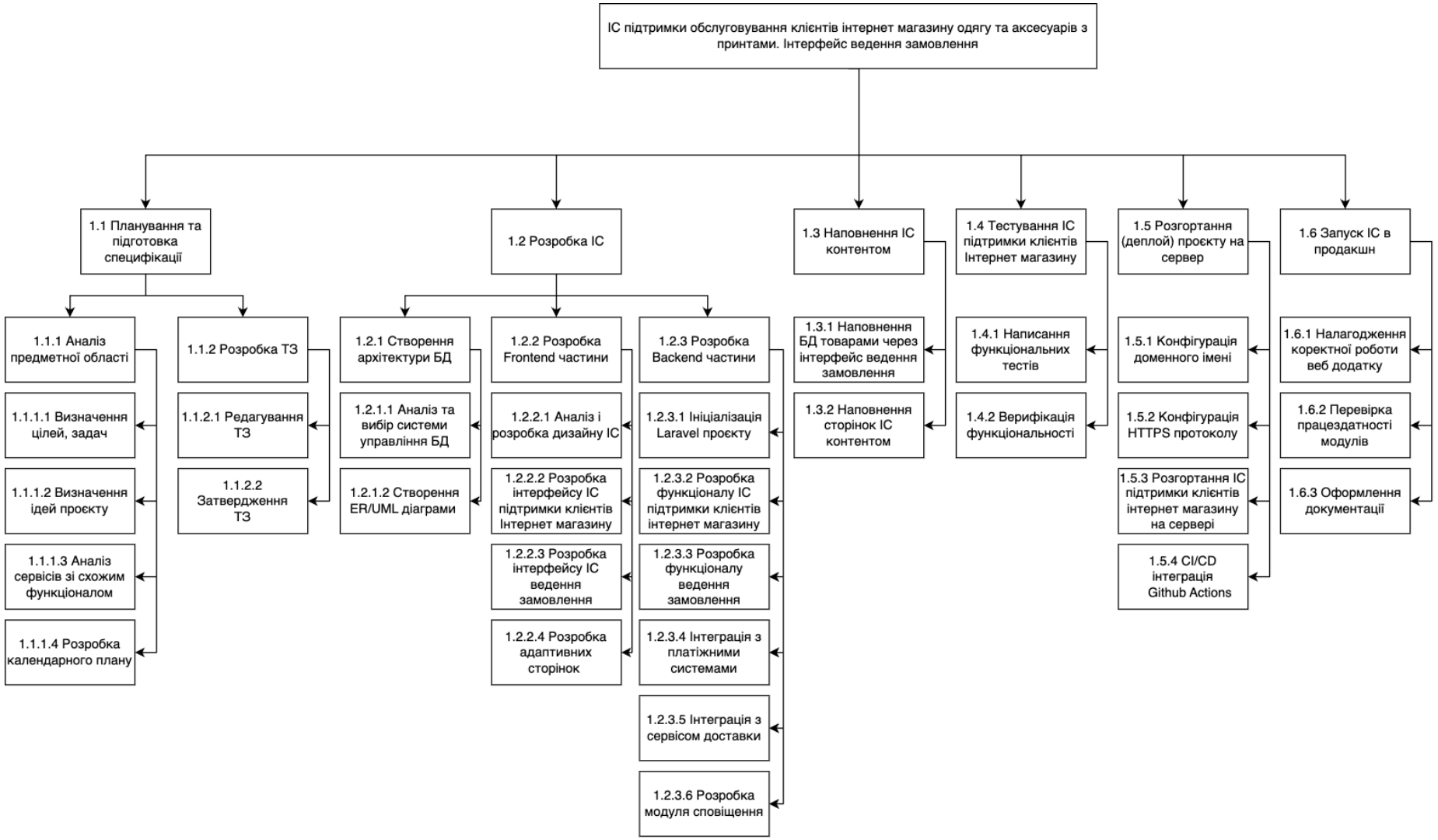


Рисунок А.1 – WBS структура робіт проекту.

Джерело: розроблено автором

Структура організаційного розподілу робіт (OBS).

OBS (Organizational Breakdown Structure) або Організаційна Структура Робіт – це інструмент, який використовується для відображення організаційної структури проекту, розподілу ролей та відповідальностей серед учасників команди. [52]Він визначає, хто за що відповідає у проекті, тим самим забезпечуючи кожному члену команди чітке розуміння своїх обов'язків. У процесі роботи було розроблено діаграму, яка визначає розподіл завдань між виконавцями проекту. Це спрощує комунікацію та координацію в команді, та забезпечує ефективне виконання кожної частини роботи. Результати розробки OBS відображені на рисунку А.2, де показана структуру керування проектом.

Згідно з розробленою OBS структурою проекту (рис.А.2), на різних рівнях структури OBS встановлені чіткі зв'язки між виконавцями та їхніми завданнями. Така деталізація сприяє кращому управлінню проектом, адже кожен виконавець точно знає свої обов'язки та як вони вписуються в загальний план розвитку інформаційної системи. Це дозволяє вчасно виявляти та вирішувати потенційні проблеми, забезпечуючи плавний прогрес та високу якість кінцевого продукту.

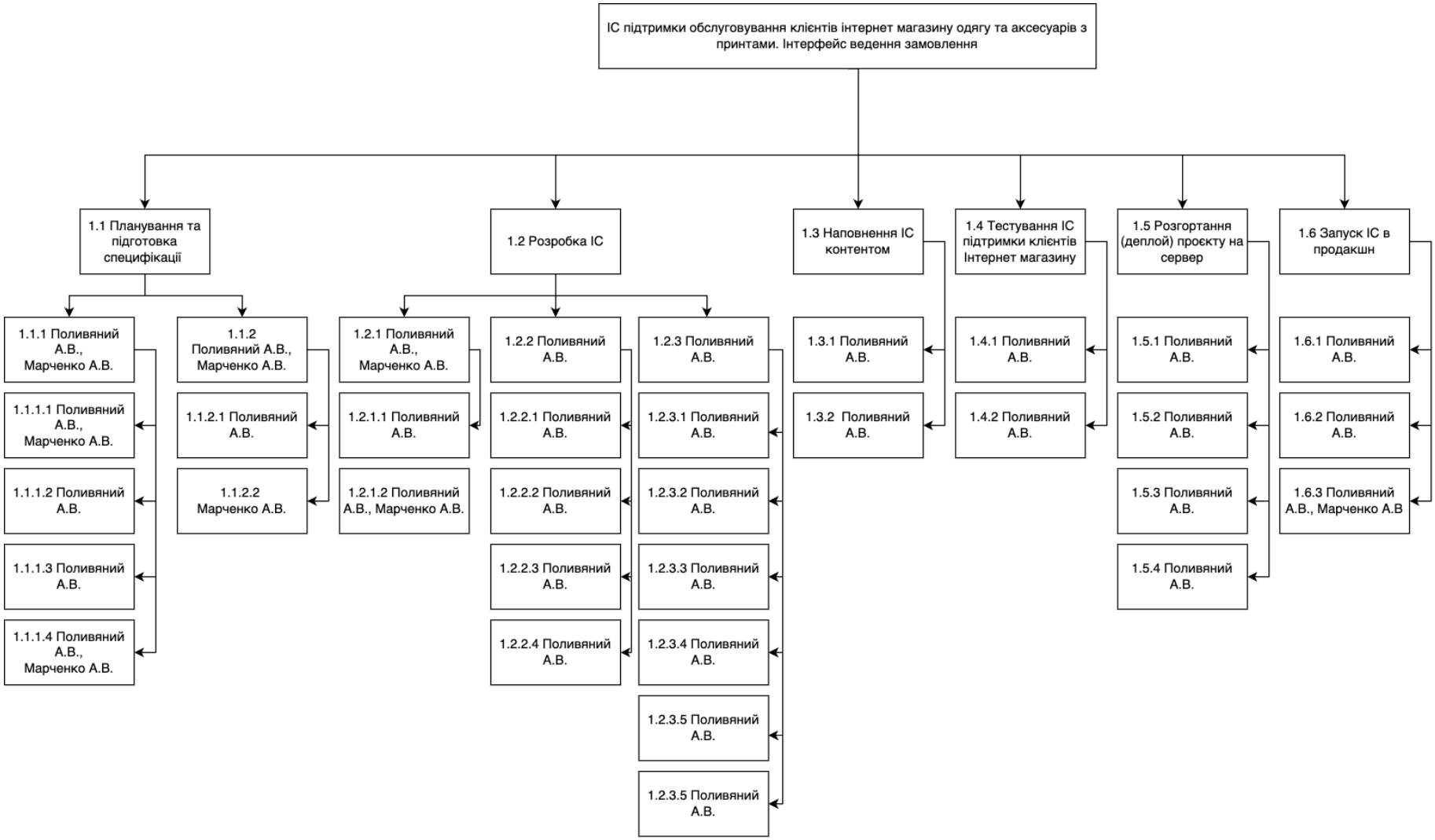
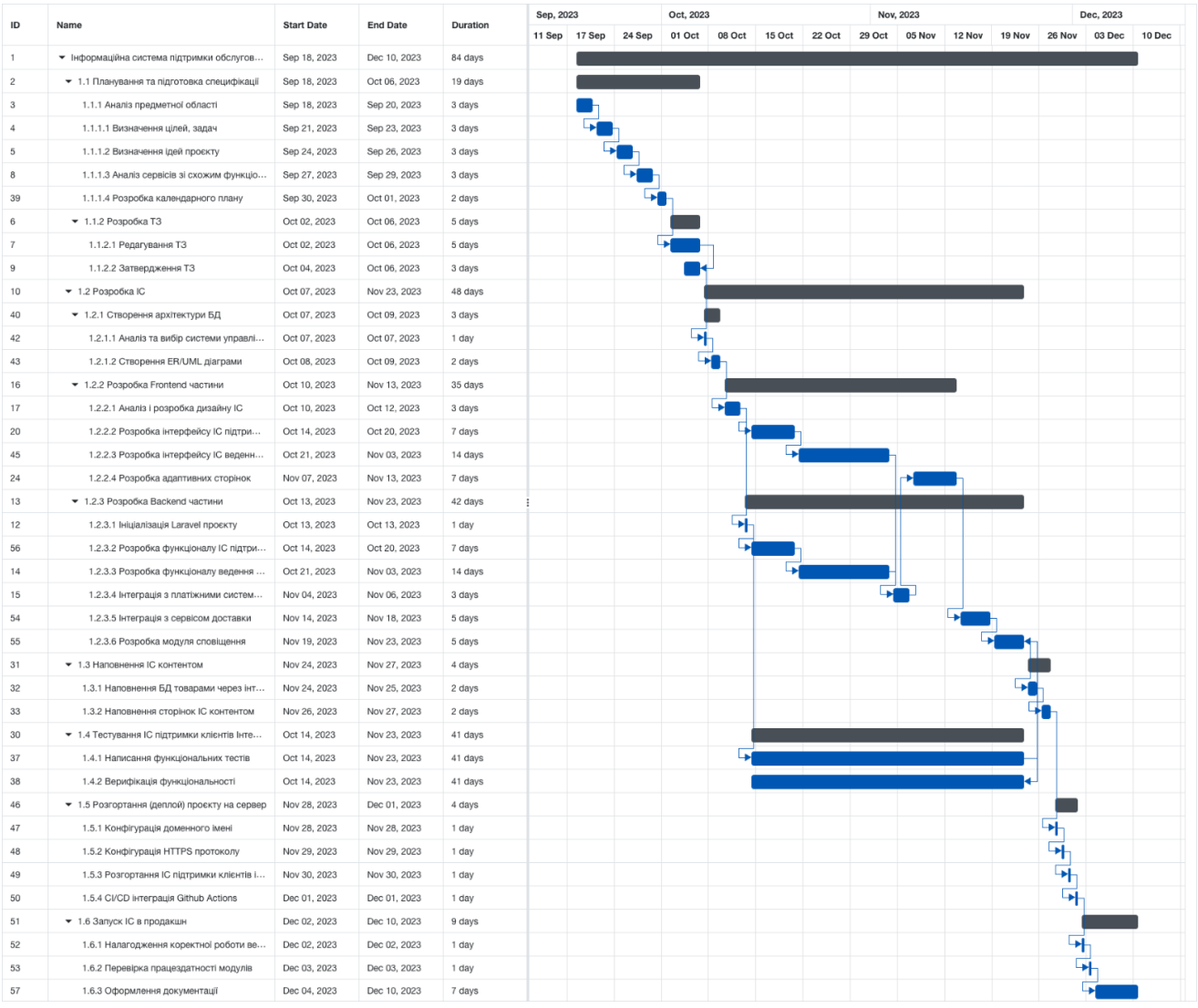


Рисунок А.2 – OBS-структура проекту.

Джерело: розроблено автором

Діаграма Ганта

У рамках проектування та планування розробки інформаційної системи було побудовано діаграму Ганта, що зображена на рис. А.3. Ця діаграма докладно відображає планування роботи над різними компонентами проекту, включаючи етапи розробки, тестування, впровадження та оцінки, забезпечуючи чітку візуалізацію часових витрат і послідовності виконання завдань.



ID	Name	Start Date	End Date	Duration
1	Інформаційна система підтримки обслугов…	Sep 18, 2023	Dec 10, 2023	84 days
2	1.1 Планування та підготовка специфікації	Sep 18, 2023	Oct 06, 2023	19 days
3	1.1.1 Аналіз предметної області	Sep 18, 2023	Sep 20, 2023	3 days
4	1.1.1.1 Визначення цілей, задач	Sep 21, 2023	Sep 23, 2023	3 days
5	1.1.1.2 Визначення ідей проєкту	Sep 24, 2023	Sep 26, 2023	3 days
8	1.1.1.3 Аналіз сервісів зі схожим функціо…	Sep 27, 2023	Sep 29, 2023	3 days
39	1.1.1.4 Розробка календарного плану	Sep 30, 2023	Oct 01, 2023	2 days
6	1.1.2 Розробка ТЗ	Oct 02, 2023	Oct 06, 2023	5 days
7	1.1.2.1 Редагування ТЗ	Oct 02, 2023	Oct 06, 2023	5 days
9	1.1.2.2 Затвердження ТЗ	Oct 04, 2023	Oct 06, 2023	3 days
10	1.2 Розробка ІС	Oct 07, 2023	Nov 23, 2023	48 days
40	1.2.1 Створення архітектури БД	Oct 07, 2023	Oct 09, 2023	3 days
42	1.2.1.1 Аналіз та вибір системи управлі…	Oct 07, 2023	Oct 07, 2023	1 day
43	1.2.1.2 Створення ER/UML діаграми	Oct 08, 2023	Oct 09, 2023	2 days
16	1.2.2 Розробка Frontend частини	Oct 10, 2023	Nov 13, 2023	35 days
17	1.2.2.1 Аналіз і розробка дизайну ІС	Oct 10, 2023	Oct 12, 2023	3 days
20	1.2.2.2 Розробка інтерфейсу ІС підтри…	Oct 14, 2023	Oct 20, 2023	7 days
45	1.2.2.3 Розробка інтерфейсу ІС веденн…	Oct 21, 2023	Nov 03, 2023	14 days
24	1.2.2.4 Розробка адаптивних сторінок	Nov 07, 2023	Nov 13, 2023	7 days
13	1.2.3 Розробка Backend частини	Oct 13, 2023	Nov 23, 2023	42 days
12	1.2.3.1 Ініціалізація Laravel проєкту	Oct 13, 2023	Oct 13, 2023	1 day
56	1.2.3.2 Розробка функціоналу ІС підтри…	Oct 14, 2023	Oct 20, 2023	7 days
14	1.2.3.3 Розробка функціоналу ведення …	Oct 21, 2023	Nov 03, 2023	14 days
15	1.2.3.4 Інтеграція з платіжними систем…	Nov 04, 2023	Nov 06, 2023	3 days
54	1.2.3.5 Інтеграція з сервісом доставки	Nov 14, 2023	Nov 18, 2023	5 days
55	1.2.3.6 Розробка модуля сповіщення	Nov 19, 2023	Nov 23, 2023	5 days
31	1.3 Наповнення ІС контентом	Nov 24, 2023	Nov 27, 2023	4 days
32	1.3.1 Наповнення БД товарами через інт…	Nov 24, 2023	Nov 25, 2023	2 days
33	1.3.2 Наповнення сторінок ІС контентом	Nov 26, 2023	Nov 27, 2023	2 days
30	1.4 Тестування ІС підтримки клієнтів Інте…	Oct 14, 2023	Nov 23, 2023	41 days
37	1.4.1 Написання функціональних тестів	Oct 14, 2023	Nov 23, 2023	41 days
38	1.4.2 Верифікація функціональності	Oct 14, 2023	Nov 23, 2023	41 days
46	1.5 Розгортання (деплой) проєкту на сервер	Nov 28, 2023	Dec 01, 2023	4 days
47	1.5.1 Конфігурація доменного імені	Nov 28, 2023	Nov 28, 2023	1 day
48	1.5.2 Конфігурація HTTPS протоколу	Nov 29, 2023	Nov 29, 2023	1 day
49	1.5.3 Розгортання ІС підтримки клієнтів і…	Nov 30, 2023	Nov 30, 2023	1 day
50	1.5.4 CI/CD інтеграція Github Actions	Dec 01, 2023	Dec 01, 2023	1 day
51	1.6 Запуск ІС в продакшн	Dec 02, 2023	Dec 10, 2023	9 days
52	1.6.1 Налагодження коректної роботи ве…	Dec 02, 2023	Dec 02, 2023	1 day
53	1.6.2 Перевірка працездатності модулів	Dec 03, 2023	Dec 03, 2023	1 day
57	1.6.3 Оформлення документації	Dec 04, 2023	Dec 10, 2023	7 days

Рисунок А.3 – Діаграма Ганта для проекту розробки інформаційної системи підтримки обслуговування клієнтів інтернет-магазину одягу та аксесуарів з принтами.

Джерело: розроблено автором

Керування ризиками

В контексті управління проектами, особливу увагу слід звернути на важливість оцінки ризиків. Це вимагає визначення таких ризиків, що потребують термінового втручання. Залежно від серйозності кожного ризику, розробляється стратегія для його мінімізації чи усунення. Після цього настає фаза кількісного аналізу ризиків, який може бути виконаний одночасно з якісним оцінюванням або окремо, відповідно до рівня розвитку та ресурсів проекту. В табл. А.2 описано перелік можливих ризиків, що стосуються цього проекту.

Таблиця А.2 – Перелік ризиків проекту

№ ризику	Назва ризику
1	Обмежені технічні навички розробників.
2	Затримка з поставками необхідного обладнання та програмного забезпечення.
3	Зміна вимог до проекту з боку замовника.
4	Непередбачені технічні проблеми під час розробки.
5	Недооцінка часу, необхідного для завершення проекту.
6	Відсутність залучення з боку користувачів під час тестування системи.
7	Неспроможність системи адаптуватися до зростаючого числа користувачів.
8	Складності інтеграції з існуючими системами управління логістики та відслідковування транзакцій.
9	Ризики безпеки даних через потенційні кібератаки або витоки інформації.
10	Зміни в трендах, які можуть зробити систему застарілою незабаром після запуску

11	Проблеми з масштабуванням системи у відповідності до зростаючого обсягу замовлень.
12	Невідповідність інтерфейсу очікуванням користувачів, що може призвести до зниження задоволеності клієнтів та зменшення повторних покупок.
13	Збої в системі платежів, що можуть призвести до затримок у обробці замовлень та фінансових втрат для клієнтів і компанії.
14	Неправильна інтеграція з іншими системами (наприклад, з логістичними сервісами), що може спричинити помилки в доставці.
15	Високий рівень конкуренції на ринку

Джерело: розроблено автором

Проаналізувавши даний перелік ризиків, можна зробити висновок, що успішність розробки та впровадження інформаційної системи підтримки обслуговування клієнтів в значній мірі залежить від передбачливості та гнучкості управлінської стратегії. Затримки з поставками необхідного обладнання та програмного забезпечення, зміна вимог з боку замовника, а також непередбачені технічні проблеми можуть суттєво відтермінувати терміни реалізації проекту. Недооцінка часу, необхідного для завершення проекту, відсутність залучення користувачів під час тестування, та нездатність системи адаптуватися до зростання кількості користувачів можуть призвести до перевищення бюджету та низької задоволеності користувачів.

Оцінка ризиків дозволяє швидко та ефективно реагувати, зменшуючи вплив на проект. В табл. А.3 представлено результати оцінок цих ризиків.

Таблиця А.3 – результати оцінки ймовірності, впливу та рейтингу ризиків проекту.

№ ризику	**Назва (опис) ризику**	**Ймовірність (0,1-0,9)**	**Вплив (0,05-0,8)**	**Ранг**
1	Обмежені технічні навички розробників.	0.3	0.7	низький

2	Затримка з поставками необхідного обладнання та програмного забезпечення.	0.2	0.7	низький
3	Зміна вимог до проекту з боку замовника.	0.4	0.4	низький
4	Непередбачені технічні проблеми під час розробки.	0.6	0.5	середній
5	Недооцінка часу, необхідного для завершення проекту.	0.4	0.7	високий
6	Відсутність залучення з боку користувачів під час тестування системи.	0.2	0.4	середній
7	Неспроможність системи адаптуватися до зростаючого числа користувачів.	0.2	0.6	середній
8	Складності інтеграції з платіжними системами та системами управління логістики.	0.3	0.6	середній
9	Ризики безпеки даних через потенційні кібератаки або витоки інформації.	0.1	0.6	низький
10	Зміни в трендах, які можуть зробити систему застарілою незабаром після запуску	0.1	0.4	низький
11	Проблеми з масштабуванням системи у відповідності до зростаючого обсягу замовлень.	0.3	0.6	середній
12	Невідповідність інтерфейсу очікуванням користувачів, що може призвести до зниження задоволеності клієнтів та зменшення повторних покупок.	0.1	0.7	середній
13	Збої в системі платежів, що можуть призвести до затримок у обробці замовлень та фінансових втрат для клієнтів і компанії.	0.1	0.8	середній
14	Неправильна інтеграція з іншими системами (наприклад, з логістичними сервісами), що може спричинити помилки в доставці.	0.1	0.5	низький
15	Високий рівень конкуренції на ринку	0.6	0.4	високий

Джерело: розроблено автором

У таблиці А.4 представлено систематичний підхід до класифікації потенційних ризиків проекту. Ризики ранжовані за критеріями, що включають їх ймовірність та вплив, що дозволяє визначити, які з них є прийнятними, які вимагають виправданих заходів реагування, та які є недопустимими і потребують негайного втручання.

Таблиця А.4 – Шкала оцінювання ризиків за ймовірністю виникнення та величиною впливу

Оцінка	**Ймовірність виникнення**	**Вплив ризику**	**Тип ризику**
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Джерело: розроблено автором

Для мінімізації впливу ризиків на проект також необхідно розробити план їх управління. Цей процес включає створення стратегій їх усунення чи зменшення. В результаті була створена матриця, яка ілюструє ймовірність появи ризиків та їх потенційний вплив (табл. А.5). У цій матриці використано кольорову маркування: зелений колір вказує на ризики, які можна прийняти, жовтий – на ті, які є обґрунтованими, а червоний позначає ризики, які є неприпустимими.

Таблиця А.5 – Матриця ймовірності та впливу.

Ймовірність виникнення ризику	**Вплив ризику**								
	0.05	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**
0.9	0.045	0.09	0.18	0.27	0.36	0.45	0.54	0.63	0.72
0.8	0.040	0.08	0.16	0.24	0.32	0.4	0.48	0.56	0.64
0.7	0.035	0.07	0.14	0.21	0.28	0.35	0.42	0.49	0.56
0.6	0.030	0.06	0.12	0.18	0.24 **R15**	0.30 **R4**	0.36	0.42	0.48
0.5	0.025	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40
0.4	0.020	0.04	0.08	0.12	0.16	0.20	0.24	0.28	0.32

					R3			**R5**	
0.3	0.015	0.03	0.06	0.09	0.12	0.15	0.18 **R8, R11**	0.21 **R1**	0.24
0.2	0.010	0.02	0.04	0.06	0.08 **R6**	0.10	0.12 **R7**	0.14 **R2**	0.16
0.1	0.005	0.01	0.02	0.03	0.04 **R10**	0.05 **R14**	0.06 **R9**	0.07 **R12**	0.08 **R13**

Джерело: розроблено автором

У таблиці А.6 визначено кількісні межі для категоризації ризиків. Ризики поділені на три рівні: "Прийнятні", "Виправдані" та "Недопустимі", залежно від їх обчисленої ймовірності виникнення та впливу на проект. Завдяки цьому поділу можна визначити, які ризики потребують моніторингу, засобів мінімізації чи негайних дій для забезпечення успішного виконання проекту.

Таблиця А.6 - Шкала оцінювання ризику за рівнем.

№	Назва	Межі	Ризики, які входять
1	Прийнятні	$0.005 \leq R \leq 0.12$	6, 10, 14, 9
2	Виправдані	$0.12 < R \leq 0.25$	12, 13, 7, 2, 11, 8 ,1 ,3, 15
3	Недопустимі	$0.25 < R \leq 0.72$	4, 5

Джерело: розроблено автором

Таблиця А.7 містить деталізацію цих ризиків та їх оцінки за шкалою ймовірності та впливу, а також ранг кожного ризику. Для кожного з них розроблено типові стратегії реагування, що включають первинні та альтернативні плани дій, аби забезпечити проактивне управління ризиками в процесі розробки проекту.

Таблиця А.7 – Ризики та стратегії реагування на них

ID	Стату с ризик у	Опис ризику	Ймовір ність виникн ення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А	План Б
R1	Новий	Обмежені технічні навички розробників.	Низька	Висок ий	0.21	Передача	Проведення додаткових тренінгів, огляд документації, перегляд курсів	Залучення менторів, викладачів
R2	Новий	Затримка з поставками необхідного обладнання та програмного забезпечення.	Низька	Висок ий	0.14	Зменшення	Покупка підписки на ПО для розробки - PhpStorm	Використання безкоштовного ПО – IDE VSCode
R3	Новий	Зміна вимог до проекту з боку замовника.	Середня	Середня	0.16	Адаптація	Регулярні зустрічі з керівником проекту для перегляду вимог	Розробка модульної системи для легкого внесення змін
R4	Новий	Непередбачені технічні проблеми під час розробки.	Висок ий	Висок ий	0.30	Уникнення	Ретельне планування та аналіз ризиків на етапі проектування	Створення гнучкого плану розвитку продукту
R5	Новий	Недооцінка часу, необхідного для	Середня	Висок ий	0.28	Зменшення	Використання агільних методологій для кращого управління часом	Виділення додаткового часу для кожного етапу

		завершення проекту.						
R6	Новий	Відсутність залучення з боку користувачів під час тестування системи.	Низька	Середня	0.08	Прийняття	Впровадження бета-тестування з користувачами	Залучення зовнішніх експертів для оцінки юзабіліті
R7	Новий	Неспроможність системи адаптуватися до зростаючого числа користувачів.	Низька	Високий	0.12	Уникнення	Тестування системи на масштабованості	Розробка плану масштабування ресурсів
R8	Новий	Складності інтеграції з платіжними системами та системами управління логістики.	Низька	Високий	0.18	Зменшення	Ретельне тестування інтеграції до початку використання	Розробка запасного плану для ручного управління замовленнями
R9	Новий	Ризики безпеки даних через потенційні кібератаки або витоки інформації.	Низька	Високий	0.06	Передача	Впровадження рішень для кібербезпеки	Страхування від кіберризиків

R10	Новий	Зміни в трендах, які можуть зробити систему застарілою незабаром після запуску	Низька	Середня	0.04	Прийняття	Моніторинг трендів і швидке адаптування	Розробка стратегії постійного оновлення
R11	Новий	Проблеми з масштабуванням системи у відповідності до зростаючого обсягу замовлень.	Низька	Високий	0.18	Уникнення	Використання хмарних технологій для масштабування	Розробка плану нарощування апаратних ресурсів
R12	Новий	Невідповідність інтерфейсу очікуванням користувачів, що може призвести до зниження задоволеності клієнтів та зменшення повторних покупок.	Низька	Високий	0.07	Зменшення	Прототипування та залучення користувачів до процесу дизайну	Розробка системи зворотного зв’язку для поліпшення інтерфейсу
R13	Новий	Збої в системі платежів, що можуть	Низька	Високий	0.08	Зменшення	Впровадження надійних платіжних шлюзів	Встановлення процедур швидкого

		призвести до затримок у обробці замовлень та фінансових втрат для клієнтів і компанії.						відновлення після збоїв
R14	Новий	Неправильна інтеграція з іншими системами (з логістичними сервісами), що може спричинити помилки в доставці.	Низька	Середня	0.05	Уникнення	Тестування інтеграції з логістичними сервісами	Створення альтернативних маршрутів доставки
R15	Новий	Високий рівень конкуренції на ринку	Низька	Середня	0.24	Адаптація	Аналіз конкурентів і диференціація продукту	Розробка унікальної пропозиції для користувачів

Джерело: розроблено автором

ДОДАТОК Б

Лістинг коду «CheckoutController.php»:

```
<?php

namespace App\Http\Controllers\Payment;

use App\Contracts\CartServiceInterface;
use App\Contracts\CustomerServiceInterface;
use App\Contracts\DeliveryServiceInterface;
use App\Contracts\OrderServiceInterface;
use App\Contracts\PaymentMethodRepositoryInterface;
use App\Contracts\PaymentServiceInterface;
use App\Contracts\PhoneConfirmationServiceInterface;
use App\Events\OrderCreated;
use App\Http\Controllers\BaseController;
use App\Http\Requests\SubmitCheckoutRequest;
use App\Http\Resources\DeliveryResource;
use App\Http\Resources\PaymentMethodResource;
use App\Services\PhoneConfirmationService;
use Auth;

class CheckoutController extends BaseController
{
    /**
     * Display a listing of the resource.
     */
    public function index(
        DeliveryServiceInterface $deliveryService,
        PaymentServiceInterface $paymentMethod,
        PhoneConfirmationServiceInterface $phoneConfirmationService
    ) {
        return $this->showView('Payment/Checkout', [
            'deliveryMethods' => DeliveryResource::collection($deliveryService->getMethods()),
            'paymentMethods' => PaymentMethodResource::collection($paymentMethod->getMethods()),
```

```
            'isVerifiedPhone' => session('isVerifiedPhone'),
            'cachedPhones' => $phoneConfirmationService->getCachedPhones()
        ]);
    }

    public function submitCheckoutForm(
        SubmitCheckoutRequest $request,
        CustomerServiceInterface $customerService,
        PhoneConfirmationService $phoneConfirmationService,
        OrderServiceInterface $orderService,
        CartServiceInterface $cartService,
        PaymentServiceInterface $paymentService,
        PaymentMethodRepositoryInterface $paymentMethodRepository
    ) {
        $data = $request->validated();

        if (!$phoneConfirmationService->checkPhoneVerification($data['phone'],
$data['confirm_phone_code'])) {
            return back()->withInput()->withErrors(['phone' =>
__('messages.phone_is_not_verified')]);
        }
        $customer = $customerService->createCustomerIfNotExist([
            "user_id" => Auth::id(),
            'name' => $data['lastName'] . " " . $data['firstName'],
            'phone' => $data['phone'],
            'email' => $data['email'],
            'address' => $customerService->generateAddress(),
            'city' => $customerService->getCity(),
            'state' => $customerService->getState(),
            'country' => $customerService->getCountry(),
            'postal_code' => $customerService->getPostalCode(),
            'ip' => $customerService->getIpAddress(),
            'currency' => $customerService->getCurrency(),
            'coordinates' => $customerService->getCoordinates(),
        ]);
        $order = $orderService->createOrder($customer, $data);
        if (!$order) {
```

```
            return back()->with('error', __('messages.order_not_created'));
        }

        $paymentMethod = $paymentMethodRepository->getPaymentMethodByKey($data['paymentMethod']);
        $paymentService->createPayment($order, $paymentMethod);

        $cartService->clearCart();

        event(new OrderCreated($order));

        if ($data['call_me_back']) {
            return                                  redirect(route('products.index'))->with('success',
__('messages.order_successfully_created'));
        }

        return redirect(route('products.index'))->with(
            'success',
            __('messages.order_successfully_created_without_callback')
        );
    }
}
```

Лістинг коду OrderService.php:

```
<?php

namespace App\Services;

use App\Contracts\CartServiceInterface;
use App\Contracts\DeliveryServiceInterface;
use App\Contracts\NovaPoshtaServiceInterface;
use App\Contracts\OrderItemRepositoryInterface;
use App\Contracts\OrderRepositoryInterface;
use App\Contracts\OrderServiceInterface;
use App\Contracts\PaymentServiceInterface;
use App\Contracts\ProductRepositoryInterface;
use App\Enums\CurrencyEnum;
use App\Enums\DeliveryEnum;
use App\Enums\OrderStatusEnum;
use App\Events\OrderCanceled;
use App\Events\OrderConfirmed;
use App\Events\OrderDelivered;
use App\Events\OrderDone;
use App\Events\OrderShipped;
use App\Events\PendingPaymentForOrder;
use App\Models\Customer;
use App\Models\Order;
use App\Models\Product;
use Illuminate\Support\Collection;

class OrderService implements OrderServiceInterface
{
    private OrderRepositoryInterface $repository;
    private CartServiceInterface $cartService;
    private DeliveryServiceInterface $deliveryService;
    private ProductRepositoryInterface $productRepository;
    private OrderItemRepositoryInterface $orderItemRepository;
    private array $quantityMapper = [];
    private Collection $productsInCart;
```

```
    private PaymentServiceInterface $paymentService;
    /**
     *                                                                                                    @var
NovaPoshtaServiceInterface|(NovaPoshtaServiceInterface&\Illuminate\Contracts\Foundation\Application)|
\Illuminate\Contracts\Foundation\Application|\Illuminate\Foundation\Application|mixed
     */
    private mixed $novaPoshtaService;

    public function __construct()
    {
        $this->repository = app(OrderRepositoryInterface::class);
        $this->cartService = app(CartServiceInterface::class);
        $this->deliveryService = app(DeliveryServiceInterface::class);
        $this->productRepository = app(ProductRepositoryInterface::class);
        $this->orderItemRepository = app(OrderItemRepositoryInterface::class);
        $this->paymentService = app(PaymentServiceInterface::class);
        $this->novaPoshtaService = app(NovaPoshtaServiceInterface::class);
        $this->initOrderProducts();
    }

    /**
     * @return void
     */
    private function initOrderProducts(): void
    {
        $cart = $this->cartService->getItems();
        $ids = collect($cart->items)->map(function ($cartItemResource) {
            if (is_array($cartItemResource)) {
                $this->quantityMapper[$cartItemResource['product']['id']]                          =
$cartItemResource['quantity'];
                return $cartItemResource['product']['id'];
            }
            $this->quantityMapper[$cartItemResource->resource->product_id]    =    $cartItemResource-
>resource->quantity;
            return $cartItemResource->resource->product_id;
        })->toArray();
        $this->productsInCart = $this->productRepository->getProductsById($ids);
```

```
    }

    /**
     * Generate an order number for the given customer.
     *
     * @param Customer $customer The customer for whom the order number is generated.
     * @return string The generated order number.
     */
    public function generateOrderNumber(Customer $customer): string
    {
        return 'ORD-' . $customer->id . '-' . substr(str_shuffle('ABCDEFGHIJKLMNOPQRSTUVWXYZ'), 0, 4);
    }

    public function createOrder(Customer $customer, array $data)
    {
        $delivery = $this->deliveryService->findDeliveryByKey($data['deliveryMethod']);
        $orderData = [
            'customer_id' => $customer->id,
            'delivery_id' => $delivery?->id,
            'delivery_details' => $this->getDeliveryDetailsJson($data),
            'order_number' => $this->generateOrderNumber($customer),
            'total_price'      =>      $this->countOrderTotalPrice($this->productsInCart,      $this-
>quantityMapper),
            'currency_name' => CurrencyEnum::UAH->value,
            'notes' => $data['notes'] ?? null,
            'callback' => $data['call_me_back'] ?? true,
            'status' => OrderStatusEnum::PENDING->value,
        ];

        $order = $this->repository->createOrder($orderData);
        $orderItemsData    =    $this->getOrderItemsData($order,    $this->productsInCart,    $this-
>quantityMapper);

        $this->orderItemRepository->createOrderItems($orderItemsData);

        return $order;
    }
```

```
/**
 * @param Product[] $products
 * @param array $quantityMapper
 * @return int
 */
public function countOrderTotalPrice(array|Collection $products, array $quantityMapper): int
{
    $totalPrice = 0;
    foreach ($products as $product) {
        $totalPrice += $this->cartService->calcProductPrice(
            $product,
            $quantityMapper[$product->id]
        );
    }
    return $totalPrice;
}

private function getDeliveryDetailsJson($data): array
{
    switch ($data['deliveryMethod']) {
        case DeliveryEnum::NOVAPOSHTA_DEPARTMENT->value:
            return [
                'deliveryDepartment' => $data['deliveryDepartment'],
                'deliveryDepartmentCity' => $data['deliveryDepartmentCity']
            ];
        default:
        case DeliveryEnum::NOVAPOSHTA_COURIER->value:
            return [
                'city' => $data['city'],
                'street' => $data['street'],
                'house' => $data['house'],
                'flat' => $data['flat']
            ];
    }
}
```

```
/**
 * @param Order $order
 * @param Collection $products
 * @param array $quantityMapper
 * @return array
 */
public function getOrderItemsData(Order $order, Collection $products, array $quantityMapper): array
{
    $orderItems = [];
    $timestamp = now();
    foreach ($products as $product) {
        $quantity = $quantityMapper[$product->id];
        $price = $this->cartService->calcProductPrice($product, $quantity);
        $orderItems[] = [
            'order_id' => $order->id,
            'product_id' => $product->id,
            'quantity' => $quantity,
            'price' => $price,
            'created_at' => $timestamp,
            'updated_at' => $timestamp,
        ];
    }

    return $orderItems;
}

public function setOrderCanceled(Order $order, string $reason, bool $notify_customer = true): void
{
    $this->paymentService->cancelPayment($order->payment);
    $order->status = OrderStatusEnum::CANCELED->value;
    $order->reason = $reason;
    $order->save();
    event(new OrderCanceled($order, $notify_customer));
}

public function confirmOrder(Order $order): bool
```

```
{
    $payment = $order->payment;
    if (!$payment->paymentMethod->isCash() && !$payment->isSuccess()) {
        $order->status = OrderStatusEnum::PENDING_PAYMENT->value;
        if ($saved = $order->save()) {
            event(new PendingPaymentForOrder($order));
            return $saved;
        }
    } else {
        if ($payment->paymentMethod->isCash() || $payment->isSuccess()) {
            $order->status = OrderStatusEnum::PROCESSING->value;
            if ($saved = $order->save()) {
                event(new OrderConfirmed($order));
                return $saved;
            }
        }
    }
    return false;
}

public function setOrderShipped(Order $order, string $status_details): void
{
    $order->status = OrderStatusEnum::SHIPPED->value;
    $order->status_details = $status_details;
    $order->save();
    event(new OrderShipped($order));
}

public function setOrderDelivered(Order $order, string $status_details): void
{
    $order->status = OrderStatusEnum::DELIVERED->value;
    $order->status_details = $status_details;
    $order->save();
    event(new OrderDelivered($order));
}

public function setOrderDone(Order $order, string $status_details): void
```

```
    {
        $order->status = OrderStatusEnum::DONE->value;
        $order->status_details = $status_details;
        $order->save();
        event(new OrderDone($order));
    }

    public function updateOrderStatusDetails(Order $order, string $orderStatusDetails): void
    {
        $order->status_details = $orderStatusDetails;
        $order->save();
    }
}
```

Лістинг коду docker-compose.yml:

```
version: '3'
services:
    web:
        build:
            context: ./docker/8.2
            dockerfile: Dockerfile
            args:
                WWWGROUP: '${WWWGROUP}'
        image: sail-8.2/app
        extra_hosts:
            - 'host.docker.internal:host-gateway'
        ports:
            - '${APP_PORT:-80}:80'
            - '${VITE_PORT:-5173}:${VITE_PORT:-5173}'
        environment:
            WWWUSER: '${WWWUSER}'
            LARAVEL_SAIL: 1
            XDEBUG_MODE: '${SAIL_XDEBUG_MODE:-off}'
            XDEBUG_CONFIG: '${SAIL_XDEBUG_CONFIG:-client_host=host.docker.internal}'
            PHP_IDE_CONFIG: 'serverName=sail'
        volumes:
            - '.:/var/www/html'
        networks:
            - sail
        depends_on:
            - mysql
            - redis
            - mailpit
    mysql:
        image: 'mysql/mysql-server:8.0'
        ports:
            - '${FORWARD_DB_PORT:-3306}:3306'
        environment:
            MYSQL_ROOT_PASSWORD: '${DB_PASSWORD}'
```

```
            MYSQL_ROOT_HOST: '%'
            MYSQL_DATABASE: '${DB_DATABASE}'
            MYSQL_USER: '${DB_USERNAME}'
            MYSQL_PASSWORD: '${DB_PASSWORD}'
            MYSQL_ALLOW_EMPTY_PASSWORD: 1
        volumes:
            - 'sail-mysql:/var/lib/mysql'
            -    './vendor/laravel/sail/database/mysql/create-testing-database.sh:/docker-entrypoint-
initdb.d/10-create-testing-database.sh'
        networks:
            - sail
        healthcheck:
            test:
                - CMD
                - mysqladmin
                - ping
                - '-p${DB_PASSWORD}'
            retries: 3
            timeout: 5s
    redis:
        image: 'redis:alpine'
        ports:
            - '${FORWARD_REDIS_PORT:-6379}:6379'
        volumes:
            - 'sail-redis:/data'
        networks:
            - sail
        healthcheck:
            test:
                - CMD
                - redis-cli
                - ping
            retries: 3
            timeout: 5s
    mailpit:
        image: 'axllent/mailpit:latest'
        ports:
```

```
            - '${FORWARD_MAILPIT_PORT:-1025}:1025'
            - '${FORWARD_MAILPIT_DASHBOARD_PORT:-8025}:8025'
        networks:
            - sail
networks:
    sail:
        driver: bridge
volumes:
    sail-mysql:
        driver: local
    sail-redis:
        driver: local
```

Лістинг коду app.js:

```
import "./bootstrap";
// import '../css/app.css';
import { createSSRApp, h } from "vue";
import { createInertiaApp } from "@inertiajs/vue3";
import { resolvePageComponent } from "laravel-vite-plugin/inertia-helpers";
import { ZiggyVue } from "../../vendor/tightenco/ziggy/dist/vue.m";
import { translations } from "@/Mixins/translations.js";
import common from "@/Mixins/common.js";
import { store } from "@/store.js";
import Notifications from "@kyvg/vue3-notification";
import { $request } from "@/Services/RequestService";
import notifier from "@/Mixins/notifier.js";
import { $search } from "@/Services/SearchService";
import FloatingVue from "floating-vue";
import "floating-vue/dist/style.css";

const appName =
    window.document.getElementsByTagName("title")[0]?.innerText || "Laravel";

const app = createInertiaApp({
    title: (title) => `${title} - ${appName}`,
    resolve: (name) =>
        resolvePageComponent(
            `./Pages/${name}.vue`,
            import.meta.glob("./Pages/**/*.vue")
        ),
    setup({ el, App, props, plugin }) {
        const app = createSSRApp({ render: () => h(App, props) })
            .use(plugin)
            .mixin(translations)
            .mixin(common)
            .mixin(notifier)
            .use(ZiggyVue, Ziggy)
            .use(Notifications)
```

```
            .use(FloatingVue)
            .use(store);

        app.config.globalProperties.$request = $request;
        app.config.globalProperties.$search = $search;
        app.mount(el);
        return app;
    },
    progress: {
        color: "#ff5722",
    },
});
```

Лістинг коду Wishlist.vue:

```
<template>
  <div>
    <Head :title="__('products.head')" />
    <Breadcrumbs v-bind="{breadcrumbs}" />
    <section class="section">
      <div class="container small-size">
        <template v-if="wishlist.data.length > 0">
          <ProductList
            :products="wishlist.data"
            :meta="wishlist.meta"
            :links="wishlist.links"
          />
        </template>
        <div
          v-else
          class="text-center"
        >
          {{ __('wishlist.empty_wishlist') }}
        </div>
      </div>
    </section>
  </div>
</template>

<script>
import Breadcrumbs from '@/Components/layouts/Breadcrumbs.vue';
import BaseLayout from '@/Layouts/BaseLayout.vue';
import ProductList from '@/Components/Shop/Products/ProductList.vue';

export default {
    name: 'Wishlist',
    components: {
        Breadcrumbs,
        ProductList,
```

```
    },
    layout: [BaseLayout],
    props: {
        wishlist: {
            type: Object,
        },
    },
    computed: {
        breadcrumbs() {
            return [
                {icon: 'ti-home', to: this.route('index')},
                {label: this.__('main.wishlist'), to: this.route('wishlist.index')},
            ];
        },
    },
};
</script>

<style scoped></style>
```

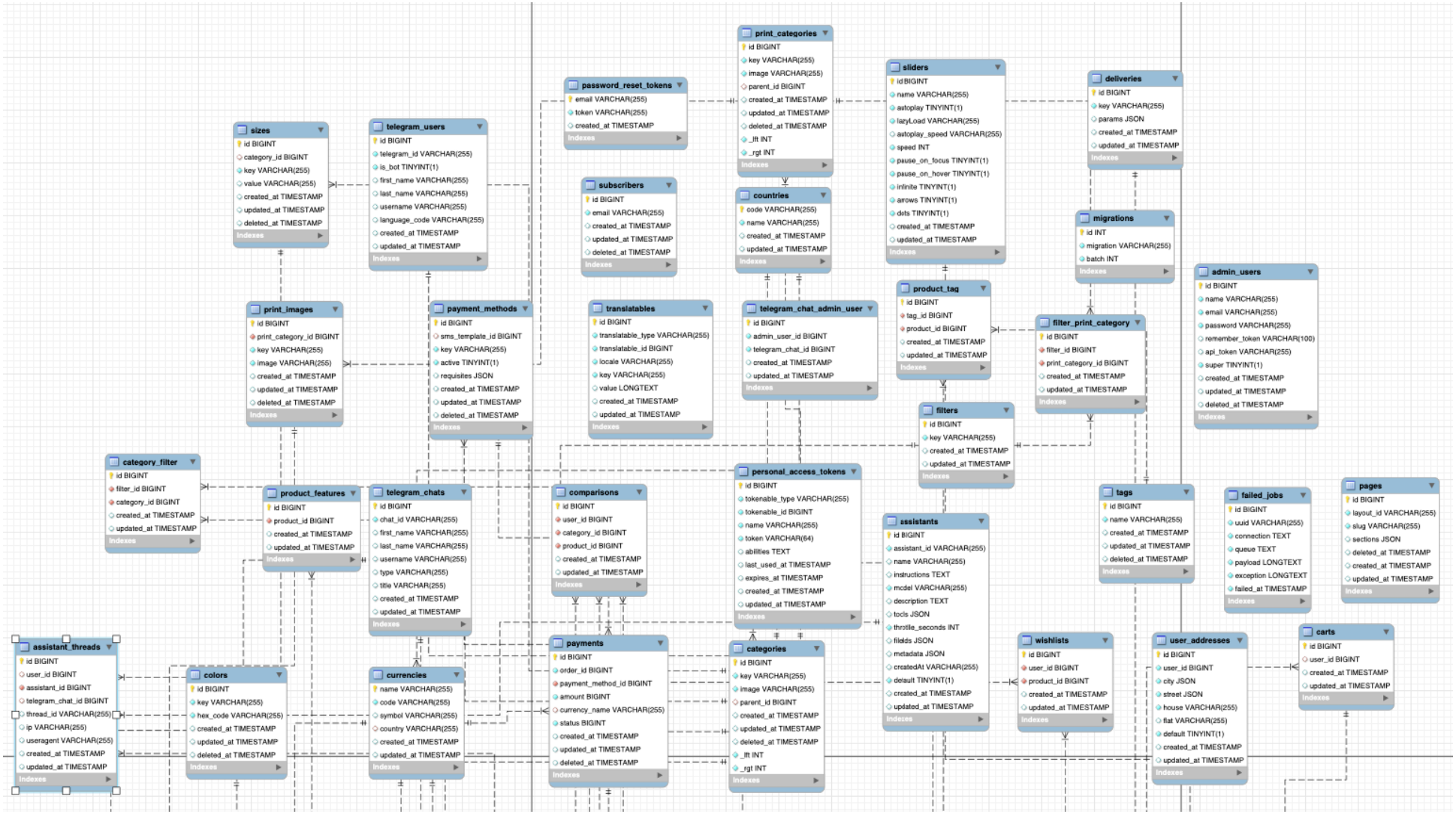


Рисунок Б.1– Фізична модель бази даних ІС підтримки обслуговування клієнтів інтернет-магазину “printcrafters.com.ua”. Частина 1.

Джерело: розроблено автором

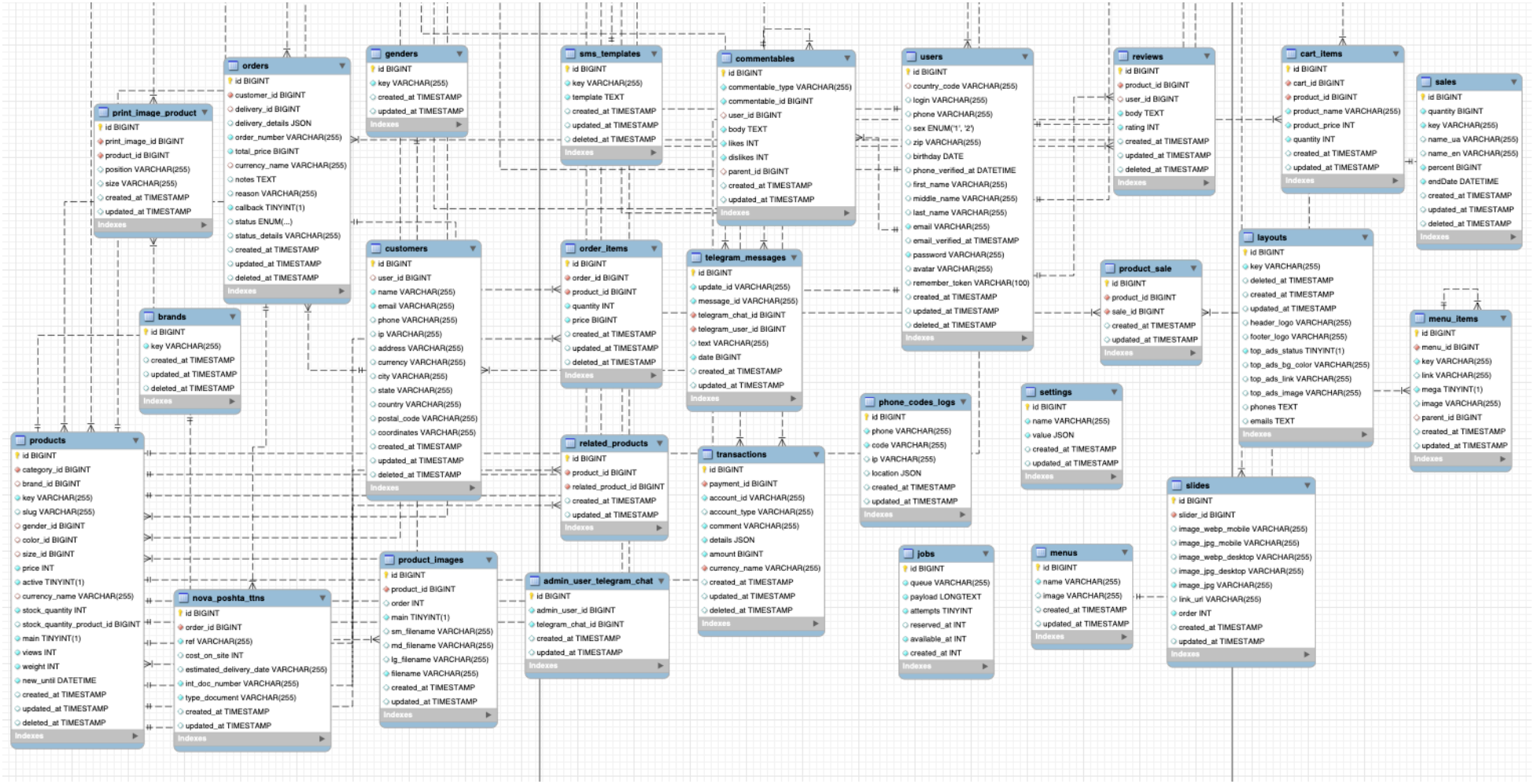


Рисунок Б.2– Фізична модель бази даних ІС підтримки обслуговування клієнтів інтернет-магазину “printcrafters.com.ua”. Частина 2.

Джерело: розроблено автором