МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Навчально-науковий інститут бізнесу, економіки та менеджменту "БіЕМ"

(повна назва інституту/факультету)

Кафедра економічної кібернетики

(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

_____ Віталія КОЙБІЧУК

(підпис) (Ім'я та ПРІЗВИЩЕ)

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

(бакалавр / магістр)

зі спеціальності 051 «Економіка»

(код та назва)

освітньо-професійної програми Економічна кібернетика та бізнес аналітика

(освітньо-професійної / освітньо-наукової) (назва програми)

на тему: Розробка веборієнтованої інформаційної системи магазину годинників

Здобувача (ки) групи ЕК-01а Дидишка Андрія Михайловича

(шифр груп) (прізвище, ім'я, по батькові)

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

_____ Андрій ДИДИШКО

(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доцент, канд. техн. наук. Валерій ЯЦЕНКО _____

(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2024

АНОТАЦІЯ

кваліфікаційної роботи бакалавра
на тему

«РОЗРОБКА ВЕБОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ МАГАЗИНУ ГОДИННИКІВ»

студента Дидишко Андрія Михайловича
(прізвище, ім’я, по-батькові)

Актуальність теми, обраної теми визначається тим, що у сучасних умовах розвитку інформаційних технологій бізнес все частіше переходить у цифровий формат. Веборієнтовані інформаційні системи забезпечують зручний та ефективний спосіб управління підприємством..

Метою даної роботи є розробка веборієнтованої інформаційної системи для магазину годинників.

Об’єкт дослідження – процеси автоматизації торгівлі та управління бізнесом у сфері роздрібного продажу годинників.

Предметом дослідження є методи та засоби розробки веборієнтованих інформаційних систем для автоматизації процесів управління магазином годинників.

Методи дослідження для досягнення поставлених завдань: аналіз, узагальнення, моделювання, аналогія, методи і технології розробки програмного забезпечення за допомогою фреймворків Laravel.

Результати роботи можуть бути використані для створення ефективних веборієнтованих інформаційних систем для магазинів різних товарів, зокрема, для магазинів годинників.

Завдання до кваліфікаційної роботи:

– проаналізувати тенденції розвитку електронної комерції та ринку годинників;

– сформулювати вимоги до веборієнтованої інформаційної системи для магазину годинників;

– побудувати моделі бізнес-процесів магазину годинників;

– описати архітектуру інформаційної системи та технології розв’язання поставлених задач;

– описати структуру та особливості реалізації інформаційного забезпечення;

– описати структуру та особливості реалізації алгоритмічного забезпечення;

– реалізувати веборієнтовану інформаційну систему для магазину годинників;

– розробити інструкції з використання інформаційної системи.

Ключові слова: веборієнтована система, автоматизація, продаж годинників, бізнес-процеси, веббраузер, MySQL, PHP, Laravel.

Зміст кваліфікаційної роботи викладено на 96 сторінках. Список використаних джерел із 43 найменувань, розміщений на 4 сторінках.

Робота містить 2 таблиці, 26 рисунків, а також 5 додатків, розміщених на 29 сторінках.

Рік виконання кваліфікаційної роботи – 2024 рік.

Рік захисту роботи – 2024 рік.

Міністерство освіти і науки України
Сумський державний університет
Навчально-науковий інститут бізнесу, економіки та менеджменту
Кафедра економічної кібернетики

ЗАТВЕРДЖУЮ
Завідувачка кафедри
к.е.н., доцентка
_____ Віталія КОЙБІЧУК
“26” березня 2024 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
БАКАЛАВРА
спеціальність 051 «Економіка (Економічна кібернетика)»студенту *IV* курсу, групи *ЕК-01а*

Дидишко Андрію Михайловичу
(прізвище, ім’я, по батькові студента)

1. Тема роботи *Розробка веборієнтованої інформаційної системи магазину годинників*

затверджена наказом по університету від « 26»березня2024 року №0486-VI

2. Термін подання студентом закінченої роботи «27» травня 2024 року

3. Мета кваліфікаційної роботи *розробка інтернет-магазину годинників, що базується на сучасних технологіях і забезпечує високу якість обслуговування клієнтів*

4. Об’єкт дослідження *процеси автоматизації торгівлі та управління бізнесом у сфері роздрібного продажу годинників*

5. Предмет дослідження *розробка веборієнтованої інформаційної системи для інтернет магазину годинників*

6. Кваліфікаційна робота виконується на матеріалах

7. Орієнтовний план кваліфікаційної роботи, терміни подання розділів керівникові та зміст завдань для виконання поставленої мети

Розділ 1. *Дослідження тенденцій електронної комерції та формування вимог до інформаційної системи* *15.04.2024*
(назва – термін подання)

У розділі 1 *Проаналізувати тенденції розвитку електронної комерції та ринку годинників. Сформулювати вимоги до веборієнтованої інформаційної системи для магазину годинників. Побудувати моделі бізнес-процесів магазину годинників. Описати архітектуру інформаційної системи та технології розв’язання поставлених задач.*

(зміст конкретних завдань до розділу, які має виконати студент)

Розділ 2. *Реалізація прототипу веборієнтованої інформаційної системи для магазину годинників* *13.05.2024*
(назва – термін подання)

У розділі 2 *Описати структуру та особливості реалізації інформаційного забезпечення. Описати структуру та особливості реалізації алгоритмічного забезпечення. Реалізувати веборієнтовану інформаційну систему для магазину годинників. Розробити інструкції з використання інформаційної системи.*

(зміст конкретних завдань до розділу, які повинен виконати студент)

8. Консультації з роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1			
2			
3			

9. Дата видачі завдання: «01» квітня 2024 року

Керівник кваліфікаційної роботи _____ Яценко В.В
(підпис) (ініціали, прізвище)

Завдання до виконання одержав _____ Дидишко А.М
(підпис) (ініціали, прізви)

ЗМІСТ

ВСТУП 7

1 ДОСЛІДЖЕННЯ ТЕНДЕНЦІЙ ЕЛЕКТРОННОЇ КОМЕРЦІЇ ТА ФОРМУВАННЯ ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ 10

1.1. Трансформація вітчизняного ринку систем електронної комерції 10

1.2. Вимоги до веборієнтованої інформаційної системи 14

1.3. Моделі бізнес-процесів 22

1.4. Архітектура інформаційної системи та технології розв’язання 30

поставлених задач 30

2 РЕАЛІЗАЦІЯ ПРОТОТИПУ ВЕБОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ МАГАЗИНУ ГОДИННИКІВ 38

2.1. Структура та особливості реалізації інформаційного забезпечення .. 38

2.2. Структура та особливості реалізації алгоритмічного забезпечення ... 47

2.3 Реалізація прототипу інформаційної системи та інструкції з використання 51

ВИСНОВКИ 62

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 64

ДОДАТКИ 68

ДОДАТОК А 69

ДОДАТОК Б 70

ДОДАТОК В 78

ДОДАТОК Г 84

ДОДАТОК Д 96

ВСТУП

Актуальність дослідження. Сучасний ринок споживчих товарів зазнає значних змін під впливом швидкого розвитку інформаційних технологій. Однією з найважливіших складових цього процесу є впровадження веборієнтованих інформаційних систем, які сприяють підвищенню ефективності управління бізнес-процесами та забезпечують зручний доступ до інформації для споживачів. Особливо це актуально для роздрібної торгівлі, де якісна взаємодія з клієнтами відіграє ключову роль у забезпеченні конкурентоспроможності підприємства.

Магазини годинників, як частина індустрії розкоші, потребують особливої уваги до деталей у процесі обслуговування клієнтів. Високий рівень конкуренції та специфічні вимоги споживачів вимагають створення інформаційних систем, які забезпечать зручний доступ до асортименту товарів, інформації про їх характеристики, наявність та ціни, а також можливість здійснення покупок онлайн.

Успішна реалізація даного проекту сприятиме підвищенню конкурентоспроможності магазину годинників, зростанню обсягів продажів та покращенню задоволеності клієнтів.

У сучасних умовах стрімкого розвитку електронної комерції все більше підприємств переходять до онлайн-продажів, що обумовлює необхідність створення ефективних та зручних у використанні інтернет-магазинів. Для магазину годинників, як і для багатьох інших спеціалізованих торгових точок, наявність інтернет-магазину відкриває нові можливості для розширення ринку збуту, залучення нових клієнтів та підвищення рівня продажів.

Розробка веборієнтованої інформаційної системи для магазину годинників є актуальною через декілька ключових причин. Веборієнтована інформаційна система автоматизує багато рутинних операцій, таких як управління товарними запасами, обробка замовлень, взаємодія з постачальниками та клієнтами. Це дозволяє зменшити витрати на персонал, знизити ймовірність помилок та підвищити ефективність роботи підприємства.

Інтернет-магазин надає можливість збирати та аналізувати дані про поведінку користувачів, їхні уподобання та потреби. Це дозволяє розробляти ефективні маркетингові стратегії, адаптувати асортимент товарів та покращувати якість обслуговування.

Мета роботи – розробка інтернет-магазину годинників, що буде базуватися на сучасних вебтехнологіях і забезпечить високу якість обслуговування клієнтів, автоматизацію процесів управління товарними запасами та інтеграцію з платіжними системами.

Об’єкт дослідження – процеси автоматизації торгівлі та управління бізнесом у сфері роздрібного продажу годинників.

Предмет – розробка веборієнтованої інформаційної системи для інтернет-магазину годинників.

Відповідно до мети були визначені наступні завдання:

– проаналізувати тенденції розвитку електронної комерції та ринку годинників;

– сформулювати вимоги до веборієнтованої інформаційної системи для магазину годинників;

– побудувати моделі бізнес-процесів магазину годинників;

– описати архітектуру інформаційної системи та технології розв’язання поставлених задач;

– описати структуру та особливості реалізації інформаційного забезпечення;

– описати структуру та особливості реалізації алгоритмічного забезпечення;

– реалізувати веборієнтовану інформаційну систему для магазину годинників;

– розробити інструкції з використання інформаційної системи.

Для розв’язання поставлених завдань нами були використані такі методи дослідження. теоретико-критичний аналіз літератури з теми дослідження; зіставлення, узагальнення і синтезування здобутої інформації тощо.

Робота може бути використана студентами ВНЗ для підготовки до семінарських занять, також може бути використана викладачами для проведення лекції, практик тощо.

Структура роботи. Робота складається зі вступу, двох розділів, висновків, списку використаних джерел, що містить 43 найменування. Повний обсяг роботи 96 сторінок.

1 ДОСЛІДЖЕННЯ ТЕНДЕНЦІЙ ЕЛЕКТРОННОЇ КОМЕРЦІЇ ТА ФОРМУВАННЯ ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1. Трансформація вітчизняного ринку систем електронної комерції

Ринок електронної комерції в Україні демонструє значне зростання останніми роками, що пов'язано зі збільшенням доступу до інтернету та розвитком цифрової інфраструктури. Кількість користувачів інтернету постійно зростає, що стимулює зростання онлайн-продажів. Зокрема, війна прискорила перехід до онлайн-шопінгу, оскільки багато фізичних магазинів були тимчасово закриті, а споживачі почали більше покладатися на електронну комерцію. Українські інтернет-магазини, такі як Rozetka, Prom.ua, та інші, зміцнили свої позиції на ринку, розширюючи асортимент товарів та покращуючи умови доставки і сервісу.

Ринок годинників в Україні також демонструє позитивну динаміку. Годинники залишаються популярним товаром серед українських споживачів, причому зростає попит як на традиційні механічні годинники, так і на розумні годинники. Серед традиційних годинників популярні такі бренди, як Casio, Seiko, Tissot та інші. Розумні годинники, такі як Apple Watch, Samsung Galaxy Watch та Xiaomi Mi Band, стають все більш затребуваними завдяки своїм багатофункціональним можливостям, включаючи моніторинг здоров'я, фітнес-трекінг та інтеграцію зі смартфонами. Онлайн-продажі годинників займають значну частку ринку, оскільки споживачі цінують зручність замовлення товарів через інтернет з можливістю детально ознайомитись з характеристиками продукту та прочитати відгуки.

Загалом ринок електронної комерції в Україні продовжує розвиватися, і очікується, що ця тенденція збережеться в найближчі роки. Зростання попиту на годинники, як традиційні, так і розумні, свідчить про стабільний інтерес споживачів до цього сегменту товарів.

Завдяки розвитку систем електронної комерції кардинально змінюється вітчизняний ринок, це призводить до помітних змін як на національному рівні, так і серед окремих господарств. Електронна комерція, в свою чергу, характеризується переліком відмінностей, серед яких:

– Змінюється традиційний механізм продажу товарів.

– Відбувається поширення вебплатформ для реалізації ключових бізнес-процесів.

– Спостерігається зміна поведінки суб'єктів електронної комерції.

– Цифрові технології виступають посередником між підприємством та споживачем, спрощуючи взаємодію та усуваючи необхідність безпосереднього контакту.

З початком війни з’явились серйозні виклики для ринку електронної комерції. Станом на 22 лютого 2023 року, середньоринковий рівень відвідування магазинів знизився на 82,7%, а середній дохід впав на 92%. Протягом лютого-березня 2022 року існувала тенденція зниження доходу на 1 покупця. Незважаючи на позитивні прогнози щодо відновлення ринкового доходу до 52% від довоєнного рівня у 2023 році, очікується, що рівень доходу на кінець 2027 року лише досягне показників 2020 року (рис. 1.1.).

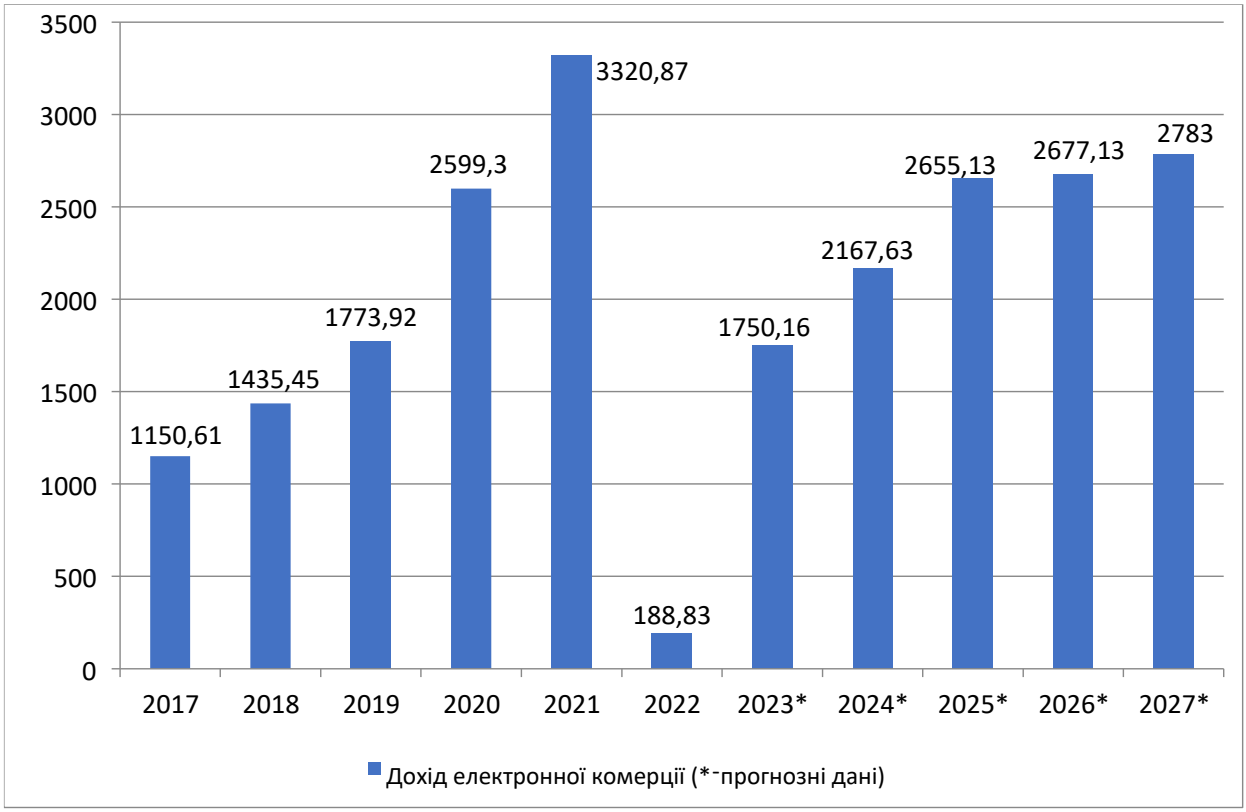

Рисунок 1.1 – Реальні та прогнозовані доходи ринку електронної комерції в Україні в період з 2017 до 2027 року, млн дол. США

В 2023 році частка електронної комерції на ринку становила близько 11%, що свідчить про подальший розвиток цього ринку. Це спричинено тим, що електронна комерція зазнала шокове падіння з початком повномасштабного вторгнення, а після цього відбувся сплеск попиту на деякі категорії товарів, і лише після цього всього досягнули певної стабілізації після масової як внутрішньої так і зовнішньої міграції людей, переміщення складів та виробництв до більш безпечних середовищ.

Стабілізації ринку сприяли продажі товарів, яких споживачі найбільше потребували: господарські товари, гігієнічні засоби, товари для тварин тощо. Саме завдяки цим категоріям вдалося досягти стабільності на ринку під час спричиненої кризи.

До війни вітчизняному ринку електронної комерції було характерне зростання частки населення з доступом до Інтернету. За даними звіту EcommerceEurope, у 2022 році очікувалося, що 67% населення користуватимуться

Інтернетом, тоді як у 2020 і 2021 роках цей показник становив 62% і 65% відповідно[4]. Проте, внаслідок значної кількості внутрішньо переміщених осіб, міграції та тимчасової окупації частини території України, частка населення знизилася приблизно на 15% у 2022 році. На сьогодні з 22,1 млн громадян України близько 19 млн мають доступ до Інтернету, що складає приблизно 85%. Частка тих, хто купує товари та послуги за допомогою Інтернету, зазвичай була нижчою і становила 44% станом на 2021 рік. Тобто, населення зменшилося приблизно на 23%, а кількість інтернет-користувачів – на 22,5%

Варто відзначити, що одночасно відбулося суттєве зменшення частки населення, яке фактично проживає в Україні, та зниження частки тих, хто купує товари в Інтернеті та має достатню купівельну спроможність, що в значній мірі спричинено інфляційними процесами. У майбутньому це може призвести до уповільнення розвитку ринку електронної комерції[1].

З поширенням електронної комерції прогнозується збільшення частоти покупок, що спричинить посилення конкуренції за увагу та лояльність споживачів за допомогою SMM інструментів. Використання автоматизованих маркетингових інструментів дозволить застосовувати персоналізовані підходи до потреб споживачів, підвищуючи їхню лояльність та активність.

Ринок годинників в Україні рухається відповідно до світових тенденцій і проявляє високу динаміку. До 2020 року спостерігалося значне зростання виробництва та продажу годинників, зокрема швейцарських. Проте, пандемія коронавірусу призвела до повного замороження цього ринку у всьому світі. Згідно з даними Федерації швейцарської годинникової промисловості, у першому півріччі 2020 року поставки годинників знизилися на 36%, досягнувши 6.9 мільярда швейцарських франків. Прогнозується, що до кінця 2020 року експорт швейцарських годинників продовжить зменшуватися (близько 30%), оскільки кількість випадків захворювання на COVID-19 продовжує зростати, що призводить до закриття ринків збуту[2]. Можливо, 2021 рік принесе більшу стабільність швейцарській годинниковій індустрії, переважно завдяки піднесенню китайського ринку, який поступово набирає обертів.

1.2. Вимоги до веборієнтованої інформаційної системи

Сайт повинен мати привабливий вигляд з використанням привабливих кольорів та графічних елементів, що стимулюють користувачів до перегляду та покупки товарів. Важливо, щоб дизайн передавав позитивний настрій та відображав тематику годинників.

Сайт має мати легку та зрозумілу навігацію, що дозволяє користувачам швидко знаходити потрібні товари. Для цього варто використовувати зручне меню, фільтри, пошукову систему та категорії.

Важливо, щоб сайт був адаптивним та зручно відображався на будь-яких пристроях, включаючи комп'ютери, планшети та мобільні телефони. Це забезпечить зручність перегляду та покупки товарів незалежно від пристрою.

Головна сторінка магазину має містити ключову інформацію та привертати увагу користувачів. Рекомендується відображати популярні товари, нові надходження, акції та спеціальні пропозиції.

Дизайн сайту повинен передбачати зручний функціонал кошика, де користувачі можуть додавати товари та керувати їх кількістю. Оформлення замовлення повинно бути простим і зрозумілим, з можливістю вказати доставку та оплату [3].

При створенні дизайну для інтернет-магазину з продажу годинників, настійно рекомендується мати візуальне представлення кожного товару, що включає в себе зображення високої якості, докладні описи і можливість докладного розгляду товару у різних ракурсах. Крім того, використання захисних протоколів та сертифікатів безпеки є невід'ємною частиною створення безпечного середовища для користувачів, забезпечуючи конфіденційність та захист їх особистої інформації. Також варто розглянути можливість включення відгуків користувачів, рейтингів товарів та символів довіри, які підвищують впевненість покупців у магазині.

З урахуванням цих вимог до дизайну, можна створити привабливий та функціональний інтернет-магазин з продажу годинників, який задовольнятиме потреби користувачів.

Основний функціонал сайту включає:

1. Керування товарами. Створення нового товару з вказанням основних характеристик, отримання списку всіх товарів та інформації про конкретний товар за його ідентифікатором.

2. Керування типами товарів та брендами. Створення нових типів товарів і брендів з можливістю управління списком інформації [32].

3. Керування рейтингом та кошиком. Додавання та управління рейтингом для товарів, а також створення та управління кошиком з можливістю додавання товарів.

4. Завантаження файлів. Можливість користувачів завантажувати зображення для товарів з подальшою обробкою та збереженням на сервері.

5. Управління замовленнями. Реалізація функціоналу для оформлення та керування замовленнями користувачів. Це включає в себе можливість перегляду списку замовлень, їх детальної інформації, а також можливість зміни статусу замовлення.

6. Пошук товарів. Додавання можливості пошуку товарів за різними критеріями, такими як назва, категорія, бренд тощо. Це забезпечить користувачам зручний і швидкий спосіб знаходження потрібних товарів.

7. Система коментарів та відгуків. Реалізація можливості додавання коментарів та відгуків до товарів користувачами. Це дозволить покупцям обмінюватися враженнями про товари, а також збільшить впевненість інших користувачів у покупці.

8. Система знижок та промокодів. Впровадження системи знижок та промокодів для стимулювання покупок. Це може бути реалізовано шляхом автоматичного застосування знижок до товарів або можливості введення промокоду при оформленні замовлення.

9. Інтеграція з платіжними системами. Підключення різних платіжних систем для зручного та безпечного здійснення оплати замовлень. Це може включати підтримку кредитних карт, електронних гаманців та інших платіжних методів.

10. Статистика та аналітика. Додавання функціоналу для збору та аналізу статистичних даних про продажі, активність користувачів та інші важливі метрики. Це допоможе власникам магазину приймати обґрунтовані рішення щодо стратегії розвитку бізнесу.

Розробка дизайну для Інтернет-магазину, спеціалізованого на продажі годинників, відіграє важливу роль у привертанні та задоволенні потреб цільової аудиторії. Створення привабливого інтерфейсу, інтуїтивно зрозумілої навігації та адаптивного дизайну є ключовими аспектами, які сприяють приверненню та утриманню користувачів на сайті. Функціональність кошика та зручне оформлення замовлення роблять процес покупки максимально зручним. Окрім цього, важливо забезпечити безпеку та довіру користувачів за допомогою захисних протоколів та символів довіри[4].

Основною метою дизайну є створення приємного та зручного середовища для покупок, яке спонукає користувачів повертатися та залишатися задоволеними від взаємодії з магазином. Досвід користувача повинен бути максимально позитивним, щоб стимулювати покупки та побудувати довгострокові відносини з клієнтами.

Головна сторінка сайту повинна бути інформативною і привабливою, оскільки саме з неї починається взаємодія користувача з інтернет-магазином. На головній сторінці повинні бути розміщені основні категорії товарів, спеціальні пропозиції, новинки та популярні продукти. Важливо передбачити зручну навігаційну панель, що містить посилання на основні розділи сайту, такі як каталог товарів, сторінка про нас, контакти, допомога та кошик покупок.

Каталог товарів повинен бути структурованим за категоріями, підкатегоріями і фільтрами, що дозволяють швидко знаходити потрібні моделі годинників. Кожна категорія повинна мати власну сторінку з відповідними фільтрами, які дозволяють сортувати товари за ціною, брендом, типом механізму, матеріалом корпусу та іншими характеристиками. Сторінки товарів повинні містити детальні описи,

високоякісні зображення, інформацію про наявність, ціну, можливі знижки та відгуки покупців.

Важливим аспектом є адаптивний дизайн сайту, що забезпечує коректне відображення та функціонування на багатьох пристроях, таких як: персональні комп'ютери, планшети та смартфони. Це дозволяє користувачам зручно переглядати сайт та здійснювати покупки з будь-якого пристрою [5].

Наступним кроком буде аналіз конкретних програм для роботи з PHP-кодом у різних категоріях. Кожен інструмент, наведений у подальшому переліку, має свої переваги та можливості і є одним з лідерів у світовому ринку.

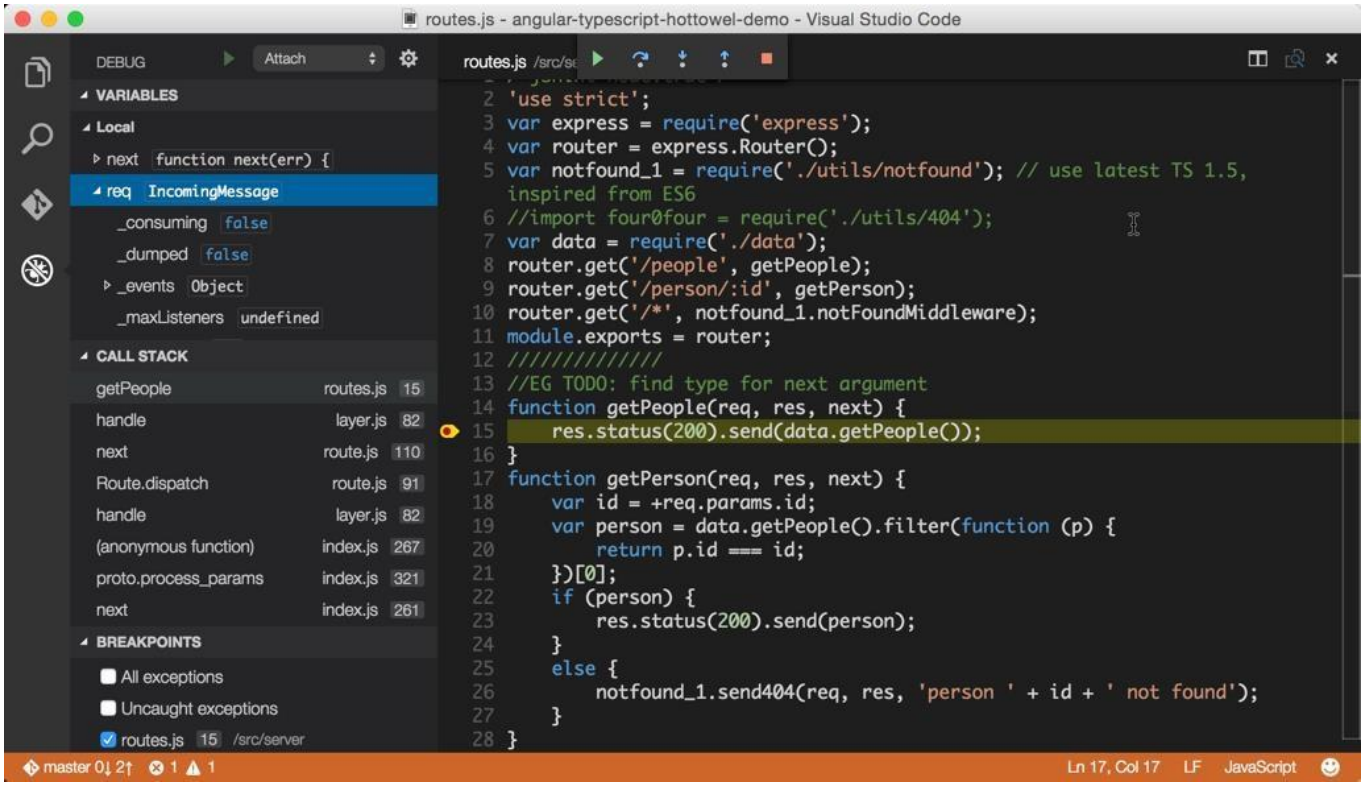

Рисунок 1.2 – Робочий вигляд Visual Studio Code

Згаданий текстовий редактор, відомий також як VS Code, є повністю безкоштовним із відкритим вихідним кодом, розробленим корпорацією Microsoft. Останнім часом це рішення набуває все більшої популярності як серед початківців, так і серед професійних розробників. Він володіє вбудованою підтримкою PHP-синтаксису та можливістю розширення завдяки VS Code Marketplace, що дозволяє адаптувати конфігурацію до потреб користувача [6].

Особливою перевагою є детальна налаштованість програми, включаючи відкритий доступ для зміни кольорової схеми та призначення кожної клавіші для конкретної функції. Це суттєво підвищує комфорт використання програмного забезпечення.

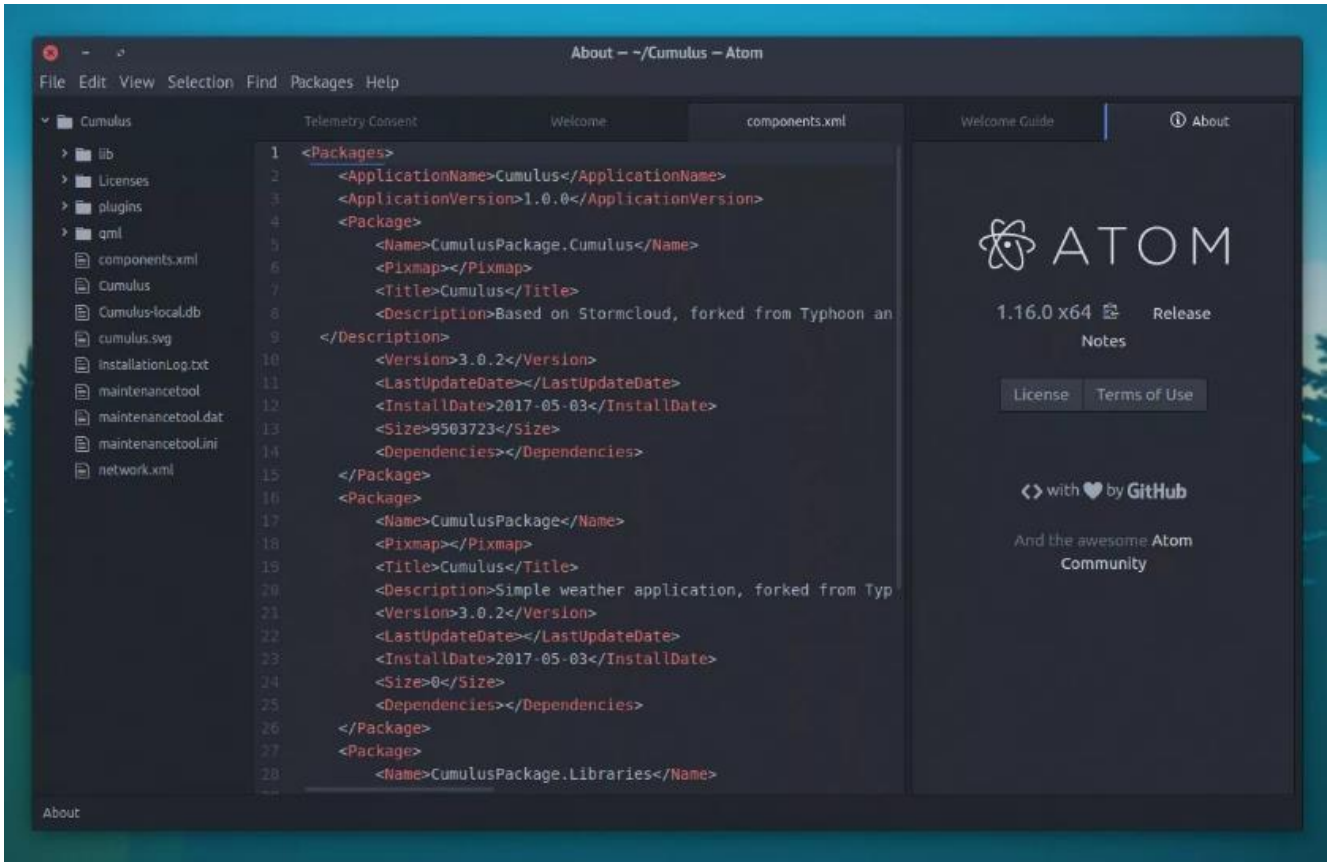

Рисунок 1.3 - Вигляд редактору ATOM

Atom є текстовим редактором для PHP, спочатку розробленим як внутрішній інструмент у системі GitHub, пізніше став відкритим проектом з відкритим вихідним кодом, проте залишаючись вірним своїм кореням. Його гасло - "зламайте редактор", що означає, що будь-який користувач може налаштувати його на свій смак, роблячи його ідеальним інструментом для розв'язання різноманітних завдань, пов'язаних з розробкою на мові програмування, такій як PHP.

Однією з істотних переваг Atom перед конкурентами є його широкі можливості розширення. Наразі існує понад дев'ять тисяч пакунків, які додають різні функціональні можливості до редактора. Серед них є і спеціалізовані

інструменти для налагодження, рефакторингу, анотацій, лінтингу та багато інших, що вирішують різноманітні завдання, пов'язані з розробкою програмного забезпечення [7].

Під час розробки вебсайту було використано ряд інших інструментів та технологій, зокрема Node.js. Це середовище виконання JavaScript, яке дозволяє запускати JavaScript-код на сервері. Node.js є ідеальним вибором для розробки веб та мережевих додатків завдяки своїм особливостям, таким як однопотоковість та неблокуючий ввод/вивід, що дозволяють обробляти багато запитів без блокування виконання [8].

Node.js відображає високу продуктивність та ефективне використання ресурсів через розподілену архітектуру та активне використання неблокуючих операцій вводу/виводу.

Завдяки своєму пакетному менеджеру npm (Node Package Manager), Node.js дозволяє легко керувати залежностями проектів та має величезну кількість розширень для різноманітних завдань.

Використання Node.js не обмежується лише створенням вебсерверів чи API. Воно включає в себе створення різноманітних додатків, таких як мережеві, стрімінгові додатки, чат-боти, та інструменти розробки [9].

Один із популярних фреймворків для розробки вебдодатків на Node.js - це Express.js. Він надає простий та елегантний спосіб для обробки маршрутів, створення обробників запитів та керування middleware.

Express.js підтримує мінімалістичний підхід до розробки та надає потужний механізм маршрутизації та middleware, що робить його популярним вибором для розробки серверної частини вебдодатків.

Додаток Sequelize - це ORM для роботи з базами даних, яке надає зручний спосіб взаємодії з базами даних у JavaScript-додатках, дозволяючи розробникам працювати з базами даних, використовуючи об'єктно-орієнтовані підходи.

Основні характеристики та можливості Sequelize:

1. Відображення об'єктів та таблиць. Sequelize дозволяє визначати моделі, які відображають об'єкти додатку та автоматично зв'язувати їх з таблицями бази даних.

Це спрощує взаємодію з даними, дозволяючи працювати з ними у вигляді об'єктів, а не прямо з SQL-запитами та таблицями.

2. Міграції бази даних. Sequelize надає механізм міграцій, який дозволяє легко керувати структурою бази даних. З його допомогою можна визначати міграції для створення, зміни або видалення таблиць, стовпців та інших елементів бази даних, що забезпечує синхронізацію структури бази даних з розвитком додатку.

3. Підтримка різних СУБД. Sequelize підтримує різні системи управління базами даних (СУБД), включаючи PostgreSQL, SQLite, MySQL та Microsoft SQL Server. Це дозволяє вам працювати з обраною вами СУБД, не змінюючи способу взаємодії з даними в вашому додатку[10].

Sequelize є потужним інструментом, який спрощує взаємодію з базами даних у JavaScript-додатках, надаючи розширені можливості для керування базою даних.

Основні характеристики та можливості PostgreSQL:

1. Розширюваність. PostgreSQL дозволяє додавати новий функціонал та розширення через використання розширень та власних функцій, що робить її гнучкою та розширюваною [24].

2. Розширені типи даних. PostgreSQL надає широкий набір вбудованих типів даних та дозволяє розробникам визначати власні типи даних та оператори.

3. Підтримка SQL. PostgreSQL повністю сумісний з мовою SQL та підтримує багато розширень та розширених функцій SQL, що дозволяє виконувати складні операції з даними.

PostgreSQL є потужною та розширеною системою управління базами даних, яка підходить для різних типів проектів та надає багатий функціонал та гнучкість для ефективної роботи з даними.

Bcrypt - це бібліотека для хешування паролів, яка забезпечує міцний криптографічний захист паролів шляхом використання солі та повільного обчислення хеш-значення. Цей алгоритм має високу опірність до атак перебору та рейнбов-таблиць, що робить його ідеальним для захисту паролів у системах.

JSON Web Token (JWT) - це стандарт для створення та передачі токенів авторизації, який використовується для аутентифікації користувачів і захисту

маршрутів у розподілених системах. Цей стандарт дозволяє безпечно передавати та перевіряти інформацію у форматі JSON між різними сторонами. JWT складається з трьох основних частин. заголовка, пейлоаду та підпису[11].

JWT використовується для підтвердження ідентифікації користувача та надання доступу до ресурсів після успішної аутентифікації. Після вдалої аутентифікації сервер створює JWT токен, який потім може бути переданий клієнту. Клієнт включає цей токен у заголовок або запит, щоб довести свою автентичність до сервера. Після цього сервер перевіряє підпис токена, розкодовує пейлоад та перевіряє, чи користувач має необхідні права доступу.

JWT є легким та мобільним механізмом для передачі даних, оскільки всі необхідні дані зберігаються у самому токені. Це робить його ідеальним для обміну даними між різними службами або мікросервісами, спрощуючи процес автентифікації та авторизації [12].

Технології, які використовуються у процесі розробки інтернет-магазинів, включають MongoDB, Express.js, React.js та Node.js. Ось коротка інформація про кожну з них:

1. MongoDB. Реляційна база даних, що використовується для зберігання даних про товари та користувачів.

2. Express.js. Фреймворк для створення вебдодатків на Node.js, який дозволяє легко обробляти запити та створювати API для взаємодії з базою даних.

3. React.js. Бібліотека JavaScript для створення користувацького інтерфейсу, яка дозволяє розробникам створювати інтерактивні елементи вебсторінок.

4. Node.js. Серверне середовище для виконання JavaScript, яке дозволяє розробникам створювати швидкі та ефективні вебдодатки.

Додатково, у процесі розробки використовуються такі інструменти, як dotenv для зберігання конфіденційних даних, CORS для настроювання політики обміну ресурсами між різними доменами, Express-fileupload для завантаження файлів на сервер, Nodemon для автоматичного перезавантаження сервера та Postman для тестування та налагодження API. Також використовується система контролю версій Git та платформа GitHub для зберігання коду проекту та спільної розробки.

Інтегроване середовище розробки Visual Studio Code використовується для написання коду, налагодження та керування проектом.

Express.js є програмним каркасом, спрямованим на полегшення розробки серверної частини вебзастосунків для Node.js. Він надає різноманітні готові абстракції, які спрощують процес розробки сервера та серверної логіки, такі як обробка форм, робота з куками та CORS (Cross-Origin Resource Sharing).

Node.js - це платформа, завдяки якій виконуються JavaScript-скрипти на сервері та відправляються користувачеві результати їх роботи. Розробка в ній відбувається мовою програмування JavaScript, це дозволяє як розробляти високопродуктивні мережеві застосунки, так і створювати клієнтські і серверні програми [13].

Node.js використовує рушій V8, розроблений компанією Google, та імплементує принципи асинхронного програмування, що дозволяє уникнути блокування потоків та забезпечити високу продуктивність вебзастосунків.

1.3. Моделі бізнес-процесів

У цьому розділі детально розглядаються моделі процесів веборієнтованої інформаційної системи у вигляді інтернет-магазину годинників. Описані процеси охоплюють ключові аспекти функціонування магазину, починаючи від реєстрації користувачів і додавання товарів до каталогу, до пошуку, вибору та покупки товарів. Кожен бізнес-процес представлений у вигляді послідовних дій, які забезпечують ефективну та безперебійну роботу інтернет-магазину.

Рисунок 1.4 – Базова структура сайту

Далі було розглянуто взаємодія нашого сайту з сервером та базою даних. Для цього були обрані наступні процеси:

1. Відображення товарів на сторінках сайту та надання детальної інформації про них.

2. Додавання товарів у кошик користувачем.

3. Оформлення замовлення.

4. Оплата замовлення.

Детальний опис процесів 1 та 2 наведено на рисунку 1.5.

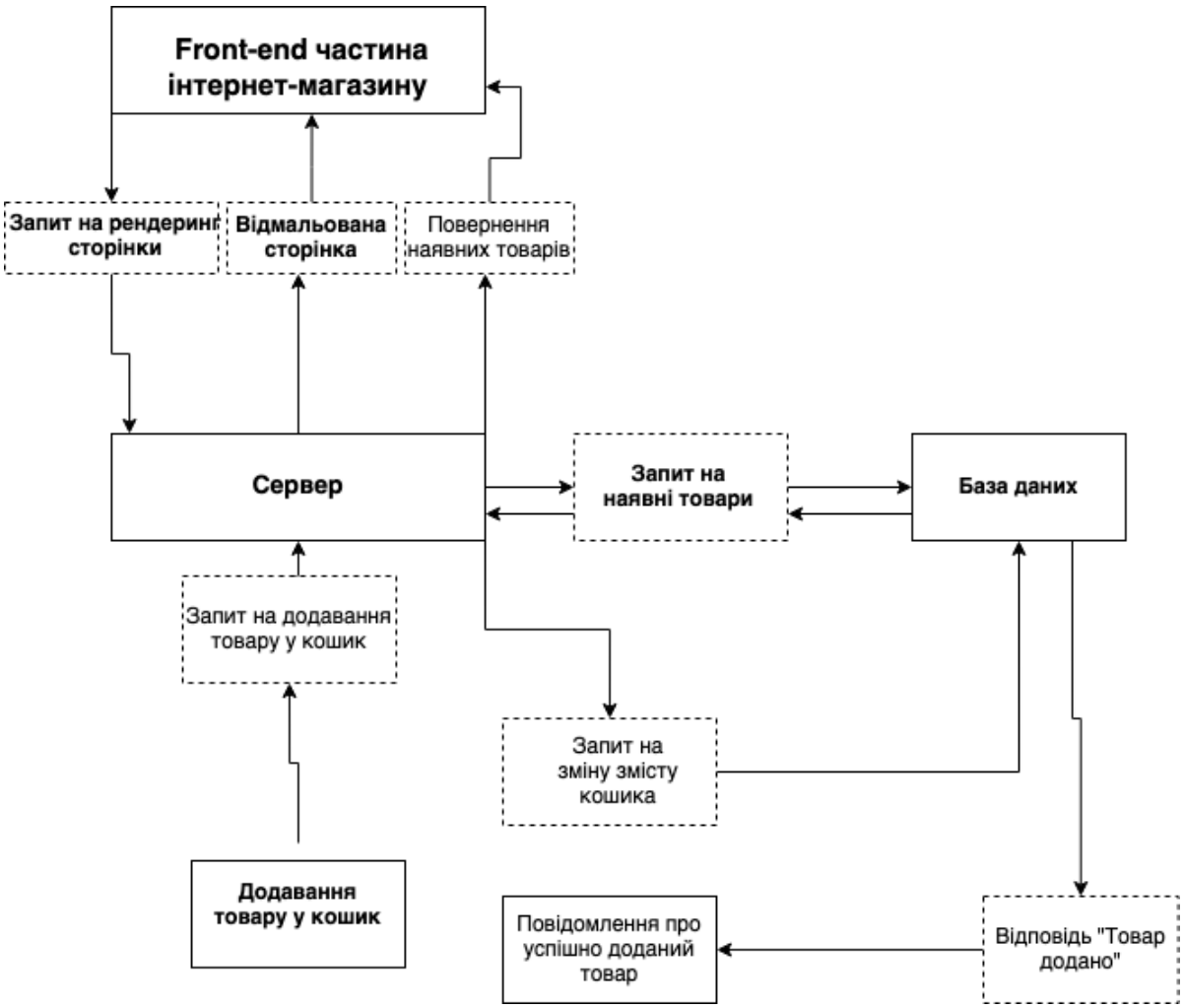

Рисунок 1.5 – Деталізація процесів «Поява товарів на сторінках сайту та деталей про них» та «Додавання у кошик користувачем»

Детальний опис процесів 3 та 4 наведено на рисунку 1.6.

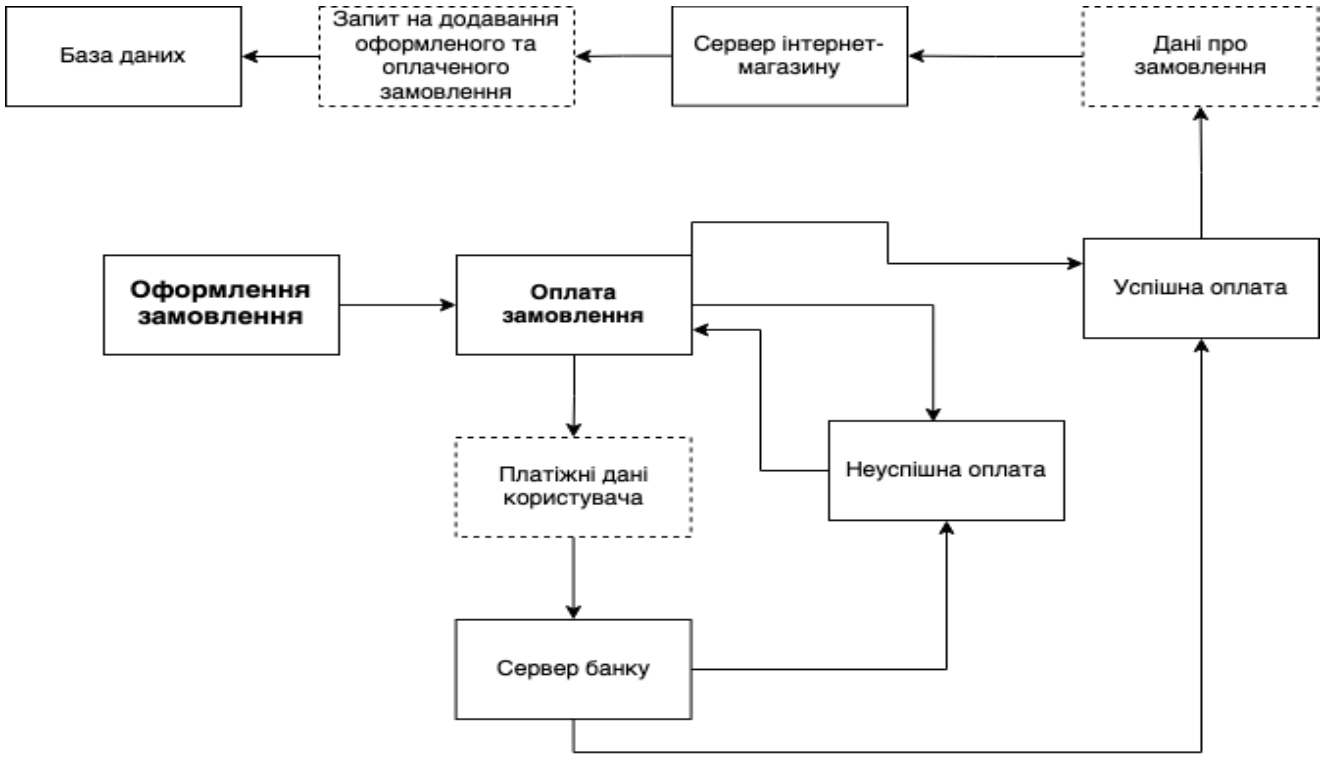

Рисунок 1.6 – Деталізація процесів «Оформлення замовлення» та «Оплата замовлення»

Інформація процесів вказує на те, що наш вебсайт буде базуватися на класичній архітектурі клієнт-сервер RESTful, де клієнт направляє запит на сервер, а сервер обробляє його та повертає відповідь клієнту для відображення даних. Це все буде працювати асинхронно, що означає, що клієнт не буде чекати на повне завантаження, а замість цього буде рендерити головну сторінку і поступово заповнювати її товарами

З урахуванням розробленої та ретельно продуманої структури сайту, перейдемо до реалізації нашого інтернет-магазину. Діаграма варіантів використання ілюструє взаємодію покупця з системою. Відповідно до висунутих вимог, було розглянуто діаграму варіантів використання (рис. 1.7).

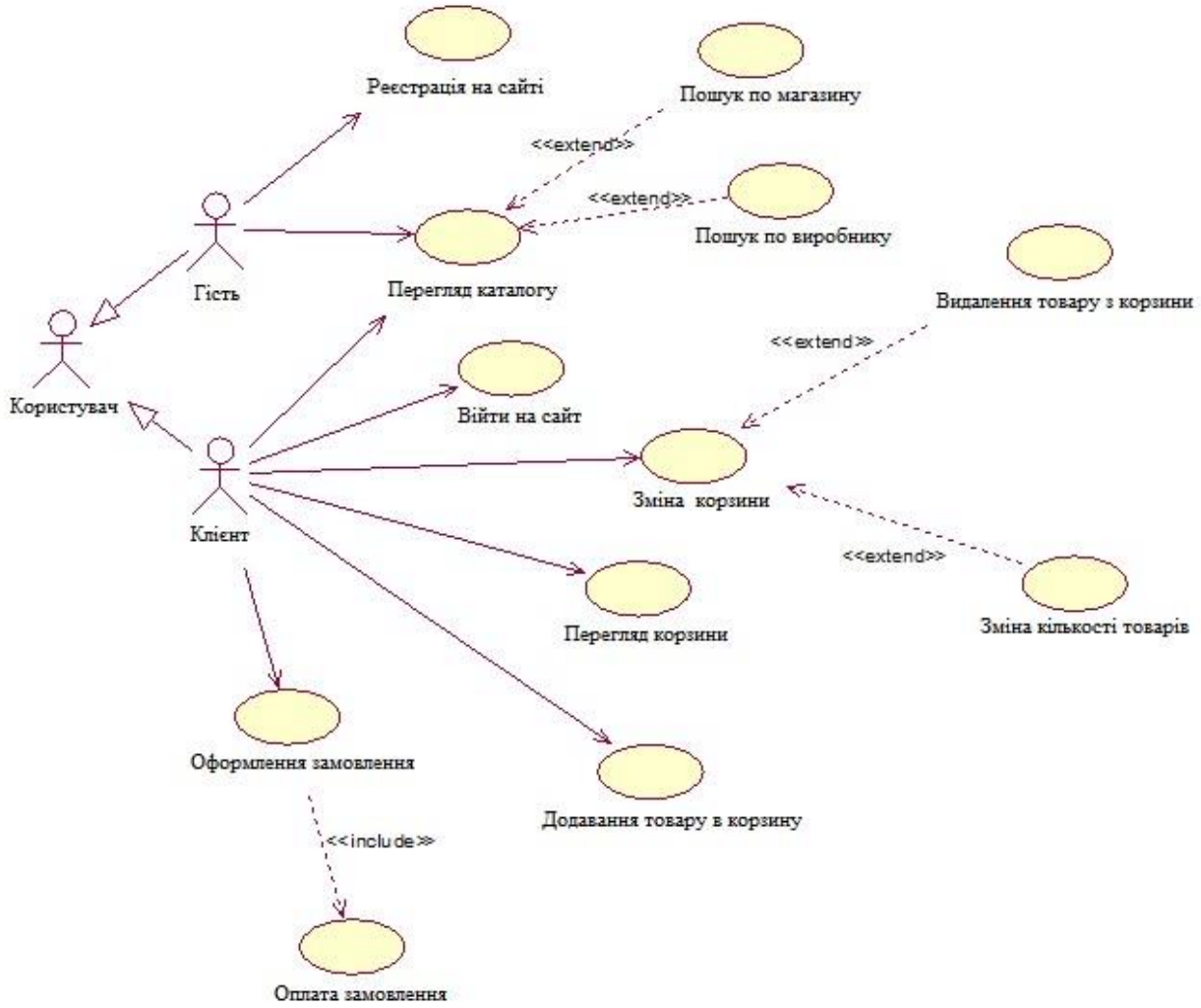

Рисунок 1.7 – Діаграма варіантів використання

Система сайту організована за модульним принципом. Спостереження чітко показали, що коли всі елементи, що мають спільні характеристики, групуються разом, це сприяє привабливості вебсайту. Тому на рисунку 1.8 наведено модульну структуру сайту.

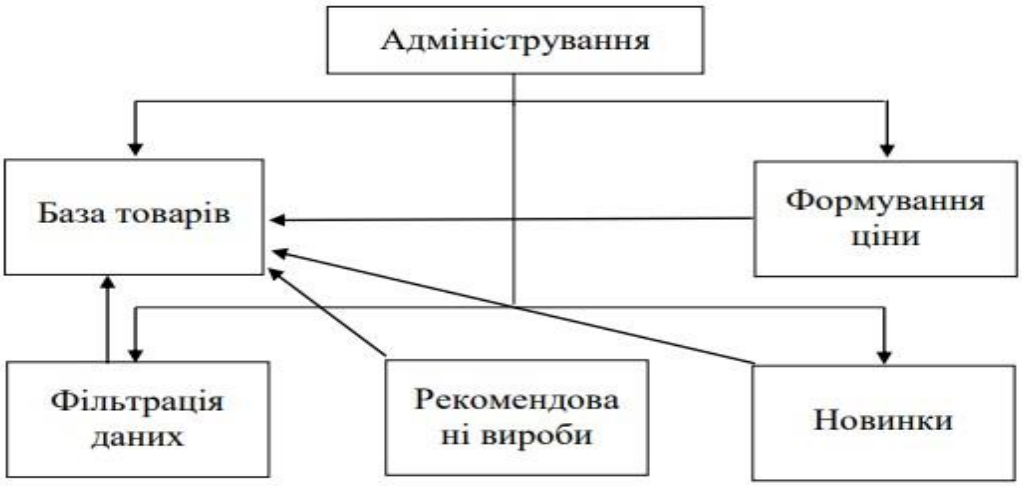

Рисунок 1.8 - Структура модулів сайту

Все вищесказане відображає процес розробки та реалізації інтернет-магазину, починаючи з проектування його інтерфейсу та функціоналу, а завершуючи детальним аналізом архітектури сайту і його взаємодії з сервером та базою даних.

Загальна мета – створення привабливого, функціонального та безпечного середовища для покупок онлайн, яке задовольнятиме потреби користувачів та забезпечить їхнє позитивне враження від взаємодії з магазином.

Моделювання роботи – це процес створення абстрактної моделі, яка відображає суб'єктивне сприйняття потоку робіт через систему взаємопов'язаних операцій. У контексті моделювання інтернет-магазину цей процес включав розробку моделі функціонування на основі стандарту IDEF0 та моделі взаємодії з сайтом з використанням UML.

При розгляді на рівні A0, процес представлений як функціональний блок з усіма відповідними керуючими та робочими об'єктами. Ця діаграма також включає всю потрібну інформацію в процесі виконання замовлення товару (рис. 1.9).

Рисунок 1.9 – Контекстна модель

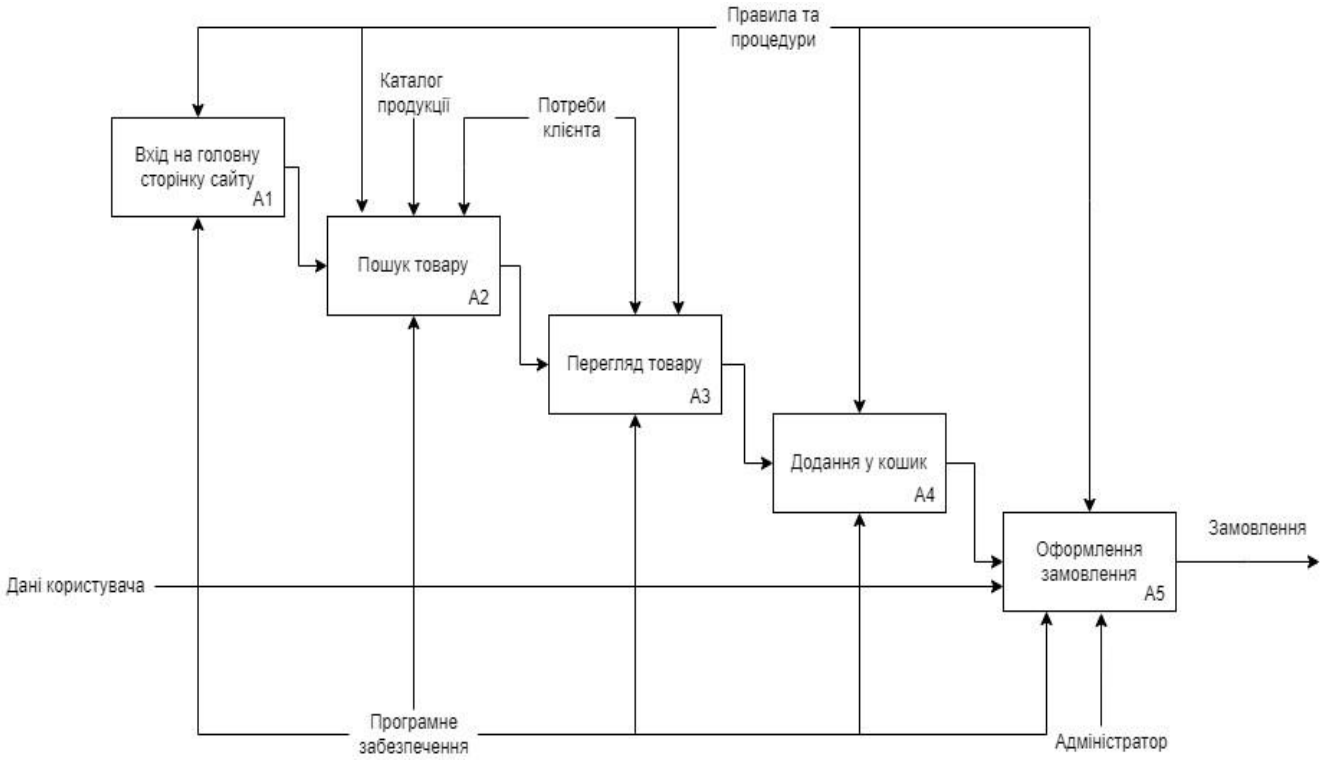

Рисунок 1.10 – Модель декомпозиції

Діаграма першого рівня розгортає, деталізує функцію обробки на нульовому рівні та декомпонує функціональний блок A0 на набір взаємопов'язаних підфункцій. У даному проекті варіант діаграми зображений на рисунку 1.10. Після аналізу функціонування веб-додатку було ідентифіковано акторів у табличці 1.1 та описано різноманітні сценарії використання у табличці 1.2.

Таблиця 1.1 – Опис акторів

Назва	Опис
Актори-користувачі	
Користувач	Користувач інтернет-магазину.
Адміністратор	Адміністратор інтернет-магазину.
Актори-зовнішні системи	
База даних	База даних, що зберігає інформацію.

Таблиця 1.2 – Опис варіантів використання

Назва	Опис
Редагування даних	Адміністратор має можливість редагувати дані на сайті
Додавання товару	Адміністратор має можливість додавати товари на сайт
Перегляд замовлень	Адміністратор має можливість переглянути всі замовлення
Підтвердження замовлення	Адміністратор повинен підтвердити замовлення
Замовлення товару	Користувач має можливість замовити товар, який йому сподобався
Оформлення замовлення	Заявка замовлення товару

Продовження таблиці 1.2

Оплата	Надана можливість вибору способу оплати
Використання плагіна для оплати криптовалютою	Під час оплати використовується інструмент для оплати криптовалютою
Перегляд товару	Користувач має можливість переглянути товари на сайті
Пошук товару	Функція, що надає можливість пошуку товарів на сайті.

Кожен бізнес-процес представлений у вигляді послідовних дій, які забезпечують ефективну та безперебійну роботу інтернет-магазину. Це дозволяє створити зручний і функціональний ресурс як для користувачів, так і для адміністраторів, забезпечуючи високу якість обслуговування, легкість у використанні та ефективність управління всіма аспектами діяльності магазину. У подальшому розділі буде докладно описано кожен з цих процесів.

1.4. Архітектура інформаційної системи та технології розв'язання поставлених задач

Архітектура веборієнтованої інформаційної системи у вигляді інтернет-магазину годинників базується на клієнт-серверній моделі. Вона включає кілька ключових компонентів: клієнтський інтерфейс, сервер додатків, базу даних та інтеграційні сервіси.

Тип архітектури та моделі веб-додатків є одним із найскладніших та найважливіших завдань у їхній розробці. Архітектура веб-додатків представляє собою схему взаємодії між їх компонентами. Спосіб, яким взаємодіють ці компоненти, визначає майбутню ефективність, стійкість та безпеку веб-додатка [13].

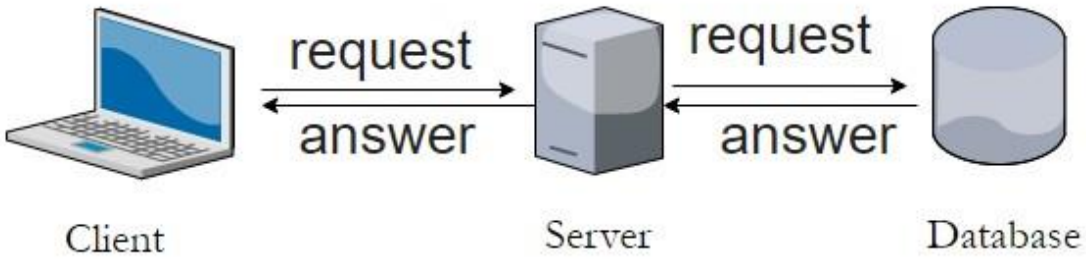

Рисунок 1.11 – Клієнт-серверна архітектура інтернет-магазину

В клієнт-серверній архітектурі відносини між комп'ютерами поділяються на дві ролі:

– клієнт, з конкретними запитами послуг або ресурсів;

– сервер, основним завданням якого є виконання запитів, відповідаючи на запитувані послуги чи ресурси.

Головною особливістю цього підходу є те, що сам вебдодаток розташовується на сервері і вбудь’яка взаємодія з ним відбувається з використанням серверу програми. Клієнт, в свою чергу, отримує лише результати своєї роботи. Працює це таким чином:

1) користувач надсилає запит через Інтеренет.
2) сервер отримує його.
3) сервер обробляє його.
4) сервер надсилає його назад користувачу.
5) користувач отримує, лише результат роботи серверу.

Головною перевагою даного методу є те, що не має різниці в оперційній системі конкретного користувача, що робить вебдодатки кроссплатформенними.

Певна програма, як от браузери, зазвичай відповідають за відображення результатів запитів клієнтів та обробку даних, які він отримує та відправляє на сервер. Однією з ключових функцій браузера є інтерпретація даних, отриманих з Інтернету, у форматі веб-сторінки, яка написана мовою HTML.

Браузер та сервер обробляють декілька різних видів кодів. В свою чергу вибір мови програмування має обґрунтовуватись взаємодією цих кодів. Розробники які

працюють в цій галузі, користуються наступними правилами при виборі мови на стороні сервера:

– зберігає сталі дані такі як: профілі користувачів, сторінки, тощо;

– користувач не має можливості напряму взаємодіяти з сервером;

– має можливість відправляти лише конкретні HTTP запити для певної URL-адреси, а не будь-який ввід від користувача;

Клієнтський інтерфейс. цей компонент відповідає за взаємодію з користувачем. Він реалізується за допомогою вебтехнологій HTML, CSS, та сучасних фронтенд-фреймворків, таких як React або Angular. Інтерфейс забезпечує зручну навігацію, пошук, фільтрацію товарів, управління кошиком та оформлення замовлень. Адаптивний дизайн забезпечує коректне відображення на різних пристроях, включаючи смартфони та планшети.

Сервер додатків. цей компонент обробляє запити від клієнтського інтерфейсу, виконує бізнес-логіку та взаємодіє з базою даних. Сервер додатків може бути реалізований за допомогою технологій, таких як Node.js, Python (Django, Flask). Він забезпечує обробку реєстрації та авторизації користувачів, управління товарами, обробку замовлень, а також відправлення повідомлень користувачам.

База даних. для зберігання інформації про користувачів, товари, замовлення та інші дані використовується реляційна база даних, наприклад, MySQL, PostgreSQL або Oracle. База даних забезпечує надійне та швидке зберігання даних, підтримує складні запити та транзакції.

Таким чином, архітектура веборієнтованої інформаційної системи інтернет-магазину годинників є комплексною та забезпечує надійність, масштабованість і безпеку, дозволяючи ефективно розв'язувати поставлені задачі. Клієнт-серверна модель забезпечує чіткий поділ відповідальності між інтерфейсом користувача та серверною логікою, що спрощує розробку, тестування та підтримку системи. Використання сучасних технологій та інструментів для фронтенду, бекенду, баз даних, інтеграції, кешування, моніторингу та логування дозволяє створити

високопродуктивний і зручний інтернет-магазин, який задовольняє потреби як користувачів, так і адміністраторів.

PHP є однією з найбільш популярних мов програмування у сфері веброзробки. Ця мова використовується для створення різноманітних вебсайтів та вебдодатків будь-якої складності, починаючи від простих лендингів та блогів і закінчуючи складними інтернет-магазинами та браузерними іграми.

Однією з головних переваг PHP є його широкі можливості та висока захищеність завдяки закритому вихідному коду. Щодо недоліків, раніше однією з них було те, що код не працював без наявності сервера з інтерпретатором, але ця проблема була виправлена у нових версіях мови [25].

Для початку роботи з PHP створюється сторінка з розширенням .php або .html, залежно від конфігурації сервера. Вихідний код цієї сторінки містить теги розмітки HTML та вбудовані PHP-команди, які укладаються між операторами.

PHP надає можливості для використання функцій, проведення математичних обчислень, роботи зі змінними та масивами, умовних конструкцій, створення об'єктів тощо. Частіше за все PHP використовується для:

– відправлення форм;
– роботи з базами даних;
– створення динамічних сторінок;
– використання сесій та cookies;
– завантаження та обробки файлів;
– створення зображень;

Наприклад, для отримання даних, введених користувачем у форму, потрібно використовувати PHP, оскільки HTML сам по собі не здатний обробити ці дані. З форми, наприклад, на сторінку auth.php будуть передані змінні login та password, які потраплять у супермасив POST або GET. На сторінці auth.php можна буде провести з ними різноманітні операції, включаючи перевірку наявності цих даних у базі даних.

Підключення бази даних до вебсайту на PHP значно спрощує процес розробки. Інформація зберігається у таблицях бази даних, що дозволяє ефективно

редагувати та додавати дані без втручання в основний код. Наприклад, інформація про користувачів може бути збережена в базі даних замість занесення її безпосередньо в код, що сприяє підвищенню швидкості обробки та зменшенню завантаження сервера [16].

Використання PHP дозволяє створити лише один файл, який керує різними частинами вебсайту, такими як головна сторінка, каталог товарів і т. д. Для цього можна використовувати метод GET.

PHP є мовою програмування, яка дозволяє реалізувати різноманітний функціонал. Навіть якщо деякі розробники не виявляють до неї великої симпатії, вона все одно відмінно справляється із своїми завданнями. Ігнорувати PHP просто через загальний настрій не раціонально - досить просто адаптуватися до кількох його недоліків, щоб скористатися його численними перевагами.

Щоб ефективно працювати з високоякісним PHP-кодом, важливо обрати для себе редактор PHP, який відповідає вашим потребам та особистим уподобанням. Наразі є безліч варіантів, але кожен з них має свої унікальні особливості та призначення. Далі розглянуто найпопулярніші PHP-редактори та середовища розробки, які не лише дозволять зручно редагувати код, але й нададуть можливість тестувати, налагоджувати та розробляти за допомогою них з нуля.

Зараз розробники PHP активно використовують два основних типи інструментів для написання коду - редактори та інтегровані середовища розробки (Integrated Development Environments, IDE). Основна відмінність між ними полягає в складності та функціональності.

Редактори, по суті, є спрощеними інструментами, призначеними для редагування коду без надмірної складності. Вони забезпечують необхідний функціонал для роботи з кодом невисокого рівня складності, мають зрозумілий інтерфейс та швидкий запуск, що робить їх ідеальними для початківців.

З іншого боку, IDE є більш розширеними інструментами, що надають розширений набір функцій для досвідчених програмістів. Вони забезпечують більше можливостей для роботи з кодом, такі як автодоповнення, вбудовані відладчики та системи керування версіями [31, с. 67].

Технічно, можливе редагування PHP-файлів навіть за допомогою звичайного текстового редактора, такого як Блокнот у Windows. Проте це може бути складним завданням через відсутність підсвічування синтаксису, що робить код менш зрозумілим та збільшує кількість помилок. Тому редактори програмного коду, які розпізнають синтаксис PHP та надають підсвічування синтаксису, стають необхідними для зручної роботи.

Підсвічування синтаксису є ключовою функцією будь-якого редактора PHP, але кращі з них мають додаткові функції, такі як автоматичне доповнення коду та вбудовані інструменти для відладки, що полегшує написання та редагування коду.

Важливо також зазначити, що багато редакторів PHP працюють не лише з PHP-файлами, а й з іншими типами файлів, такими як HTML. Це робить їх універсальними інструментами для розробки вебдодатків, які потребують роботи з різними мовами програмування.

IDE, або Integrated Development Environment, - це програмні засоби, які об'єднують у собі редактори коду разом з іншими інструментами для розробки програмного забезпечення. Як уже зазначалося, ці інструменти мають ті ж базові можливості, що й редактори коду, але вони надають користувачам більший набір функцій та можливостей [24 ,с. 87].

Кращі IDE дозволяють розробникам використовувати шаблони коду, налаштовувати інтерфейс для оптимальної роботи та надають безліч вбудованих інструментів. Наприклад, вони можуть включати інтегрований термінал, вебінтерфейси, що завантажуються безпосередньо з IDE, та інші корисні інструменти.

Також окремо слід зазначити хмарні сервіси, тому що вони дозволяють створювати розширене середовище для програмування прямо у веббраузері та зберігати файли на відокремленому сервері.

Важливо зауважити, що, крім великої кількості функцій, такі IDE вимагають значних навичок та досвіду для коректного використання. Якщо вам не потрібні всі додаткові можливості IDE, можливо, варто скористатися простішими редакторами.

Однак, для великих та складних проектів IDE може стати незамінним інструментом.

Варто відзначити, що з кожним роком різниця між вебредакторами та повнофункціональними IDE стає все менш помітною, адже редактори додають все більше і більше функціональності, раніше доступної лише у складних IDE.

Laravel – популярний фреймворк для PHP, який надає розробникам інструменти для швидкої та ефективної розробки вебдодатків. Він базується на архітектурі Model-View-Controller (MVC), що розділяє логіку додатка на моделі, види та контролери, забезпечуючи кращу організацію коду.

Основні можливості Laravel включають:

– Маршрутизація: Проста та гнучка система визначення маршрутів для вебзапитів.

– Eloquent ORM: Інтуїтивний об'єктно-реляційний мапер для роботи з базою даних через PHP-об'єкти.

– Blade шаблонізатор: Потужний інструмент для створення динамічних вебсторінок з простим синтаксисом.

– Міграції та схеми баз даних: Інструмент для управління структурою бази даних за допомогою PHP-коду.

Переваги Laravel включають високу продуктивність розробки завдяки вбудованим функціям, активну спільноту, що забезпечує підтримку та ресурси, а також вбудовані механізми безпеки, такі як захист від CSRF, SQL-ін'єкцій та хешування паролів.

Laravel використовується для розробки різноманітних вебдодатків, включаючи інтернет-магазини, системи управління контентом (CMS) та соціальні мережі, завдяки своїй гнучкості та можливостям для масштабування.

Laravel вирізняється серед інших PHP-фреймворків своєю простотою використання та багатим набором вбудованих функцій. Інтуїтивно зрозумілий синтаксис і потужні інструменти, такі як Eloquent ORM, Blade шаблонізатор, міграції баз даних та вбудована аутентифікація, роблять його привабливим для розробників різного рівня. Laravel також має одну з найбільших та найактивніших

спільнот, що забезпечує доступ до великої кількості ресурсів і підтримки, а також пропонує потужні інструменти для роботи з фронтендом через Laravel Mix.

Крім того, Laravel забезпечує високу продуктивність та масштабованість завдяки використанню кешування, черг та інших оптимізацій. Вбудовані засоби для тестування, такі як PHPUnit і Laravel Dusk, спрощують процес забезпечення якості додатка. Модульна архітектура Laravel дозволяє легко додавати або змінювати функціональність додатків, що робить його гнучким та розширюваним фреймворком. Ці переваги роблять Laravel ідеальним вибором для створення сучасних, безпечних та масштабованих вебдодатків.

2 РЕАЛІЗАЦІЯ ПРОТОТИПУ ВЕБОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ МАГАЗИНУ ГОДИННИКІВ

2.1. Структура та особливості реалізації інформаційного забезпечення

Процес розробки головної сторінки інтернет-магазину, спеціалізованого на продажу годинників, включає низку етапів та функціональних елементів, спрямованих на привернення та зацікавлення відвідувачів сайту. Першим етапом є проектування інтерфейсу, що передбачає врахування тематики магазину та використання привабливих кольорів, шрифтів та графічних елементів для створення атмосфери, що відображає концепцію бізнесу. Ключові елементи, такі як логотип, пошукова панель, категорії товарів та рекомендації, розташовуються з урахуванням зручності та естетики.

Другий етап передбачає використання динамічних даних, для чого застосовується Laravel Eloquent ORM для отримання даних з бази даних. Наприклад, можна відображати нові надходження, популярні товари, акції або рекомендації на головній сторінці, щоб зацікавити відвідувачів та стимулювати їх до покупок.

Третій етап передбачає розміщення категорій товарів і фільтрів на головній сторінці, щоб користувачі могли швидко знаходити потрібні товари. Це сприяє підвищенню зручності використання та покращує користувацький досвід.

Описана система дозволяє приступити безпосередньо до розробки дизайну інтернет-магазину, завдяки встановленій структурі та ретельній деталізації процесів. Важливо, щоб дизайн відповідав сучасним тенденціям UI/UX, привертав увагу користувача своїм зовнішнім виглядом, мав інтуїтивний інтерфейс та не навантажував його зайвою інформацією. Наукові дослідження та аналіз інформації допомагають у створенні ефективних інтерфейсів, що сприяють позитивному користувацькому досвіду та збільшенню конверсії [41].

На рисунку 2.1 зображено приклад головної сторінки інтернет-магазину, який ілюструє зазначені елементи та принципи їх розміщення.

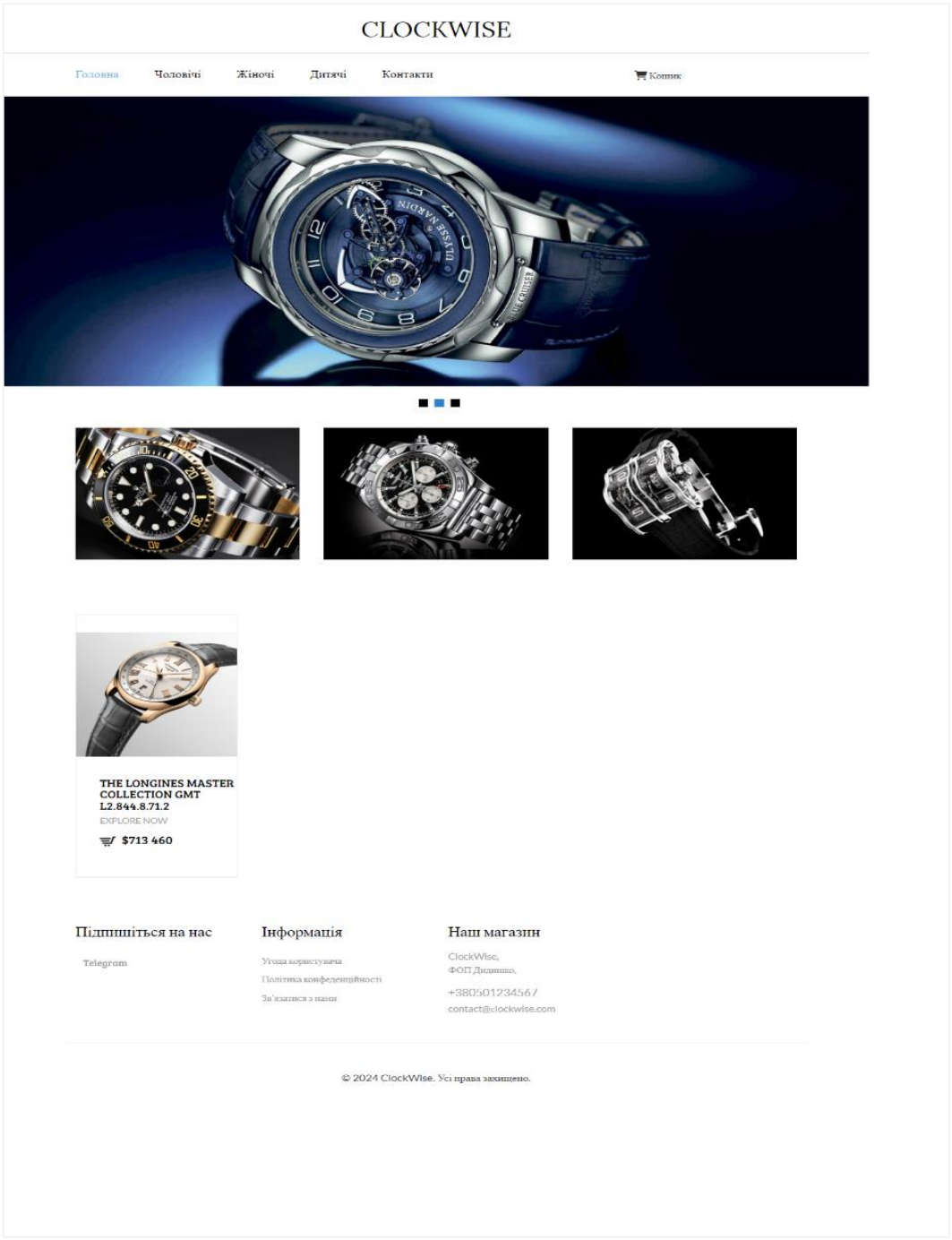

Рисунок 2.1 – Головна сторінка сайту

Головна сторінка інтернет-магазину була розроблена з використанням передових технологій, таких як Laravel. Цей сучасний дизайн дозволяє створити зручний та привабливий інтерфейс для користувачів.

Для подальшого вдосконалення роботи інтернет-магазину важливо розробити ефективну базу даних, яка буде складовою фундаменту цього сервісу. Проектування бази даних має включати в себе аналіз потреб користувачів,

структуру даних і їх зв'язки, а також механізми забезпечення безпеки та надійності інформації.

Після успішної розробки головної сторінки і впровадження бази даних інтернет-магазину, наступним кроком буде розширення функціональності та оптимізація роботи сайту. Це може включати в себе розробку додаткових сторінок для товарів, удосконалення системи пошуку та фільтрації, впровадження зручної системи оплати та доставки [32].

Крім того, важливо продовжити вдосконалення інтерфейсу користувача з урахуванням сучасних тенденцій у дизайні та вимог ергономіки. Розробка адаптивності сайту для різних пристроїв та платформ також може покращити користувацький досвід.

Необхідно також звернути увагу на безпеку даних та захист персональної інформації користувачів. Впровадження заходів з шифрування та забезпечення конфіденційності даних є важливою складовою успішної роботи інтернет-магазину.

Постійне вдосконалення та розвиток інтернет-магазину є ключовими факторами для забезпечення конкурентоспроможності та задоволення потреб користувачів.

Вебдизайн – це процес створення та розробки веб-сайтів, який об'єднує в собі елементи графічного дизайну, інтерфейсу користувача та програмування для створення ефективної та зручної веб-присутності. Він включає в себе вибір кольорової палітри, шрифтів, композиції сторінок, а також розташування та взаємодію елементів на веб-сторінках. Вебдизайн спрямований на створення привабливого та функціонального веб-сайту, який не лише відображає інформацію, але й забезпечує зручність користування та позитивне враження від взаємодії з ним.

На цьому етапі встановлюються загальні вимоги до дизайну вебінтерфейсу.

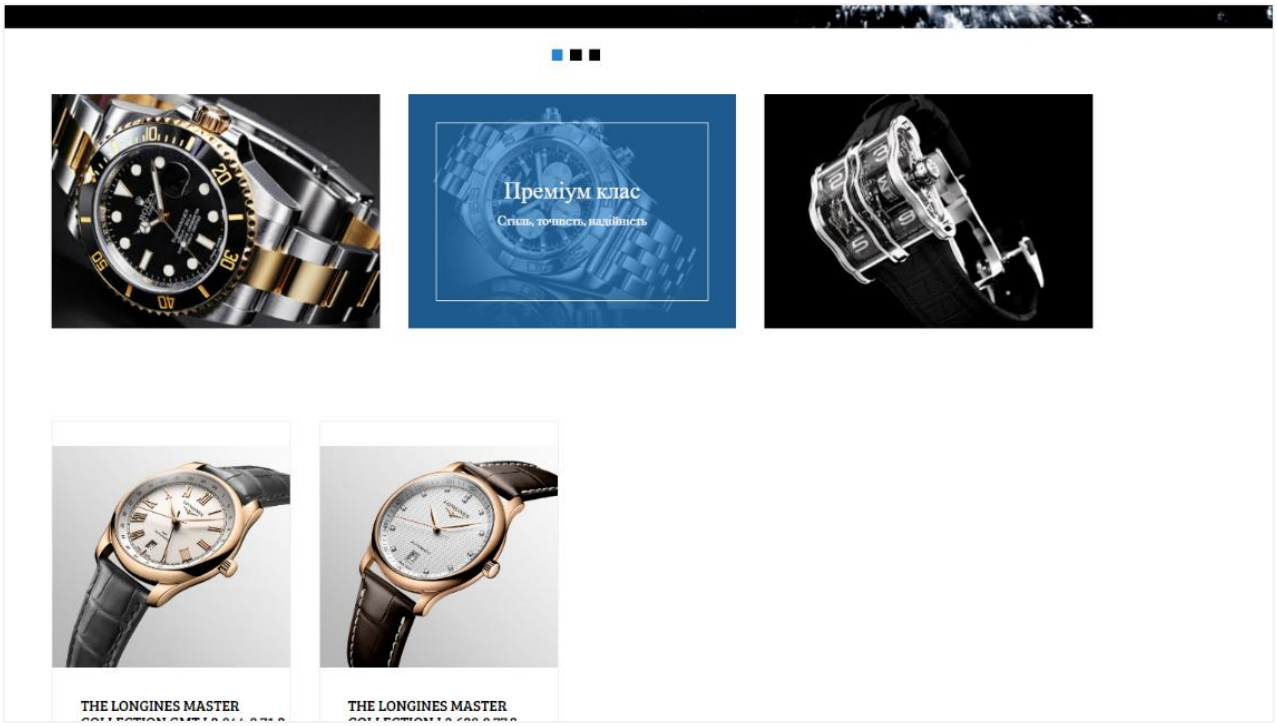

Рисунок 2.2 – Елементи дизайну інтерфейсу категорій (Преміум клас)

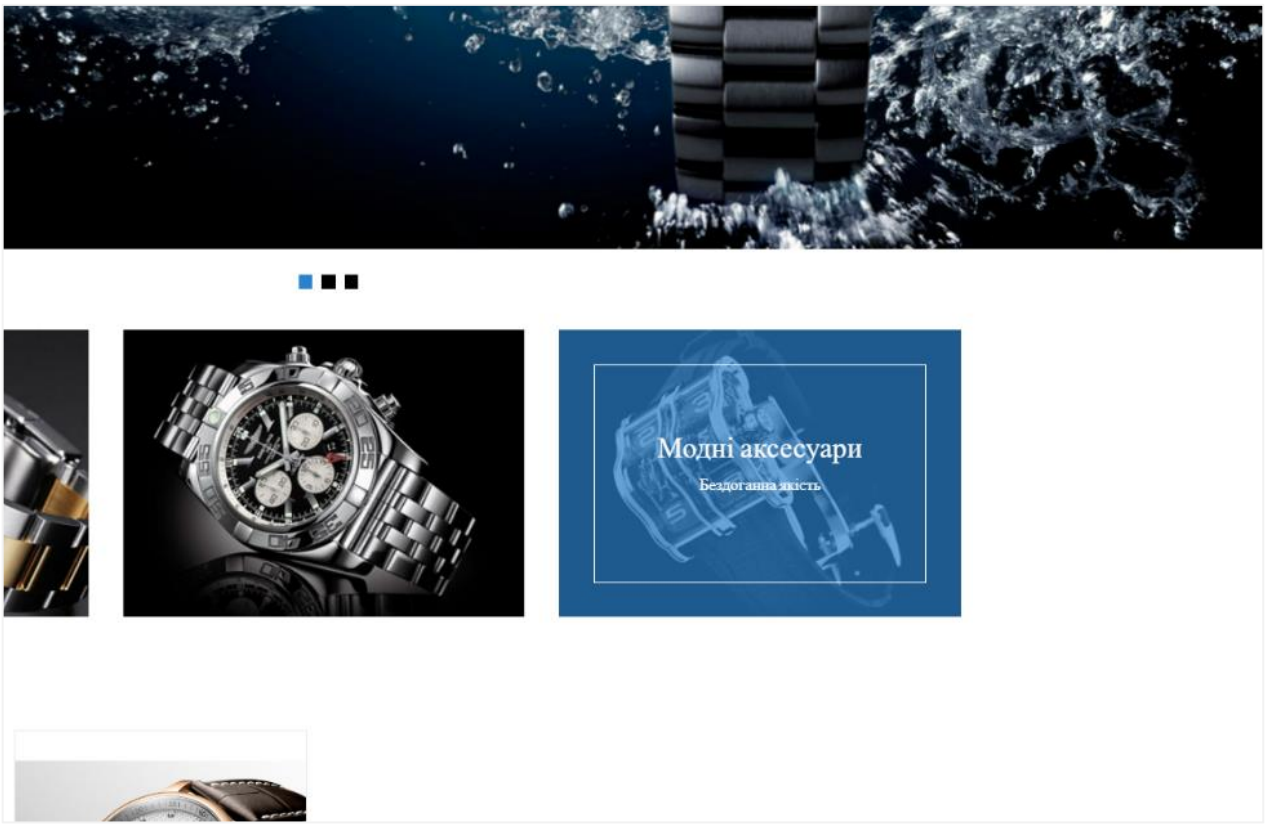

Рисунок 2.3 – Елементи дизайну інтерфейсу категорій (Модні аксесуари)

Карта сайту представляє собою візуальну структуру вебсистеми, що відображає всі розділи, підрозділи та сторінки, впорядковані у формі дерева. Цей інструмент використовуються для створення інформаційної архітектури продукту.

Схема навігації є іншим підходом до відображення структури сайту. Вона також включає розділи та сторінки, але вони групуються за різними меню.

Призначення карти сайту та схеми навігації включає наступне [24]:

– створення зрозумілої та зручної для користувача інформаційної структури;

– визначення шляхів навігації по сайту з мінімізацією кількості кроків, необхідних для досягнення бажаної сторінки.

Діаграми переходів між сторінками представляють собою схеми, які показують процес взаємодії користувача з функціональністю системи. Ці діаграми зосереджуються на конкретному модулі системи та показують послідовність дій користувача при переході від однієї сторінки до іншої. Вони допомагають зрозуміти, як користувач виконує свої завдання та взаємодіє з функціоналом системи.

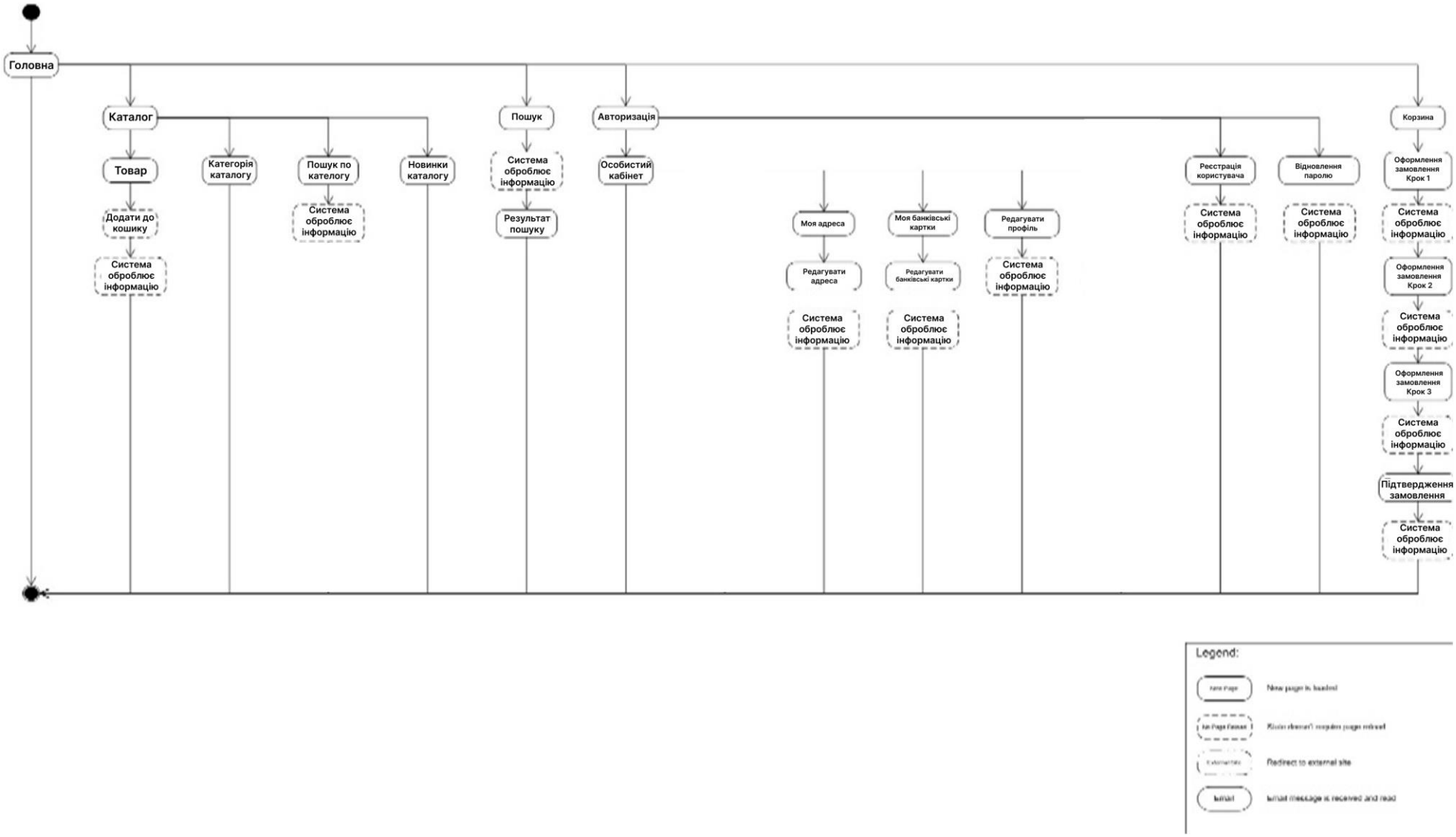

Рисунок 2.4 – Приклад діаграми переходів між сторінками

На наступному етапі процесу веброзробки проводиться створення структури БД. Для цього необхідно запустити скрипт, котрий встановить усі міграції (це механізм для управління базою даних, який дозволяє змінювати структуру бази даних програмно. Вони дозволяють створювати, змінювати та видаляти таблиці та індекси, а також робити інші операції з базою даних за допомогою PHP-коду.)

Після завершення процесу нам необхідно запустити тестовий веб сервер та ініціалізувати npm, який у свою чергу запустить усі скрипти необхідні для роботи сайту з боку NodeJS.

Панель навігації сайту містить наступні кнопки, які перенаправляють користувача на відповідні сторінки. "Головна", "Чоловічі", "Жіночі", "Дитячі", "Контакти" (рис. 2.5).

Рисунок 2.5 – Панель навігації інтернет-магазину у шапці

"Кошик" в інтернет-магазині служить для збереження товарів, які були вибрані користувачем. Ця сторінка відображає загальну вартість товарів у замовленні та дозволяє вказати кількість кожного товару. Після цього користувач може натиснути кнопку "Оформити замовлення", яка перенаправить його на форму для оформлення замовлення (рис. 2.6).

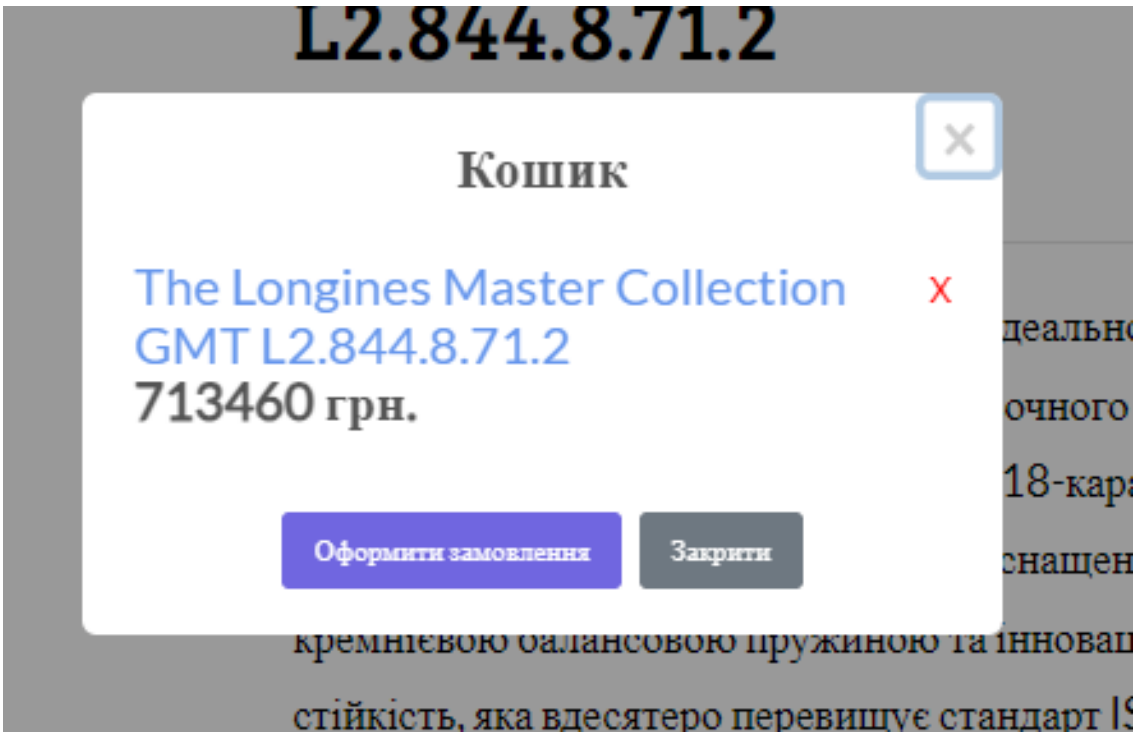

Рис. 2.6. – сторінка «Кошик» з обраними товарами

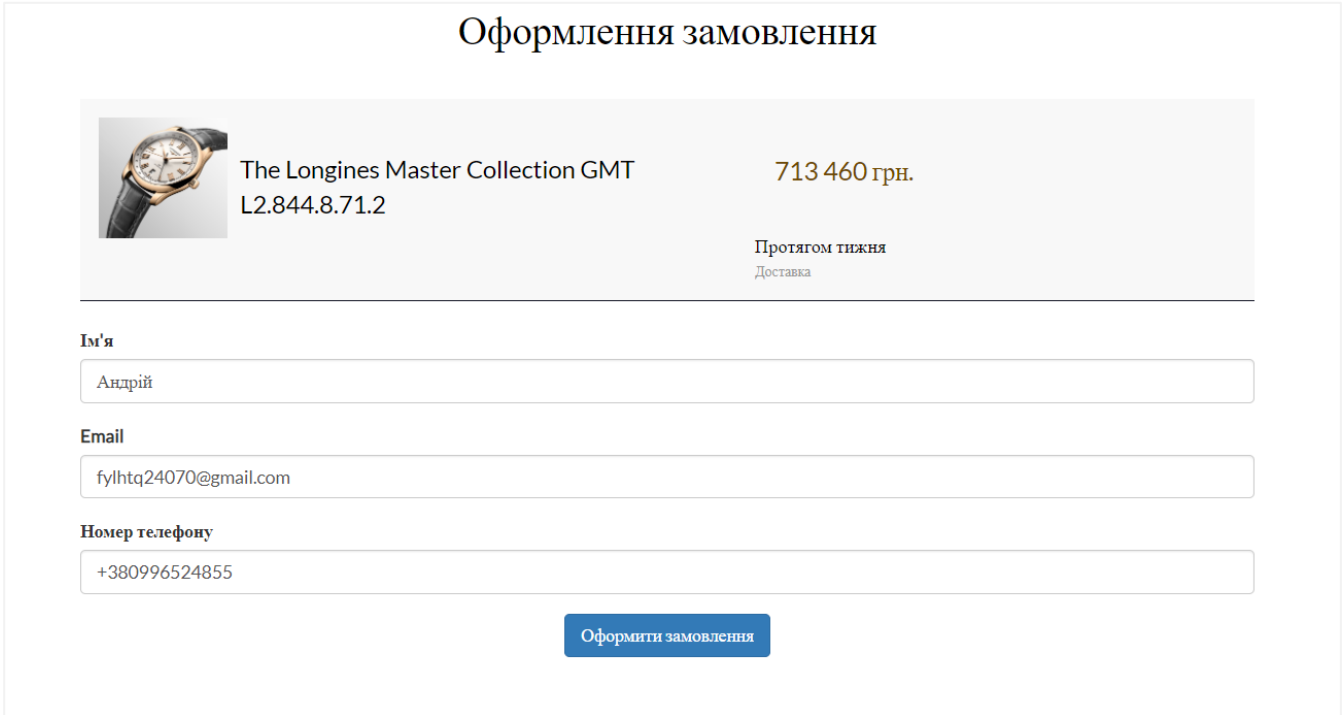

Рисунок 2.7 – Меню оформлення замовлення

Після того, як користувач введе всі необхідні дані на формі замовлення, включаючи контактну інформацію та адресу доставки, він переходить до важливого

етапу - вибору методу оплати. На цьому етапі він має можливість вибрати з різних варіантів оплати, які надає інтернет-магазин. Серед них можуть бути оплата карткою, платіжним переказом, оплата при отриманні або інші доступні методи. Це важлива опція, яка дозволяє користувачеві вибрати найзручніший для нього спосіб оплати за замовлення.

Після вибору методу оплати і подальшого підтвердження замовлення, користувач завершує процес покупки і отримує підтвердження про замовлення [25]. Тим часом інформація про замовлення надсилається за допомогою Telegram Bot API у чат менеджерів. Це автоматизує процес збирання інформації щодо замовлень від всіх клієнтів, та покращить швидкість реагування робітника інтернет-магазину на замовлення.

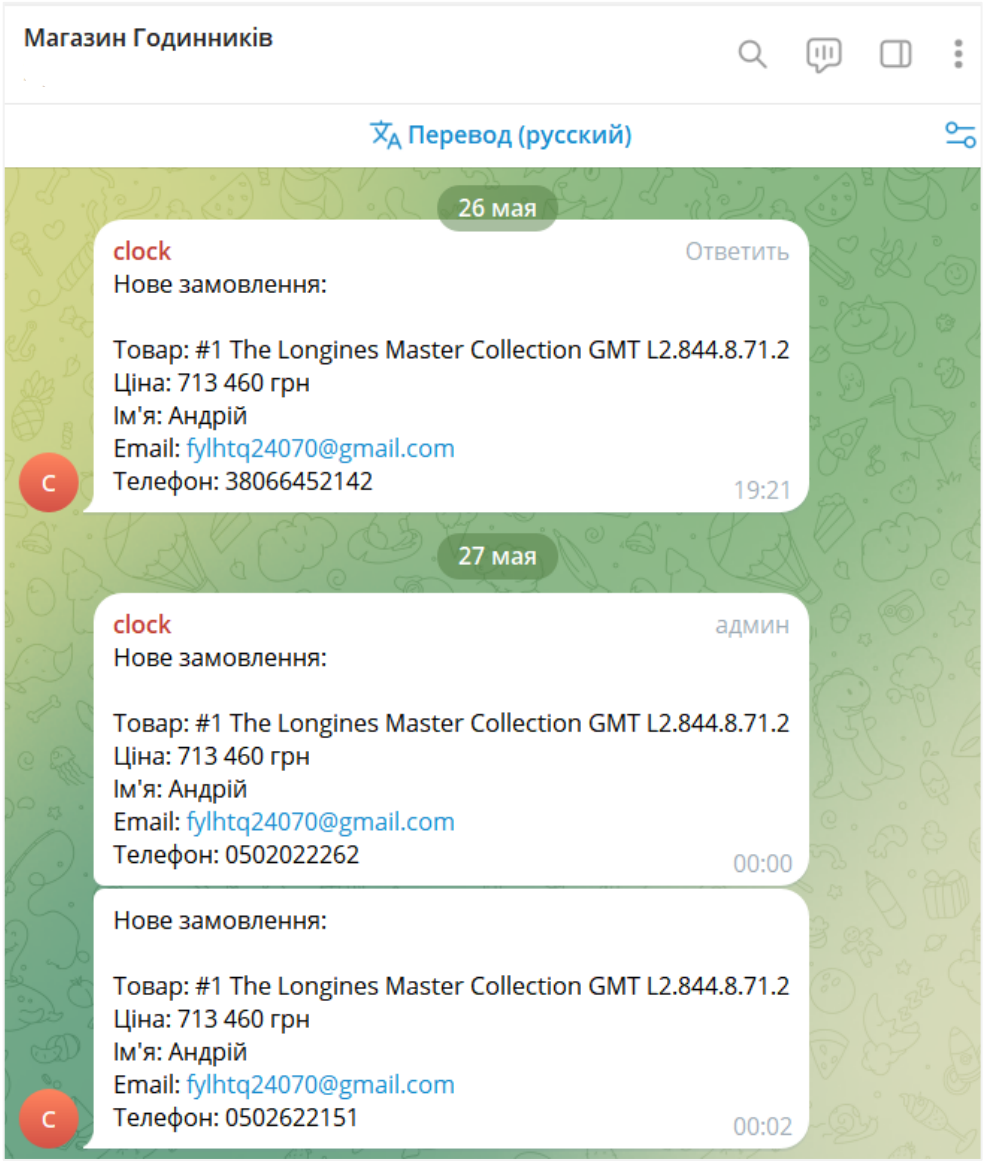

Рисунок 2.8 – Чат для прийому замовлень

Перед тим як додавати товари, першочергово створюються категорії. Категорії мають ієрархічну структуру, і хоча умовно їх глибина вкладеності може бути нескінченною, фактично це обмеження визначається системними вимогами. Вимог до створення категорій немає, і вони формуються виключно на основі логіки та практичності для групування товарів. Слід зауважити, що існують інші способи групування, такі як артикули, проте для більшої зрозумілості та логічності переважною залишаються категорії.

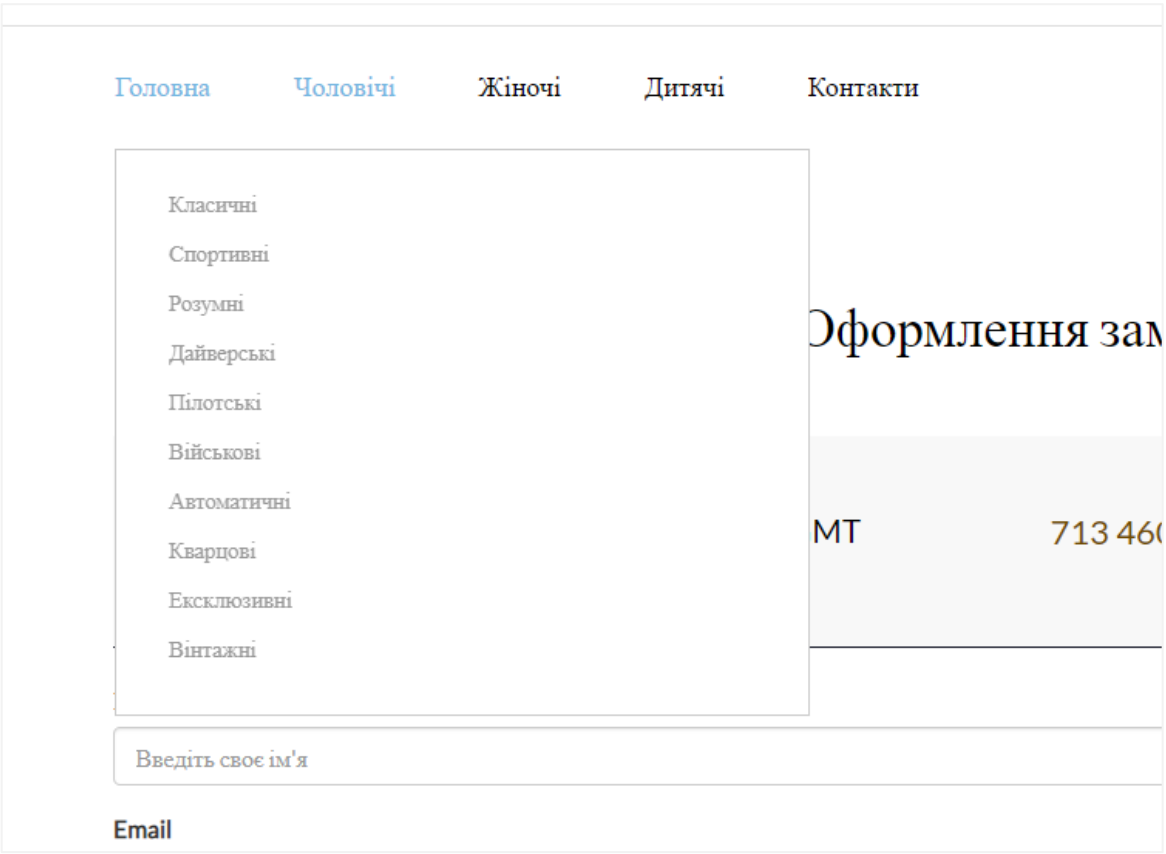

Рисунок 2.9 – Створені категорії товарів

2.2. Структура та особливості реалізації алгоритмічного забезпечення

Процес проектування структури та бази даних для інтернет-магазину з продажу годинників у MySQL передбачає кілька ключових етапів. Починаючи з аналізу вимог, визначаються основні сутності, такі як категорії, підкатегорії, товар,

колір товару, зображення товару тощо. Потім встановлюються взаємозв'язки між цими сутностями, відповідно до їхньої природи та потреб бізнесу.

За допомогою визначених сутностей та їх взаємозв'язків розробляється схема бази даних, яка відображає структуру та організацію даних в системі. Для товару це може включати атрибути, такі як назва, бренд, ціна, матеріал корпусу тощо. Товари можуть мати атрибути, такі як артикул (ID), посилання на підкатегорію, назву, назву у URL форматі, вартість, знижку, опис, додаткову інформацію, коли створено товар та коли останній раз редаговано.

Для забезпечення ефективного доступу до даних і оптимізації запитів, можна розглядати індексацію відповідних полів, таких як унікальні ідентифікатори, інформацію, яка часто використовується для пошуку, тощо. Також потрібно розглянути питання безпеки, такі як захист від SQL-ін'єкцій, аутентифікація та авторизація користувачів. Завдяки Laravel та Eloquent ORM користувачі захищені від SQL-ін'єкцій.

Після створення схеми бази даних необхідно реалізувати цю структуру в MySQL, використовуючи відповідні команди для створення таблиць, відносин між ними, індексів тощо. Важливо також регулярно вносити зміни в базу даних відповідно до зростання потреб бізнесу та вимог користувачів, забезпечуючи її масштабованість та ефективність.

Виділення та подальший розгляд основних бізнес-процесів в системі інтернет-магазину є критичним етапом для розуміння функціонування та ефективності цієї організаційної структури. Незважаючи на те, що бізнес-процеси магазину можуть мати загальні риси з процесами, які спостерігаються в будь-якому підприємстві у сфері торгівлі, наявні специфічні особливості, що визначають їхню унікальність. Отже, ретельне вивчення цих процесів та їх взаємозв'язків необхідне для розробки ефективних стратегій управління та оптимізації роботи магазину [24].

Таким чином, розгляд основних бізнес-процесів в системі інтернет-магазину є складним завданням, яке потребує системного підходу та комплексного аналізу. Тільки враховуючи всі аспекти функціонування магазину можна розробити

ефективні стратегії управління та забезпечити його успішну діяльність у конкурентному електронному ринковому середовищі.

Важливо відзначити, що однією з ключових складових витрат є доставка товару. Інтернет-магазин може заощадити кошти шляхом оптимізації витрат на доставку. Одним із методів оптимізації є впровадження вебсервісу для оптимізації маршрутів доставки, що може бути реалізовано як окремий модуль інформаційної системи магазину [32].

Цей сервіс повинен вирішувати ряд завдань, включаючи встановлення оптимальних маршрутів доставки, розрахунок планованого пробігу і часу на кожен маршрут, автоматичне коригування маршрутних аркушів для водіїв, а також моніторинг руху машин у режимі онлайн та облік фактичного маршруту та пробігу автомобіля.

Проте, важливо врахувати, що в контексті дослідження оптимізація транспортної логістики може бути менш ефективною з ряду об'єктивних причин.

На практиці, у інтернет-магазинах проблеми частіше виникають на рівні логістики та складського господарства, ніж на програмному рівні. Система, яка програмно управляється та оптимізована для роботи з клієнтами та менеджерами, може бути недостатньо ефективною для вирішення завдань, пов'язаних зі зберіганням, переміщенням та доставкою товарів.

Таким чином, для підвищення ефективності інтернет-магазину важливо розглянути не лише програмні аспекти, але й проблеми, пов'язані з логістикою та складським господарством, та впровадити відповідні заходи для їх вирішення.

Центральний склад, що є ключовим ланцюгом виробничого процесу, є осередком інформаційних викликів у контексті ефективного функціонування. У центральному складі зосереджені два важливі бізнес-процеси. З одного боку, він виступає як вхідний інтерфейс у процесі замовлення та одержання брендового одягу від виробників, а з іншого боку, він є вихідним інтерфейсом, формуючи відправлення замовлень для покупців згідно з отриманими заявками [18].

Створення та підключення до бази даних MySQL здійснюється за допомогою інструментів Laravel. На рисунках 2.10, 2.11 та 2.12 наведений вихідний код кожної

з таблиць, що дозволяє краще зрозуміти їхню структуру та взаємозв'язки між ними. Аналіз цього коду може допомогти розкрити технічні деталі і реалізаційні аспекти використання бази даних MySQL у контексті конкретного проекту.

```php
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create( table: 'categories', function (Blueprint $table) {
            $table->id();
            $table->string( column: 'slug');
            $table->string( column: 'name');
        });
    }
```

Рисунок 2.10 – Вихідний код таблиці categories

```php
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create( table: 'products', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger( column: 'subcategory');
            $table->string( column: 'name');
            $table->string( column: 'slug');
            $table->integer( column: 'price');
            $table->integer( column: 'discount');
            $table->text( column: 'description')->nullable();
            $table->text( column: 'extra')->nullable();
            $table->timestamps();
            $table->foreign( columns: 'subcategory')->references( columns: 'id')->on( table: 'subcategories')->onDelete( action: 'cascade');
        });
    }
```

Рисунок 2.11 – Вихідний код таблиці products

Процеси зміни та видалення товару можуть бути ключовими складовими ефективного управління інвентарем в магазині, сприяючи підтримці актуальності та точності даних про наявний асортимент. Створення та підключення до бази даних MySQL через Laravel є сучасним та потужним підходом до забезпечення надійного зберігання та обробки даних. Аналіз вихідного коду таблиць дозволяє

розкрити внутрішню структуру даних та визначити оптимальні стратегії управління ними [45].

Вищесказане підкреслює важливість ефективного управління товарним асортиментом в магазині, включаючи процеси додавання, зміни та видалення товарів. Використання сучасних технологій, таких як база даних MySQL та розробка за допомогою Laravel, сприяє створенню надійних та масштабованих систем управління. Аналіз структури даних дозволяє оптимізувати процеси роботи з ними, що в свою чергу сприяє підвищенню ефективності та точності обслуговування клієнтів.

2.3 Реалізація прототипу інформаційної системи та інструкції з використання

Фізична модель бази даних є ключовим етапом в розробці системи, оскільки вона надає детальний опис усіх об'єктів, що входять до бази даних, і визначає їхню внутрішню структуру та зв'язки між ними. Крім того, у фізичній моделі бази даних відображаються всі проміжні таблиці, а також типи даних атрибутів, що використовуються для представлення інформації.

Наприклад, у зазначеній фізичній моделі бази даних розглядаються чотири основні таблиці. "Категорії", "Підкатегорії", "Товар", "Зображення товару", та "Кольори товару". Кожна з цих таблиць відповідає певній сутності системи та містить у собі відповідні поля та взаємозв'язки.

Детальний аналіз фізичної моделі бази даних дозволяє розкрити всі аспекти її структури, а також визначити оптимальний спосіб організації даних для досягнення максимальної ефективності та швидкодії системи. База даних є програмним забезпеченням на сервері, яке забезпечує зберігання та надання даних у потрібний момент. Наприклад, у випадку форуму або блогу, дані, такі як пости, коментарі, новини, зберігаються в базі даних. База даних зазвичай розташовується

на сервері. Серверна частина вебдодатка взаємодіє з базою даних, витягуючи необхідну інформацію для формування сторінок, запитаних користувачем.

База даних ‘watches’(рис. 2.12) містить кілька таблиць, які використовуються для зберігання даних про годинники та їх категорії. Основні таблиці цієї бази даних є:

– categories;

– subcategories;

– products;

– products_colors;

– products_images;

– migrations;

– cache;

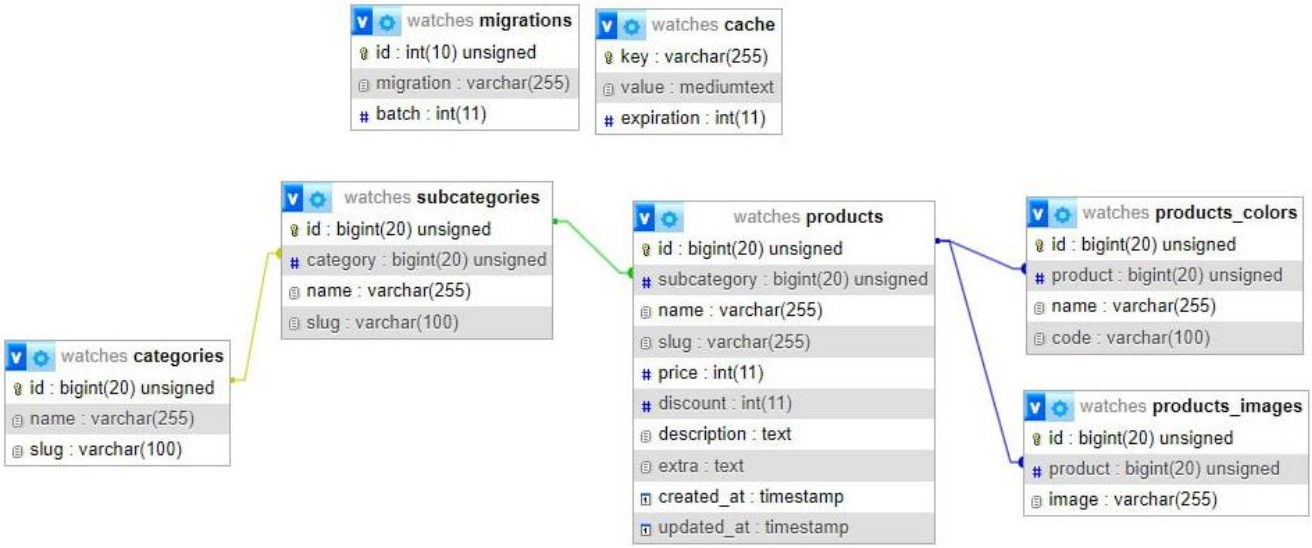

Рисунок 2.12 – Структура бази даних

На схемі показано структуру таблиць і зв'язки між ними. Нижче представлено детальний опис кожної таблиці та їх зв'язків.

Таблиця categories:

– id: Ідентифікатор категорії (тип: bigint(20) unsigned)

– name: Назва категорії (тип: varchar(255))

– slug: Унікальний ідентифікатор категорії для URL (тип: varchar(100))

Таблиця subcategories:

– id: Ідентифікатор підкатегорії (тип: bigint(20) unsigned);

– category: Ідентифікатор категорії, до якої належить підкатегорія (тип: bigint(20) unsigned);

– name: Назва підкатегорії (тип: varchar(255));

– slug: Унікальний ідентифікатор підкатегорії для URL (тип: varchar(100))

Пов'язана з таблицею categories через поле category.

Таблиця products:

– id: Ідентифікатор продукту (тип: bigint(20) unsigned);

– subcategory: Ідентифікатор підкатегорії, до якої належить продукт (тип: bigint(20) unsigned);

– name: Назва продукту (тип: varchar(255));

– slug: Унікальний ідентифікатор продукту для URL (тип: varchar(255));

– price: Ціна продукту (тип: int(11));

– discount: Відсоток знижки (тип: int(11));

– description: Опис продукту (тип: text);

– extra: Додаткова інформація (тип: text);

– created_at: Дата та час створення запису (тип: timestamp);

– updated_at: Дата та час останнього оновлення запису (тип: timestamp);

Пов'язана з таблицею subcategories через поле subcategory.

Таблиця products_colors:

– id: Ідентифікатор кольору продукту (тип: bigint(20) unsigned);

– product: Ідентифікатор продукту (тип: bigint(20) unsigned);

– name: Назва кольору (тип: varchar(255));

– code: Код кольору (тип: varchar(100));

Пов'язана з таблицею products через поле product.

Таблиця products_images:

– id: Ідентифікатор зображення продукту (тип: bigint(20) unsigned);

– product: Ідентифікатор продукту (тип: bigint(20) unsigned);

– image: URL зображення (тип: varchar(255));

Пов'язана з таблицею products через поле product.

Таблиця migrations:

– id: Ідентифікатор міграції (тип: int(10) unsigned);

– migration: Назва міграції (тип: varchar(255));

– batch: Номер батча міграції (тип: int(11));

Таблиця cache:

– id: Ідентифікатор кешу (тип: int(11));

– key: Ключ кешу (тип: varchar(255));

– value: Значення кешу (тип: mediumtext);

– expiration: Час життя кешу (тип: int(11)).

Аналіз структури таблиць в базі даних ‘watches’ показує, що вона добре організована для зберігання та управління інформацією про продукти, їх категорії, кольори та зображення. Зв'язки між таблицями дозволяють ефективно структурувати дані та забезпечують цілісність інформації. Використання phpMyAdmin дозволяє зручно керувати цією інформацією та виконувати різні операції з базою даних MySQL.

Окрему увагу приділено тестуванню баз даних, що зберігають інформацію про товари, замовлення та клієнтів. Перевірялися операції з додавання, оновлення, видалення та вибірки даних для забезпечення їхньої коректності та цілісності. Було важливо переконатися, що бази даних можуть обробляти запити ефективно, навіть за умови високого навантаження. Для цього використовувалися спеціальні інструменти для симуляції множинних запитів одночасно. Було проведено аналіз продуктивності баз даних для виявлення можливих вузьких місць та оптимізації запитів. Крім того, тестування включало перевірку безпеки баз даних, щоб запобігти несанкціонованому доступу та втраті даних.

Оцінювалися різні функціональні можливості системи, включаючи модулі статистики, які надають важливу аналітичну інформацію про відвідуваність сайту, поведінку користувачів і популярність товарів. Ці дані використовуються для покращення маркетингових стратегій та управління асортиментом. Системи оплати і доставки також пройшли ретельну перевірку для забезпечення безперебійної

роботи з різними платіжними шлюзами і кур'єрськими службами. Перевірялися різні сценарії, включаючи відміну замовлень, повернення товарів та обробку спірних ситуацій.

Тестування інтерфейсу адміністратора включало перевірку функціональності управління замовленнями, клієнтами, а також налаштування знижок та спеціальних пропозицій. Аналіз функціональних можливостей системи також включав перевірку коректності виведення статистичних даних, таких як обсяги продажів, динаміка відвідувань та інші важливі метрики. В результаті тестування були виявлені та виправлені різні помилки, що підвищило загальну стабільність та ефективність роботи інтернет-магазину [40].

Було переглянуто цін на товари в розділі "Чоловічі годинники" перед їх додаванням на сайт, а також проведено аналіз цін у інших магазинах. На основі цього були сформульовані рекомендації щодо ціноутворення.

У режимі асинхронного доступу були досліджені наступні аспекти:

1. Всі замовлення, зроблені покупцями, були вивчені (рис. 2.13).

2. Розглянуті всі дані про покупців, а також статистика по рухові товару і замовленням.

3. На основі отриманих даних сформульовані рекомендації щодо придбання товарів для магазину.

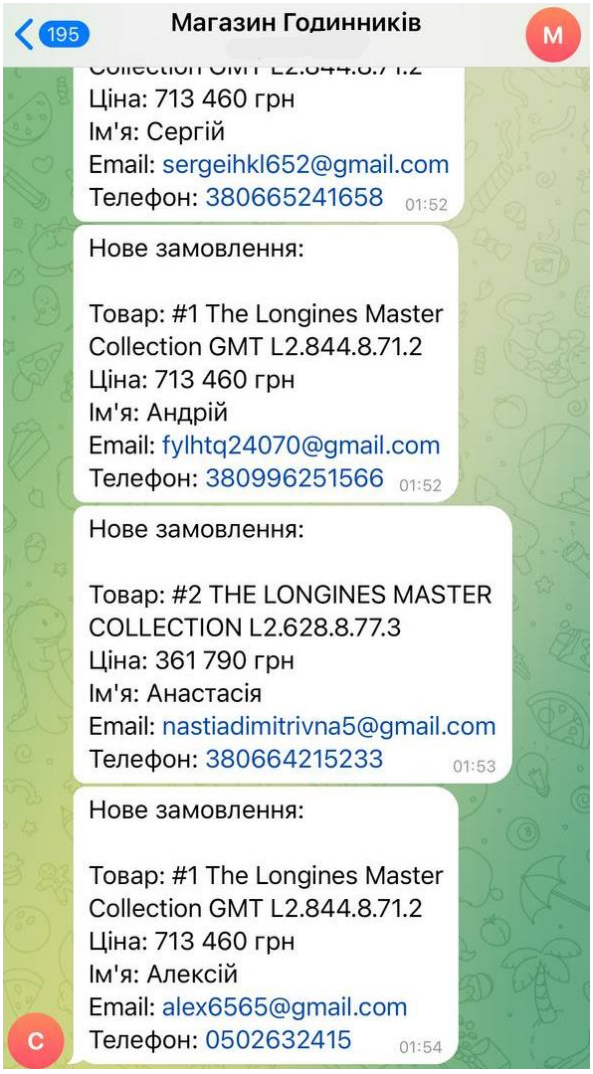

Рисунок 2.13 – Інформація про замовлення

Таким чином, у режимі менеджера були перевірені всі функції. Система працює стабільно та коректно. Під час перевірки було підтверджено правильність роботи авторизації, а також безпомилкову роботу інших функцій, таких як робота з товарами та статистикою.

Отже, можна зазначити, що система функціонує надійно та ефективно. Головні процеси здійснюються менеджером після входу до системи статистики. Їх можна виконувати як у синхронному, так і у асинхронному режимі, що забезпечує оптимальне навантаження сервера[32].

Головна панель адміністратора є центром інформації. PHPMyAdmin – це вебінтерфейс для управління базами даних MySQL, який надає можливість виконувати різноманітні операції з базами даних через зручний вебінтерфейс.

Завдяки PHPMyAdmin можна створювати, редагувати, видаляти бази даних, таблиці, виконувати SQL-запити, керувати користувачами та їх правами доступу, і багато іншого.

Однією з головних переваг PHPMyAdmin є його простота в установці та використанні. Він підтримується на багатьох хостинг-провайдерах і може бути легко встановлений на власному сервері. Крім того, PHPMyAdmin має user-friendly інтерфейс, що дозволяє навіть користувачам без досвіду ефективно керувати базами даних без необхідності знати SQL.

Ще однією важливою особливістю PHPMyAdmin є можливість візуального редагування даних у таблицях. Користувач може переглядати дані у вигляді таблиці, редагувати окремі записи, додавати нові записи або видаляти існуючі без використання SQL-запитів. Це дуже зручно для тих, хто не хоче або не може писати складні SQL-запити, але потребує доступу до даних у базі.

Крім того, PHPMyAdmin має ряд інших корисних функцій, таких як можливість імпорту та експорту даних у різноманітних форматах (наприклад, CSV, SQL, XML), генерація звітів про структуру бази даних, аналіз використання місця на диску та багато іншого.

PHPMyAdmin також має вбудовану підтримку для роботи з різними версіями MySQL, включаючи нові функції та покращення, які можуть з'являтися у випусках бази даних. Це дозволяє користувачам PHPMyAdmin використовувати останні технологічні досягнення без необхідності ручної настройки або оновлення програмного забезпечення.

Однією з ключових переваг PHPMyAdmin є також активна спільнота користувачів і розробників, яка постійно розвиває цей інструмент і надає підтримку в онлайн-форумах, чат-каналах та інших ресурсах. Це робить PHPMyAdmin не просто інструментом, а справжньою спільнотою, де можна отримати допомогу, поради та взаємодіяти з іншими користувачами.

Необхідно також відзначити, що PHPMyAdmin є відкритим програмним забезпеченням з вільною ліцензією GNU GPL. Це означає, що може використовувати, змінювати і поширювати PHPMyAdmin без обмежень,

забезпечуючи вільний доступ до його коду та сприяючи подальшому розвитку і вдосконаленню цього інструменту. Саме з його допомогою відбувається взаємодія з сайтом(рис. 2.14).

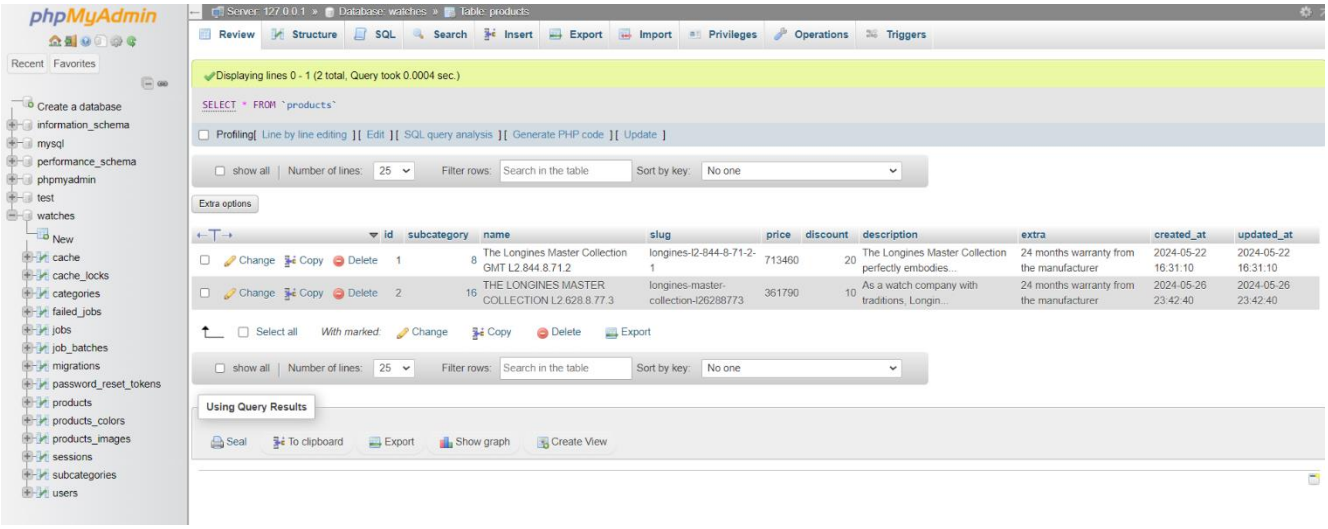

Рисунок 2.14 – Інформація про товари з PHPMyAdmin

Для ефективного управління даними в вебдодатках використовується база даних, яка дозволяє зберігати та обробляти інформацію. Більш детально про структуру бази даних `watches`, а також приведено аналіз таблиці `products`, що зберігає інформацію про продукти.

База даних `watches` містить кілька таблиць, які використовуються для зберігання даних про годинники та їх категорії. Однією з ключових таблиць у цій базі даних є таблиця `products`, яка містить інформацію про окремі продукти.

Таблиця `products` містить такі поля:

– id: Ідентифікатор продукту (цілісне число);
– subcategory: Ідентифікатор підкатегорії (цілісне число);
– name: Назва продукту (текстовий рядок);
– slug: Унікальний ідентифікатор продукту для URL (текстовий рядок);
– price: Ціна продукту (цілісне число);
– discount: Відсоток знижки (цілісне число);
– description: Опис продукту (текстовий рядок).
– extra: Додаткова інформація (текстовий рядок).

– created_at: Дата та час створення запису (мітка часу).

– updated_at: Дата та час останнього оновлення запису (мітка часу).

На момент аналізу таблиця `products` містить два записи. Нижче представлено детальний опис кожного з них.

Перший запис:

– id: 1

– subcategory: 8

– name: The Longines Master Collection GMT L2.844.8.71.2

– slug: longines-l2-844-8-71-2-1

– price: 713460

– discount: 20%

– description: Колекція Longines Master Collection ідеально втілює традиції...

– extra: 24 місяців гарантії від виробника

– created_at: 2024-05-22 16:31:10

– updated_at: 2024-05-22 16:31:10

Другий запис:

– id: 2

– subcategory: 16

– name: THE LONGINES MASTER COLLECTION L2.628.8.77.3

– slug: longines-master-collection-l26288773

– price: 361790

– discount: 10%

– description: Будучи годинниковою компанією з традиціями...

– extra: 24 місяців гарантії від виробника

– created_at: 2024-05-26 23:42:40

– updated_at: 2024-05-26 23:42:40

Аналіз структури таблиці `products` в базі даних `watches` показує, що вона містить всі необхідні поля для зберігання інформації про продукти, включаючи їх ідентифікатори, назви, ціну, знижку, опис та додаткову інформацію. Використання

phpMyAdmin дозволяє легко керувати цією інформацією та забезпечує зручний інтерфейс для роботи з базами даних MySQL.

Вся взаємодія з товарами на сайті працює за допомогою PHPMyAdmin. Для того, щоб додати новий товар вам необхідно натиснути «Insert» на верхній панелі та перейти до розділу з додаванням інформації до БД (рис. 2.15).

Для того щоб додати новий запис необхідно заповнити наступні поля:

–id: Ідентифікатор продукту (тип: bigint(20) unsigned).

–subcategory: Ідентифікатор підкатегорії (тип: bigint(20) unsigned).

–name: Назва продукту (тип: varchar(255)).

–slug: Унікальний ідентифікатор продукту для URL (тип: varchar(255)).

–price: Ціна продукту (тип: int(11)).

–discount: Відсоток знижки (тип: int(11)).

–description: Опис продукту (тип: text).

–extra: Додаткова інформація (тип: text).

–created_at: Дата та час створення запису (тип: timestamp).

–updated_at: Дата та час останнього оновлення запису (тип: timestamp).

Інтерфейс phpMyAdmin дозволяє виконувати різні операції з базами даних MySQL, такі як перегляд, редагування, додавання та видалення записів у таблицях. На зображенні показано екран для додавання нового запису до таблиці products. Користувач може заповнити відповідні поля, зазначивши значення для кожного з них, та зберегти новий запис у базі даних.

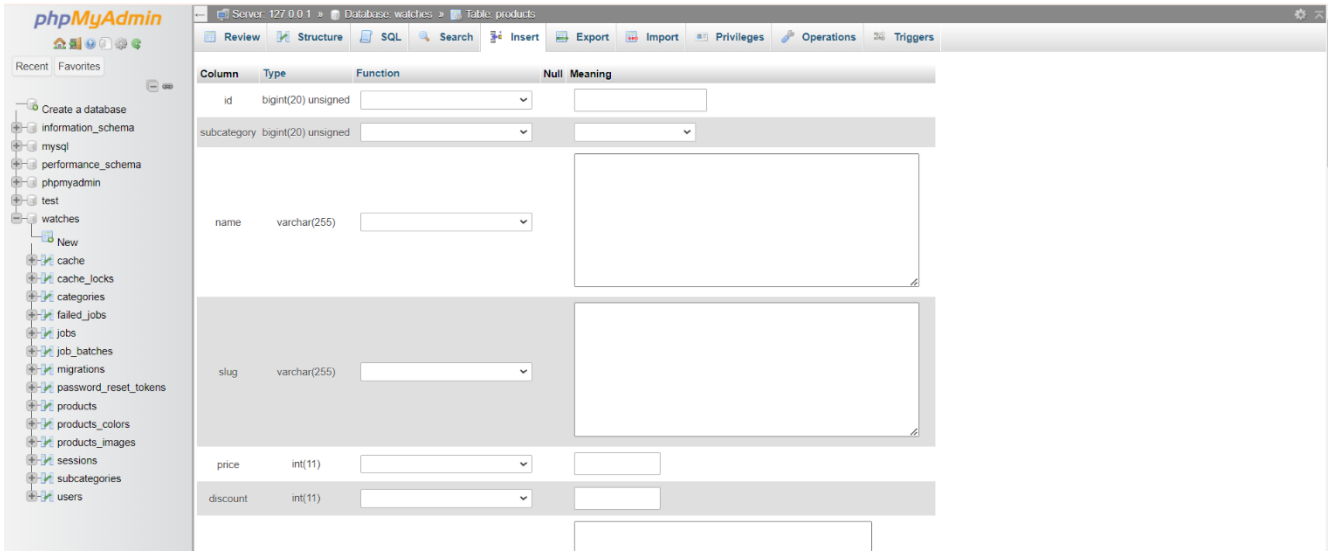

Рисунок 2.15 – Розділ додавання нової продукції

ВИСНОВКИ

Таким чином, проведений аналіз сучасних рішень у сфері електронної комерції дозволив визначити основні вимоги до функціональності інтернет-магазину годинників, що включають зручний інтерфейс користувача, безпеку даних, інтеграцію з платіжними системами та системами доставки. Розроблена архітектура системи забезпечує високу надійність та масштабованість, що дозволяє адаптувати її до зростаючих потреб бізнесу та збільшення навантаження. В процесі розробки були реалізовані основні функціональні модулі, такі як каталог товарів, кошик для покупок, система управління замовленнями та інтеграція з платіжними сервісами, що забезпечує повний цикл обслуговування клієнтів від вибору товару до здійснення покупки та доставки.

Проведене тестування системи підтвердило її працездатність та відповідність вимогам, що були поставлені на етапі проектування. Особлива увага була приділена оптимізації продуктивності та забезпеченню високого рівня безпеки даних користувачів. Розроблена документація для користувачів та адміністративного персоналу сприяє швидкому впровадженню та ефективному використанню системи. Таким чином, створена веборієнтована інформаційна система для інтернет-магазину годинників дозволяє автоматизувати основні бізнес-процеси, покращити взаємодію з клієнтами та підвищити ефективність управління підприємством. Це сприятиме підвищенню конкурентоспроможності магазину на ринку, залученню нових клієнтів та збільшенню обсягів продажів, що є важливими факторами успіху в умовах сучасної економіки.

У першому розділі було проаналізовано тенденції розвитку електронної комерції та ринку годинників, сформульовані вимоги до веборієнтованої інформаційної системи для магазину годинників та побудовані моделі бізнес-процесів магазину годинників. Окремо в цьому ж розділі описано архітектуру інформаційної системи та технології розв’язання поставлених задач.

У другому ж розділі описана структура та особливості реалізації інформаційного забезпечення, структуру та особливості реалізації алгоритмічного забезпечення, а також реалізована веборієнтована інформаційна система для магазину годинників та розроблені інструкції з використання інформаційної системи.

Підсумовуючи, розробка веборієнтованої інформаційної системи магазину годинників стала успішним проектом, який відповідає сучасним вимогам ринку та забезпечує високу якість обслуговування клієнтів.

Розроблена система відповідає сучасним вимогам та стандартам якості, забезпечуючи зручний інтерфейс користувача та ефективне управління бізнес-процесами магазину. Основні функціональні модулі, такі як каталог товарів, кошик для покупок, система управління замовленнями та інтеграція з платіжними сервісами, були успішно реалізовані, забезпечуючи повний цикл обслуговування клієнтів. Додатково, проведене тестування системи підтвердило її працездатність та надійність, а економічний аналіз підтвердив ефективність реалізації проекту. Отже, розробка веборієнтованої інформаційної системи для інтернет-магазину годинників є важливим кроком у напрямку розвитку електронної комерції, сприяючи покращенню обслуговування клієнтів та підвищенню конкурентоспроможності підприємства.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Абдуліна І., Березанська В. Роблячи ставки на розробку програмного забезпечення, 2012. 8-12 с.

2. Ткачук В.М. Алгоритми і структура даних, Івано-Франківськ : Прикарпатський національний університет імені Василя Стефаника, 2016. 286 с.

3. Коротєєва Т.О. Алгоритми та структури даних. Львів, Львівська політехніка, 2014. 280 с.

4. Алексенко О.В. Технології програмування та створення програмних продуктів, Суми, Сумський державний університет, 2013. 133 с.

5. Баран С.В. Основи web-програмування, Кривий Ріг, Державний університет економіки і технологій, 2023. 316 с.

6. Березко О.Л., Пелещишин А.М., Жежнич П.І., Концепція створення вебсайта Національного університету „Львівська політехніка” URL: http://science.lpnu.ua/sites/default/files/journalpaper/2017/jun/3146/12berezko-57-65.pdf (дата звернення 15.05.2024).

7. Бородкіна І. Л., Бородкін Г. О., Web-технології та Web-дизайн: застосування мови HTML для створення електронних ресурсів, Ліра-К, 2020. 210 с.

8. Глоба Л. С., Кот Т. М. Розробка інформаційних ресурсів та систем, Київ, НТУУ "КПІ", 2014. 318 с.

9. Григорович В. Г. Методичні вказівки до виконання лабораторних і практичних завдань з курсу «Алгоритмізація та програмування», Дрогобич, Редакційно-видавничий відділ Дрогобицького державного педагогічного університету імені Івана Франка, 2016. 1400 с.

10. Г.В Табунщик, Р.К. Кудерметов, Т.І. Брагіна, Інженерія якості програмного забезпечення Запоріжжя: ЗНТУ, 2013. 180 с.

11. Кравець П, Об'єктно-орієнтоване програмування, Львів, Видавництво Львівської політехніки, 2012. 624 с.

12. Микитишин А.Г., Митник М.М., Стухляк П.Д., Пасічник В.В. Компютерні мережі : навч. посіб, Львів Магнолія, 2013, 250 с.

13. Ю.Я. Томка та інші, Основи роботи із системою контролю версій GIT Чернівці: Технодрук, 2022. 200с.

14. Отеро, Карлос, Виклики дизайну програмного забезпечення Покращення продуктивності IT. ТОВ "Тейлор і Френсіс", 2017. 280 с.

15. Павленко П.М., Основи математичного моделювання систем і процесів: Книжкове видавництво НАУ, 2014. 274 с.

16. Пасічник В. В., Пасічник О. В., Вебдизайн, Львів, Магнолія 2006, 2020. 518 с.

17. Пасічник В. В., Пасічник О. В., Угрин Д. І., Вебтехнології Львів Магнолія, 2013, 215 с.

18. М. Ю. Карпенко, Н. О. Манакова, І. О. Гавриленко , Технології створення програмних продуктів та інформаційних систем, Харків : ХНУМГ ім. О. М. Бекетова, 2017. 93 с.

19. Трофименко О. Г. та інші. Вебтехнології та вебдизайн Одеса, Фенікс, 2019. 284 с.

20. Цвіркун Л.І. Глобальні комп’ютерні мережі. Програмування мовою PHP: Національний гірничий університет, 2013. 239 с.

21. Цвіркун Л.І., Євстігнєєва А.А., Панферова Я.В., Розробка програмного забезпечення комп’ютерних систем, Дніпро: НТУ «ДП», 2019.

22. Шевчук І. Б. Інформаційні технології в регіональній економіці: теорія і практика впровадження та використання, Видавництво ННВК "АТБ", 2018. 448 с.

23. Ян Шмідт; В. Климченко, Das Neue Nets Markmale, Praktiken und Folgen des Web 2.0, Київ, Академія Української Преси, Центр Вільної Преси, 2013. 283 с

24. Abowd G.Dт. та інші. Towards a Better Understanding of Context and Context-Awareness. In: Gellersen HW. (eds) Handheld and Ubiquitous Computing. HUC 1999. Lecture Notes in Computer Science,.pp. 304-307.

25. Cristiana Bolchini, та інші. A data-oriented survey of context models. ACM SIGMOD Record, 4 (December 2007), pp. 19-26

26. Csaba Szepesvari, Algorithms for Reinforcement Learning, Morgan and Claypool Publishers, 2010. 104 p.

27. David L. Poole, Alan K. Mackworth, Artificial Intelligence: Foundations of Computational Agents, Cambridge University Press, 2017. 820 p.

28. Fonseca, C.M.; Fleming, P.J. Genetic algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Proceedings of the Fifth International Conference on Genetic Algorithms, UrbanaChampaign, USA, 17–22 July 1993. pp. 416–423.

29. Grifoni P., D'Ulizia A., Ferri F., Context-Awareness in Location Based Services in the Big Data Era, In: Skourletopoulos G., Mastorakis G., Mavromoustakis C., Dobre C., Pallis E. (eds) Mobile Big Data. Lecture Notes on Data Engineering and Communications Technologies, Springer, 2018. pp. 85–127.

30. Guckkenheimer S., Peter J. Software Engineering With Microsoft Visual Studio. Team System. – Adison Wesley, 2006. – 273 p.

31. IEEE Standard Glossary of Software Engineering Terminology, Глосарій. IEEE Std 610.12-1990.

32. J. Rosell, N. Munoz, A. Gambin, "Robot tasks sequence planning using Petri nets," Proceedings of the IEEE International Symposium on Assembly and Task Planning, Besancon, France, 2003. pp. 24-29.

33. L. Zhang et al., "Adaptive quantum genetic algorithm for task sequence planning of complex assembly systems," in Electronics Letters, no. 14, 2018. pp. 870-872.

34. Multiagent Systems, by Gerhard Weiss (Editor), The MIT Press, 2013. p. 920

35. Nicholas Capurs, A survey on key fields of context awareness for mobile devices. Journal of Network and Computer Applications, 2018. pp. 44-60/

36. Perera C., Zaslavsky A., Christen P, D. Georgakopoulos, "Context Aware Computing for The Internet of Things: A Survey," Communications Surveys & Tutorials, no. 1, First Quarter 2014, pp. 414-454.

37. Pinheiro Francisco A. C., Goguen Joseph A. An Object-Oriented tool for Tracing Requirements, Mach 1996.– № 3.

38. Richard S. Sutton, Andrew G. Barto, Reinforcement Learning: An Introduction, A Bradford Book, 2018. 532 p.

39. Schilit B., Adams N., Want R., Context-aware computing applications, in Proceedings of the IEEE Workshop on “Mobile Computing Systems and Applications”, IEEE Computer Society, 1994. – pp. 85-90.

40. Stuart Russell, Peter Norvig, Artificial Intelligence: A Modern Approach, Pearson, 2020. – 1136 p.

41. Systems and software engineering, Software Life Cycle Processes, 122 c.

42. Tiehua Cao, Arthur C. Sanderson, Intelligent Task Planning Using Fuzzy Petri Nets, World Scientific, 1996 – 192 p.

43. Tierney K., Browne J., Task sequence planning, In: Bernhardt R., Dillman R., Hörmann K., Tierney K. (eds) Integration of Robots into CIM. Springer, 1992. pp. 36-44.

ДОДАТКИ

ДОДАТОК А

SUMMARY

Dydyshko A.M. Development of a Web-Oriented Information System for a Watch Store – Bachelor's Qualification Work, Sumy State University, Sumy, 2024.

This work examines the process of creating an information system for an online watch store. The advantages and main requirements for such a system are identified. An analysis of the information flows and business processes of the enterprise is conducted. A web-oriented system is developed to improve the comfort of product selection and optimize the ordering process for users, as well as to increase the store's recognition.

Keywords: web-oriented system, automation, watch sales, business processes, web browser, MySQL, PHP, Laravel.

АНОТАЦІЯ

Дидишко А.М. Розробка веборієнтованої інформаційної системи для магазину годинників – кваліфікаційна робота бакалавра, Сумський державний університет, Суми, 2024 р.

У роботі досліджено процес створення інформаційної системи для інтернет-магазину. Визначено переваги та основні вимоги до такої системи. Проведено аналіз інформаційних потоків та бізнес-процесів підприємства. Розроблено веборієнтовану систему, яка покращує комфорт вибору товарів та оптимізує процес замовлення для користувачів, а також підвищує впізнаваність магазину.

Ключові слова: веборієнтована система, автоматизація, продаж годинників, бізнес-процеси, веббраузер, MySQL, PHP, Laravel.

ДОДАТОК Б

(Controllers/Api/ ProductsController.php)

```
<?php

use App\Http\Controllers\ContactsController;
use App\Http\Controllers\MainController;
use App\Http\Controllers\OrderController;
use App\Http\Controllers\ProductController;
use App\Http\Controllers\ProductsController;
use Illuminate\Support\Facades\Route;

//Route::get('/', function () {
//    return view('welcome');
//});

Route::get('/', [MainController::class, 'index']);
Route::get('/{category}/{subcategory}', [ProductsController::class, 'index']);
Route::get('/{category}/{subcategory}/{product_name}',
[ProductController::class, 'index']);
Route::get('/order', [OrderController::class, 'index']);
Route::post('/order/end', [OrderController::class, 'end']);
Route::get('/contacts', [ContactsController::class, 'index']);
```

(Controllers/ContactsController.php)

```
?php
```

```
namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Category;
use Illuminate\Http\Request;

class ContactsController extends Controller
{
    public function index() {
        return view('contacts');
    }
}
```

(Controllers/ Controller.php)

```
<?php

namespace App\Http\Controllers;

abstract class Controller
{
    //
}
```

(Controllers/MainController.php)

```
<?php

namespace App\Http\Controllers;
```

```
use App\Http\Controllers\Controller;
use App\Models\Category;
use App\Models\Product;
use App\Models\SubCategory;
use Illuminate\Http\Request;

class MainController extends Controller
{
   public function index() {
      $categories = Category::all();
      $productsWithHighestDiscounts = Product::orderBy('discount', 'desc')-
>take(8)->get();

      return view('index', compact('categories', 'productsWithHighestDiscounts'));
   }

}
```

(Controllers/OrderController.php)

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Category;
use App\Models\Product;
use App\Models\SubCategory;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
```

```
class OrderController extends Controller
{
    public function index(Request $request)
    {
        // Отримаємо ID товару із GET-запиту
        $productId = $request->query('art');
        if (!$productId) $productId = 1;
        // Знаходимо товар по ID
        $product = Product::with(['colors', 'images'])->findOrFail($productId);

        // Передаємо данні товару во вью
        return view('order', compact('product'));
    }
    public function end(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'id' => 'required|integer|exists:products,id',
            'name' => 'required|string|max:255',
            'email' => 'required|email|max:255',
            'phone' => 'required|digits_between:8,15',
        ]);

        if ($validator->fails()) {
            return redirect()->back()->withErrors($validator)->withInput();
        }

        $product = Product::findOrFail($request->input('id'));
        $name = $request->input('name');
        $email = $request->input('email');
```

```
        $phone = $request->input('phone');

        // Відправка повідомлення в Telegram
        $telegramToken = '6810967317:AAE90hqn-mnN1S3ySwg2XjdJjjoJOv03LpA';
        $chatId = '-1002116255736';
        $message = "Нове замовлення:\n\n" .
            "Товар: #{$product->id} {$product->name}\n" .
            "Ціна: " . number_format($product->price, 0, '.', ' ') . " грн\n" .
            "Ім'я: {$name}\n" .
            "Email: {$email}\n" .
            "Телефон: {$phone}";

        $url = "https://api.telegram.org/bot{$telegramToken}/sendMessage?" . http_build_query([
            'chat_id' => $chatId,
            'text' => $message,
        ]);

        file_get_contents($url);
        return view('end', compact('product'));
    }
}
```

(Controllers/ ProductController.php)

```
<?php

namespace App\Http\Controllers;
```

```
use App\Http\Controllers\Controller;
use App\Models\Category;
use App\Models\Product;
use App\Models\SubCategory;
use Illuminate\Http\Request;

class ProductController extends Controller
{
    public function index($categorySlug, $subcategorySlug, $productSlug) {
        // Отримання категорії
        $category = Category::where('slug', $categorySlug)->firstOrFail();

        // Отримання підкатегорії
        $subcategory = SubCategory::where('slug', $subcategorySlug)->where('category', $category->id)->firstOrFail();

        // Отримання продукту з фото и кольорами
        $product = Product::where('slug', $productSlug)
            ->where('subcategory', $subcategory->id)
            ->with(['colors', 'images'])
            ->firstOrFail();
        return view('product', compact('category', 'subcategory', 'product'));
    }
}
```

(Controllers/ProductsController.php)

```
<?php

namespace App\Http\Controllers;
```

```
use App\Models\Category;
use App\Models\Product;
use App\Models\ProductColor;
use App\Models\SubCategory;
use Illuminate\Http\Request;

class ProductsController extends Controller
{
    public function index($categorySlug, $subcategorySlug, Request $request)
    {
        $category = Category::where('slug', $categorySlug)->firstOrFail();
        $subcategory = SubCategory::where('slug', $subcategorySlug)->where('category', $category->id)->firstOrFail();

        // Отримання унікальних кольорів
        $colors = ProductColor::select('code')->distinct()->get();
        // Отримання товарів з фільтрацією по кольору, якщо параметр color вказан
        $colorFilter = $request->query('color');
        $productsQuery = Product::where('subcategory', $subcategory->id)->with(['colors', 'images']);

        if ($colorFilter) {
            $productsQuery = $productsQuery->whereHas('colors', function ($query) use ($colorFilter) {
                $query->where('code', $colorFilter);
            });
        }
```

```
        $products = $productsQuery->get()->map(function ($product) {
            $product->first_color = $product->colors->first();
            $product->first_image = $product->images->first();
            return $product;
        });

        return view('products', compact('category', 'subcategory', 'colors', 'products'));
    }
}
```

ДОДАТОК В

(Models/ Category.php)

```
?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
    use HasFactory;
    protected $guarded = [];
    protected $table = 'categories';
    public static function getAll() {
        return self::all();
    }

}
```

(Models/Product.php)

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
```

```
use Illuminate\Database\Eloquent\Model;

class Product extends Model
{
    use HasFactory;
    protected $guarded = [];
    protected $table = 'products';

    public function colors()
    {
        return $this->hasMany(ProductColor::class, 'product');
    }

    public function images()
    {
        return $this->hasMany(ProductImage::class, 'product');
    }
}
```

(Models/ProductColor.php)

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class ProductColor extends Model
{
```

```
    use HasFactory;
    protected $guarded = [];
    protected $table = 'products_colors';

    public function product() {
        return $this->belongsTo(Product::class, 'product');
```

(Models/ProductImage.php)

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class ProductImage extends Model
{
    use HasFactory;
    protected $guarded = [];
    protected $table = 'products_images';
}
```

(Models/SubCategory.php)

```
<?php

namespace App\Models;
```

```
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class SubCategory extends Model
{
    use HasFactory;
    protected $guarded = [];
    protected $table = 'subcategories';

    public static function getByCategory($category_id) {
        return self::where('category', $category_id)->get();
    }
}
```

(Models/User.php)

```
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use HasFactory, Notifiable;

    /**
```

```
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * Get the attributes that should be cast.
     *
     * @return array<string, string>
     */
    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
```

```
        ];
    }
}
```

ДОДАТОК Г

(migrations/ 0001_01_01_000000_create_users_table.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.
     */
    public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });

        Schema::create('password_reset_tokens', function (Blueprint $table) {
            $table->string('email')->primary();
```

```
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });

        Schema::create('sessions', function (Blueprint $table) {
            $table->string('id')->primary();
            $table->foreignId('user_id')->nullable()->index();
            $table->string('ip_address', 45)->nullable();
            $table->text('user_agent')->nullable();
            $table->longText('payload');
            $table->integer('last_activity')->index();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('users');
        Schema::dropIfExists('password_reset_tokens');
        Schema::dropIfExists('sessions');
    }
};
```

(migrations/0001_01_01_000001_create_cache_table.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.
     */
    public function up(): void
    {
        Schema::create('cache', function (Blueprint $table) {
            $table->string('key')->primary();
            $table->mediumText('value');
            $table->integer('expiration');
        });

        Schema::create('cache_locks', function (Blueprint $table) {
            $table->string('key')->primary();
            $table->string('owner');
            $table->integer('expiration');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('cache');
        Schema::dropIfExists('cache_locks');
```

```
    }
}
```

(migrations/0001_01_01_000002_create_jobs_table.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.
     */
    public function up(): void
    {
        Schema::create('jobs', function (Blueprint $table) {
            $table->id();
            $table->string('queue')->index();
            $table->longText('payload');
            $table->unsignedTinyInteger('attempts');
            $table->unsignedInteger('reserved_at')->nullable();
            $table->unsignedInteger('available_at');
            $table->unsignedInteger('created_at');
        });

        Schema::create('job_batches', function (Blueprint $table) {
            $table->string('id')->primary();
```

```
            $table->string('name');
            $table->integer('total_jobs');
            $table->integer('pending_jobs');
            $table->integer('failed_jobs');
            $table->longText('failed_job_ids');
            $table->mediumText('options')->nullable();
            $table->integer('cancelled_at')->nullable();
            $table->integer('created_at');
            $table->integer('finished_at')->nullable();
        });

        Schema::create('failed_jobs', function (Blueprint $table) {
            $table->id();
            $table->string('uuid')->unique();
            $table->text('connection');
            $table->text('queue');
            $table->longText('payload');
            $table->longText('exception');
            $table->timestamp('failed_at')->useCurrent();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('jobs');
        Schema::dropIfExists('job_batches');
        Schema::dropIfExists('failed_jobs');
```

```
    }
};
```

(migrations/ 2024_04_25_233441_create_categories_table.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.
     */
    public function up(): void
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->id();
            $table->string('slug');
            $table->string('name');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
```

```
        Schema::dropIfExists('categories');
    }
};
```

(migrations/2024_04_25_235127_create_subcategories_table.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.
     */
    public function up(): void
    {
        Schema::create('subcategories', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('category');
            $table->string('name');
            $table->string('slug');
            $table->foreign('category')->references('id')->on('categories');
        });
    }

    /**
     * Reverse the migrations.
```

```
    */
    public function down(): void
    {
        Schema::dropIfExists('subcategories');
    }
};
```

(migrations/ 2024_04_26_000714_create_products_table.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.
     */
    public function up(): void
    {
        Schema::create('products', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('subcategory');
            $table->string('name');
            $table->string('slug');
            $table->integer('price');
            $table->integer('discount');
            $table->text('description')->nullable();
```

```
            $table->text('extra')->nullable();
            $table->timestamps();
            $table->foreign('subcategory')->references('id')->on('subcategories')-
>onDelete('cascade');
        });
    }

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

(migrations/ 2024_04_26_001654_create_products_colors_table.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.
     */
    public function up(): void
```

```
        {
            Schema::create('products_colors', function (Blueprint $table) {
                $table->id();
                $table->unsignedBigInteger('product');
                $table->string('name');
                $table->string('code');
                $table->foreign('product')->references('id')->on('products')-
>onDelete('cascade');
            });
        }

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

(migrations/2024_05_22_132552_create_products_images_table.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.
     */
    public function up(): void
    {
        Schema::create('products_images', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('product');
            $table->string('image');
            $table->foreign('product')->references('id')->on('products')-
>onDelete('cascade');
        });
    }

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

(migrations/ 2024_05_26_103233_create_personal_access_tokens_table.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.
     */
    public function up(): void
    {
        Schema::create('personal_access_tokens', function (Blueprint $table) {
            $table->id();
            $table->morphs('tokenable');
            $table->string('name');
            $table->string('token', 64)->unique();
            $table->text('abilities')->nullable();
            $table->timestamp('last_used_at')->nullable();
            $table->timestamp('expires_at')->nullable();
            $table->timestamps();
        });
    }

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

ДОДАТОК Д

(routes/api.php)

```
<?php


use App\Http\Controllers\Api\ProductController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

Route::get('/products/{id}', [ProductController::class, 'get']);
```

(routes/ console.php)

```
<?php

use Illuminate\Foundation\Inspiring;
use Illuminate\Support\Facades\Artisan;

Artisan::command('inspire', function () {
    $this->comment(Inspiring::quote());
})->purpose('Display an inspiring quote')->hourly();
```

(routes/web.php)

```
<?php

use App\Http\Controllers\ContactsController;
```

```
use App\Http\Controllers\MainController;
use App\Http\Controllers\OrderController;
use App\Http\Controllers\ProductController;
use App\Http\Controllers\ProductsController;
use Illuminate\Support\Facades\Route;

//Route::get('/', function () {
//    return view('welcome');
//});

Route::get('/', [MainController::class, 'index']);
Route::get('/{category}/{subcategory}', [ProductsController::class, 'index']);
Route::get('/{category}/{subcategory}/{product_name}',
[ProductController::class, 'index']);
Route::get('/order', [OrderController::class, 'index']);
Route::post('/order/end', [OrderController::class, 'end']);
Route::get('/contacts', [ContactsController::class, 'index']);
```