

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Інтернет магазин одягу»

Виконав студент групи ІПЗ-21-5
КАЛАТИНЕЦЬ Віталій Миколайович

Керівник роботи:
ВАКАЛЮК Тетяна Анатоліївна

Рецензент:
ТОЛСТОЙ Ігор Анатолійович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна Вакалюк

«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КАЛАТИНЕЦЬ Віталій Миколайович**

Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**

Тема роботи: **«Інтернет магазин одягу»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: Об'єктом дослідження є веб-платформа інтернет магазину одягу. Предметом дослідження є програмні засоби та методи розробки функціонального і зручного для користувача онлайн-магазину.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т.А.	27.02.2025р.	27.02.2025р.
2	Вакалюк Т.А.	27.02.2025р.	27.02.2025р.
3	Вакалюк Т.А.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з розробленої веб-платформи «Інтернет-магазин одягу» та пояснювальної записки. Пояснювальна записка містить 70 сторінки, 10 ілюстрацій, 11 таблиць, 4 додатки і 30 використаних джерел літератури.

Метою роботи є створення сучасного онлайн-магазину одягу з інтуїтивно зрозумілим інтерфейсом, системою зручного вибору товарів, інтеграцією способів оплати та доставки, а також базовою системою керування користувачами. Платформа надає користувачам можливість ефективного пошуку товарів за категоріями, ціною, розмірами та брендами, а також швидкого оформлення замовлення.

У реалізації проєкту використано фреймворк Laravel для серверної логіки, шаблонізатор Blade для відображення інтерфейсу, MySQL для зберігання інформації, API Нової пошти для доставки, платіжну систему LiqPay та Bootstrap для стилізації. Упроваджено валідацію форм, захист від CSRF, перевірку підтвердження email та захищене оновлення пароля. Середовище розробки — PhpStorm, розгортання локально здійснювалось через Open Server.

Проєкт демонструє приклад повноцінного функціонального рішення для онлайн-продажу одягу, яке може бути застосоване у реальному бізнесі або масштабоване для великих інтернет-магазинів.

КЛЮЧОВІ СЛОВА: ІНТЕРНЕТ-МАГАЗИН, ОДЯГ, LARAVEL, BLADE, MYSQL, LIQPAY, NOVA POSHTA, BOOTSTRAP, PHPSTORM.

					ІПЗ.КР.Б – 121 – 25 – ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата	Інтернет магазин одягу				Лім.	Арк.	Аркушів
Розроб.		Калатинець В.М.									
Керівник		Вакалюк Т.А.			Пояснювальна записка				Житомирська політехніка, група ІПЗ-21-5		
Рецензент		Толстой І.А.									
Зав. каф.		Вакалюк Т.А.									

ABSTRACT

The bachelor's qualification thesis consists of a developed web platform “Online Clothing Store” and an explanatory note. The explanatory note contains 70 pages, 10 illustrations, 11 tables, 4 appendices, and 30 literature sources.

The aim of the project is to create a modern and functional e-commerce platform for selling clothing online, providing users with an intuitive interface, advanced product filtering, integrated payment and delivery options, and essential user account management. The system allows customers to easily browse items by category, price, size, and brand, and place orders with minimal effort.

The solution was built using the Laravel framework for backend logic, Blade as the templating engine, MySQL for data storage, Nova Poshta API for shipping integration, LiqPay for online payments, and Bootstrap for front-end styling. Key features include form validation, CSRF protection, email verification, and secure password update. Development was carried out in PhpStorm, with local deployment via Open Server.

This project represents a robust and practical implementation of an online clothing store, suitable for real business use or further scaling.

KEYWORDS: ONLINE STORE, CLOTHING, LARAVEL, BLADE, MYSQL, LIQPAY, NOVA POSHTA, BOOTSTRAP, PHPSTORM.

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ	9
1.1. Постановка задачі	9
1.2. Аналіз аналогів програмного продукту	11
1.3. Вибір архітектури веб-орієнтованої системи	15
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	17
Висновки до першого розділу	21
РОЗДІЛ 2. ПРОЄКТУВАННЯ ОНЛАЙН-ПЛАТФОРМИ ІНТЕРНЕТ- МАГАЗИНУ ОДЯГУ	22
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	22
2.2. Розробка бази даних систем	28
2.3. Проєктування та реалізація алгоритмів роботи платформи	33
2.4. Реалізація функціоналу платформи	35
Висновки до другого розділу	37
РОЗДІЛ 3. ОПИС РОЗРОБЛЕНОЇ ІНТЕРНЕТ-ПЛАТФОРМИ ЗООМАГАЗИНУ	39
3.1. Порядок встановлення та налаштування параметрів платформи	39
3.2. Структура інтерфейсу платформи Інтернет-магазину одягу	41
Висновки до третього розділу	45
ВИСНОВКИ	47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	50
ДОДАТКИ	53

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

MVC – Model-View-Controller

CRUD – Create, Read, Update, Delete

ORM – Object-Relational Mapping

CSRF – Cross-Site Request Forgery

UI – User Interface

UX – User Experience

DOM – Document Object Model

JSON – JavaScript Object Notation

PHP – Hypertext Preprocessor

SQL – Structured Query Language

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

JS – JavaScript

ВСТУП

На сучасному етапі розвитку цифрових технологій усе більше сфер людської діяльності переходить у віртуальний простір. Особливо помітним це стало у сфері торгівлі, де інтернет-магазини поступово витісняють традиційні точки продажу. Онлайн-торгівля стала звичним явищем для більшості споживачів — вона дає можливість швидко та зручно обирати необхідні товари, порівнювати ціни, читати відгуки та здійснювати покупки, не виходячи з дому. Серед усіх напрямів електронної комерції інтернет-магазини одягу займають одне з провідних місць, оскільки попит на одяг є постійним і не залежить від сезону чи економічної ситуації.

Актуальність теми зумовлена не лише динамічним розвитком інтернет-торгівлі, а й потребою малого та середнього бізнесу в доступних інструментах для цифрової трансформації. Веб-платформи, які дозволяють створювати функціональні онлайн-магазини з повноцінною системою фільтрації товарів, обробки замовлень, онлайн-оплати та доставки, мають значну практичну цінність. Саме тому розробка інтернет-магазину одягу, що поєднує зручність, безпеку та функціональність, є актуальним і доцільним напрямом дослідження в межах підготовки бакалаврської кваліфікаційної роботи.

Об'єктом дослідження є веб-платформа інтернет-магазину одягу, яка забезпечує повний цикл обслуговування користувачів — від перегляду товарів до оформлення та оплати замовлень. Предметом дослідження є методи, підходи та інструменти створення функціонального веб-додатку для електронної комерції, включаючи реалізацію інтерфейсу, обробку запитів, захист даних, роботу з API доставки та платіжних систем.

Метою даної роботи є створення зручної, функціональної та захищеної веб-платформи для продажу одягу, яка б дозволяла користувачам легко знаходити потрібний товар, швидко його замовляти, оплачувати онлайн та отримувати з доставкою, при цьому гарантуючи безпеку персональних даних і стабільність роботи сервісу. Реалізація такої мети вимагає вирішення низки завдань, серед яких можна виокремити аналіз аналогічних рішень та архітектур

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						7

веб-платформ, обґрунтування вибору технологій, проєктування структури бази даних і інтерфейсу, розробку функціоналу користувача й адміністратора, інтеграцію з платіжними та логістичними API, а також тестування стабільності та зручності користування.

У межах цієї роботи розроблено прототип інтернет-магазину одягу, що передбачає реєстрацію користувачів, авторизацію, перегляд і фільтрацію товарів, вибір розміру, додавання до кошика, оформлення замовлення, інтеграцію з сервісом доставки (Нова пошта) та оплату через платіжну систему (LiqPay). Окрема увага приділена захисту даних, перевірці email-адрес, валідації форм, а також зручності інтерфейсу.

Технічною основою проєкту став фреймворк Laravel, що дозволяє реалізувати архітектурну модель MVC та забезпечити гнучку структуру коду. Веб-інтерфейс створено за допомогою Blade та Bootstrap 5, а для інтерактивності використовувався jQuery. Усі запити на оплату і доставку реалізовано через LiqPay API та Nova Poshta API, що дозволяє проводити реальні транзакції та формувати ТТН. Дані зберігаються у реляційній базі даних MySQL, а розробка велася у середовищі PhpStorm, з використанням Open Server як локального середовища.

Запропоноване рішення орієнтоване на малий і середній бізнес у сфері роздрібної торгівлі одягом, а також на підприємців, які планують запустити власний онлайн-магазин без залучення великих інвестицій у розробку. Така система може бути адаптована під різні сегменти ринку, розширена за потреби, а також розгорнута на реальному хостингу для комерційного використання.

Отже, дана робота поєднує в собі теоретичні знання та практичну реалізацію актуального й прикладного завдання — створення сучасного інтернет-магазину, який відповідає вимогам зручності, безпеки та ефективності.

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

1.1. Постановка задачі

У сучасному інформаційному суспільстві торгівля набуває нових форм, дедалі активніше переходячи у цифрове середовище. Онлайн-магазини стали невід'ємною частиною повсякденного життя мільйонів людей по всьому світу. Сьогодні користувач очікує не просто сторінку з товарами, а повноцінну платформу, що дозволяє комфортно переглядати асортимент, легко знаходити необхідне, швидко оформлювати замовлення і безпечно здійснювати оплату. Саме тому перед розробником постає низка завдань, що виходять далеко за межі простого представлення інформації. Вони охоплюють різні аспекти — від побудови логіки роботи до забезпечення взаємодії з зовнішніми сервісами, таких як служби доставки або платіжні системи.

У межах цієї кваліфікаційної роботи постає завдання розробити інтернет-магазин одягу — платформу, яка б дозволяла кінцевому користувачу зручно і швидко купувати одяг онлайн, водночас забезпечуючи просте й гнучке адміністрування для власника ресурсу. На перший погляд, це може здаватися відносно нескладним процесом. Однак, якщо глянути глибше, то виявляється, що навіть базова реалізація подібного ресурсу потребує вирішення низки критично важливих питань.

Першим і ключовим аспектом у постановці задачі є розуміння кінцевої мети — створити веб-платформу, яка буде функціональною, безпечною, стабільною і при цьому зручною для користувача. Це означає, що необхідно забезпечити логічно продуману структуру каталогу товарів, яка дозволить легко орієнтуватися в асортименті. Користувач повинен мати змогу відсортувати одяг за категоріями, брендами, розмірами, кольорами, а також ціновими діапазонами. Окрім цього, важливо враховувати можливість пошуку товарів за ключовими словами.

Другим не менш важливим завданням є реалізація системи реєстрації та авторизації користувачів. При цьому слід подбати не лише про зручність заповнення форм, але й про захист особистих даних. Користувач повинен мати можливість створити обліковий запис, підтвердити електронну адресу, увійти до системи і при потребі — змінити свій пароль. Також слід передбачити базову логіку керування профілем: оновлення контактних даних, перегляд історії замовлень та відстеження статусів доставки.

Третім напрямком є реалізація кошика та оформлення замовлення. Це — один із найвідповідальніших етапів, адже саме тут відбувається фінальна взаємодія користувача з товаром перед оплатою. Система повинна дозволяти додавати товари до кошика, змінювати їх кількість, видаляти позиції, переглядати підсумкову вартість замовлення. На етапі оформлення замовлення необхідно запропонувати вибір міста, відділення доставки, способу оплати, а також можливість ввести контактні дані одержувача. При цьому особливу увагу слід приділити інтеграції з API Нової пошти, що дозволить автоматично підвантажувати список міст і відділень для уникнення помилок і підвищення точності введення даних.

Наступним логічним кроком у постановці задачі є організація прийому онлайн-платежів. Сьогодні це вже не просто опція, а очікувана функціональність будь-якого інтернет-магазину. Саме тому у цьому проєкті необхідно реалізувати інтеграцію з LiqPay — популярною платіжною системою, що дозволяє приймати платежі з банківських карток. Для цього потрібно налаштувати взаємодію між сайтом і сервісом LiqPay, передавати параметри замовлення, обробляти відповіді API і відображати відповідні повідомлення про успіх чи помилку оплати.

Також варто врахувати вимоги до адміністративної частини проєкту. Адміністратор повинен мати змогу керувати товарами — додавати нові позиції, редагувати існуючі, змінювати їхню ціну, статус наявності, а також переглядати замовлення, що надходять від користувачів. Для цього необхідно реалізувати просту, але ефективну внутрішню панель керування з можливістю виконання основних CRUD-операцій.

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						10

Ще одним важливим моментом у постановці задачі є вибір інструментів і середовища розробки. З урахуванням особливостей проєкту було прийнято рішення використовувати фреймворк Laravel як потужний інструмент для побудови серверної частини, Blade — для генерації шаблонів інтерфейсу, а також Bootstrap і jQuery — для візуальної та інтерактивної частини. Усі дані зберігатимуться в базі даних MySQL, а розробка вестиметься у PhpStorm з використанням Open Server як локального середовища.

Зрештою, ще однією задачею є забезпечення безпеки платформи. Сюди входить реалізація захисту від CSRF-атак, правильне налаштування валідації форм, обмеження доступу до певних маршрутів лише для авторизованих користувачів, а також збереження даних із дотриманням принципів конфіденційності.

У підсумку, перед розробником постає багатокomпонентне завдання — створити не просто сайт, а цілісну, функціональну, сучасну й адаптовану до реальних бізнес-процесів систему електронної комерції. З урахуванням усього вищезазначеного, дана кваліфікаційна робота спрямована на створення повноцінного інтернет-магазину одягу, здатного задовольнити як потреби користувачів, так і вимоги власника бізнесу.

1.2. Аналіз аналогів програмного продукту

Перед початком розробки власного інтернет-магазину одягу необхідно провести аналіз уже існуючих рішень, які активно використовуються на ринку. Такий підхід дозволяє не лише краще зрозуміти очікування цільової аудиторії, але й виявити сильні та слабкі сторони конкурентів. Успішні реалізації подібних систем дають змогу сформувавши чітке уявлення про те, яким повинен бути інтерфейс користувача, як має працювати система замовлень, які функціональні блоки є обов'язковими, а які — бажаними. Також аналіз аналогів дозволяє уникнути повторення типових помилок та неефективних рішень, які могли бути реалізовані в інших проєктах.

Для аналізу було обрано три популярні платформи, які надають користувачам можливість замовляти одяг онлайн: інтернет-магазин Answear.ua,

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						11

платформа Zalando, а також вітчизняний проєкт Intertop.ua. Усі три сайти відрізняються один від одного як візуально, так і функціонально, що дозволяє зробити широку та об’єктивну оцінку.

Answer.ua — один із найпопулярніших онлайн-магазинів одягу в Україні, який пропонує широкий вибір брендів і товарів для чоловіків, жінок та дітей. Головна перевага цієї платформи полягає у простому та швидкому фільтруванні товарів за категоріями, брендами, розмірами та навіть кольорами. Також варто відзначити наявність мобільної адаптації, зручного особистого кабінету, швидкого замовлення та оплати онлайн. Інтерфейс сайту чистий, з великим акцентом на фото товарів та мінімумом відволікаючих елементів. Система лояльності з накопиченням бонусів та частими знижками додатково приваблює клієнтів.

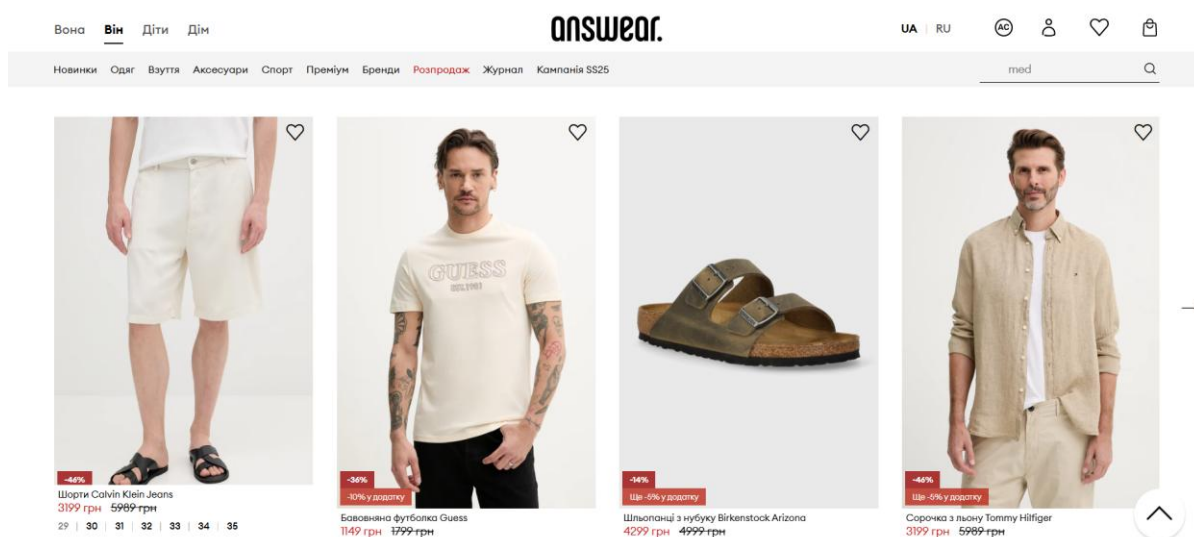


Рис. 1.1 – Головна сторінка інтернет-магазину Answer.ua.

Zalando — європейський гігант у сфері онлайн-торгівлі одягом, взуттям та акcesуарами. Цей ресурс пропонує безкоштовну доставку і повернення товарів, підтримку великої кількості мов та валют, а також гнучку систему фільтрації. Важливою особливістю платформи є персоналізовані рекомендації на основі попередніх переглядів і покупок. Zalando реалізує також складну інтеграцію зі складськими системами, що дозволяє контролювати наявність товарів у режимі реального часу. Крім того, присутній розширений фільтр за еко-брендами, матеріалами, і навіть ступенем відповідності розміру.

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						12

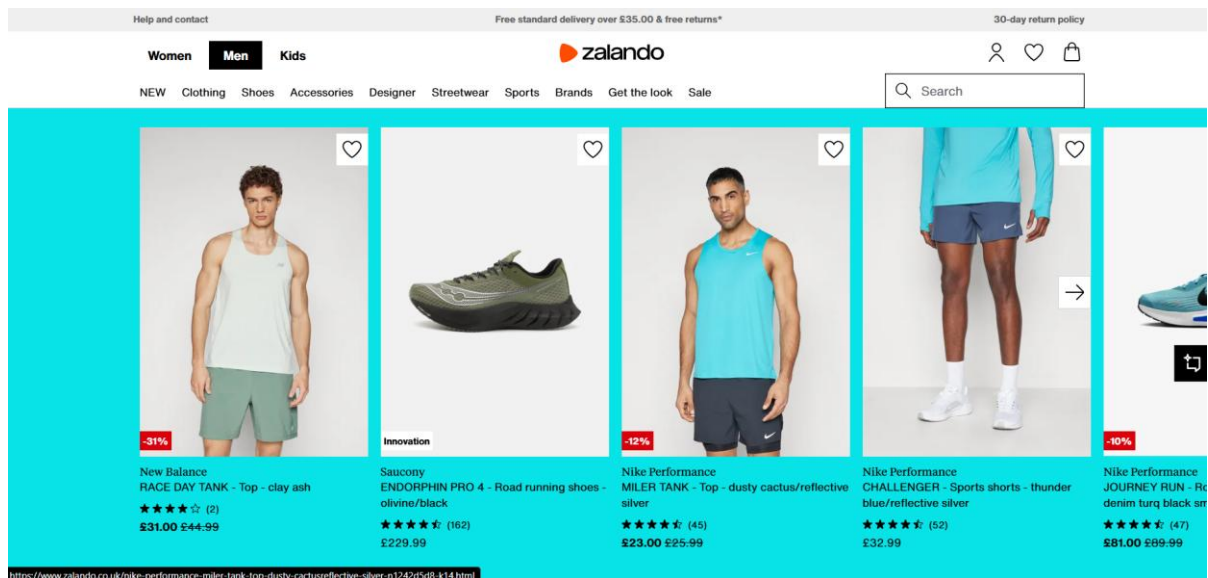


Рис. 1.3 – Інтерфейс пошуку товарів на Zalando.

Intertop.ua — відомий український інтернет-магазин, який поєднує торгівлю одягом, взуттям та аксесуарами. Його головна особливість — це можливість поєднання онлайн-замовлення з офлайн-магазинами, тобто клієнт може замовити товар онлайн і забрати його в найближчому магазині. Крім того, Intertop має зручну систему оформлення замовлення, підключення до українських служб доставки (зокрема, Нової пошти), і підтримує декілька способів оплати, включаючи оплату при отриманні. Інтерфейс дещо перевантажений графічними елементами, проте він адаптований до мобільних пристроїв і підтримує українську та російську мови.

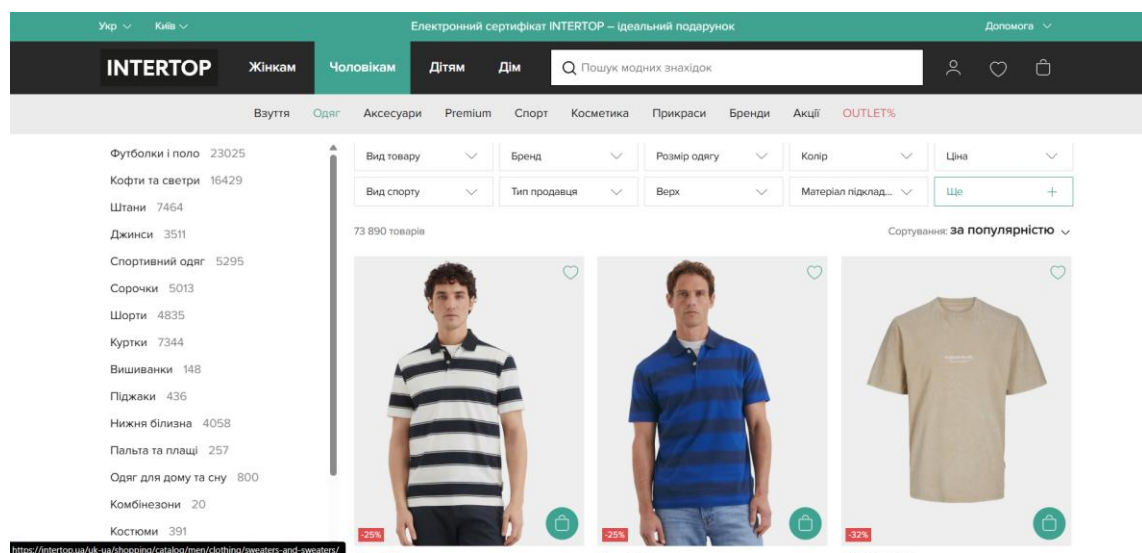


Рис. 1.3 – Головна сторінка інтернет-магазину Intertop.ua з елементами пошуку та фільтрації товарів.

Щоб краще зрозуміти переваги та недоліки цих аналогів, нижче наведено порівняльну таблицю основних характеристик (Таблиця 1.1).

Таблиця 1.1

Порівняння функціональності існуючих інтернет-магазинів одягу

Характеристика	Answer.ua	Zalando	Intertop.ua
Пошук і фільтри	Розширені	Дуже детальні	Середні
Інтеграція з доставкою	Так (Нова пошта)	Так (європейські служби)	Так (Нова пошта, кур'єри)
Оплата онлайн	Так	Так	Так
Персональні рекомендації	Обмежені	Так	Немає
Мобільна адаптація	Так	Так	Так
Система лояльності	Бонуси	Індивідуальні знижки	Частково
Інтеграція з офлайн-магазинами	Немає	Немає	Так
Мультимовність	UA/EN	20+ мов	UA/RU
Інтерфейс	Простий	Мінімалістичний	Перевантажений

Як видно з порівняння, кожен із розглянутих аналогів має свої сильні сторони. Zalando вражає глибиною персоналізації та зручністю для клієнта, Answer.ua надає простий і швидкий доступ до товарів, а Intertop.ua інтегрує онлайн-торгівлю з фізичними магазинами. Однак при цьому не всі з них враховують специфіку українського ринку, зокрема в контексті підтримки локальних служб доставки, гнучкості оплати чи мовної адаптації.

На основі цього аналізу можна зробити висновок, що розробка власної платформи повинна поєднувати переваги кожного з наведених рішень. Доцільно взяти простоту інтерфейсу від Answer.ua, персоналізовані функції від Zalando,

а також гнучкість взаємодії з локальними службами — як у Intertop.ua. Унікальність розроблюваної системи може полягати саме у вдалій адаптації типових рішень до українських реалій, включаючи інтеграцію з API Нової пошти, підтримку LiqPay, багатомовність, мобільну адаптацію та зручне керування товарами з боку адміністратора.

1.3. Вибір архітектури веб-орієнтованої системи

Створення будь-якої сучасної веб-платформи неможливе без чіткого визначення її архітектурної структури. Архітектура системи — це її логічна основа, яка визначає способи організації коду, взаємодію між компонентами, обробку запитів та даних, а також розмежування відповідальності між частинами програми. Вибір правильної архітектури на ранньому етапі розробки має ключове значення, адже саме вона забезпечує масштабованість, зрозумілість структури, можливість подальшої підтримки, розширення функціоналу та адаптацію під змінні вимоги.

У межах реалізації інтернет-магазину одягу було прийнято рішення використовувати архітектурну модель MVC (Model-View-Controller). Ця модель є однією з найпоширеніших і перевірених часом парадигм у розробці веб-додатків. Вона дозволяє чітко розмежувати логіку представлення, логіку бізнес-процесів та обробку запитів. У рамках цієї архітектури модель відповідає за взаємодію з базою даних, контролер — за прийом і обробку запитів користувача, а представлення — за виведення інформації на екран.

Причини вибору саме архітектури MVC криються у кількох аспектах. Насамперед, вона дозволяє зменшити взаємозалежність між компонентами системи, що значно спрощує тестування, відлагодження та оновлення проєкту. Наприклад, змінивши вигляд сторінки або логіку відображення даних у шаблоні, розробник не зачіпає код контролера чи моделі. Це забезпечує гнучкість та стабільність системи. Крім того, розділення на модулі дозволяє працювати кільком розробникам одночасно над різними частинами проєкту, що особливо важливо в командній розробці.

Платформою, яка реалізує архітектуру MVC у межах цього проєкту, обрано Laravel — сучасний PHP-фреймворк, що відзначається високим рівнем абстракції, зручним API, потужними інструментами для роботи з базами даних, системою маршрутизації та вбудованою підтримкою шаблонізатора Blade. Laravel чудово підходить для реалізації середніх і великих проєктів, таких як інтернет-магазини, оскільки має багато вбудованих механізмів для забезпечення безпеки, валідації, роботи з сесіями, авторизації, а також підтримку RESTful-архітектури.

У якості представлення (View) було обрано Blade — вбудований шаблонізатор Laravel, який дозволяє зручно поєднувати HTML-розмітку з динамічними даними. Це дозволяє створити адаптивний інтерфейс для користувача, який змінюється залежно від дій у системі. Інтерфейс побудований із використанням Bootstrap 5, що забезпечує сучасний зовнішній вигляд, кросбраузерну сумісність та адаптацію під мобільні пристрої. Для додаткової інтерактивності застосовано бібліотеку jQuery.

Моделі (Model) у системі відповідають за взаємодію з базою даних MySQL. За допомогою ORM Eloquent, яка є частиною Laravel, усі запити до бази формуються через об'єктно-орієнтований підхід, що значно підвищує безпеку та знижує ймовірність помилок. Наприклад, доступ до даних про товари, категорії, користувачів або замовлення реалізується через окремі класи-моделі, кожен з яких відповідає відповідній таблиці в базі.

Контролери (Controller) виступають проміжною ланкою між запитом користувача та даними. Вони відповідають за прийом запиту, перевірку даних, обробку логіки та передачу результату до представлення. У випадку інтернет-магазину це можуть бути дії на кшталт додавання товару до кошика, фільтрації за параметрами, оформлення замовлення або відправлення підтвердження на email. Завдяки можливості розділяти контролери за функціональними блоками (наприклад, ProductController, OrderController, AuthController), структура коду стає логічною і передбачуваною.

Окремої уваги заслуговує реалізація API-інтеграцій. Веб-орієнтована система має працювати з зовнішніми сервісами, такими як Nova Poshta API — для отримання списку міст і відділень при оформленні замовлення, а також LiqPay — для обробки онлайн-платежів. Ці зовнішні взаємодії реалізовано через окремі сервіси у структурі Laravel, що підтримує принцип розділення відповідальностей. Кожен сервіс відповідає за окремий зовнішній інтерфейс, що дозволяє у майбутньому легко оновлювати або змінювати логіку інтеграції без втручання в основну структуру проєкту.

Таким чином, вибір архітектури MVC у поєднанні з фреймворком Laravel забезпечує зручність підтримки, зрозумілу логіку побудови, легкість масштабування та гнучкість у реалізації додаткового функціоналу. Це дозволяє реалізувати веб-систему, яка не тільки задовольняє початкові функціональні вимоги, але й залишається відкритою для подальшого розвитку, удосконалення та адаптації під нові задачі.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Проектування та розробка сучасного веб-застосунку неможливі без відповідного вибору інструментальних засобів, які забезпечують зручність, продуктивність і надійність усіх етапів розробки — від створення коду до його тестування та запуску в робочому середовищі. У межах цього проєкту, присвяченого створенню інтернет-магазину одягу, використовувалися перевірені та стабільні технології, що поєднують простоту, функціональність та активну підтримку спільноти.

У якості основного фреймворку для серверної частини обрано Laravel. Це сучасний PHP-фреймворк, який реалізує архітектуру MVC та надає розробнику великий набір інструментів для побудови повноцінного веб-застосунку. Laravel має зрозумілу документацію, активну спільноту, вбудовані рішення для роботи з базами даних, чергами, файлами, формами, системами автентифікації та іншими базовими речами. Завдяки системі маршрутів, middleware, об'єктно-орієнтованій реалізації моделей та можливості підключення сторонніх бібліотек

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						17

через Composer, Laravel дозволяє гнучко розробляти і масштабувати веб-додатки будь-якої складності.

Для зберігання даних обрано MySQL, оскільки ця система управління базами даних є однією з найпопулярніших у світі. Вона добре інтегрується з Laravel через ORM Eloquent, забезпечує швидкий доступ до даних, підтримку транзакцій, індексів, зовнішніх ключів та реплікації. MySQL легко встановлюється, має відкритий вихідний код, регулярно оновлюється та не вимагає значних ресурсів для функціонування на середніх навантаженнях.

Для побудови користувацького інтерфейсу використано Blade — вбудований шаблонізатор Laravel. Завдяки йому можна легко передавати дані з контролерів у HTML-шаблони, використовуючи простий та інтуїтивний синтаксис. Blade дозволяє створювати умовні конструкції, цикли, компоненти та макети, що знижує кількість дубльованого коду та прискорює розробку.

У візуальній частині було зроблено ставку на Bootstrap 5, який є найпопулярнішим CSS-фреймворком для побудови адаптивного інтерфейсу. Bootstrap дозволяє без особливих зусиль створювати сучасні, привабливі та мобільно-орієнтовані веб-сторінки. Він включає набір готових компонентів — кнопки, форми, модальні вікна, навігацію — які забезпечують єдиний стиль у всьому проекті.

Для реалізації інтерактивності на клієнтській стороні використовувався jQuery — легка бібліотека JavaScript, що значно спрощує обробку подій, взаємодію з DOM. Зокрема, через jQuery реалізована динамічна підгрузка міст і відділень Нової пошти, а також автоматичне оновлення елементів форми на сторінці оформлення замовлення.

Важливою частиною проекту є інтеграція з API Нової пошти, що дозволяє реалізувати зручну форму доставки товару. Замість ручного введення адреси користувач отримує можливість обрати місто та відділення зі списку, що динамічно підвантажується. Це мінімізує кількість помилок, покращує UX та пришвидшує процес оформлення.

Оплата замовлень реалізується через LiqPay — одну з найпоширеніших платіжних систем в Україні. Цей сервіс дозволяє приймати платежі через банківські картки та Apple Pay, а також забезпечує високий рівень безпеки. Взаємодія з LiqPay відбувається через їхнє API, що легко інтегрується з Laravel, завдяки чому можна швидко створити форму оплати, обробити відповідь і зберегти статус платежу у базі даних.

Проект розроблявся у середовищі PhpStorm, яке є одним із найпотужніших IDE для PHP. Цей інструмент надає підсвічування синтаксису, рефакторинг коду, зручну роботу з базами даних, систему контролю версій, автозаповнення, інтеграцію з Laravel та Blade, що значно підвищує швидкість і якість написання коду. PhpStorm також підтримує Docker, що дозволяє розробникам працювати з контейнерами безпосередньо в межах середовища розробки.

Для локального запуску проекту було використано Open Server — програмний комплекс, що містить Apache/Nginx, PHP, MySQL та інші потрібні інструменти. Цей сервер ідеально підходить для середовища Windows і дозволяє швидко налаштувати робоче середовище без складних конфігурацій. Усі частини системи — база даних, сервер, поштовий агент — запускаються локально, що забезпечує стабільне тестування без необхідності підключення до зовнішніх хостингів.

Щодо вимог до апаратного забезпечення, то для локальної розробки і тестування достатньо комп'ютера середнього рівня. Мінімально необхідні характеристики включають процесор із двома або чотирма ядрами (Intel i5 або AMD Ryzen 3), не менше 8 ГБ оперативної пам'яті, SSD-диск для пришвидшення роботи з файлами та доступу до бази даних, а також стабільне підключення до Інтернету для використання API та зовнішніх бібліотек. Рекомендовано мати також резервну копію проекту, яка зберігається або у хмарному сховищі, або в репозиторії, наприклад, на GitHub або GitLab.

Для продуктивного серверного середовища (продакшену), де планується розміщення платформи після завершення розробки, вимоги залежать від навантаження, однак базовий VPS-сервер із 2 ядрами, 2–4 ГБ оперативної

пам'яті, 40–60 ГБ диску та передвстановленими PHP 8+, MySQL і Nginx або Apache — буде цілком достатнім для початку.

Таким чином, обраний набір інструментальних засобів дозволяє реалізувати усі функціональні вимоги до веб-системи: побудову інтерфейсу, обробку даних, взаємодію з користувачем, оплату та доставку. Він забезпечує швидкість розробки, стабільність роботи та готовність до масштабування у майбутньому.

Окрім згаданих інструментів, у процесі розробки також було використано систему контролю версій Git, що дозволила ефективно відслідковувати зміни в коді, працювати в команді, створювати гілки для нових функціональностей та забезпечувати зручне об'єднання результатів роботи. Репозиторій проєкту розміщено на GitHub, що забезпечує доступ до коду з будь-якого місця, можливість відкочування до попередніх версій та ведення історії розробки.

У процесі тестування застосовувалися як ручні, так і автоматизовані підходи. Laravel надає вбудовані інструменти для написання юніт- та інтеграційних тестів за допомогою PHPUnit. Це дозволяє перевірити коректність бізнес-логіки, взаємодію з базою даних, роботу API та загальну стабільність системи. Ретельне тестування є важливою складовою забезпечення якості та надійності веб-додатку.

Ще одним важливим аспектом став захист даних користувача. У проєкті реалізовано стандартні засоби безпеки: валідація вхідних даних, захист від CSRF-атак, хешування паролів через bcrypt, обмеження доступу до маршрутів за ролями та автентифікація користувачів. Це дозволяє гарантувати, що особисті та платіжні дані залишаються в безпеці.

Таким чином, проєкт є результатом інтеграції сучасних технологій, що забезпечують повноцінну, безпечну та масштабовану платформу для електронної комерції.

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						20

Висновки до першого розділу

У межах першого розділу було проведено всебічний аналіз передумов, рішень та підходів, необхідних для створення сучасної веб-орієнтованої системи, орієнтованої на онлайн-продаж одягу. Ретельно сформульовано основну мету розробки — створити інтернет-магазин, який поєднує зручність для користувачів, ефективність для власника ресурсу та технічну надійність у роботі. Було визначено ключові функціональні завдання, серед яких: каталог товарів з фільтрами, система реєстрації та авторизації користувачів, можливість оформлення замовлення, підключення доставки та онлайн-оплати, а також базове адміністрування.

У ході аналізу аналогічних програмних продуктів було розглянуто приклади трьох популярних інтернет-магазинів — Answer.ua, Zalando та Intertop.ua. Кожен із них має свою специфіку та рівень реалізації функцій. Це дозволило виявити найкращі практики, які можуть бути адаптовані у власному рішенні, а також вказало на типові недоліки, яких бажано уникнути.

Архітектура проєкту побудована на основі моделі MVC, що забезпечує логічну структуру, зрозумілий розподіл відповідальностей між компонентами та можливість подальшого розширення системи. Використання Laravel як основного фреймворку є обґрунтованим з погляду підтримки безпеки, інтеграції з базою даних, ефективності обробки запитів і загальної стабільності розробки.

Підбір інструментальних засобів, таких як Laravel, Blade, Bootstrap, jQuery, MySQL, Nova Poshta API, LiqPay, PhpStorm і Open Server, дозволяє реалізувати усі поставлені функціональні вимоги з урахуванням сучасних стандартів розробки. Крім того, проаналізовано технічні вимоги до апаратного забезпечення як для локальної розробки, так і для можливого розгортання на сервері.

Усе це створює надійне підґрунтя для подальшої реалізації програмної частини проєкту в наступних розділах.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ОНЛАЙН-ПЛАТФОРМИ ІНТЕРНЕТ-МАГАЗИНУ ОДЯГУ

Проектування є одним із найважливіших етапів у створенні будь-якої програмної системи, оскільки саме на цьому етапі формуються фундаментальні рішення щодо структури, логіки та майбутньої реалізації. Від того, наскільки грамотно буде спроектовано архітектуру, маршрути, базу даних і логіку взаємодії компонентів, залежить ефективність і стабільність роботи всієї системи. У межах даного розділу здійснюється проектування онлайн-платформи інтернет-магазину одягу — від структури бази даних до моделі обробки запитів, валідації та бізнес-логіки.

Мета цього етапу полягає в тому, щоб створити логічну, зручну та масштабовану модель роботи майбутнього застосунку, яка б максимально враховувала потреби як кінцевого користувача, так і адміністратора ресурсу. Усі складові проектування — починаючи з вибору назви таблиць у базі даних і закінчуючи логікою оплати — повинні відповідати сучасним стандартам веб-розробки, враховувати потреби безпеки та адаптуватися до особливостей українського ринку.

У цьому розділі буде розглянуто структура даних, взаємодія між таблицями, реалізація ключових модулів, таких як реєстрація користувачів, оформлення замовлення, інтеграція з API доставки та оплати. Окрему увагу приділено опису механізмів валідації даних, побудові маршрутів, організації шаблонів представлення та взаємодії з фронтендом.

Таким чином, розділ 2 є логічним продовженням аналітичної частини дипломної роботи та містить конкретні технічні рішення, які лежать в основі майбутньої реалізації інтернет-магазину одягу.

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Перед безпосередньою реалізацією веб-системи необхідно провести аналіз і проектування її поведінки з точки зору взаємодії користувачів із системою.

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						22

Визначення варіантів використання (use cases) дозволяє описати, як саме користувачі будуть взаємодіяти з платформою, які операції вони зможуть виконувати та які наслідки матимуть ті чи інші дії. Цей підхід дозволяє не лише структурувати логіку, але й закласти підґрунтя для створення об'єктно-орієнтованої архітектури системи.

Об'єктно-орієнтована модель — це концепція, яка дозволяє розглядати систему як сукупність об'єктів, кожен з яких має власний стан та поведінку. У межах реалізації інтернет-магазину одягу такими об'єктами є користувачі, товари, замовлення, коментарі, категорії, розміри, кольори, доставка, тощо. Визначення їхніх ролей, зв'язків і поведінки дозволяє чітко зрозуміти структуру системи ще до початку її реалізації.

Першим ключовим варіантом використання є процес реєстрації та авторизації. Цей процес є початковою точкою взаємодії будь-якого користувача з платформою. Користувач має змогу створити обліковий запис, надавши мінімальний набір обов'язкових даних: ім'я, електронну адресу та пароль. Після заповнення форми на вказану адресу надсилається лист із підтвердженням — тільки після верифікації email користувач отримує повний доступ до функціональності системи. Авторизація виконується за допомогою пари email і пароль. У випадку неправильно введених даних система інформує користувача про помилку, а при потребі можна скористатися механізмом відновлення пароля.

Таблиця 2.1

Варіант використання: реєстрація та авторизація

Актор	Користувач
Мета	Отримати доступ до особистого кабінету
Передумови	Відсутність акаунта для реєстрації або наявність — для входу
Опис	Користувач заповнює форму, підтверджує email, авторизується
Результат	Успішна авторизація або верифікація облікового запису

Наступним функціональним блоком є керування профілем користувача. Після входу до системи користувач отримує доступ до сторінки профілю, де може змінити персональні дані. До них належать прізвище, ім'я, по батькові, номер телефону, а також інформація про місто і відділення Нової пошти. Крім цього, у профілі доступна можливість змінити пароль, а також повністю видалити акаунт. Видалення облікового запису призводить до повного стирання персональних даних і втрачених замовлень, тому ця дія потребує повторного підтвердження.

Таблиця 2.2

Варіант використання: керування профілем користувача

Актор	Авторизований користувач
Мета	Оновити або видалити особисті дані
Передумови	Успішна авторизація
Опис	Користувач редагує ПІБ, телефон, дані доставки; змінює пароль або видаляє акаунт
Результат	Дані оновлено або акаунт видалено

Окремим важливим сценарієм є перегляд каталогу товарів із можливістю фільтрації. Каталог повинен бути зручним у навігації, дозволяти користувачу сортувати та фільтрувати товари за ключовими характеристиками: назвою, категорією, кольором, розміром, а також ціновим діапазоном. Це дозволяє значно полегшити пошук необхідного товару в умовах великого асортименту.

Таблиця 2.3

Варіант використання: перегляд і фільтрація товарів

Актор	Гість або зареєстрований користувач
Мета	Знайти потрібний товар
Передумови	Наявність товарів у системі
Опис	Користувач обирає фільтри, система показує відповідні товари
Результат	Список фільтрованих товарів

Процес оформлення замовлення — один із найважливіших етапів усього функціоналу. Система повинна забезпечувати просту та логічну послідовність дій. Якщо користувач заповнив профіль — дані автоматично підтягуються у форму замовлення. У разі відсутності таких даних, користувач має можливість заповнити їх вручну. Після цього він обирає відділення доставки та переходить до оплати. Оформлення завершується підтвердженням, і замовлення потрапляє в систему.

Таблиця 2.4

Варіант використання: оформлення замовлення

Актор	Авторизований користувач
Мета	Купити один або кілька товарів
Передумови	Наявність товарів у кошику
Опис	Користувач підтверджує або вводить дані, обирає доставку, здійснює оплату
Результат	Замовлення збережено у базі даних

Додатковою функціональністю, яка сприяє взаємодії між клієнтами і збільшує довіру до магазину, є система коментарів. Авторизований користувач може залишати відгуки до товарів, редагувати або видаляти їх. Це дає змогу підтримувати актуальність відгуків. Адміністратор платформи має повний контроль над усіма коментарями, зокрема може їх видаляти у випадку порушення правил.

Таблиця 2.5

Варіант використання: коментарі до товарів

Актор	Користувач / Адміністратор
Мета	Надати або модерувати відгук
Передумови	Авторизація (для користувача), права доступу (для адміністратора)

Продовження таблиці 2.5

Опис	Користувач додає або змінює коментар, адмін може його видалити
Результат	Коментар опубліковано або видалено

Фінальним і найбільш критичним для адміністрування проєкту є функціонал адмін-панелі. У цій частині платформи адміністратор має змогу створювати, редагувати та видаляти категорії, кольори, розміри, а також додавати нові товари. Окрім цього, він може переглядати надіслані замовлення, контролювати їхній статус і взаємодіяти з клієнтами за потреби. Доступ до цієї панелі має лише користувач з відповідними правами, що гарантує безпеку керування контентом магазину.

Таблиця 2.6

Варіант використання: адміністративна панель

Актор	Адміністратор
Мета	Керувати даними в системі
Передумови	Авторизація адміністратора
Опис	Адмін додає, редагує, видаляє сутності (категорії, товари), переглядає замовлення
Результат	Дані системи оновлено

Усі розглянуті варіанти використання не лише описують зовнішню поведінку системи з точки зору користувача, але й тісно пов'язані з внутрішньою структурою веб-додатку, яка реалізується за допомогою об'єктно-орієнтованого програмування. Кожна бізнес-сутність системи (наприклад, товар, користувач, замовлення) реалізується як окрема модель, що відображає відповідну таблицю

в базі даних. Саме завдяки цим моделям система забезпечує збереження, обробку та взаємодію з даними.

Сучасний фреймворк Laravel, який було обрано для реалізації цього проєкту, пропонує зручний механізм ORM — Eloquent, що дозволяє працювати з моделями як з об'єктами, не використовуючи вручну SQL-запити. Це значно спрощує розробку та покращує читабельність коду. Наприклад, модель User має зв'язок hasMany з моделями Order та Comment, що логічно відображає ситуацію, коли один користувач може мати декілька замовлень і залишати кілька відгуків.

Аналогічно, модель Product належить до Category, має багато Comment і може бути пов'язана через проміжну таблицю з Size та Color, якщо реалізується система варіантів товару. Такий підхід дозволяє масштабувати систему у майбутньому, наприклад, додавши багатомовність, систему знижок, історію змін або кешування популярних товарів. Розширення структури не вимагатиме кардинальної зміни всього коду — достатньо буде додати нову модель або зв'язок.

Крім моделей, у системі також важливу роль відіграють контролери — класи, що відповідають за обробку запитів користувача. Наприклад, контролер OrderController приймає дані з форми, перевіряє їхню коректність через механізм валідації, створює нове замовлення та перенаправляє користувача на сторінку підтвердження. Така побудова дозволяє чітко відокремити логіку обробки від логіки збереження та відображення, що і є однією з головних цілей об'єктно-орієнтованого програмування.

Об'єктно-орієнтована структура проєкту також значно полегшує тестування та налагодження. Завдяки тому, що кожен об'єкт є ізольованою сутністю, можна перевіряти його роботу окремо, створювати мок-дані та легко відстежувати помилки. У майбутньому це забезпечує кращу підтримку коду та мінімізує ризики при розширенні функціоналу.

У підсумку можна зазначити, що визначення варіантів використання в поєднанні з об'єктно-орієнтованою структурою дає можливість побудувати логічну, масштабовану та підтримувану систему. Такий підхід забезпечує

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						27

чіткість у реалізації, передбачуваність у поведінці та гнучкість для подальшого розвитку проєкту.

2.2. Розробка бази даних систем

Однією з фундаментальних складових будь-якої веб-платформи є її база даних. Саме вона відповідає за зберігання та структуровану обробку всіх даних, які надходять у процесі взаємодії користувача із системою. Під час розробки інтернет-магазину одягу особливу увагу було приділено правильному проектуванню бази даних, оскільки вона повинна враховувати великий обсяг змінних даних: користувачів, товари, замовлення, коментарі, атрибути товарів (розмір, колір), а також взаємозв'язки між цими сутностями.

Основним принципом, на якому базувалося проєктування, є нормалізація — поділ даних на логічні частини з метою зменшення дублювання та забезпечення цілісності. У структурі бази чітко розділено сутності: окремо зберігаються дані про користувачів, товари, категорії, атрибути, замовлення, коментарі, та відповідні допоміжні зв'язки. Це дозволяє легко масштабувати систему, додавати нові функції, забезпечувати ефективну індексацію та швидкий доступ до потрібної інформації.

На рисунку (Рис. 2.1) нижче наведено загальну структуру бази даних системи, реалізовану відповідно до стандартів реляційного моделювання.

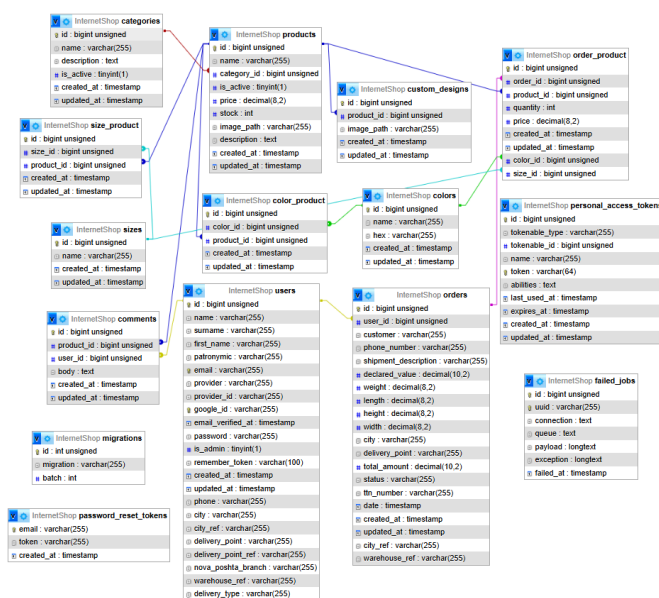


Рис. 2.1 – Схема бази даних онлайн-магазину одягу

Однією з ключових таблиць є таблиця users, яка відповідає за збереження облікових записів користувачів. Вона містить як основні поля для авторизації (електронна пошта, пароль, підтвердження email), так і додаткову інформацію, пов'язану з доставкою (місто, відділення, номер телефону). Також таблиця передбачає підтримку OAuth-провайдерів (Google), що може бути використано для входу через соціальні мережі. Завдяки полям is_admin і email_verified_at система може розрізняти ролі користувачів і керувати доступом до адміністративних функцій.

Таблиця 2.1

Структура таблиці users

Назва поля	Тип даних	Призначення
id	bigint UNSIGNED	Унікальний ідентифікатор користувача
name	varchar(255)	Ім'я користувача
surname	varchar(255)	Прізвище користувача
first_name	varchar(255)	Ім'я (як окреме поле)
patronymic	varchar(255)	По батькові
email	varchar(255)	Електронна адреса користувача
provider	varchar(255)	OAuth-провайдер (Google, Facebook тощо)
provider_id	varchar(255)	ID користувача у зовнішньому провайдері
google_id	varchar(255)	Google ID, якщо автентифікація через Google
email_verified_at	timestamp	Час підтвердження електронної пошти
password	varchar(255)	Хеш пароля
is_admin	tinyint(1)	Прапорець адміністратора (1 – адмін)

Продовження таблиці 2.1

remember_token	varchar(100)	Токен для автологіну
phone	varchar(255)	Телефон користувача
city	varchar(255)	Місто доставки
city_ref	varchar(255)	Ref міста (Nova Poshta API)
delivery_point	varchar(255)	Назва відділення
delivery_point_ref	varchar(255)	Ref відділення (Nova Poshta API)
nova_poshta_branch	varchar(255)	Текстовий опис відділення
warehouse_ref	varchar(255)	Унікальний код відділення
delivery_type	varchar(255)	Тип доставки (відділення, кур'єр)
created_at	timestamp	Дата створення
updated_at	timestamp	Дата оновлення

Ще однією важливою таблицею є orders. Вона містить повний набір інформації, необхідний для обробки замовлення: контактні дані покупця, обрані параметри доставки, загальна сума, габарити посилки, а також номер ТТН після формування через API Нової пошти. Поля city_ref і warehouse_ref є посиланнями на внутрішні довідники Нової пошти, які необхідні для коректної генерації електронної накладної.

Таблиця 2.2

Структура таблиці orders

Назва поля	Тип даних	Призначення
id	bigint UNSIGNED	Унікальний ідентифікатор замовлення
user_id	bigint UNSIGNED	Зовнішній ключ на користувача

Продовження таблиці 2.2

customer	varchar(255)	ПІБ замовника
phone_number	varchar(255)	Телефон для зв'язку
shipment_description	varchar(255)	Опис відправлення (для ТТН)
declared_value	decimal(10,2)	Оголошена вартість посилки
weight	decimal(8,2)	Вага посилки
length	decimal(8,2)	Довжина упаковки
height	decimal(8,2)	Висота упаковки
width	decimal(8,2)	Ширина упаковки
city	varchar(255)	Місто доставки
delivery_point	varchar(255)	Відділення доставки
city_ref	varchar(255)	Ref міста (Nova Poshta API)
warehouse_ref	varchar(255)	Ref відділення (Nova Poshta API)
total_amount	decimal(10,2)	Сума до оплати
status	varchar(255)	Статус замовлення (new, paid, sent тощо)
ttn_number	varchar(255)	Номер ТТН (якщо створено)
date	timestamp	Дата оформлення замовлення
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення

Таблиця products є центральною у системі торгівлі. Вона містить основні атрибути товару: назву, категорію, ціну, залишок на складі, фото, опис. Через зовнішні таблиці товар може бути зв'язаний із розмірами, кольорами, а також

користувачькими дизайнами або варіантами. Наявність поля is_active дозволяє швидко змінювати статус товару без його видалення з бази.

Таблиця 2.3

Структура таблиці products

Назва поля	Тип даних	Призначення
id	bigint UNSIGNED	Унікальний ідентифікатор товару
name	varchar(255)	Назва товару
category_id	bigint UNSIGNED	Посилання на категорію
is_active	tinyint(1)	Статус (1 – активний, 0 – прихований)
price	decimal(8,2)	Ціна товару
stock	int	Кількість товару на складі
image_path	varchar(255)	Шлях до фото товару
description	text	Опис товару
created_at	timestamp	Дата створення товару
updated_at	timestamp	Дата останнього оновлення

Структура бази даних побудована таким чином, щоб дозволити створення зв'язків один-до-багатьох (наприклад, користувач може мати кілька замовлень) і багато-до-багатьох (товари можуть мати кілька кольорів і розмірів). Для цього використовуються проміжні таблиці, такі як size_product, color_product, order_product.

У підсумку, розроблена база даних забезпечує повноцінну підтримку всіх функцій інтернет-магазину, включаючи авторизацію, керування профілем, додавання товарів, фільтрацію, оформлення замовлення, відгуки, а також адміністрування. Така структура є гнучкою, надійною і відкритою до подальшого розширення.

2.3. Проєктування та реалізація алгоритмів роботи платформи

Проєктування алгоритмів — це серце логіки будь-якої системи. Навіть найгарніший інтерфейс не буде мати цінності, якщо за ним не стоятиме зрозуміла та надійна система обробки дій користувача. У випадку інтернет-магазину одягу важливо передбачити, яким чином відбуваються основні процеси: від реєстрації користувача до оформлення замовлення, взаємодії з доставкою й оплатою. Усе це охоплюється набором алгоритмів, які повинні бути чіткими, логічними та максимально автоматизованими.

Кожен із базових сценаріїв платформи реалізовано у вигляді окремого маршруту з відповідним контролером у Laravel. Наприклад, коли користувач заповнює форму замовлення, система спершу перевіряє, чи авторизований він. Якщо так — підтягуються збережені у профілі дані (контактна інформація, місто, відділення). Усі поля, які залишаються порожніми, користувач заповнює вручну. Далі відбувається валідація даних, і, якщо все коректно, формується нове замовлення.

Алгоритм оформлення замовлення поєднує в собі декілька важливих частин. По-перше, це взаємодія з базою даних: збереження основних атрибутів замовлення — ПІБ, номер телефону, місто, доставка, загальна сума. По-друге, створення зв'язку між користувачем і конкретними товарами, які він замовив. По-третє, інтеграція з API Нової пошти: після створення замовлення формується запит на отримання ТТН (номер накладної), який повертається у відповідь і зберігається в полі `ttn_number`. Якщо API повертає помилку, система не зберігає замовлення до усунення проблеми.

Нижче представлено діаграму послідовності, яка демонструє послідовність дій під час оформлення замовлення авторизованим користувачем:

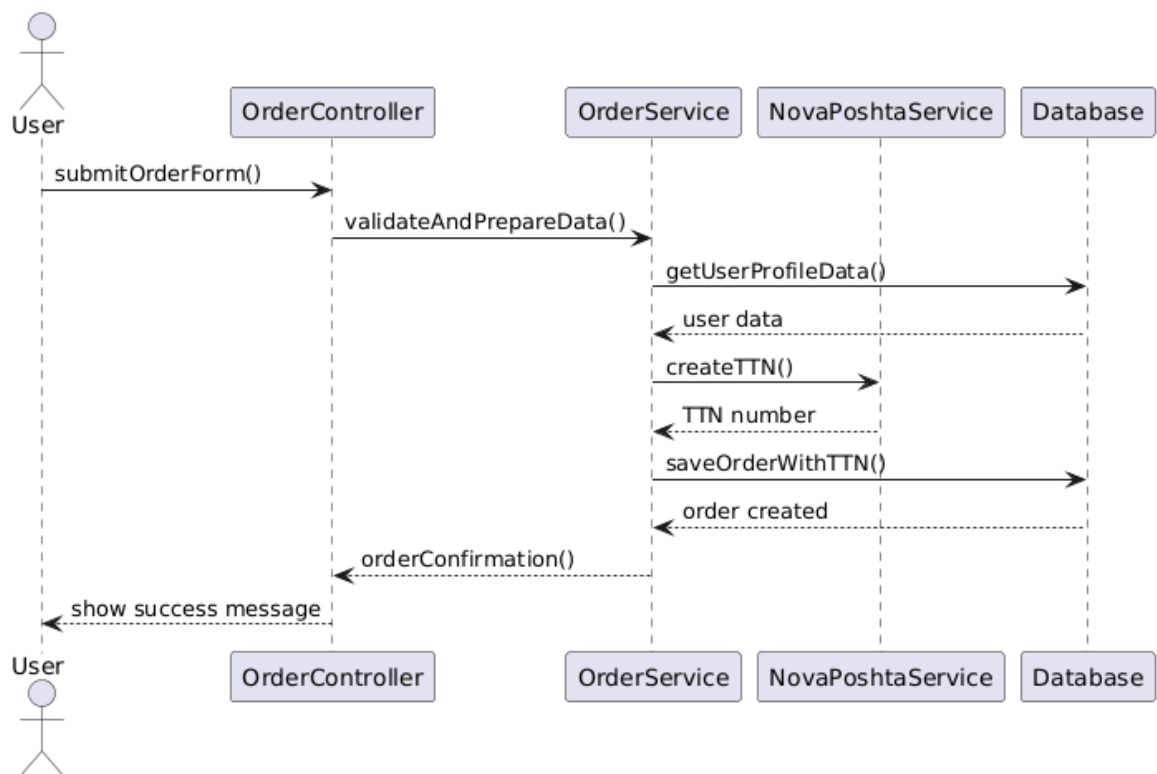


Рис. 2.2 – Діаграма послідовності оформлення замовлення

У цьому сценарії видно, як система виконує серію перевірок і взаємодій. Користувач взаємодіє лише з одним маршрутом, але під капотом система виконує декілька асинхронних або послідовних дій: валідацію, роботу з базою, звернення до зовнішнього API. Така побудова дозволяє централізувати логіку в окремих сервісах, таких як NovaPoshtaService, які легко тестувати та підтримувати.

Окрім процесу замовлення, подібним чином реалізовано інші сценарії. Наприклад, алгоритм реєстрації користувача включає створення облікового запису, відправлення листа підтвердження та очікування активації. Тільки після цього користувач отримує доступ до функцій магазину. Зміна пароля або контактної інформації — окремі запити, які також проходять валідацію, перевірку прав доступу та оновлення у базі.

Особливо важливою є поведінка адміністративної частини платформи. Усі дії адміністратора — створення категорій, редагування товарів, перегляд замовлень — захищені middleware, які не дозволяють доступ без відповідного рівня прав. У контролерах обробка таких дій винесена у сервісні класи, що

зберігає архітектурну чистоту й дозволяє масштабувати адмінку без дублювання коду.

Таким чином, алгоритми платформи реалізовані у відповідності до принципів SOLID і MVC. Усі ключові бізнес-процеси поділено на незалежні логічні блоки: перевірка даних, обробка бізнес-логіки, робота з API, збереження в базі. Завдяки цьому досягається стабільна й передбачувана поведінка системи навіть за змінних умов.

2.4. Реалізація функціоналу платформи

Після етапів проектування, аналізу архітектури та структурування бази даних переходять до безпосередньої реалізації функціоналу системи. На цьому етапі вся логіка, що раніше була описана у вигляді сценаріїв та алгоритмів, починає втілюватися в реальні маршрути, контролери, шаблони та сервіси. Основне завдання розробника — забезпечити не лише працездатність платформи, але й її стійкість до помилок, зручність для користувачів, адаптивність і захищеність.

Система реалізована на основі фреймворку Laravel, який дозволяє організувати структуру у відповідності до принципів MVC. Вся функціональність розподілена по контролерах: SocialiteController, ProfileController, OrderController, ProductController тощо. Кожен із цих контролерів обслуговує певну частину маршруту і відповідає за логіку взаємодії з відповідними моделями та представленнями.

Реєстрація користувачів реалізована за допомогою стандартного Laravel Auth із підтримкою підтвердження електронної пошти. Після заповнення форми користувачеві надсилається лист з посиланням для активації акаунта. Тільки після підтвердження email користувач отримує доступ до функціональності — перегляду профілю, оформлення замовлень, залишення коментарів. Такий підхід дозволяє захистити платформу від спаму та неактуальних акаунтів.

Авторизовані користувачі мають доступ до сторінки профілю, де реалізовано можливість змінювати персональні дані — ім'я, прізвище, номер телефону, місто, відділення Нової пошти. Оновлення відбувається через форму,

в якій валідація виконується як на клієнтській, так і на серверній стороні. Крім цього, доступна функція зміни пароля та можливість повного видалення акаунта. Ці дії виконуються тільки після повторного підтвердження дій користувача, що підвищує рівень безпеки.

Усі товари на платформі виводяться на головній сторінці та в каталозі з можливістю фільтрації. Фільтри реалізовано через параметри в URL, що дозволяє зберігати стан сортування при переході між сторінками. Пошук працює за назвою, категорією, кольором, розміром і ціновим діапазоном. При виборі кількох фільтрів одночасно результати оновлюються динамічно. Сторінки товарів містять фотографії, опис, доступні розміри та кольори.

Форма замовлення передбачає заповнення контактної інформації, вибір міста та відділення Нової пошти. Якщо користувач вже вказав ці дані в профілі — вони автоматично підтягуються у форму. При зміні міста відбувається динамічне оновлення доступних відділень за допомогою Select2 до Nova Poshta API. Після заповнення всіх полів і натискання кнопки «Оформити замовлення» дані проходять серверну валідацію, і тільки після успішної обробки викликається метод створення ТТН. Якщо API повертає валідний номер накладної, дані зберігаються в таблицю orders, а користувач бачить повідомлення про успішне оформлення.

Інтеграція з LiqPay дозволяє користувачам оплачувати замовлення одразу після підтвердження. Laravel формує динамічну кнопку оплати з параметрами суми, опису та callback-посиланням для підтвердження статусу платежу. Після завершення оплати LiqPay переадресовує користувача назад на сайт, і статус замовлення оновлюється до «сплачено».

Система коментарів реалізована просто та зручно: будь-який авторизований користувач може залишити коментар до товару, змінити його або видалити. Адміністратор бачить усі коментарі, незалежно від їх автора, і може видалити будь-який у разі порушення правил чи залишення некоректної інформації. Усі коментарі зберігаються в таблиці comments, пов'язуючись через зовнішній ключ із конкретним товаром.

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						36

Окремо реалізовано адмін-панель, доступ до якої мають лише користувачі з прапорцем `is_admin = 1`. У цій панелі адміністратор може створювати нові категорії, розміри, кольори, додавати товари, редагувати їх описи, завантажувати фотографії. Для перегляду замовлень створено окрему вкладку, де можна бачити усі замовлення, їх статус, дату, номер ТТН і контактні дані клієнтів. Це дозволяє оперативно реагувати на нові покупки, обробляти їх і формувати відправлення.

Для реалізації всіх функцій використовуються окремі сервіси, що відповідають за бізнес-логіку (наприклад `NovaPoshtaService`, `LiqPayService`). Така структура дозволяє зберігати контролери максимально «чистими», переміщуючи основну логіку у відповідні класи, які легко тестуються та змінюються. Усі запити захищено middleware, CSRF-токенами, валідацією форм і правами доступу.

Таким чином, реалізований функціонал платформи повністю відповідає попередньому проєктному етапу. Всі ключові сценарії: авторизація, перегляд товарів, фільтрація, оформлення замовлень, оплата, коментування й адміністрування — реалізовано з дотриманням стандартів Laravel і принципів безпечної веб-розробки. Платформа є завершеним, працездатним рішенням, готовим до запуску в реальних умовах.

Висновки до другого розділу

У другому розділі було детально розглянуто процес проєктування онлайн-платформи інтернет-магазину одягу. Робота над цим етапом відіграє критично важливу роль у створенні стабільного, логічного та розширюваного програмного забезпечення. Від правильного проєктування залежить не лише зручність використання системи, а й її подальше обслуговування, масштабування, адаптація до нових вимог, а також безпека і швидкодія.

На початковому етапі було сформовано основні сценарії використання, які враховують всі типові взаємодії користувача з платформою: реєстрація, авторизація, керування профілем, перегляд товарів, застосування фільтрів, створення замовлення, залишення коментарів та адміністрування магазину. Всі

ці дії були описані у вигляді варіантів використання, що дозволило чітко визначити функціональні вимоги до кожного модуля.

Далі було проєктовано структуру бази даних з урахуванням реляційних зв'язків між сутностями. Основна увага була приділена таблицям users, orders і products, які відіграють ключову роль у системі. Ретельне нормалізоване проєктування дозволяє уникнути дублювання даних, зменшити навантаження на систему, а також спростити подальший аналіз інформації. Розроблена база є достатньо гнучкою для подальшого розширення, наприклад, додавання нових атрибутів товару чи способів доставки.

У підпункті, присвяченому алгоритмам роботи платформи, було продемонстровано, яким чином реалізовано взаємодію між користувачем, внутрішніми сервісами та зовнішніми API. Особливу увагу приділено послідовності дій при оформленні замовлення: від валідації введених даних до створення ТТН через Nova Poshta API та обробки платежів через LiqPay. Для демонстрації було побудовано діаграму послідовності, яка відображає реальні виклики між модулями системи.

Результатом цього розділу стала реалізація повноцінного функціонального ядра платформи. Реєстрація, фільтрація, робота з профілем, оформлення замовлень, інтеграція з доставкою та оплатою, система коментарів і адміністрування — всі ці модулі реалізовано відповідно до сучасних стандартів. Така структура забезпечує не лише стабільність роботи, але й відкритість до вдосконалення.

Розділ 2 фактично заклав технічний фундамент платформи та довів, що обрані підходи та засоби проєктування є ефективними для реалізації повнофункціонального інтернет-магазину одягу.

РОЗДІЛ 3. ОПИС РОЗРОБЛЕНОЇ ІНТЕРНЕТ-ПЛАТФОРМИ ЗООМАГАЗИНУ

3.1. Порядок встановлення та налаштування параметрів платформи

Після завершення етапів проєктування, розробки та тестування функціоналу, платформа інтернет-магазину одягу переходить у фазу встановлення та налаштування. Це важливий етап, який забезпечує коректну роботу всієї системи на реальному сервері або локальному середовищі. Для того щоб платформа працювала стабільно, необхідно виконати серію послідовних дій, які включають встановлення залежностей, створення бази даних, налаштування середовища, генерацію ключів безпеки, запуск міграцій та конфігурацію API-сервісів.

У процесі розробки для локального розгортання використовувалася зручна програма Open Server, яка забезпечує запуск PHP, MySQL, Apache або Nginx без складних інсталяцій. Завдяки цьому всі компоненти системи були об'єднані в одному контрольованому середовищі. Це дозволяє як розробнику, так і майбутньому адміністратору сайту відтворити умови реального хостингу навіть без доступу до інтернету. Запуск локального проєкту передбачає старт Open Server, після чого проєкт доступний за локальним доменом internet-shop.com.

Після того як середовище встановлено, виконується копіювання вихідного коду платформи до папки domains або www. Для коректної роботи Laravel-платформи потрібна наявність залежностей, які встановлюються через менеджер пакетів Composer. Однією з перших команд, яку необхідно виконати у терміналі, є `composer install`, що забезпечує завантаження всіх необхідних бібліотек — починаючи від базових компонентів Laravel і закінчуючи бібліотеками для API-інтеграцій з LiqPay та Nova Poshta.

Одночасно з цим створюється копія конфігураційного файлу `.env.example`, який перейменовується у `.env`. У цьому файлі задаються всі конфіденційні параметри: дані для з'єднання з базою даних, ключ API Нової пошти, публічний і приватний ключі для LiqPay, параметри поштового сервісу для надсилання

листів, посилання на фронтенд, тип доставки за замовчуванням, а також інші змінні середовища. У цьому ж файлі можна визначити локальний режим роботи, встановити валюту та часовий пояс, обрати тип кешування.

Щоб забезпечити безпеку проєкту, після налаштування `.env` генерується унікальний ключ додатку за допомогою команди `php artisan key:generate`. Цей ключ використовується для шифрування сесій і токенів. Далі виконується налаштування бази даних. У MySQL створюється нова база, наприклад, `clothing_shop`, а доступ до неї прописується у змінних `DB_DATABASE`, `DB_USERNAME` та `DB_PASSWORD`. Потім запускаються міграції (`php artisan migrate`) — команда, що створює всі необхідні таблиці відповідно до попередньо спроектованої структури.

Для наповнення бази початковими даними — наприклад, категоріями товарів, розмірами, кольорами або тестовими товарами — використовуються сидери (`php artisan db:seed`). У рамках проєкту також реалізовано механізм генерації адміністратора, тому після сидів можна увійти в адмін-панель із тестовими обліковими даними. Це дає змогу відразу перевірити функціональність додавання товарів, перегляду замовлень і керування атрибутами.

Окрему увагу приділено конфігурації API-інтеграцій. Для коректної взаємодії з Nova Poshta API потрібно у `.env` вказати ключ, отриманий у кабінеті користувача Нової пошти. Те саме стосується платіжної системи LiqPay: вона потребує ключів для генерації платіжних форм і перевірки транзакцій. Після встановлення цих параметрів система готова до повноцінної взаємодії з реальними зовнішніми сервісами.

Таким чином, порядок встановлення системи є логічним і чітко структурованим. Завдяки використанню Laravel та Open Server, процес розгортання значно спрощено й формалізовано, що зменшує кількість людських помилок і прискорює запуск платформи як у локальному середовищі, так і на віддаленому сервері. Це робить систему не лише ефективною з точки зору коду, але й зручною для супроводу, підтримки та відтворення.

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						40

3.2. Структура інтерфейсу платформи Інтернет-магазину одягу

Інтерфейс користувача є ключовим елементом платформи, що відповідає за перше враження, зручність взаємодії та загальне сприйняття системи. Саме завдяки інтерфейсу платформа стає доступною, інтуїтивною і дружньою до клієнта. У випадку інтернет-магазину одягу важливо було забезпечити поєднання простоти, естетики та повної функціональності. Розробка інтерфейсу відбувалась із урахуванням сучасних UX/UI-принципів, а також адаптації під мобільні пристрої.

Початкова сторінка платформи відкривається з промоційного банера, який задає настрій усього сайту та водночас підкреслює унікальність: можливість створити індивідуальний образ або обрати вже готовий. На фоні затемненого градієнту розміщено яскравий слоган «Створи свій стиль» і кнопку, що веде на сторінку кастомізації одягу. Це одразу підкреслює позицію магазину як сучасної та модної платформи.

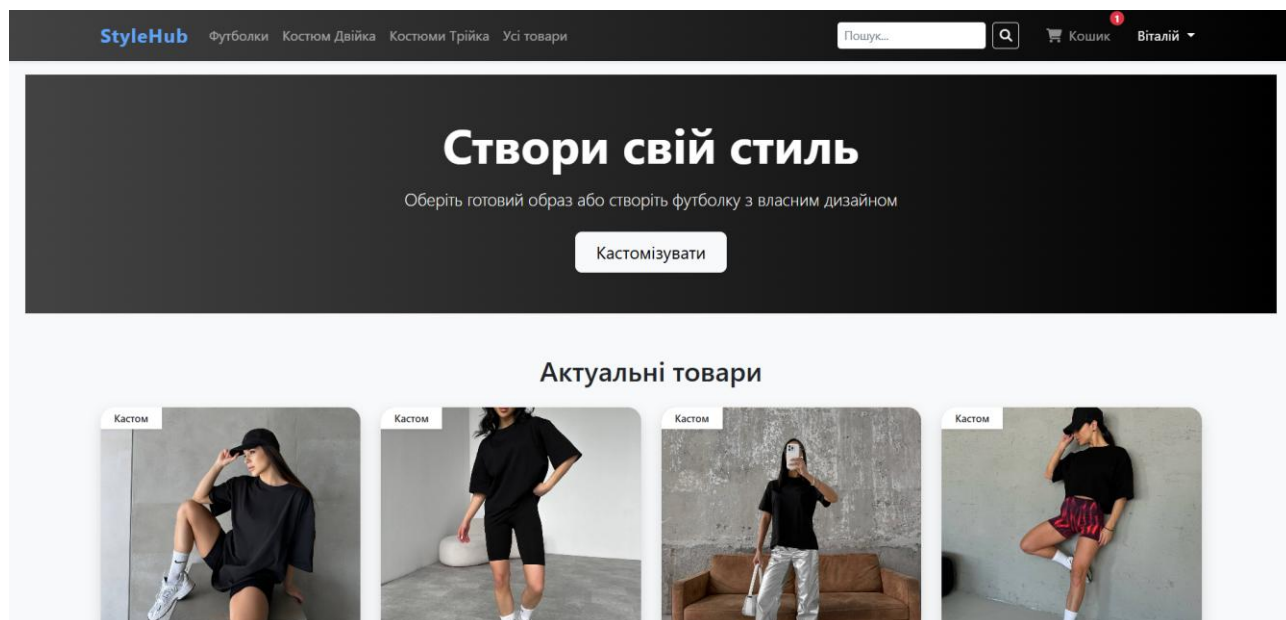


Рис. 3.1 – Головна сторінка з акцентом на індивідуальний стиль

Нижче розташована секція «Актуальні товари», де виводяться найбільш популярні позиції. Кожна картка містить зображення товару, його категорію, іконку «Кастом», якщо товар створено вручну, та інші візуальні акценти. Підтримується єдиний стиль відображення, що дозволяє користувачу швидко зорієнтуватися в асортименті.

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						41

На сторінці каталогу товарів реалізовано повноцінну систему фільтрів (Рис 3.2). Вона розташована в лівій частині екрана та дозволяє обирати категорії, кольори, розміри й ціновий діапазон. Користувач може комбінувати кілька фільтрів, що значно полегшує пошук. Особливо зручно реалізовано сортування — за ціною чи назвою в обох напрямках.

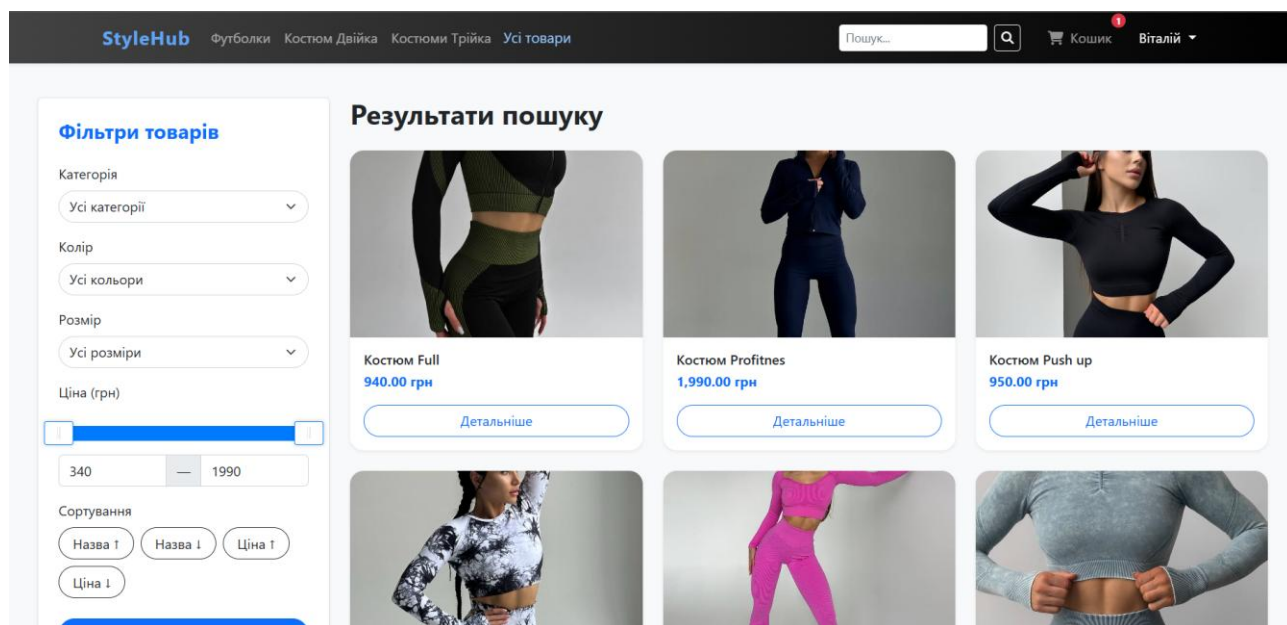


Рис. 3.2 – Каталог товарів з інтерактивними фільтрами

Карточки товарів містять яскраві фото, назву, ціну та кнопку «Детальніше», яка веде на сторінку з детальною інформацією про продукт. Таким чином, інтерфейс не перевантажений зайвою інформацією, але містить усе необхідне для швидкого прийняття рішення про купівлю.

Кошик — це ще одна важлива частина інтерфейсу. Тут реалізовано список товарів із усіма характеристиками: фото, назва, колір, розмір, ціна, кількість. Є можливість оновити кількість товару або видалити його зі списку. Загальна сума автоматично підраховується внизу, а кнопки «Оформити замовлення» та «Очистити кошик» розміщені зручним і логічним чином (Рис. 3.3).

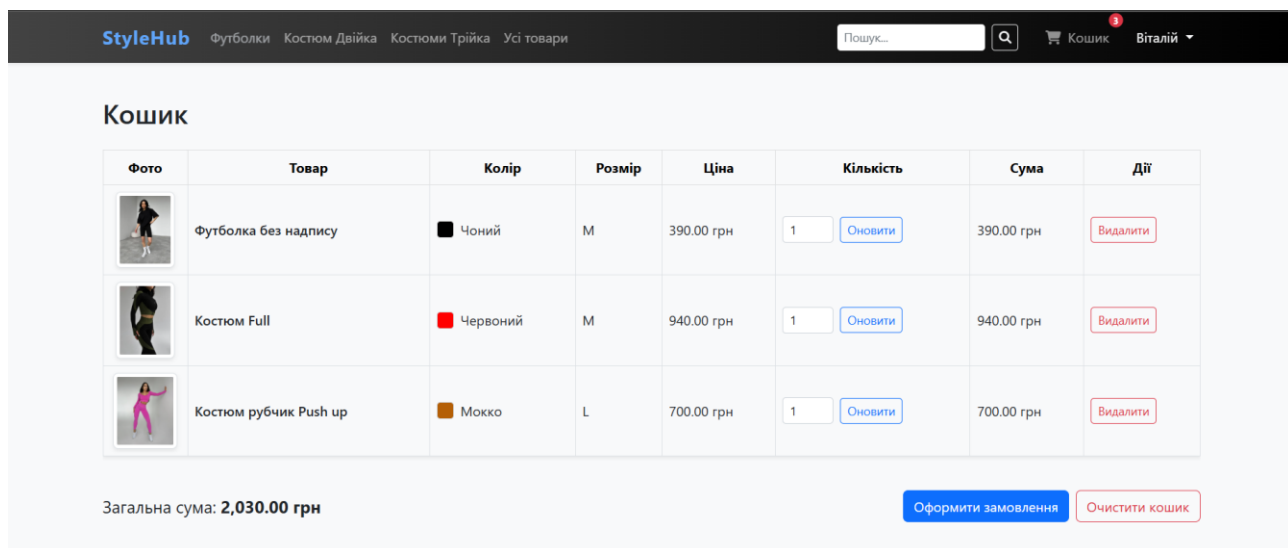


Рис. 3.3 – Кошик із можливістю редагування замовлення

Профіль користувача також реалізовано зрозуміло та логічно (Рис 3.4). У правій частині сторінки розміщено блок для зміни пароля, а в лівій — особисті дані: прізвище, ім'я, по батькові, телефон, місто, тип доставки, відділення. У нижній частині сторінки передбачено можливість видалити акаунт. Всі дії підтверджуються через введення пароля, що додає рівень захисту.

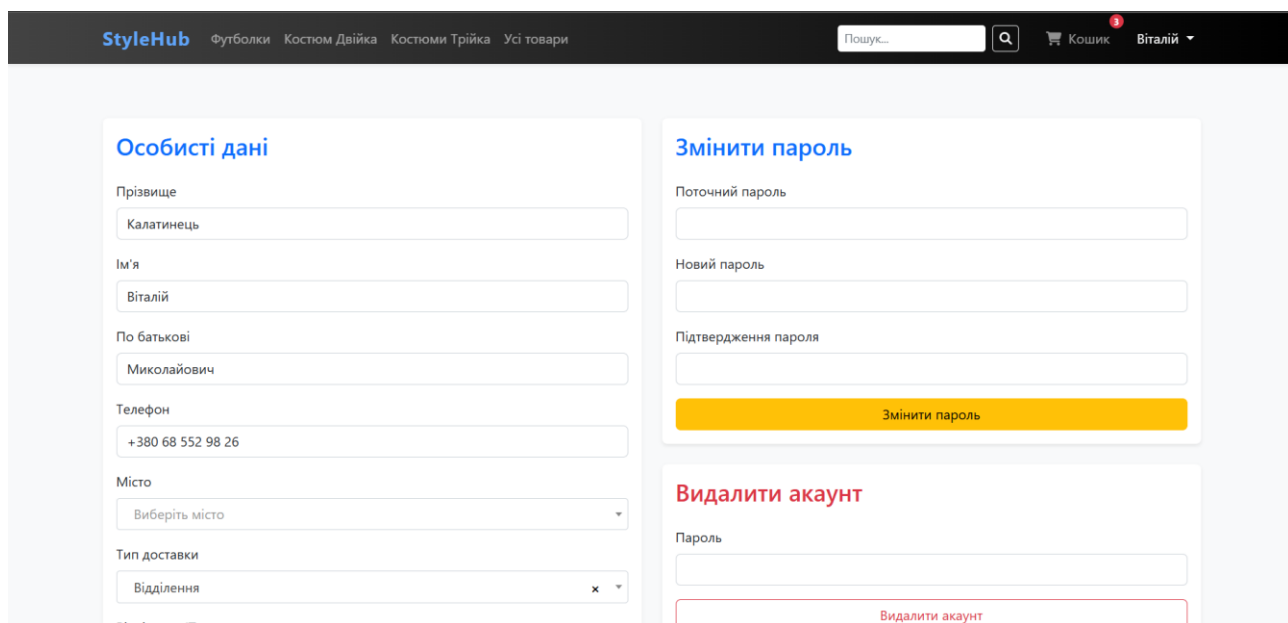


Рис. 3.4 – Особистий кабінет користувача з можливістю редагування даних

Особливої уваги заслуговує адміністративна панель. Вона відокремлена від основної частини сайту, але виконана в єдиному стилі, що підтримує загальну візуальну ідентичність. Зліва розташоване меню з пунктами: Товари, Категорії,

Кольори, Розміри, Замовлення, Перехід на сайт. У правій частині виводиться таблиця з відповідними даними. Для кожного запису передбачено дії: редагування або видалення. Завдяки цьому адміністратор може швидко реагувати на зміну асортименту, оновлювати наявність, контролювати кольори та розміри.

ID	Назва	Категорія	Ціна	Наявність	Кольори	Розміри	Статус	Дії
18	Костюм Profitnes	Костюми Трійка	1,990.00 грн	54	Чорний, Червоний, Білий, Апельсин, Малина, Електрик, Фіолет, Коричневий, Мокко	S, M, L	Активний	[Редагувати] [Видалити]
17	Костюм Undies	Костюми Трійка	1,390.00 грн	53	Чорний, Білий, Яблуко, Голубий, Електрик, Коричневий, Мокко	S, M, L	Активний	[Редагувати] [Видалити]
16	Костюм Full	Костюм Двійка	940.00 грн	786	Чорний, Червоний, Бежевий, Жовтий, Яблуко, Електрик, Фіолет, Коричневий, Мокко	S, M, L	Активний	[Редагувати] [Видалити]
15	Костюм рубчик Push up	Костюм Двійка	700.00 грн	76	Червоний, Бежевий, Яблуко, Неон, Малина, Фіолет, Коричневий, Мокко	S, M, L	Активний	[Редагувати] [Видалити]
14	Костюм Тай-дай	Костюм Двійка	1,000.00 грн	899	Бежевий, Білий, Яблуко, Голубий, Неон, Апельсин, Електрик, Фіолет, Мокко	S, M, L	Активний	[Редагувати] [Видалити]
13	Костюм Push up	Костюм Двійка	950.00 грн	98	Чорний, Червоний, Жовтий, Голубий, Фіолет, Мокко	S, M, L	Активний	[Редагувати] [Видалити]

Рис. 3.5 – Адміністративна панель керування товарами

Крім базового відображення, весь інтерфейс має адаптивний дизайн. Це означає, що на мобільних пристроях усі елементи масштабуються, змінюють положення і залишаються зручними для користування. Наприклад, у каталозі фільтри зсуваються вгору або відкриваються через випадаюче меню, кнопки стають більшими, а поля для введення спрощуються.

Інтерфейс розроблено з використанням Bootstrap 5, що дозволяє забезпечити кросбраузерну сумісність і швидку реакцію на події. Додаткові ефекти — анімація кнопок, фокус на полях, підказки — реалізовано через JavaScript і jQuery.

У підсумку можна стверджувати, що інтерфейс розробленої платформи є вдалим прикладом гармонійного поєднання сучасного дизайну, інтуїтивної взаємодії та високого рівня користувацької зручності. Усі елементи сайту — від першого візуального враження на головній сторінці до фінального натискання кнопки «Оформити замовлення» — були спроектовані з чітким фокусом на

реального користувача. Тут немає нічого зайвого чи випадкового: кожна кнопка, кожен відступ, кожен шрифт спрямовані на те, щоб зробити процес перегляду товарів і покупки максимально комфортним та приємним.

Структура інтерфейсу дозволяє користувачу не замислюватись над тим, куди потрібно натискати або як знайти бажаний товар — усе логічно розміщено, зрозуміло підписано та адаптується під будь-який екран. Це особливо важливо з урахуванням того, що дедалі більше людей здійснюють покупки саме з мобільних пристроїв. Інтерфейс платформи відповідає цим очікуванням: гнучкий, легкий, привабливий. Він не просто відображає дані, а супроводжує користувача протягом усього шляху — від перегляду до покупки, залишаючи при цьому позитивне враження та бажання повернутись.

Завдяки адаптивності, сучасним фронтенд-технологіям і дбайливо продуманій логіці відображення, ця система є не просто функціональним інструментом для продажу одягу, а й повноцінним сервісом, що задовольняє запити різних категорій користувачів — від молоді, яка прагне виглядати стильно, до дорослих покупців, яким важлива надійність, простота і прозорість кожного кроку. Саме тому інтерфейс платформи можна вважати не лише ефективним, а й конкурентоспроможним — таким, що здатен підтримати розвиток повноцінного онлайн-бізнесу.

Висновки до третього розділу

У третьому розділі було зосереджено увагу на практичному аспекті розробки платформи інтернет-магазину — а саме на її встановленні, налаштуванні та візуальній реалізації. Результати, отримані на цьому етапі, підтверджують успішну інтеграцію технічної частини з інтерфейсною, що й дозволило досягти цілісної, функціональної та зручної системи.

Перш за все, було розглянуто процес розгортання платформи в локальному середовищі. Усі ключові кроки, такі як налаштування середовища Open Server, створення бази даних, виконання міграцій та налаштування конфігураційних параметрів через .env, були описані як частина загального процесу впровадження. Особливу увагу приділено безпеці системи: реалізовано CSRF-

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						45

захист, валідацію даних, перевірку прав доступу, підтвердження електронної пошти та обмеження адміністративних прав.

Інтерфейс користувача детально проаналізовано з точки зору зручності, доступності та сучасності. Розглянуто кожен важливий розділ: головну сторінку, каталог товарів із фільтрами, детальну сторінку товару, кошик, оформлення замовлення та профіль користувача. Для адміністратора було реалізовано окрему панель керування, де передбачено всі необхідні дії з товарами, категоріями, кольорами, розмірами та замовленнями. Це дозволяє ефективно управляти магазином без потреби втручання у код чи базу даних.

Значну увагу приділено адаптивності інтерфейсу. Завдяки використанню фреймворку Bootstrap 5 та динамічної логіки через jQuery й AJAX, система однаково зручно працює як на десктопах, так і на мобільних пристроях. Це забезпечує ширше охоплення користувачів та покращує загальне враження від платформи.

Загалом, можна зробити висновок, що розроблений інтерфейс повністю відповідає поставленим вимогам. Він логічний, естетичний, інтуїтивно зрозумілий та виконує всі функціональні задачі. Разом із коректним налаштуванням середовища, це дозволяє розглядати платформу як готовий до впровадження інструмент для запуску повноцінного інтернет-магазину одягу.

ВИСНОВКИ

У межах даної кваліфікаційної роботи було виконано повний цикл розробки платформи інтернет-магазину одягу — від аналізу сучасних рішень і вибору архітектури до технічної реалізації та опису готового інтерфейсу. Отримані результати свідчать про досягнення поставленої мети — створити зручну, безпечну, адаптивну систему, яка дозволяє користувачеві зручно здійснювати онлайн-замовлення, а адміністратору — ефективно керувати асортиментом, замовленнями та структурою магазину.

У першому розділі було проаналізовано особливості використання інформаційних технологій у контексті розробки веб-систем для електронної торгівлі. Ретельний аналіз аналогічних платформ — таких як Intertop, Zalando та Answer — дозволив визначити найкращі практики, які варто запозичити, а також типові недоліки, яких слід уникати. Було сформульовано технічні та функціональні вимоги до системи, серед яких: підтримка реєстрації користувачів із верифікацією email, особистий кабінет з редагуванням даних, фільтрація товарів за різними параметрами, оформлення та оплата замовлення, а також система адміністрування. Обґрунтування вибору архітектури MVC на базі фреймворку Laravel дозволило закласти фундамент для побудови масштабованої, розширюваної системи. Також було визначено оптимальні інструментальні засоби — середовище PhpStorm, сервер Open Server, базу даних MySQL, платіжну систему LiqPay та API Нової пошти для інтеграції з логістикою.

Другий розділ був присвячений детальному технічному проєктуванню. У ньому були визначені варіанти використання системи з боку користувача, адміністратора та взаємодії з зовнішніми сервісами. На їхній основі сформована об'єктно-орієнтована модель системи, що стала основою для побудови структури бази даних. Була проаналізована схема зв'язків між ключовими таблицями — такими як users, orders, products, comments, sizes, colors — та їх допоміжними елементами. Усі таблиці нормалізовані, забезпечують цілісність і мінімізують дублювання інформації. Алгоритми роботи системи реалізовано

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						47

через контролери та сервіси Laravel: обробку замовлень, інтеграцію з API доставки й оплати, створення накладних та підтвердження платежів. Це було проілюстровано за допомогою діаграми послідовності, яка демонструє логіку роботи системи при оформленні замовлення.

У третьому розділі акцент зроблено на практичному аспекті — розгортанні, конфігурації та безпосередньому використанні платформи. Було описано порядок встановлення системи на локальному сервері з використанням Open Server, конфігурацію змінних середовища .env, виконання міграцій та сидів, генерацію ключів безпеки. Особливу увагу приділено інтеграції з Nova Poshta API та LiqPay, що забезпечує можливість створення замовлення з реальним номером ТТН і оплатою онлайн. Інтерфейс користувача спроектовано у відповідності до сучасних UX/UI-тенденцій: чистий, адаптивний, мінімалістичний. Реалізовано зручну фільтрацію, персональний кабінет, кошик із оновленням без перезавантаження та сторінку оформлення з автоматичним підтягуванням особистих даних. Інтерфейс адміністрування дозволяє додавати нові товари, категорії, розміри, кольори, керувати замовленнями, бачити їх статус і зміст.

Усі функціональні модулі працюють у взаємозв'язку, забезпечуючи повноцінну комерційну модель. Система підтримує основні бізнес-процеси: від ознайомлення з товаром — до оплати та доставки. Завдяки використанню сучасного технологічного стеку (Laravel, Bootstrap, MySQL, API), платформа є гнучкою та відкритою до масштабування: у майбутньому можна додати підтримку мультивалютності, реєстрацію продавців, систему знижок або чат із менеджером.

Узагальнюючи, можна сказати, що проєктна мета була повністю досягнута. Платформа вийшла функціональною, надійною, зручна в користуванні як для покупця, так і для адміністратора. Було дотримано вимог безпеки (використання CSRF-захисту, перевірка прав доступу, підтвердження email), принципів об'єктно-орієнтованої розробки, а також забезпечено чітку структуру даних.

Система протестована в умовах локального середовища та готова до подальшого розгортання в реальному продакшені.

Таким чином, виконана робота демонструє не лише технічні знання та навички в розробці сучасних веб-систем, а й уміння цілісно підходити до проєкту: від ідеї — до реалізації, документування та підготовки до впровадження. Розроблена платформа може бути ефективно використана в реальному секторі малого та середнього бізнесу, а також слугувати основою для подальших навчальних і комерційних ініціатив.

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						49

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційний сайт «Laravel» [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com>
2. Офіційний сайт «PHP» [Електронний ресурс] – Режим доступу до ресурсу: <https://www.php.net>
3. Документація Laravel [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs>
4. Офіційний сайт «Blade Templates» [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/blade>
5. Офіційний сайт «Bootstrap» [Електронний ресурс] – Режим доступу до ресурсу: <https://getbootstrap.com>
6. Документація Bootstrap 5 [Електронний ресурс] – Режим доступу до ресурсу: <https://getbootstrap.com/docs/5.0/getting-started/introduction>
7. Офіційний сайт «jQuery» [Електронний ресурс] – Режим доступу до ресурсу: <https://jquery.com>
8. Документація jQuery [Електронний ресурс] – Режим доступу до ресурсу: <https://api.jquery.com>
9. Офіційний сайт «MySQL» [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mysql.com>
10. Документація MySQL 8.0 [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc>
11. Офіційний сайт «PhpStorm» [Електронний ресурс] – Режим доступу до ресурсу: <https://www.jetbrains.com/phpstorm>
12. Документація PhpStorm [Електронний ресурс] – Режим доступу до ресурсу: <https://www.jetbrains.com/help/phpstorm>
13. Офіційний сайт «Open Server» [Електронний ресурс] – Режим доступу до ресурсу: <https://ospanel.io>
14. Документація Open Server Panel [Електронний ресурс] – Режим доступу до ресурсу: <https://ospanel.io/docs>

15. Офіційний сайт «LiqPay» [Електронний ресурс] – Режим доступу до ресурсу: <https://www.liqpay.ua>

16. Документація LiqPay API [Електронний ресурс] – Режим доступу до ресурсу: <https://www.liqpay.ua/documentation>

17. Офіційний сайт «Nova Poshta» [Електронний ресурс] – Режим доступу до ресурсу: <https://novaposhta.ua>

18. Документація Nova Poshta API [Електронний ресурс] – Режим доступу до ресурсу: <https://developers.novaposhta.ua>

19. Офіційний сайт «Select2» [Електронний ресурс] – Режим доступу до ресурсу: <https://select2.org>

20. Документація Select2 [Електронний ресурс] – Режим доступу до ресурсу: <https://select2.org/data-sources/ajax>

21. Документація Laravel Validation [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/validation>

22. Документація Laravel Localization [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/localization>

23. Документація Laravel Authentication [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/authentication>

24. Документація Laravel CSRF Protection [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/csrf>

25. Документація Laravel Routing [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/routing>

26. Документація Laravel Eloquent ORM [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/eloquent>

27. Документація Laravel Middleware [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/middleware>

28. Офіційний сайт «Composer» [Електронний ресурс] – Режим доступу до ресурсу: <https://getcomposer.org>

29. Документація RESTful API [Електронний ресурс] – Режим доступу до ресурсу: <https://restfulapi.net>

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						51

30. Стаття «MVC Pattern in Web Development» [Електронний ресурс] –
Режим доступу до ресурсу: <https://www.geeksforgeeks.org/mvc-design-pattern>

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						52

ДОДАТКИ

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						53

Контролер профілю

<?php

namespace App\Http\Controllers;

use App\Services\NovaPoshtaService;

use Illuminate\Http\Request;

use Illuminate\Support\Arr;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Log;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Validator;

class ProfileController extends Controller

{

protected \$novaPoshtaService;

public function __construct(NovaPoshtaService \$novaPoshtaService)

{

\$this->novaPoshtaService = \$novaPoshtaService;

}

public function edit()

{

return view('profile.edit', ['user' => Auth::user()]);

}

public function updatePassword(Request \$request)

{

\$user = Auth::user();

\$validator = Validator::make(

\$request->all(),

[

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						54

```

        'current_password' => ['required'],
        'password' => ['required', 'string', 'min:8', 'confirmed'],
    ],
    [
        'current_password.required' => 'Поле поточний пароль є обов’язковим.',
        'password.required' => 'Поле новий пароль є обов’язковим.',
        'password.min' => 'Новий пароль має містити щонайменше :min символів.',
        'password.confirmed' => 'Поле підтвердження пароля не збігається.',
    ],
    [
        'current_password' => 'поточний пароль',
        'password' => 'новий пароль',
    ]
);

if ($validator->fails()) {
    return back()->withErrors($validator, 'updatePassword')->withInput();
}

if (!Hash::check($request->current_password, $user->password)) {
    return back()->withErrors(['current_password' => 'Поточний пароль невірний.'],
'updatePassword');
}

$user->update([
    'password' => Hash::make($request->password),
]);

return back()->with('status', 'password-updated');
}

public function destroy(Request $request)
{
    $user = Auth::user();

```

```

$validator = Validator::make(
    $request->all(),
    [
        'password' => ['required', 'current_password'],
    ],
    [
        'password.required' => 'Поле пароль є обов'язковим.',
        'password.current_password' => 'Вказаний пароль невірний.',
    ],
    [
        'password' => 'пароль',
    ]
);

if ($validator->fails()) {
    return back()->withErrors($validator, 'deleteAccount')->withInput();
}

Auth::logout();
$user->delete();
$request->session()->invalidate();
$request->session()->regenerateToken();

return redirect('/')->with('success', 'Ваш акаунт було успішно видалено.');
```

```

public function update(Request $request)
{
    $user = Auth::user();
    $validated = $request->validate([
        'surname' => ['required', 'string', 'max:255'],
        'first_name' => ['required', 'string', 'max:255'],
        'patronymic' => ['required', 'string', 'max:255'],
    ]

```

```

'phone' => ['required', 'string', 'max:255'],
'city_ref' => ['required', 'string', 'max:255'],
'city' => ['nullable', 'string', 'max:255'],
'warehouse_ref' => ['required', 'string', 'max:255'],
'delivery_point' => ['nullable', 'string', 'max:255'],
'delivery_type' => ['required', 'in:warehouse,postomat'],
], [
    'required' => 'Поле :attribute є обов'язковим.',
    'string' => 'Поле :attribute має бути текстовим.',
    'max' => 'Поле :attribute не повинно перевищувати :max символів.',
    'in' => 'Поле :attribute містить недопустиме значення.',
]);

Log::debug('Profile update validated data', $validated);

$cityName = $validated['city'];
if ($validated['city_ref']) {
    $cityData = $this->novaPoshtaService->getCitiesByRef($validated['city_ref']);
    $cityName = !empty($cityData) && isset($cityData[0]['Description']) ?
$cityData[0]['Description'] : ($validated['city'] ?? 'Невідоме місто');
    Log::debug('City data for city_ref', ['city_ref' => $validated['city_ref'], 'city_data' =>
$cityData, 'city_name' => $cityName]);
}

$warehouseName = $validated['delivery_point'];
if ($validated['warehouse_ref']) {
    $warehouseData = $this->novaPoshtaService->getWarehousesByRef($validated['city_ref'],
$validated['warehouse_ref'], $validated['delivery_type']);
    $warehouseName = !empty($warehouseData) && isset($warehouseData[0]['Description']) ?
$warehouseData[0]['Description'] : ($validated['delivery_point'] ?? 'Невідоме відділення');
    Log::debug('Warehouse data for warehouse_ref', [
        'city_ref' => $validated['city_ref'],
        'warehouse_ref' => $validated['warehouse_ref'],
        'delivery_type' => $validated['delivery_type'],
        'warehouse_data' => $warehouseData,

```

```

        'warehouse_name' => $warehouseName
    ]);
}

$updateData = array_merge(
    Arr::except($validated, ['city', 'delivery_point']),
    [
        'city' => $cityName,
        'delivery_point' => $warehouseName,
    ]
);

Log::debug('Profile update data before saving', $updateData);

$user->update($updateData);

Log::info('Profile updated successfully', ['user_id' => $user->id, 'updated_data' =>
$updateData]);

return redirect()->route('profile.edit')->with('status', 'profile-updated');
}
}

```

Контролер товарів

<?php

namespace App\Http\Controllers;

use App\Models\Category;

use App\Models\Color;

use App\Models\Product;

use App\Models\Size;

use Illuminate\Http\Request;

class ProductController extends Controller

{

public function index(Request \$request)

{

\$categories = Category::where('is_active', true)->get();

\$colors = Color::all();

\$sizes = Size::all();

\$query = Product::where('is_active', true)->with(['colors', 'sizes', 'category'])

->whereHas('category', function (\$q) {

\$q->where('is_active', true);

});

if (\$request->has('category_id') && \$request->category_id) {

\$query->where('category_id', \$request->category_id);

}

if (\$request->has('color_id') && \$request->color_id) {

\$query->whereHas('colors', function (\$q) use (\$request) {

\$q->where('colors.id', \$request->color_id);

});

}

if (\$request->has('size_id') && \$request->size_id) {

\$query->whereHas('sizes', function (\$q) use (\$request) {

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						59

```

        $q->where('sizes.id', $request->size_id);
    });
}
if ($request->has('min_price') && is_numeric($request->min_price)) {
    $query->where('price', '>=', $request->min_price);
}
if ($request->has('max_price') && is_numeric($request->max_price)) {
    $query->where('price', '<=', $request->max_price);
}

$sort = $request->input('sort', 'name_asc');
switch ($sort) {
    case 'price_asc':
        $query->orderBy('price', 'asc');
        break;
    case 'price_desc':
        $query->orderBy('price', 'desc');
        break;
    case 'name_desc':
        $query->orderBy('name', 'desc');
        break;
    default:
        $query->orderBy('name', 'asc');
}

$products = $query->paginate(12);

$minPrice = Product::where('is_active', true)->min('price') ?? 0;
$maxPrice = Product::where('is_active', true)->max('price') ?? 0;

return view('client.products.index', compact('products', 'categories', 'colors', 'sizes', 'minPrice',
'maxPrice'));
}

public function search(Request $request)

```

```

{
    $queryString = $request->input('query');
    $categories = Category::where('is_active', true)->get();
    $colors = Color::all();
    $sizes = Size::all();

    $query = Product::where('is_active', true)->with(['colors', 'sizes', 'category'])
        ->whereHas('category', function ($q) {
            $q->where('is_active', true);
        });

    if ($queryString) {
        $query->where(function ($q) use ($queryString) {
            $q->where('name', 'LIKE', "%{$queryString}%")
                ->orWhere('description', 'LIKE', "%{$queryString}%");
        });
    }

    $products = $query->paginate(12);

    $minPrice = Product::where('is_active', true)->min('price') ?? 0;
    $maxPrice = Product::where('is_active', true)->max('price') ?? 0;

    return view('client.products.index', compact('products', 'categories', 'colors', 'sizes',
'queryString', 'minPrice', 'maxPrice'));
}

public function showCategory($id)
{
    $category = Category::where('is_active', true)->with(['products' => function ($query) {
        $query->where('is_active', true);
    }])->findOrFail($id);
    $categories = Category::where('is_active', true)->get();

    return view('client.categories.show', compact('category', 'categories'));
}

```

```

}

public function show($id)
{
    $product = Product::where('is_active', true)
        ->with([
            'colors' => function ($query) {
                $query->select('colors.id', 'colors.name', 'colors.hex');
            },
            'sizes',
            'comments.user' // завантажити коментарі з автором
        ])
        ->findOrFail($id);

    $categories = Category::where('is_active', true)->get();

    $related = Product::where('category_id', $product->category_id)
        ->where('id', '!=', $product->id)
        ->where('is_active', true)
        ->inRandomOrder()
        ->take(4)
        ->get();

    return view('client.products.show', compact('product', 'categories', 'related'));
}

public function customize($id)
{
    $product = Product::where('is_active', true)->with(['colors' => function ($query) {
        $query->select('colors.id', 'colors.name', 'colors.hex');
    }, 'sizes'])->findOrFail($id);
    $categories = Category::where('is_active', true)->get();

    return view('client.products.customize', compact('product', 'categories'));
}

```

```

}

public function storeCustomize(Request $request, $id)
{
    $request->validate([
        'design' => 'required|image|mimes:jpg,png,jpeg|max:2048',
    ]);

    $product = Product::findOrFail($id);
    $path = $request->file('design')->store('custom_designs', 'public');

    \App\Models\CustomDesign::create([
        'product_id' => $id,
        'image_path' => $path,
    ]);

    return redirect()->route('products.show', $id)->with('success', 'Кастомізація збережена');
}
}

```

Контролер кошику

```
<?php

namespace App\Http\Controllers;

use App\Models\Color;
use App\Models\Product;
use App\Models\Size;
use App\Traits\CartTrait;
use Illuminate\Http\Request;

class CartController extends Controller
{
    use CartTrait;

    public function index()
    {
        $cart = $this->getCart();
        $total = $cart['subtotal'];

        return view('client.cart.index', [
            'cart' => $cart['items'],
            'total' => $total,
        ]);
    }

    public function store(Request $request, Product $product)
    {
        $validated = $request->validate([
            'color_id' => ['required', 'exists:colors,id'],
            'size_id' => ['required', 'exists:sizes,id'],
            'quantity' => ['required', 'integer', 'min:1'],
        ]);
    }
}
```

```

$color = Color::findOrFail($validated['color_id']);
$size = Size::findOrFail($validated['size_id']);
$availableQuantity = $product->stock !== null ? $product->stock : 1000;

if ($product->stock !== null && $validated['quantity'] > $availableQuantity) {
    return redirect()->back()->with('error', 'Недостатньо товару в наявності.');
```

```

}

$cart = $this->getCart();
$cartKey = $product->id . '_' . $validated['color_id'] . '_' . $validated['size_id'];

$existingItemKey = array_search($cartKey, array_column($cart['items'], 'cartKey'));

if ($existingItemKey !== false) {
    $newQuantity = $cart['items'][$existingItemKey]['quantity'] + $validated['quantity'];
    if ($product->stock !== null && $newQuantity > $availableQuantity) {
        return redirect()->back()->with('error', 'Недостатньо товару в наявності.');
```

```

    }
    $cart['items'][$existingItemKey]['quantity'] = $newQuantity;
} else {
    $cart['items'][] = [
        'cartKey' => $cartKey,
        'product_id' => $product->id,
        'name' => $product->name,
        'price' => $product->price,
        'quantity' => $validated['quantity'],
        'color_id' => $validated['color_id'],
        'color_name' => $color->name,
        'color_hex' => $color->hex ?? '#ffffff',
        'size_id' => $validated['size_id'],
        'size_name' => $size->name,
        'image_path' => $product->image_path,
    ];
}

```

```

$cart['subtotal'] = $this->calculateTotal($cart['items']);
session()->put('cart', $cart);

return redirect()->route('cart.index')->with('success', 'Товар додано до кошика.');
```

}

```

public function update(Request $request, $cartKey)
{
    $validated = $request->validate([
        'quantity' => ['required', 'integer', 'min:0'],
    ]);

    $cart = $this->getCart();
    $existingItemKey = array_search($cartKey, array_column($cart['items'], 'cartKey'));

    if ($existingItemKey === false) {
        return redirect()->route('cart.index')->with('error', 'Товар не знайдено.');
```

}

```

    $productId = $cart['items'][$existingItemKey]['product_id'];
    $product = Product::findOrFail($productId);
    $availableQuantity = $product->stock !== null ? $product->stock : 1000;

    if ($validated['quantity'] == 0) {
        unset($cart['items'][$existingItemKey]);
        $cart['items'] = array_values($cart['items']);
        $cart['subtotal'] = $this->calculateTotal($cart['items']);
        session()->put('cart', $cart);
        return redirect()->route('cart.index')->with('success', 'Товар видалено.');
```

}

```

    if ($product->stock !== null && $validated['quantity'] > $availableQuantity) {
        return redirect()->route('cart.index')->with('error', 'Недостатньо товару в наявності.');
```

}

```

$cart['items'][$existingItemKey]['quantity'] = $validated['quantity'];
$cart['subtotal'] = $this->calculateTotal($cart['items']);
session()->put('cart', $cart);

return redirect()->route('cart.index')->with('success', 'Кількість оновлено.');
```

}

```

public function remove($cartKey)
{
    $cart = $this->getCart();
    $existingItemKey = array_search($cartKey, array_column($cart['items'], 'cartKey'));

    if ($existingItemKey !== false) {
        unset($cart['items'][$existingItemKey]);
        $cart['items'] = array_values($cart['items']);
        $cart['subtotal'] = $this->calculateTotal($cart['items']);
        session()->put('cart', $cart);
        return redirect()->route('cart.index')->with('success', 'Товар видалено.');
```

}

```

    return redirect()->route('cart.index')->with('error', 'Товар не знайдено.');
```

}

```

public function clear()
{
    session()->forget('cart');
    return redirect()->route('cart.index')->with('success', 'Кошик очищено.');
```

}

}

Контролер редагування товарів

```

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\Category;
use App\Models\Color;
use App\Models\Product;
use App\Models\Size;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;
use Illuminate\View\View;

class ProductController extends Controller
{
    public function index(Request $request): View
    {
        $showInactive = $request->boolean('show_inactive');
        $categoryId = $request->input('category_id');

        $query = Product::query();

        if (!$showInactive) {
            $query->where('is_active', true)
                ->whereHas('category', function ($q) {
                    $q->where('is_active', true);
                });
        }

        if ($categoryId) {
            $query->where('category_id', $categoryId);
        }
    }
}

```

```

    }

    $products = $query->with('category')->latest()->get();
    $categories = Category::all();

    return view('admin.products.index', compact('products', 'categories', 'showInactive',
'categoryId'));
    }

    public function create(Request $request): View
    {
        $showInactive = $request->boolean('show_inactive');
        $categories = $showInactive ? Category::all() : Category::where('is_active', true)->get();
        $colors = Color::all();
        $sizes = Size::all();

        return view('admin.products.create', compact('categories', 'colors', 'sizes', 'showInactive'));
    }

    public function store(Request $request): RedirectResponse
    {
        $validated = $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'description' => ['required', 'string'],
            'price' => ['required', 'numeric', 'min:0'],
            'stock' => ['required', 'integer', 'min:0'],
            'category_id' => ['required', 'exists:categories,id'],
            'colors' => ['array', 'exists:colors,id'],
            'sizes' => ['array', 'exists:sizes,id'],
            'image' => ['nullable', 'image', 'max:2048'],
            'is_active' => ['boolean'],
        ]);

        $imagePath = $request->file('image')
            ? $request->file('image')->store('products', 'public')

```

: null;

```
$product = Product::create([
    'name' => $validated['name'],
    'description' => $validated['description'],
    'price' => $validated['price'],
    'stock' => $validated['stock'],
    'category_id' => $validated['category_id'],
    'image_path' => $imagePath,
    'is_active' => $validated['is_active'] ?? true,
]);
```

```
if (!empty($validated['colors'])) {
    $product->colors()->sync($validated['colors']);
}
```

```
if (!empty($validated['sizes'])) {
    $product->sizes()->sync($validated['sizes']);
}
```

```
return redirect()->route('admin.products.index')
    ->with('success', 'Товар успішно створено.');
```

```
public function edit(Product $product, Request $request): View
{
    $showInactive = $request->boolean('show_inactive');
    $categories = $showInactive ? Category::all() : Category::where('is_active', true)->get();
    $colors = Color::all();
    $sizes = Size::all();

    return view('admin.products.edit', compact('product', 'categories', 'colors', 'sizes',
'showInactive'));
}
```

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						70

```

public function update(Request $request, Product $product): RedirectResponse
{
    $validated = $request->validate([
        'name' => ['required', 'string', 'max:255'],
        'description' => ['required', 'string'],
        'price' => ['required', 'numeric', 'min:0'],
        'stock' => ['required', 'integer', 'min:0'],
        'category_id' => ['required', 'exists:categories,id'],
        'colors' => ['array', 'exists:colors,id'],
        'sizes' => ['array', 'exists:sizes,id'],
        'image' => ['nullable', 'image', 'max:2048'],
        'is_active' => ['boolean'],
    ]);

    if ($request->file('image')) {
        if ($product->image_path) {
            Storage::disk('public')->delete($product->image_path);
        }
        $validated['image_path'] = $request->file('image')->store('products', 'public');
    }

    $product->update([
        'name' => $validated['name'],
        'description' => $validated['description'],
        'price' => $validated['price'],
        'stock' => $validated['stock'],
        'category_id' => $validated['category_id'],
        'image_path' => $validated['image_path'] ?? $product->image_path,
        'is_active' => $validated['is_active'] ?? true,
    ]);

    $product->colors()->sync($validated['colors'] ?? []);
    $product->sizes()->sync($validated['sizes'] ?? []);

    return redirect()->route('admin.products.index')

```

					ДУ «Житомирська політехніка».25.121.11.000 - ПЗ	Арк.
						71

```

        ->with('success', 'Товар успішно оновлено.');
```

}

```

public function destroy(Product $product): RedirectResponse
{
    if ($product->image_path) {
        Storage::disk('public')->delete($product->image_path);
    }

    $product->delete();

    return redirect()->route('admin.products.index')
        ->with('success', 'Товар успішно видалено.');
```

}

}