

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 цифрової техніки»

Виконав студент групи ЗПЗ-22-1
ВИГІВСЬКИЙ Богдан Віталійович

Керівник роботи:
ЧИЖМОТРИЯ Олексій Володимирович

Рецензент:
БАКАЛЮК Тетяна Анатоліївна

Житомир – 2026

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

«27» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ВИГІВСЬКИЙ Богдан Віталійович**

Керівник роботи: **ЧИЖМОТРИЯ Олексій Володимирович**

Тема роботи: **«Інтернет-магазин цифрової техніки»,**

затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., № 93/с**

Вихідні дані для роботи: **процеси роздрібної торгівлі, електронного обліку та реалізації товарів у сфері електронної комерції; предмет дослідження — методи, архітектурні патерни (MVC), бази даних (MySQL) та програмні засоби розробки вебзастосунку інтернет-магазину цифрової техніки.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.В.	01.03.2026р.	01.03.2026р
2	Чижмотря О.В.	01.03.2026р	01.03.2026р
3	Чижмотря О.В.	01.03.2026р	01.03.2026р

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебзастосунку інтернет-магазину та пояснювальної записки. Випускна кваліфікаційна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із 31 найменувань (3 стор.), 3 додатків (4 стор.), 11 таблиць та 16 ілюстрацій по тексту. Повний обсяг роботи становить 69 сторінок, з них 54 сторінок основного тексту.

Метою роботи є проєктування та програмна реалізація високопродуктивного інтернет-магазину цифрової техніки на основі архітектурного патерну MVC для оптимізації процесів онлайн-продажів та управління товарним асортиментом.

Архітектура розробленої системи базується на власному MVC-фреймворку без використання сторонніх монолітних рішень, що забезпечує високу швидкодію та гнучкість. Для збереження структурованої інформації про товари, категорії та користувачів створено реляційну базу даних у СУБД MySQL, а для обробки замовлень реалізовано інтерактивний кошик покупця із підтримкою збереження даних у форматі JSON.

Вебзастосунок має зручний користувацький інтерфейс, який дозволяє клієнтам легко переглядати каталог, фільтрувати товари та оформлювати замовлення. Також розроблено захищену адміністративну панель для керування контентом. Програмний продукт рекомендується для впровадження в комерційну діяльність підприємств роздрібної торгівлі.

КЛЮЧОВІ СЛОВА: БАЗА, ВЕБЗАСТОСУНОК, ІНТЕРНЕТ-МАГАЗИН, КОМЕРЦІЯ, ПАТЕРН.

					ІПЗ.КР.Б – 121 – 26 – ПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата	Інтернет-магазин цифрової техніки	Літ.	Арк.	Аркушів
Розроб.		Вигівський Б.В.					3	69
Керівник		Чижомря О.В.			Пояснювальна записка	Житомирська політехніка, група ІПЗ-22-1		
Рецензент.		Вакалюк Т.А.						
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's qualification work consists of an online store web application and an explanatory note. The bachelor's qualification work consists of an introduction, 3 chapters, conclusions, a list of references with 31 sources (3 pages), 3 appendices (4 pages), 11 tables, and 16 illustrations in the text. The total volume of the work is 69 pages, including 54 pages of the main text.

The purpose of the work is the design and software implementation of a high-performance online store for digital equipment based on the MVC architectural pattern to optimize online sales processes and product range management.

The system architecture is based on a custom MVC framework without the use of third-party monolithic solutions, which ensures high performance and flexibility. To store structured information about products, categories, and users, a relational database in the MySQL DBMS was created, and an interactive shopping cart with JSON format support was implemented to process orders.

The web application has a user-friendly interface that allows clients to easily browse the catalog, filter products, and place orders. A secure administrative panel for content management was also developed. The software product is recommended for implementation in the commercial activities of retail enterprises.

KEYWORDS: BASE, COMMERCE, PATTERN, STORE, WEB-APPLICATION.

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У СФЕРІ ЕЛЕКТРОННОЇ КОМЕРЦІЇ ЦИФРОВОЇ ТЕХНІКИ	10
1.1. Постановка задачі.....	10
1.2. Аналіз аналогів програмного продукту	12
1.3. Формування функціональних та нефункціональних вимог до системи.....	16
Висновки до першого розділу	19
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ІНТЕРНЕТ-МАГАЗИНУ ЦИФРОВОЇ ТЕХНІКИ	21
2.1. Вибір архітектури та технологічного інструментарію розробки	21
2.2. Визначення варіантів використання та об'єктно-орієнтованої структури системи	27
2.3. Розробка бази даних системи.....	33
2.4. Проєктування та реалізація алгоритмів роботи системи.....	37
2.5. Реалізація додатку веборієнтованої системи.....	41
Висновки до другого розділу	44
РОЗДІЛ 3. ІНТЕРФЕЙС, ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ ІНТЕРНЕТ-МАГАЗИНУ ЦИФРОВОЇ ТЕХНІКИ.....	47
3.1. Порядок встановлення та налаштування параметрів системи.....	47
3.2. Опис структури інтерфейсу користувача та адміністратора	51
3.3. Тестування системи та аналіз результатів	55
Висновки до третього розділу	59
ВИСНОВКИ.....	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	63
ДОДАТКИ	66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

ТЗ – технічне завдання

API – прикладний програмний інтерфейс (Application Programming Interface)

CRUD – основні операції, що використовуються для управління даними (Create, Read, Update, Delete)

CSS – каскадні таблиці стилів (Cascading Style Sheets)

HTML – стандартизована мова гіпертекстової розмітки документів (HyperText Markup Language)

JSON – текстовий формат обміну даними (JavaScript Object Notation)

LAMP – комплекс серверного програмного забезпечення (Linux, Apache, MySQL, PHP)

MVC – архітектурний патерн, схема розділення даних додатка і керуючої логіки на три окремі компоненти (Model-View-Controller)

PDO – об'єктно-орієнтований інтерфейс доступу до баз даних у PHP (PHP Data Objects)

PHP – скриптова мова програмування для веброзробки (PHP: Hypertext Preprocessor)

SQL – мова структурованих запитів (Structured Query Language)

UI/UX – інтерфейс користувача та досвід користувача (User Interface / User Experience)

ВСТУП

Актуальність обраної теми зумовлена тим, що сучасний етап розвитку світової економіки характеризується стрімким переходом традиційних форм ведення бізнесу в цифрове середовище. Електронна комерція (e-commerce) стала одним із найбільш динамічних секторів глобального ринку, а онлайн-торгівля — невід'ємною умовою життєздатності та конкурентоспроможності комерційних підприємств. Серед різних сегментів електронної комерції ринок цифрової техніки та побутової електроніки посідає одне з провідних місць за обсягами продажів та темпами залучення нових користувачів. Це зумовлено високою стандартизацією товарів, постійним оновленням модельних рядів та потребою споживачів в оперативному отриманні детальної технічної інформації та порівнянні характеристик.

Ефективне функціонування інтернет-магазину цифрової техніки критично залежить від якості, швидкодії та архітектурної гнучкості його програмного забезпечення. Товарні позиції цієї категорії мають складну ієрархічну структуру та велику кількість специфічних атрибутів, що висуває підвищені вимоги до проєктування баз даних та алгоритмів фільтрації і пошуку. Багато сучасних торговельних платформ розгортаються на базі готових систем керування контентом (CMS) або універсальних монолітних фреймворків. Проте такі рішення часто є надмірними, перевантаженими невикористовуваним функціоналом, що призводить до зниження продуктивності вебресурсу, збільшення часу відгуку сервера та ускладнення процесу подальшого масштабування під специфічні бізнес-процеси компанії.

У зв'язку з цим актуальним є інженерне завдання проєктування та розробки легковагового, високопродуктивного вебзастосунку інтернет-магазину на основі індивідуального архітектурного каркасу, що реалізує патерн MVC (Model-View-Controller). Створення системи «з нуля» за допомогою мови серверного програмування PHP та СУБД MySQL дозволяє оптимізувати швидкість обробки запитів, забезпечити повний контроль над вихідним кодом,

підвищити рівень захищеності даних від типових уразливостей та мінімізувати апаратні вимоги до хостингової інфраструктури.

Об'єктом цього дослідження виступають процеси роздрібної торгівлі, електронного обліку, систематизації та реалізації товарів у сфері електронної комерції. Своєю чергою, предметом дослідження є методи, архітектурні патерни (MVC), реляційні бази даних та інструментальні програмні засоби розробки вебзастосунку інтернет-магазину цифрової техніки.

Головна мета роботи полягає у проектуванні та програмній реалізації високопродуктивного вебзастосунку інтернет-магазину цифрової техніки на основі архітектурного патерну MVC для оптимізації процесів онлайн-продажів, автоматизації обробки замовлень та ефективного управління товарним асортиментом підприємства.

Для досягнення поставленої мети необхідно вирішити такі завдання: проаналізувати предметну область електронної комерції, дослідити специфіку торгівлі цифровою технікою та виконати порівняльний аналіз існуючих аналогів-конкурентів на ринку; сформулювати технічне завдання, визначити функціональні та нефункціональні вимоги до розроблюваного програмного комплексу; спроектувати логічну та фізичну структуру реляційної бази даних, забезпечивши її приведення до третьої нормальної форми (3НФ); розробити архітектуру індивідуального MVC-каркасу вебзастосунку, включаючи створення підсистем маршрутизації запитів на основі регулярних виразів та безпечного підключення до бази даних; реалізувати програмні модулі клієнтської частини інтернет-магазину, зокрема динамічний каталог із пагінацією, інтерактивний кошик покупця на базі сесій та модуль оформлення замовлень із конвертацією даних у формат JSON; створити захищений адміністративний інтерфейс (Back-office) для виконання CRUD-операцій над категоріями, товарами та замовленнями з розмежуванням прав доступу користувачів; провести модульне та функціональне тестування розробленого вебзастосунку, оцінити його швидкодію та відповідність критеріям надійності.

Основними методами дослідження для вирішення поставлених завдань у роботі обрано: методи системного аналізу предметної області — для дослідження бізнес-процесів торгівлі; принципи об'єктно-орієнтованого аналізу та проєктування (ООАП) — для побудови архітектури програмного комплексу; теорію реляційних баз даних — для моделювання схеми збереження інформації; методи чорної та білої скриньки — для тестування модулів вебзастосунку.

Практичне значення отриманих результатів полягає у створенні готового до розгортання та комерційної експлуатації вебзастосунку інтернет-магазину цифрової техніки, який має високу швидкодію та гнучкість налаштування. Розроблений індивідуальний MVC-каркас може слугувати базовою платформою для створення інших галузевих систем електронної комерції. Програмний продукт рекомендується до впровадження у діяльність підприємств малого та середнього бізнесу з метою розширення каналів збуту, автоматизації обліку онлайн-замовлень та зниження витрат на технічну підтримку вебресурсів.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У СФЕРІ ЕЛЕКТРОННОЇ КОМЕРЦІЇ ЦИФРОВОЇ ТЕХНІКИ

1.1. Постановка задачі

Стрімкий розвиток електронної комерції висуває нові вимоги до якості та швидкодії програмного забезпечення. Особливо це стосується сегменту продажу цифрової та побутової техніки, який характеризується високою динамікою оновлення модельних рядів, великою кількістю технічних специфікацій для кожної одиниці товару та високим рівнем конкуренції за увагу споживача. Забезпечення ефективної взаємодії між покупцем та магазином вимагає розробки надійного, інтуїтивно зрозумілого та швидкодіючого вебзастосунку.

Об'єктом автоматизації у межах цього дослідження виступають бізнес-процеси роздрібної торгівлі цифровою технікою. Проведений детальний аналіз предметної області дозволив визначити ключові інформаційні потоки та операційні процеси, які підлягають першочерговій цифровізації. До них належить, зокрема, управління асортиментом і товарним каталогом, що передбачає внесення нових моделей техніки зі складними ієрархічними атрибутами (колір, обсяг пам'яті, тип процесора тощо), динамічне оновлення ціноутворення відповідно до ринкових умов, керування статусами наявності товарних позицій на складах, а також гнучке налаштування акційних пропозицій. Не менш важливим інженерним аспектом є автоматизація обробки замовлень та логістики, яка охоплює прийом онлайн-заявок від клієнтів у режимі реального часу, автоматичне резервування продукції, відстеження та послідовну зміну статусів виконання замовлення на всіх етапах його життєвого циклу — від комплектування до безпосередньої відправки покупцеві. Крім того, цифровізації підлягає підсистема взаємодії з клієнтами (CRM-компонент), призначена для реєстрації нових користувачів, безпечного збереження їхніх персональних даних, ведення історії покупок та надання повноцінного доступу до функціональних можливостей особистого кабінету.

Дослідження традиційного стану ведення торговельної діяльності виявило, що відсутність централізованої автоматизованої системи обліку та продажів зумовлює виникнення низки критичних проблем. У таких умовах спостерігається суттєве уповільнення процесів обробки замовлень менеджерами, а також зростає ризик негативного впливу людського фактора, що виражається, наприклад, у помилковому підтвердженні заявок на відсутній на складі товар. Додатковою деструктивною ознакою є втрата потенційних покупців через обмеженість комунікаційних каналів та відсутність можливості самостійного вибору товарів у позаробочий час. Ситуація ускладнюється високою чутливістю ринку цифрової техніки до коливань валютних курсів і динамічною зміною закупівельних цін, оскільки ручне оновлення прайс-листів та перевірка актуальності технічних специфікацій на вітрині відволікають значні ресурси персоналу. Водночас відсутність інструментів швидкого пошуку та багатокритеріальної фільтрації товарів за технічними параметрами (зокрема за обсягом оперативної пам'яті чи типом матриці) призводить до погіршення показників користувацького досвіду, змушуючи споживачів звертатися до платформ конкурентів.

Для кардинальної оптимізації зазначених інформаційних потоків та усунення операційних ризиків необхідним є перехід до автоматизованої моделі функціонування шляхом впровадження спеціалізованого вебзастосунку. Делегування системі процесів валідації вхідних даних, миттєвого розрахунку підсумкової вартості віртуального кошика та динамічного управління залишками на складі дозволяє мінімізувати рутинну операційну діяльність. Завдяки цьому формується цілісне інтерактивне середовище, у якому споживач одержує можливість безперешкодно пройти повний цикл від ознайомлення з характеристиками технічного засобу до фіксації покупки в режимі двадцять чотири на сім без обов'язкового залучення персоналу магазину. Такий підхід повністю усуває проблему розсинхронізації даних між фактичним складом та електронною вітриною, забезпечуючи безперебійне функціонування

вебресурсу. Оптимізовану логіку процесу оформлення замовлення в межах розроблюваної системи графічно відображено на схемі (рис. 1.1).

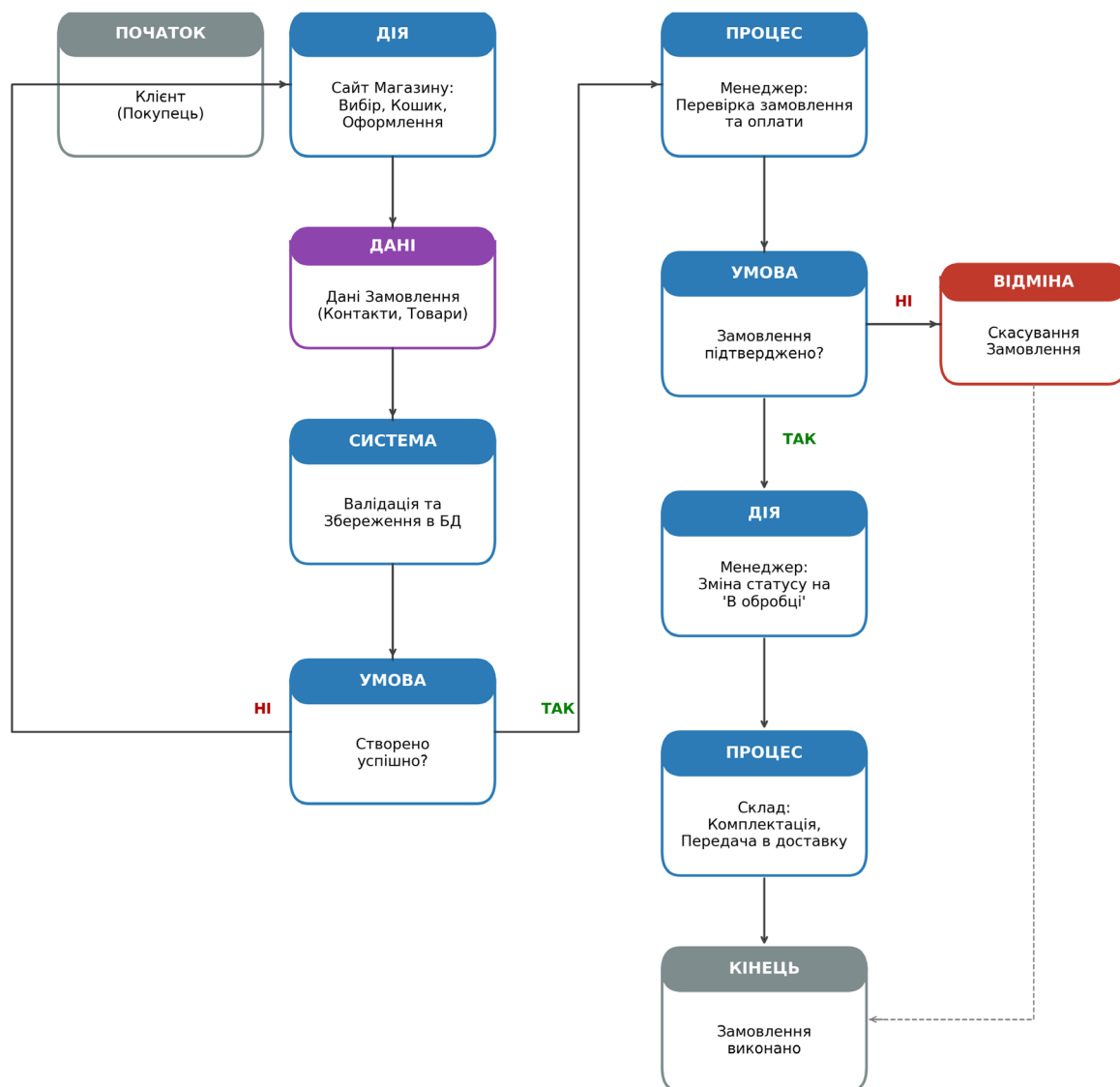


Рис. 1.1 – Модель бізнес-процесу оформлення замовлення (To-Be)

1.2. Аналіз аналогів програмного продукту

Для обґрунтування доцільності розробки індивідуального програмного забезпечення інтернет-магазину цифрової техніки необхідно провести детальне дослідження існуючих аналогів, які наразі займають лідируючі позиції на ринку електронної комерції України. Аналіз аналогів дозволяє виявити найбільш затребуваний функціонал, дослідити особливості побудови інтерфейсів користувача та визначити архітектурні обмеження монолітних систем, що підлягають оптимізації у розроблюваному вебзастосунку [10]. Як основні

об'єкти для порівняльного аналізу було обрано три провідні вітчизняні платформи електронної комерції у сегменті електроніки та цифрової техніки: інтернет-супермаркет «Rozetka», мультибрендовий онлайн-магазин «Allo» та спеціалізований вебресурс «Citrus» [11].

Інтернет-платформа «Rozetka» (рис. 1.2) є найбільшим гравцем вітчизняного ринку електронної комерції. Вона вирізняється потужною системою фільтрації за технічними атрибутами та розгалуженою системою особистих кабінетів. Водночас перехід платформи у формат глобального маркетплейсу зумовив перевантаження користувацького інтерфейсу (UI) та системи навігації, що ускладнює пошук цільової цифрової техніки. Архітектурно система базується на масивних мікросервісних рішеннях. При цьому велика кількість фонових скриптів і рекламних трекерів суттєво знижує швидкість первинного завантаження сторінки на мобільних пристроях, вимагаючи значних апаратних ресурсів від сервера та браузера клієнта [12].

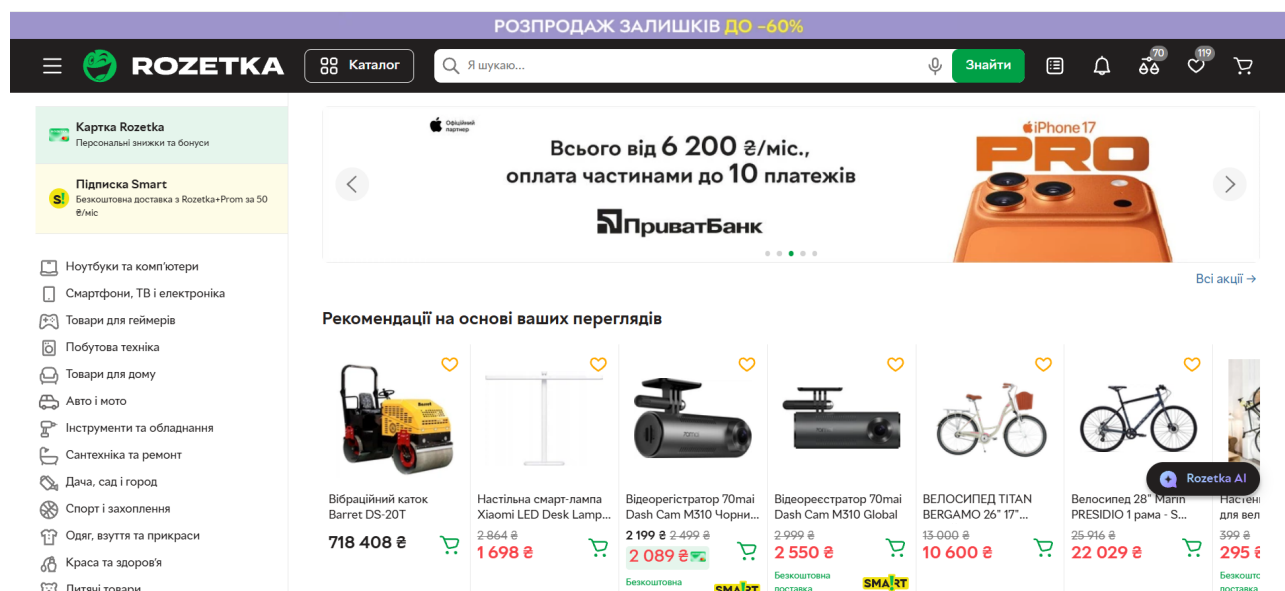


Рис. 1.2 – Інтерфейс головної сторінки інтернет-супермаркету «Rozetka»

Свою чергою, електронний магазин «Allo» (рис. 1.3) орієнтований на портативну цифрову техніку та екосистему «розумного дому». Система характеризується високою швидкістю каталогу, лаконічним дизайном та зручним оформленням замовлення з мінімальною кількістю полів. Попри ці здобутки, архітектура платформи базується на гібридному підході із жорсткою

прив'язкою до закритих Enterprise CMS для забезпечення омніканальності. Така монолітність ускладнює кастомізацію та робить вартість розробки подібного рішення економічно недоцільною для підприємств малого та середнього бізнесу [13].

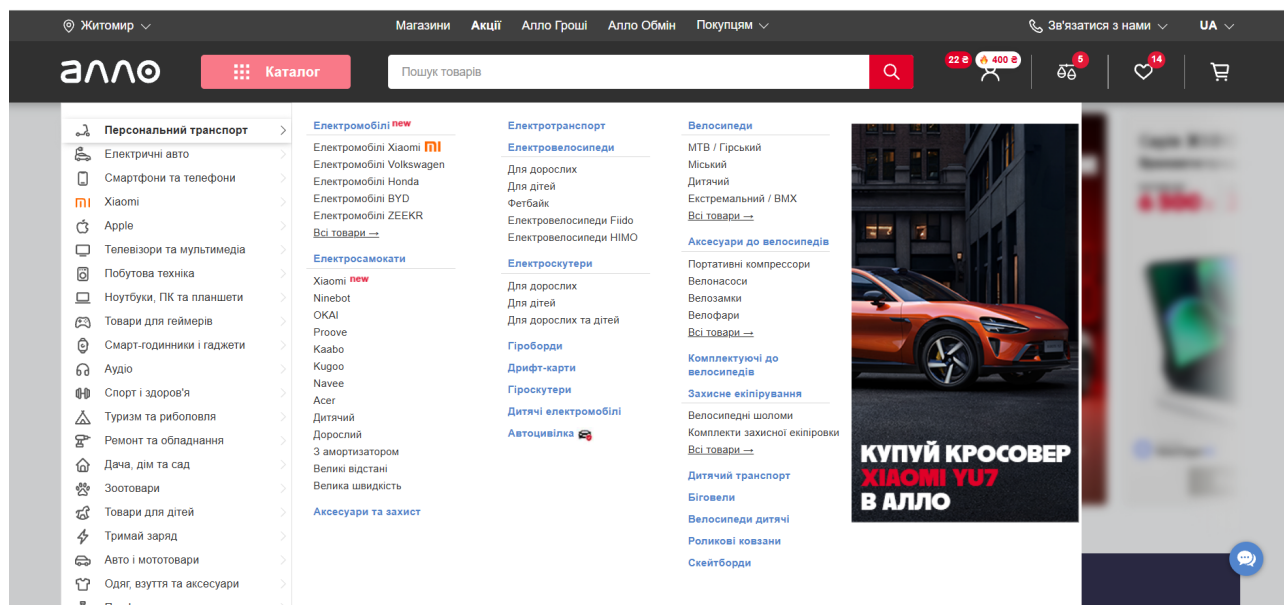


Рис. 1.3 – Відображення каталогу товарів інтернет-магазину «Алло»

Інтернет-магазин «Citrus» (рис. 1.4) відзначається нестандартним та високоінтерактивним інтерфейсом, цільовим чином розрахованим на молодіжну аудиторію. Платформа пропонує якісно реалізований модуль порівняння технічних характеристик пристроїв та зручний кошик покупця, що розгортається у форматі спливаючого вікна без необхідності переходу на окрему сторінку. Разом із тим, наслідком такої високої інтерактивності є надмірне використання масивних JavaScript-бібліотек на фронтенді в поєднанні із загальною монолітною архітектурою серверної частини. Оскільки реалізація платформи базується на принципах, наближених до Single Page Application (SPA), це призводить до істотного зниження загальної швидкості роботи на пристроях низької цінової категорії через створення надмірного навантаження на процесор та оперативну пам'ять клієнтського пристрою під час рендерингу складних анімацій [14].

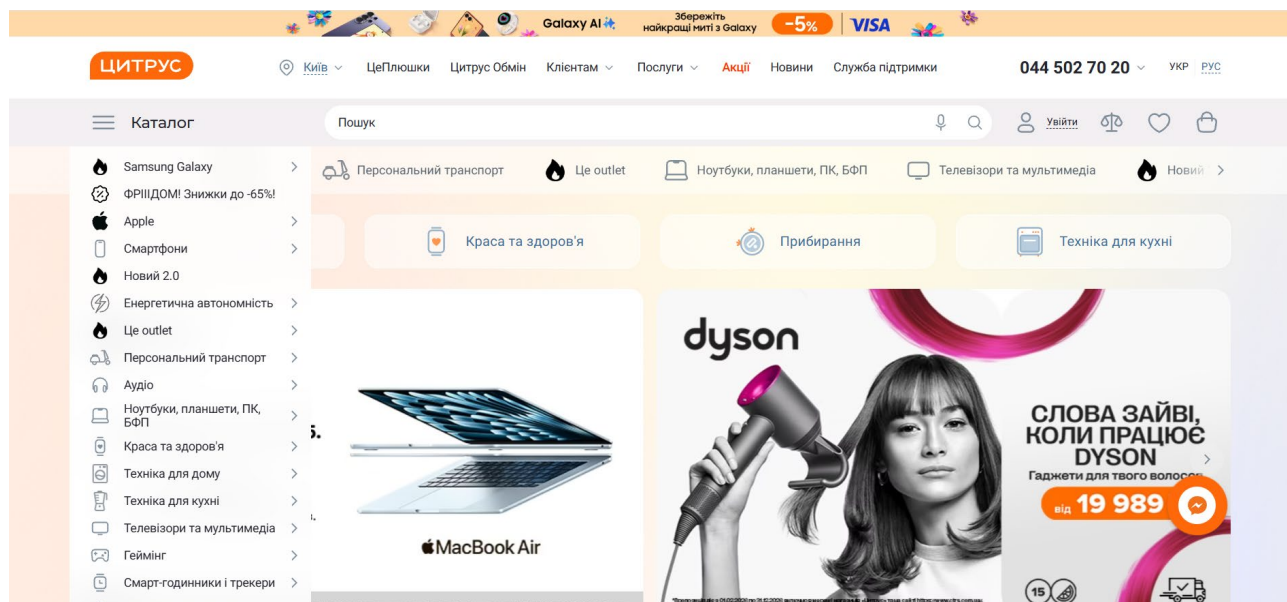


Рис. 1.4 – Інтерфейс картки товару та кошика інтернет-магазину «Citrus»

Для систематизації результатів дослідження та визначення оптимальних функціональних і нефункціональних вимог до проєктуваної системи було сформовано критерії порівняння. Зведені результати порівняльного аналізу за ключовими технічними та економічними критеріями наведено у таблиці 1.1.

Таблиця 1.1

Порівняльна характеристика аналогів

Критерій порівняння	Rozetka	Allo	Citrus	Проектована система
Спеціалізація	Універсальний маркетплейс	Цифрова техніка та дім	Гаджети та аксесуари	Цифрова техніка
Архітектура	Мікросервісна (складна)	Гібридна	Монолітна	MVC (Модульна)
Швидкість роботи	Середня	Висока	Низька (багато скриптів)	Дуже висока
Вартість розробки	Дуже висока	Висока	Висока	Низька
Гнучкість налаштувань	Низька (платформа)	Середня	Середня	Висока

Таким чином, результати аналізу існуючих платформ підтверджують, що розробка власного рішення за архітектурним патерном MVC дозволить

створити легковагову систему, оптимізовану спеціально під реалізацію цифрової техніки. Відмова від зайвого функціоналу, притаманного готовим монолітним CMS, мінімізує вимоги до серверного забезпечення та гарантує високу гнучкість налаштувань під специфічні бізнес-процеси підприємства [15].

1.3. Формування функціональних та нефункціональних вимог до системи

На основі проведеного комплексного системного аналізу предметної області та детального вивчення недоліків існуючих підходів до організації електронної комерції було сформовано розширений перелік вимог до програмного продукту. Проектована система інтернет-магазину цифрової техніки повинна забезпечувати виконання суворого комплексу функцій, які концептуально та архітектурно розподіляються за ролями доступу між клієнтською частиною та закритою адміністративною панеллю.

До функціональних вимог клієнтської частини (Front-office) належить комплекс завдань, спрямованих на забезпечення максимально зручної та інтуїтивно зрозумілої взаємодії споживача з торговельною платформою. Першочергово система повинна генерувати інтерактивний каталог товарів із можливістю гнучкої навігації та багатокритеріальної фільтрації за складною ієрархією категорій, таких як смартфони, планшети, ноутбуки, а також їхні відповідні підкатегорії та супутні аксесуари. Для швидкого знаходження потрібної позиції серед широкого асортименту передбачається реалізація модуля пошуку, який здатний обробляти запити за ключовими словами у комерційній назві пристрою або знаходити точні збіги за унікальним артикулом виробника. При переході до конкретної позиції система має динамічно формувати розгорнуту картку товару. Цей інтерфейсний модуль повинен містити детальний технічний опис специфікацій, високоякісну фотогалерею з можливістю масштабування зображень, актуальний статус наявності пристрою на складі та його роздрібну ціну.

Критично важливим аспектом функціональності клієнтської частини є механізм додавання обраних товарів до віртуального кошика. Цей обчислювальний процес повинен базуватися на надійному сесійному збереженні даних із використанням глобального серверного масиву та алгоритмів серіалізації інформації у формат JSON. Такий підхід дозволить мінімізувати кількість проміжних звернень до бази даних. Користувач повинен мати можливість динамічно редагувати кількість замовлених одиниць або повністю видаляти їх із кошика з миттєвим перерахунком загальної вартості без необхідності повного перезавантаження вебсторінки. Логічним завершенням цього процесу є оптимізований модуль оформлення замовлення, який має складатися з мінімальної кількості кроків. На цьому етапі система повинна збирати необхідні контактні дані клієнта, надавати вибір зручного способу оплати та містити текстове поле для додавання коментарів чи побажань для служби логістики. Для постійних покупців передбачено повноцінну CRM-складову, що дозволяє користувачам реєструвати власні облікові записи, безпечно авторизуватися в системі та отримувати доступ до особистого кабінету з метою відстеження статусів поточних замовлень та аналізу історії попередніх покупок.

Своєю чергою, до функціональних вимог адміністративної частини (Back-office) належить забезпечення повного та безпечного управління торговельним асортиментом через закриту панель керування. Цей інструментарій повинен охоплювати реалізацію повноцінних базових операцій управління даними, а саме створення нових карток товарів, читання та редагування існуючих описів чи цін, а також безпечне видалення неактуальних позицій з бази даних. Разом з тим вимагається наявність гнучкого механізму для керування багаторівневою ієрархією категорій каталогу, що включає створення нових тематичних розділів та зміну їхньої логічної вкладеності. Окремим модулем адміністративної панелі має стати система моніторингу та обробки клієнтських замовлень. Управління життєвим циклом кожної заявки повинно здійснюватися через зміну її статусів,

починаючи від статусу нового замовлення, переходу в стадію обробки чи очікування оплати, і закінчуючи успішним виконанням або скасуванням. Серед іншого адміністративного функціоналу варто виокремити необхідність завантаження та автоматичної оптимізації графічного контенту на сервері, а також управління обліковими записами зареєстрованих користувачів із налаштуванням прав доступу на основі рольової моделі.

Окрім функціональних характеристик, надзвичайно важливу роль відіграє група нефункціональних вимог, які визначають критерії безпеки, продуктивності та експлуатаційної надійності вебзастосунку. З точки зору інформаційної безпеки, архітектура системи повинна гарантувати абсолютний захист реляційної бази даних від атак типу SQL-ін'єкцій. Це досягається завдяки обов'язковому використанню об'єктно-орієнтованого інтерфейсу бази даних та підготовлених запитів на рівні драйвера. Усі паролі користувачів повинні зберігатися виключно у вигляді незворотних криптографічних хеш-сум.

Окрім того, вимагається досягнення високих показників продуктивності. Час генерації сторінки сервером для стандартних запитів до каталогу товарів не повинен перевищувати нормативні показники, що досягається за рахунок глибокої оптимізації SQL-запитів та раціонального використання оперативної пам'яті сервера. Обробка асинхронних клієнтських запитів також має відбуватися з мінімальними затримками для забезпечення плавності роботи інтерфейсу. Додатковою і невід'ємною умовою є дотримання принципів ергономічності завдяки впровадженню адаптивного користувацького інтерфейсу. Візуальне відображення каталогу та елементів управління повинно автоматично масштабуватися, гарантуючи коректну та зручну роботу інтернет-магазину як на широкоформатних моніторах персональних комп'ютерів, так і на екранах мобільних пристроїв. Вирішення зазначеного комплексу завдань вимагає проєктування надійної даталогічної архітектури та застосування

сучасних патернів програмування, що буде детально розглянуто в наступних підрозділах кваліфікаційної роботи.

Висновки до першого розділу

У першому розділі випускної кваліфікаційної роботи виконано комплексний системний аналіз предметної області роздрібної торгівлі цифровою технікою та сформовано інженерне обґрунтування розробки вебзастосунку. За результатами проведеного дослідження зроблено низку відповідних висновків.

Зокрема, було досліджено бізнес-процеси онлайн-комерції та визначено перелік ключових завдань, що підлягають автоматизації, серед яких: управління асортиментом товарів зі складними характеристиками, сесійне забезпечення кошика покупок, а також обробка та зміна життєвих циклів замовлень адміністратором. На основі цього сформовано детальні функціональні вимоги для двох основних ролей доступу до системи — покупця та адміністратора.

Крім того, проведено аналіз існуючих комерційних платформ-аналогів на ринку України, таких як «Rozetka», «Allo» та «Citrus». Під час аналізу виявлено їхні основні архітектурні обмеження, які полягають у надмірності коду, складності кастомізації, високій вартості супроводу та значних вимогах до обчислювальних ресурсів сервера, що, своєю чергою, підтвердило доцільність розробки індивідуального програмного рішення.

Для забезпечення слабкої зв'язності компонентів системи було обґрунтовано вибір архітектурного патерну MVC як оптимального шаблону проектування. У результаті прийнято рішення про створення індивідуального легковагового програмного ядра (фреймворку) на основі концепції Front Controller (файл index.php) та модуля маршрутизації на регулярних виразах, що гарантуватиме високу швидкодію вебресурсу та повний контроль над вихідним кодом.

Підсумовуючи, сформовано та обґрунтовано технологічний стек розробки, який базується на серверній платформі LAMP із застосуванням

СУБД MySQL та об'єктно-орієнтованого інтерфейсу PDO для надійного захисту від SQL-ін'єкцій. Для реалізації автозавантаження класів інтегровано менеджер пакетів Composer за стандартом PSR-4, що забезпечує відповідність архітектури сучасним індустріальним нормам. Водночас розроблено точну специфікацію мінімальних та рекомендованих системно-апаратних вимог до серверного і клієнтського середовища експлуатації вебзастосунку.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ІНТЕРНЕТ-МАГАЗИНУ ЦИФРОВОЇ ТЕХНІКИ

2.1. Вибір архітектури та технологічного інструментарію розробки

Проектування сучасних вебзастосунків у сфері електронної комерції вимагає обґрунтованого вибору архітектурного підходу, який безпосередньо визначає стабільність, швидкодію, безпеку та вартість подальшої технічної підтримки програмного продукту. На етапі аналізу архітектурних рішень для інтернет-магазину цифрової техніки було розглянуто три базові концепції побудови програмних систем: монолітну архітектуру, мікросервісну архітектуру та шаблон проектування Model-View-Controller (MVC) [16].

Дослідження класичної монолітної архітектури, за якої всі функціональні компоненти системи (інтерфейс користувача, бізнес-логіка, модулі авторизації, обробки замовлень та взаємодії з базою даних) об'єднані в єдиний програмний комплекс та виконуються в одному контексті, підтвердило її відносну простоту початкового розгортання, зручність наскрізного тестування та високу швидкість міжмодульної взаємодії всередині оперативної пам'яті. Однак сильна зв'язність компонентів (tight coupling) призводить до того, що внесення змін або виправлення помилок в одному модулі (наприклад, у системі кошика) може непередбачувано порушити працездатність інших підсистем. Окрім того, монолітна архітектура унеможлиблює вибіркове масштабування найбільш навантажених частин вебресурсу, що суттєво знижує загальну відмовостійкість системи під час пікових навантажень [17].

Альтернативою виступає мікросервісна архітектура, що передбачає декомпозицію вебзастосунку на набір слабкопов'язаних, автономних сервісів, кожен з яких відповідає за окрему бізнес-функцію (наприклад, окремі мікросервіси для каталогу товарів, обробки кошика та автентифікації користувачів). Кожен мікросервіс може мати власну базу даних та взаємодіяти з іншими компонентами через структуровані протоколи передачі даних, зокрема REST API або брокери повідомлень. Попри такі переваги, як висока

масштабованість, технологічна гнучкість та ізоляція помилок, цей підхід характеризується надзвичайною складністю проєктування, високими вимогами до кваліфікації команди розробників та суттєвими накладними витратами серверних ресурсів на організацію мережевої взаємодії. Саме тому для проєктів малого та середнього бізнесу впровадження мікросервісної архітектури є економічно та інженерно недоцільним [18].

З огляду на виявлені обмеження, архітектурний патерн MVC (Model-View-Controller) було визначено як оптимальний інженерний компроміс для систем електронної комерції середнього масштабу. Він поєднує переваги простоти розгортання з чітким розділенням відповідальності між трьома основними шарами системи, що мінімізує зв'язність коду (loose coupling) та полегшує його супровід [19].

Аналізуючи інструменти реалізації патерну MVC, варто зазначити, що часто розглядаються готові важкі програмні фреймворки (такі як Laravel або Symfony). Вони надають розробнику потужний інструментарій, проте містять велику кількість універсальних абстракцій та надлишкового коду, який створює додаткове навантаження на сервер, знижуючи швидкість первинної відповіді. Беручи це до уваги, для розроблюваного інтернет-магазину цифрової техніки було прийнято рішення про створення індивідуального легковагового програмного каркаса на базі чистої мови PHP, що реалізує патерн MVC. Такий підхід дозволяє досягти максимальної швидкодії ядра системи, усунути зайві рівні абстракцій та забезпечити повний контроль над безпекою вихідного коду.

Спроектowana архітектура розробленої системи базується на принципі «єдиної точки входу» (Front Controller), реалізованої через файл index.php. Цей файл перехоплює всі запити користувачів, здійснює базові налаштування середовища та ініціалізує системний маршрутизатор (Router). Маршрутизатор, своєю чергою, аналізує вхідний URI-запит, зіставляє його з конфігураційними правилами на основі регулярних виразів та передає управління відповідному контролеру [20].

Усі основні логічні компоненти архітектури зосереджені в базовій директорії `app` та чітко структуровані відповідно до парадигми MVC. Зокрема, інформаційний шар системи представлений моделями даних (класи `Product`, `Category`, `User`, `Order`), які розташовані у директорії `app/models/` та інкапсулюють у собі взаємодію з базою даних, валідацію вхідних параметрів і реалізацію ключової бізнес-логіки інтернет-магазину, як-от розрахунок вартості чи перевірка наявності товару. За формування графічного інтерфейсу користувача та візуалізацію підготовленої інформації відповідає шар представлень (`View`), що містить структуровані HTML-шаблони з мінімальними вкрапленнями PHP-коду в папці `app/views/`. Зв'язуючою ланкою виступає шар контролерів (класи `ProductController`, `CartController`, `AdminController` у директорії `app/controllers/`), які виконують роль координаторів: вони приймають HTTP-запити від модуля маршрутизації, ініціюють виклик відповідних методів моделей для обробки даних та підключають необхідні шаблони для генерації фінальної відповіді клієнту. Крім того, архітектура включає допоміжні системні компоненти ядра у папці `app/components/` (такі як `Db`, `Autoload`, `Cart`, `Router`), що забезпечують базову життєдіяльність власного фреймворку, зокрема безпечне підключення до бази даних за патерном `Singleton`, автозавантаження класів без застосування сторонніх утиліт, менеджмент сесій кошика покупця та маршрутизацію запитів [21].

Взаємозв'язок та логіка взаємодії зазначених архітектурних компонентів під час обробки запиту користувача відображені на структурній схемі (рис. 2.1).

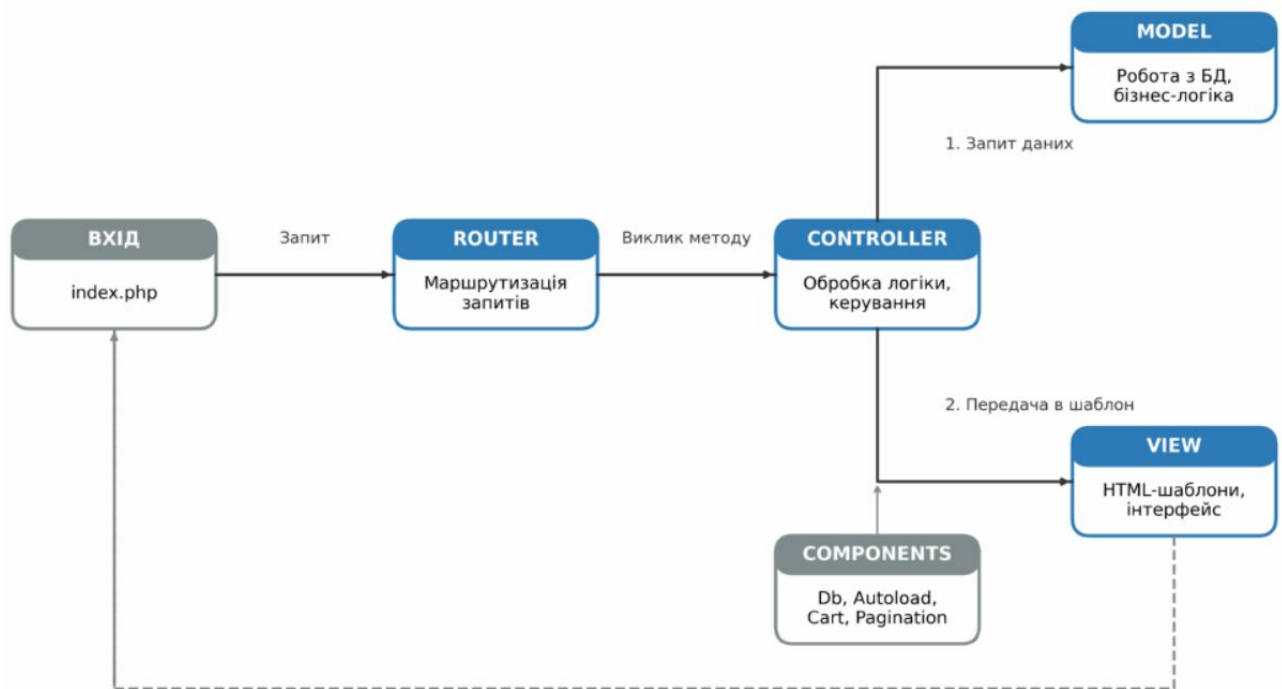


Рис.2.1 – Архітектурна схема розробленого MVC-фреймворку

Для деталізації зон відповідальності кожного елемента створеної індивідуальної архітектури вебзастосунку було сформовано узагальнену структуру компонентів програмного каркаса, яку наведено у відповідній таблиці (табл. 2.1).

Таблиця 2.1

Структура та призначення компонентів MVC-фреймворку

Компонент	Директорія	Основні класи / Файли	Призначення (Відповідальність)
Model (Модель)	app/models/	Product, Category, User, Order	Пряма взаємодія з БД, обробка даних та реалізація бізнес-логіки.
View (Вигляд)	app/views/	HTML-шаблони (.php)	Візуалізація даних, формування інтерфейсу користувача.
Controller (Контролер)	app/controllers/	ProductController, CartController, AdminController	Обробка вхідних даних, виклик методів моделей, підключення представлень.
Components (Компоненти)	app/components/	Db, Autoload, Cart, Router	Забезпечення роботи ядра системи: з'єднання з БД, маршрутизація, сесії.

Для якісної програмної реалізації високопродуктивного інтернет-магазину цифрової техніки та забезпечення його надійної експлуатації

критично важливим етапом є обґрунтований вибір технологічного стеку. Розроблюване програмне забезпечення має повною мірою відповідати сучасним індустріальним стандартам інженерії програмного забезпечення, а також суворим критеріям безпеки, кросплатформності та загальної швидкодії системи. Саме тому як базову інфраструктуру для реалізації вебзастосунку було обрано надійний стек вебтехнологій LAMP (Linux, Apache, MySQL, PHP), який на сьогоднішній день є загальновизнаним індустріальним стандартом для розгортання сучасних розподілених вебдодатків будь-якого рівня складності [22].

Базовим інструментарієм серверної розробки у межах цього проєкту виступає мова програмування PHP версії 8.x, яка фактично закріпила за собою статус стандарту де-факто для проєктування динамічних платформ електронної комерції. Ця версія забезпечує надзвичайно високу продуктивність виконання серверних сценаріїв, що стало можливим завдяки впровадженню вдосконаленого механізму JIT-компіляції (Just-In-Time). Окрім цього, PHP 8.x на рівні ядра підтримує сувору типізацію аргументів і повертаваних значень, систему атрибутів та нові синтаксичні конструкції, що сукупно дозволяє проєктувати максимально чистий, безпечний та зручний для подальшого рефакторингу об'єктно-орієнтований код. Відмова від використання важких готових монолітних фреймворків на користь нативного (чистого) PHP під час створення індивідуального каркасу дає змогу повністю усунути зайві рівні програмних абстракцій, тим самим досягаючи максимальної швидкодії системи під час обробки HTTP-запитів [23].

У ролі системи управління базами даних (СУБД) було інтегровано надійну реляційну базу даних MySQL. Таке рішення гарантує повну цілісність, збереженість та несуперечливість оброблюваної інформації відповідно до суворих принципів транзакційності ACID (Atomicity, Consistency, Isolation, Durability). Дотримання цих принципів виступає критично важливим аспектом під час виконання фінансових операцій, ведення обліку товарних залишків на

складі та фіксації замовлень з боку покупців. При цьому взаємодія серверних скриптів із базою даних реалізується виключно через об'єктно-орієнтований інтерфейс PDO (PHP Data Objects). Безпосереднє використання підготовлених запитів (Prepared Statements) на рівні розширення PDO гарантує абсолютну безпеку виконання SQL-команд та забезпечує надійний захист спроектованої системи від найбільш поширених веб-уразливостей, зокрема від несанкціонованих SQL-ін'єкцій [24].

Для організації професійної та логічної структури проєкту, а також для ефективного керування зовнішніми компонентами, до екосистеми розробки було інтегровано Composer — сучасний менеджер залежностей для мови програмування PHP. У контексті розроблюваного інтернет-магазину цей інструмент використовується з метою налаштування механізму автоматичного завантаження класів за загальноприйнятим міжнародним стандартом PSR-4 (PHP Standard Recommendation). Такий інженерний крок дозволяє повністю відмовитися від застарілих і громіздких конструкцій ручного підключення файлів (require/include), що робить архітектуру всього проєкту стандартизованою, гнучкою та максимально зручною для подальшого масштабування чи супроводу іншими командами інженерів [25].

Водночас реалізована індивідуальна архітектура MVC створює умови для чіткого структурного розділення бізнес-логіки (Models), алгоритмів маршрутизації та обробки запитів (Controllers) від користувацького інтерфейсу (Views). Такий комплексний підхід до проєктування значно спрощує процес подальшої підтримки, проведення модульного тестування та поступове розширення програмного коду розробленої платформи електронної комерції [26].

З метою забезпечення стабільного розгортання та подальшої комерційної експлуатації розробленого інтернет-магазину було визначено перелік сумісних технічних, системних та апаратних вимог, що висуваються як до серверної

інфраструктури, так і до клієнтських пристроїв кінцевих користувачів. Детальну специфікацію цих вимог наведено у відповідній таблиці (табл. 2.2).

Таблиця 2.2

Вимоги до технічного та програмного забезпечення вебзастосунку

Тип інфраструктури	Компонент забезпечення	Нормативні вимоги та характеристики
Серверна частина (Хостинг / Сервер)	Операційна система	Linux (Ubuntu Server 22.04 LTS / Debian 12) або Windows 10/11 (для розробки)
	Вебсервер	Apache 2.4 або Nginx 1.22 з модулем обробки PHP-FPM
	Програмне середовище	PHP 8.1 (або вище) з розширеннями pdo_mysql, json, session
	Система управління БД	MySQL 8.0 (або сумісна версія MariaDB 10.6)
	Процесор (CPU)	Базовий 2-ядерний процесор з частотою від 2.0 ГГц
	Оперативна пам'ять (RAM)	Не менше 2 Гб (рекомендовано 4 Гб для кешування БД)
	Дискова підсистема	Накопичувач SSD, від 5 Гб вільного простору (під СУБД та зображення)
Клієнтська частина (Користувач / Адмін)	Програмне забезпечення	Актуальна версія браузера з підтримкою HTML5/CSS3 (Chrome, Firefox, Safari)
	Апаратне забезпечення	Будь-який ПК або мобільний пристрій з RAM від 1 Гб та дисплеєм від 320px

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

Проектування сучасних розподілених вебзастосунків у сфері електронної комерції вимагає обов'язкового застосування методологій об'єктно-орієнтованого аналізу та проектування (ООАП). Першочерговим завданням цього етапу є точна формалізація вимог користувачів, аналіз архітектурних обмежень та моделювання функціональних сценаріїв взаємодії за допомогою

мови графічного моделювання UML 2.0. Такий підхід дозволяє розробнику ефективно абстрагуватися від конкретної програмно-апаратної реалізації, чітко визначити межі проєкту і логічно структурувати базову бізнес-логіку системи [27].

На етапі аналізу вимог до інтернет-магазину цифрової техніки було виділено три базові категорії зовнішніх сутностей (акторів), які безпосередньо взаємодіють із програмним комплексом. Першою категорією є гість (неавторизований користувач) — відвідувач сайту, який має доступ до публічного контенту, загального каталогу товарів, пошукових механізмів та може здійснювати виключно анонімне формування кошика покупок. Другою категорією виступає покупець (авторизований користувач) — клієнт, який успішно пройшов процедуру реєстрації та автентифікації, що відкриває йому повноцінний доступ до персональних інструментів, зокрема до особистого кабінету та детальної історії замовлень. Третьою категорією визначено адміністратора (керуючого користувача) — актора із підвищеними привілеями доступу, який безпосередньо здійснює контроль за життєвим циклом замовлень, управлінням контентом платформи та критичними налаштуваннями довідників СУБД.

Для наочної деталізації взаємодії визначених акторів із вебзастосунком було розроблено діаграму варіантів використання (Use Case Diagram), яка є графічним представленням усіх функціональних можливостей системи.

Важливою вимогою інженерного проєктування є також визначення суворих вимог до часових характеристик реакції системи на дії користувача. Для забезпечення високого рівня користувацького досвіду (UX) та мінімізації показника відмов було встановлено низку нормативних обмежень швидкодії. Зокрема, час генерації сторінки сервером (TTFB) для стандартних GET-запитів до каталогу товарів не повинен перевищувати 250 мс; час обробки асинхронних запитів (наприклад, під час додавання товару до кошика через технологію AJAX) має становити не більше 150 мс; водночас час виконання транзакційних

операцій, таких як запис замовлення у СУБД або перевірка сесії авторизації, обмежується показником до 300 мс.

Для глибокої деталізації логіки роботи вебзастосунку розроблено три розгорнуті текстові специфікації (сценарії) для ключових варіантів використання: «Оформлення замовлення» (табл. 2.3), «Управління товарним асортиментом» (табл. 2.4) та «Авторизація користувача» (табл. 2.5).

Таблиця 2.3

Оформлення замовлення

Елемент специфікації	Опис та характеристики прецеденту
Назва прецеденту	Оформлення замовлення цифрової техніки (Checkout)
Актори	Гість, Покупець
Передумови	У віртуальному кошику користувача (\$_SESSION['products']) є хоча б один товар.
Основний потік подій	1) користувач переходить за посиланням оформлення замовлення; 2) система зчитує масив сесії, розраховує загальну вартість та відображає форму введення даних; 3) користувач заповнює поля: ПІБ, номер телефону, коментар, Email; 4) користувач натискає кнопку «Підтвердити замовлення»; 5) контролер викликає методи валідації введених даних; 6) система перевіряє статус авторизації (якщо користувач увійшов, підтягується його user_id); 7) склад кошика конвертується у рядок JSON; 8) викликається метод моделі для виконання SQL-запиту INSERT у таблицю product_order; 9) сесія кошика очищується, система відправляє Email адміністратору та перенаправляє користувача.
Альтернативні потоки	Відхилення на кроці 5: якщо валідація не пройдена (некоректний формат телефону чи Email), процес переривається, а система повертає користувача на форму з виведенням масиву помилок.
Постумови	Створено новий рядок у таблиці product_order зі статусом «Нове».

Таблиця 2.4

Управління товарним асортиментом

Елемент специфікації	Опис та характеристики прецеденту
Назва прецеденту	Модерація карток товарів (Реалізація CRUD операцій)
Актори	Адміністратор
Передумови	Актор успішно пройшов автентифікацію та має активну сесію з роллю admin.
Основний потік подій	1) адміністратор переходить у розділ панелі керування «Управління товарами»; 2) система робить вибірку з таблиці product та відображає табличний список; 3) актор вибирає дію (створення, редагування або видалення); 4) при створенні або редагуванні актор заповнює форму (назва, ціна, артикул, опис, категорія); 5) після натискання кнопки «Зберегти» система валідує дані та завантажує графічний файл; 6) модель виконує відповідний SQL-запит до бази даних; 7) система повертає адміністратора до загального списку товарів.
Альтернативні потоки	Відхилення на кроці 1: якщо сесія не має прав адміністратора, система блокує доступ та виконує перенаправлення на сторінку входу.
Постумови	Фізичні зміни успішно внесені у таблицю product СУБД MySQL.

Таблиця 2.5

Авторизація користувача

Елемент специфікації	Опис та характеристики прецеденту
Назва прецеденту	Автентифікація користувача в системі (Login)
Актори	Гість, Адміністратор
Передумови	Користувач вже зареєстрований у базі даних (у таблиці user є відповідний Email).

Елемент специфікації	Опис та характеристики прецеденту
Основний потік подій	1) користувач переходить на сторінку входу та вводить Email і пароль; 2) контролер приймає POST-запит і передає дані в модель User; 3) система виконує вибірку запису та порівняння паролів (checkUserData); 4) у разі успіху система записує у сесію ідентифікатор користувача (auth); 5) відбувається перенаправлення на відповідну закриту сторінку.
Альтернативні потоки	Відхилення на кроці 3: якщо дані не збігаються, система фіксує помилку автентифікації, а форма відображається знову з текстом попередження.
Постумови	Ініціалізовано масив \$_SESSION['user'], надано відповідні права доступу.

Після деталізації функціональних сценаріїв здійснюється плавний перехід до проектування об'єктно-орієнтованої структури вебзастосунку. Статична структура та архітектурна зв'язність усіх модулів системи описується за допомогою діаграми класів (Class Diagram) згідно зі стандартом UML 2.0.

Для забезпечення модульності архітектура проєкту була чітко структурована відповідно до стандарту автозавантаження PSR-4 із залученням сучасного менеджера залежностей Composer. Системні компоненти ядра фреймворку, що відповідають за базову життєдіяльність застосунку, згруповані у просторі імен App\components. Серед них ключову роль відіграє клас Router, який безпосередньо забезпечує декомпозицію та обробку вхідних URL-адрес. Цей клас містить приватну властивість \$routes для збереження масиву маршрутів сайту та метод run(), що зчитує рядок URI, знаходить збіги за допомогою регулярних виразів (preg_match), визначає назву цільового контролера з його екшеном і динамічно запускає відповідний об'єкт через

функцію `call_user_func_array()`. Ініціалізація з'єднання з реляційною СУБД MySQL покладена на клас `Db`, чий статичний фабричний метод `getConnection()` зчитує параметри хостингу, створює новий об'єкт PDO та налаштовує кодування UTF-8 разом із суворим режимом обробки винятків. Управління тимчасовим сховищем обраних товарів реалізовано у класі `Cart`, який інкапсулює логіку роботи з сесіями (`$_SESSION['products']`) через набір статичних методів для додавання товарів, підрахунку їхньої кількості, отримання загального вмісту кошика та його повного очищення.

Сутності шару моделей даних локалізовані у просторі імен `App\models`. Зокрема, клас `Product` керує бізнес-логікою та SQL-операціями над асортиментом техніки, використовуючи такі методи, як `getLatestProducts()` для отримання останніх доданих товарів, `getProductsListByCategory()` для вибірки з урахуванням пагінації (за допомогою операторів `LIMIT` та `OFFSET`), а також `getProductById()` для виконання підготовлених запитів за первинним ключем. Менеджмент облікових записів та перевірку безпеки делеговано класу `User`, що містить методи реєстрації, автентифікації (`auth`), перевірки даних користувача та широкий набір валідаторів. Своєю чергою, за фіксацію фінальних замовлень відповідає клас `Order`, метод `save()` якого успішно трансформує масив обраних товарів у формат JSON та здійснює відповідний запис у таблицю `product_order`.

Обробка клієнтських запитів здійснюється сутностями шару контролерів, які розміщені у просторі імен `App\controllers`. Зокрема, клас `ProductController` містить метод `actionView()`, що виконує вибірку супутніх даних (категорій та поточного стану кошика) і підключає відповідний HTML-шаблон для коректного відображення картки товару. Координацію процесів оформлення покупок забезпечує `CartController`, де метод `actionCheckout()` обробляє вхідні POST-запити, ініціює валідацію введених даних через модель користувача та зберігає замовлення шляхом виклику методу збереження відповідної моделі.

Чіткий поділ обов'язків між наведеними класами та наскрізне застосування підготовлених запитів PDO гарантує системі високу безпеку та

повну відповідність структури проєкту сучасним індустріальним стандартам веброзробки.

2.3. Розробка бази даних системи

Ефективність функціонування будь-якого інтернет-магазину критично залежить від швидкості обробки інформації та надійності її збереження. Проєктування інформаційного забезпечення вебзастосунку здійснювалося на основі реляційної моделі даних з використанням системи управління базами даних MySQL та сучасного рушія збереження таблиць InnoDB, що гарантує підтримку транзакцій та забезпечення суворої цілісності зв'язків (Referential Integrity).

Процес проєктування бази даних розроблюваної платформи складався з двох послідовних етапів: інфологічного (концептуального) та даталогічного (фізичного) моделювання. На етапі нормалізації всі відношення були приведені до третьої нормальної форми (3НФ). Це дозволило успішно усунути транзитивні залежності, мінімізувати надмірність даних та гарантувати їхню несуперечливість під час виконання транзакційних операцій оновлення чи видалення.

Фізична схема спроектованої бази даних складається з чотирьох основних взаємопов'язаних таблиць, які повністю покривають усі функціональні потреби платформи електронної комерції: category (Категорії), product (Товари), user (Користувачі) та product_order (Замовлення).

Базовою сутністю для побудови ієрархії каталогу є таблиця category, яка містить перелік розділів магазину. Оптимізація структури полів та належне налаштування індексів первинних ключів дозволяють суттєво знизити навантаження на сервер СУБД при одночасному опрацюванні великої кількості запитів до деревоподібної структури меню сайту. Фізичні типи даних та загальну структуру цього відношення наведено у відповідній таблиці (табл. 2.6).

Таблиця 2.6.

Структура таблиці category (Категорії)

Поле	Тип даних	Опис та призначення атрибута
id	INT (PK)	Унікальний ідентифікатор категорії (Primary Key, Auto Increment)
name	VARCHAR(255)	Назва категорії ("Смартфони", "Ноутбуки")
sort_order	INT	Порядковий номер для сортування при виведенні в меню
status	INT	Статус видимості: відображати на сайті (1) чи приховати (0)

Центральним елементом бази даних є таблиця product, що зберігає деталізовану інформацію про кожну одиницю цифрової техніки. Ця таблиця логічно пов'язана з таблицею категорій зв'язком типу «один-до-багатьох» (1:∞) через зовнішній ключ category_id. Специфікацію структури таблиці товарів наведено нижче (табл. 2.7).

Таблиця 2.7

Структура таблиці product (Товари)

Поле	Тип даних	Опис та призначення атрибута
id	INT (PK)	Унікальний ідентифікатор товару (Primary Key)
name	VARCHAR(255)	Комерційна назва моделі техніки
code	VARCHAR(255)	Унікальний артикул виробника для внутрішнього обліку
price	FLOAT	Актуальна роздрібна ціна товару
category_id	INT	Зовнішній ключ (Foreign Key), що вказує на ID категорії
brand	VARCHAR(255)	Бренд виробника (Apple, Samsung)
description	TEXT	Детальний опис та технічні характеристики
availability	INT	Статус наявності на складі (1 — є в наявності, 0 — немає)
is_new	INT	Маркер "Новинка" для маркетингового просування (1/0)

Для візуалізації логічного зв'язку між товарним асортиментом та структурою каталогу було побудовано ER-діаграму підсистеми каталогу, яку відображено на структурній схемі (рис. 2.2).

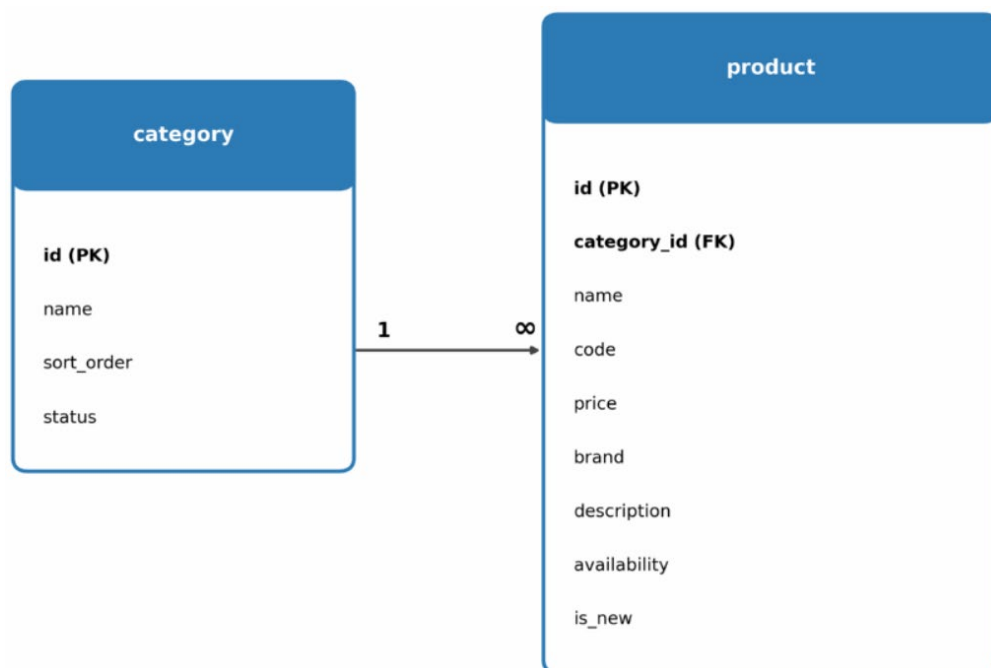


Рис. 2.2 – Схема підсистеми каталогу

Для повноцінної реалізації CRM-функцій інтернет-магазину було спроектовано таблицю user, яка безпосередньо відповідає за збереження облікових записів зареєстрованих клієнтів та адміністративного персоналу. Важливою інженерною вимогою є те, що паролі користувачів зберігаються виключно у вигляді криптографічних хеш-сум, що унеможливорює їх компрометацію у випадку несанкціонованого доступу до бази даних. Структуру цієї таблиці наведено у специфікації (табл. 2.8).

Таблиця 2.8

Структура таблиці user (Користувачі)

Поле	Тип даних	Опис та призначення атрибута
id	INT (PK)	Унікальний ідентифікатор користувача (Primary Key)
first_name	VARCHAR(255)	Ім'я користувача
email	VARCHAR(255)	Е-mail адреса (використовується як унікальний логін)

Поле	Тип даних	Опис та призначення атрибута
password	VARCHAR(255)	Криптографічний хеш пароля (для гарантування безпеки)
role	VARCHAR(50)	Роль доступу до системи ('admin' або 'user')
phone	VARCHAR(255)	Контактний телефон клієнта

Фіксація фактів здійснення покупок реалізується через таблицю `product_order`. Інженерною особливістю архітектури цієї таблиці є обґрунтована відмова від створення додаткової зв'язкової таблиці «багато-до-багатьох» для товарів. Натомість використовується текстове поле `products`, у якому масив обраних товарів та їхня кількість серіалізуються у формат JSON (наприклад, `{"45":1, "22":2}`). Таке архітектурне рішення значно прискорює виконання SQL-запитів під час оформлення покупки та суттєво спрощує загальну логіку системи. Структуру таблиці замовлень наведено у відповідному переліку (табл. 2.9).

Таблиця 2.9

Структура таблиці `product_order` (Замовлення)

Поле	Тип даних	Опис та призначення атрибута
id	INT (PK)	Номер замовлення (Primary Key)
user_name	VARCHAR(255)	ПІБ клієнта (для доставки)
user_phone	VARCHAR(255)	Контактний телефон для зв'язку
user_email	VARCHAR(255)	Електронна пошта клієнта
user_comment	TEXT	Коментар до замовлення від покупця
user_id	INT	ID зареєстрованого користувача (або NULL для гостей)
products	TEXT	JSON-масив товарів кошика
status	INT	Статус обробки (1 — нове, 2 — в роботі, 3 — виконано)

Безпосередній взаємозв'язок між зареєстрованим користувачем та його замовленнями (реалізований як відношення 1:∞ через зовнішній ключ `user_id`) наочно продемонстровано на схемі підсистеми замовлень (рис. 2.3).

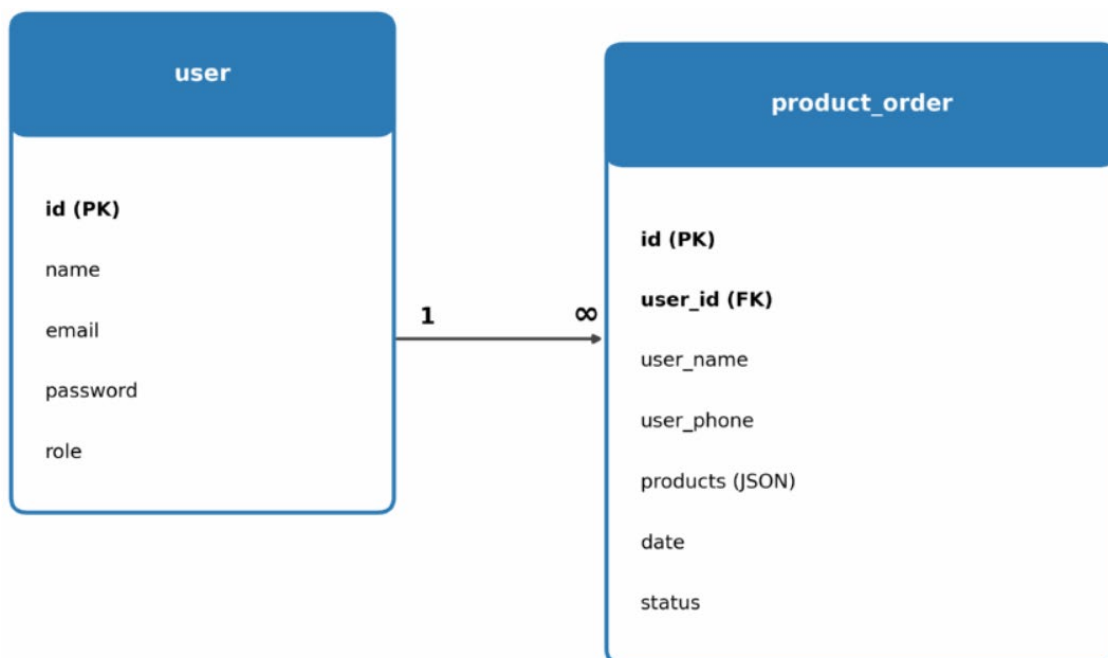


Рис. 2.3 – Схема підсистеми замовлень

Таким чином, спроектована структура бази даних є оптимальною для обраної предметної області. Усі SQL-запити до наведених таблиць здійснюються виключно через об'єктно-орієнтований інтерфейс PDO з використанням підготовлених виразів, що гарантує системі високу швидкість вибірки даних та їхню абсолютну безпеку.

2.4. Проектування та реалізація алгоритмів роботи системи

Функціонування спроектованого інтернет-магазину цифрової техніки забезпечується сукупністю алгоритмів, які реалізують бізнес-логіку платформи, маршрутизацію запитів та взаємодію з базою даних. Завдяки послідовному використанню патерну MVC, обробка даних у системі є строго декомпованою.

Основним механізмом, що ініціює роботу всього вебзастосунку, виступає алгоритм маршрутизації (Routing). Оскільки спроектована система використовує архітектурний принцип єдиної точки входу (Front Controller), абсолютно всі HTTP-запити перенаправляються вебсервером на головний файл index.php, який, своєю чергою, ініціалізує виклик методу run() системного компонента Router.

Безпосередній алгоритм маршрутизації запитів реалізується через чітку послідовність логічних операцій. На початковому етапі відбувається отримання поточного рядка запиту (URI) за допомогою виклику глобального масиву `$_SERVER['REQUEST_URI']`. Після цього здійснюється ітераційний обхід конфігураційного масиву маршрутів (`$routes`), де ключами виступають регулярні вирази (наприклад, `product/([0-9]+)`), а відповідними значеннями є внутрішні системні шляхи. Під час обходу виконується порівняння поточного URI із заданими шаблонами за допомогою вбудованої функції `preg_match()`. У разі успішного знаходження збігу система отримує відповідний внутрішній шлях та за допомогою функції `explode()` розбиває його на окремі сегменти. За логікою фреймворку, перший сегмент завжди визначає назву класу цільового контролера (наприклад, `ProductController`), другий сегмент вказує на назву методу-екшену (наприклад, `actionView`), а всі решта сегментів передаються далі як аргументи у масив параметрів (`$parameters`). На завершальному етапі маршрутизатор динамічно створює екземпляр визначеного контролера через конструкцію `new $controllerName` та викликає потрібний метод із передачею масиву параметрів за допомогою базової функції `call_user_func_array()`. Важливим аспектом експлуатаційної надійності описаного алгоритму є вбудований превентивний механізм обробки виняткових ситуацій та перевірки цілісності файлової структури. Перед ініціалізацією контролера система автоматично перевіряє фізичну наявність відповідного файлу класу в директорії застосунку. Якщо файл успішно знайдено, виконується його підключення, інакше алгоритм зупиняє подальше виконання скрипта. Крім того, у разі повної відсутності збігів поточного клієнтського запиту з будь-яким із визначених конфігураційних шаблонів, система генерує стандартну HTTP-відповідь сервера зі статусом відсутності ресурсу. При цьому користувач перенаправляється на ізольовану сторінку помилки з відповідним інформаційним повідомленням. Такий інженерний підхід унеможливорює виникнення неконтрольованих критичних збоїв програмного ядра під час спроб

звернення до неіснуючих сторінок. Візуалізовану блок-схему алгоритму маршрутизації запитів наведено на відповідному рисунку (рис. 2.4).

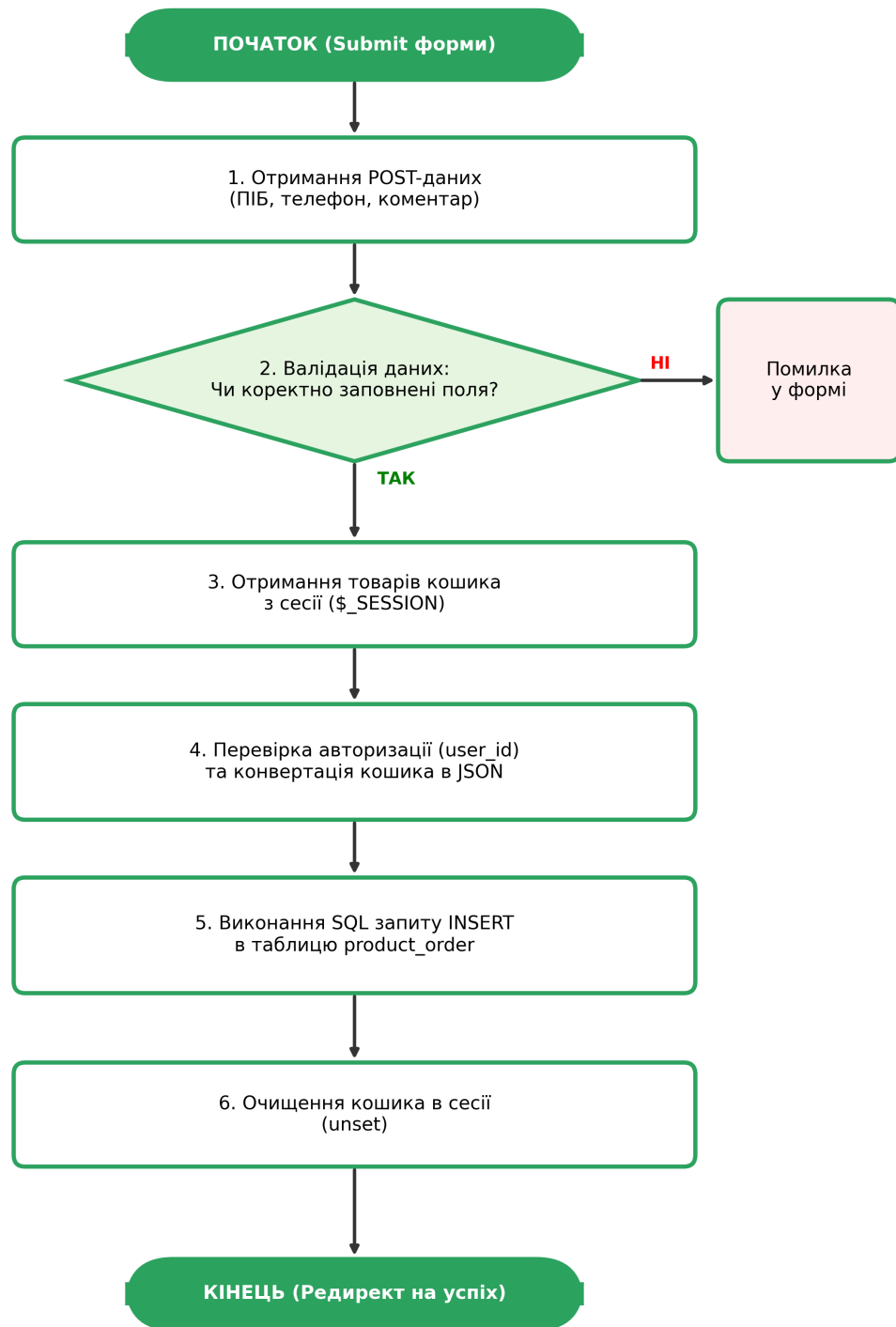


Рис. 2.4 – Блок-схема алгоритму маршрутизації запитів

Другим критично важливим структурним елементом є алгоритм функціонування віртуального кошика покупця. З метою забезпечення максимальної швидкодії системи та уникнення надлишкових звернень до бази даних, проміжне збереження обраних товарів реалізовано виключно на рівні

серверних сесій (через глобальний масив `$_SESSION['products']`) у межах спеціалізованого класу `Cart`. Алгоритм додавання товару до кошика, реалізований у методі `addProduct()`, ініціюється з отримання ідентифікатора конкретного товару (`$id`) та його обов'язкового приведення до цілочисельного типу за допомогою функції `intval()`. Далі система перевіряє наявність масиву `products` у глобальній змінній `$_SESSION`. За допомогою функції `array_key_exists()` виконується логічна перевірка щодо того, чи був цей товар доданий до кошика раніше. Якщо товар уже існує в масиві сесії, його кількісний лічильник інкрементується на одиницю через звернення `$productsInCart[$id]++`. У протилежному випадку, якщо товару ще немає, створюється новий елемент масиву із початковим значенням кількості, що дорівнює одиниці. Наприкінці оновлений масив перезаписується у поточну сесію, після чого ініціюється виклик методу `countItems()` для повернення загальної актуальної кількості товарів, що дозволяє миттєво оновити лічильник у візуальній шапці сайту за допомогою асинхронної технології AJAX.

Фінальним етапом воронки продажів виступає алгоритм оформлення та збереження замовлення (`Checkout`). Цей алгоритм акумулює попередньо збережені дані з кошика, інтегрує їх із персональною інформацією клієнта та здійснює підсумковий транзакційний запис у відповідну реляційну таблицю бази даних. Процес обробки замовлення у системі інтернет-магазину розпочинається з того, що контролер `CartController` отримує POST-запит із введеними даними клієнта (ПІБ, телефон, електронна пошта, коментар) безпосередньо з форми оформлення. Після цього система виконує серію суворих валідаційних перевірок вхідних даних через статичні методи моделі `User` (зокрема, здійснюється перевірка коректності електронної пошти через функцію `filter_var` та валідація допустимої довжини номера телефону). У разі виявлення будь-яких помилок (наприклад, введення хибного формату електронної пошти), процес негайно зупиняється, а користувачу повертається масив із відповідним текстом помилок для їх виправлення. Якщо ж дані

визнано валідними, система викликає метод `User::isGuest()`. Для авторизованих користувачів система автоматично отримує ідентифікатор `user_id` із поточної сесії, тоді як для неавторизованих гостей це значення залишається хибним (`false`). Наступним кроком викликається метод `Order::save()`, у межах якого асоціативний масив товарів із кошика серіалізується у зручний текстовий формат за допомогою функції `json_encode($products)`, що дозволяє ефективно зберегти всі товарні позиції в одне поле СУБД. Збереження здійснюється шляхом виконання підготовленого SQL-запиту `INSERT INTO` через об'єктно-орієнтований інтерфейс PDO та метод `bindParam`, що стовідсотково гарантує захист системи від SQL-ін'єкцій. Після успішного збереження даних у таблицю, система автоматично відправляє електронний лист-сповіщення адміністратору магазину за допомогою нативної функції `mail()`. Завершується алгоритм викликом методу `Cart::clear()` для повного очищення сесії кошика, після чого користувач перенаправляється на фінальну сторінку з повідомленням про успішне підтвердження замовлення.

Описані алгоритми формують надійне обчислювальне ядро програмного комплексу, яке характеризується високою стійкістю до помилок та оптимізованим використанням серверних ресурсів, що досягається завдяки мінімізації прямих звернень до системи управління базами даних на проміжних етапах взаємодії.

2.5. Реалізація додатку веборієнтованої системи

Програмна реалізація вебзастосунку інтернет-магазину цифрової техніки здійснювалася у суворій відповідності до спроектованої об'єктно-орієнтованої структури та вимог міжнародного стандарту автозавантаження класів PSR-4. Весь вихідний код серверної частини було логічно розбито на ізольовані простори імен (`namespaces`), що відповідають архітектурним верствам патерну MVC, та підпорядковано єдиному менеджеру залежностей `Composer`. Оскільки повний лістинг програмних модулів системи винесено у відповідні додатки до кваліфікаційної роботи, у даному підрозділі приведено та детально

проаналізовано найбільш складні з інженерної точки зору фрагменти коду, які безпосередньо забезпечують базову життєдіяльність вебзастосунку [29].

Першим ключовим компонентом, що програмно реалізує логіку «єдиної точки входу» (Front Controller), є системний клас Router, який розташований у просторі імен App\components. Основним завданням цього модуля виступає динамічне розпізнавання вхідних HTTP-запитів, зіставлення їх із регулярними виразами конфігураційної карти маршрутів та подальша передача управління відповідному цільовому об'єкту контролера (повний код класу наведено у Додатку А). При цьому найбільш складну обчислювальну логіку зосереджено в методі run(). Фрагмент коду, що відповідає за динамічне створення екземплярів контролерів та виклик відповідних екшенів із передачею масиву параметрів, представлено у лістингу.

```
// Динамічне створення об'єкта контролера на основі URL
$controllerName = sprintf("%sApp\controllers%s", $controllerName);
$controllerObject = new $controllerName;

// Виклик методу із передачею динамічного масиву параметрів
$result = call_user_func_array(array($controllerObject, $actionName),
$params);
```

Складність наведеного фрагмента полягає у використанні механізмів динамічного зв'язування та рефлексії мови PHP. Завдяки функціям preg_match та preg_replace здійснюється гнучке опрацювання параметризованих URL-адрес (наприклад, маршруту /product/25). Своєю чергою, функція call_user_func_array() дозволяє динамічно викликати метод об'єкта, точна назва якого заздалегідь невідома, передаючи йому варіативну кількість аргументів у вигляді індексованого масиву \$parameters. Таке архітектурне рішення мінімізує зв'язність коду та повністю усуває потребу у використанні громіздких умовних конструкцій switch-case.

Другим за важливістю архітектурним рівнем є забезпечення безпечного та високопродуктивного зв'язку із системою управління базами даних MySQL. Ініціалізація з'єднання відбувається у класі Db, який створює об'єкт розширення PDO (PHP Data Objects). Критично важливим аспектом цієї реалізації є активація суворого режиму обробки помилок шляхом встановлення

константного атрибута PDO::ERRMODE_EXCEPTION. Це дозволяє автоматично трансформувати будь-які внутрішні збої бази даних у виняткові ситуації (Exceptions), які згодом безпечно перехоплюються логікою вебзастосунку, надійно запобігаючи витoku конфіденційної технічної інформації (наприклад, назв таблиць чи полів) на екран кінцевого користувача [30].

З метою гарантування захисту системи від атак типу SQL-ін'єкцій, у моделях даних (простір імен App\models) взаємодія із базою здійснюється виключно через технологію підготовлених запитів. Як приклад, у класі Product (повний код якого міститься у Додатку Б) реалізовано метод getLatestProducts(), де на практиці застосовано типізовану прив'язку параметрів безпосередньо на рівні драйвера СУБД.

```
$sql = 'SELECT id, name, price, is_new FROM product WHERE status="1" ORDER
BY id DESC LIMIT :count';
$result = $db->prepare($sql);

// Сувора типізація вхідного параметра на рівні драйвера СУБД
$result->bindParam(':count', $count, PDO::PARAM_INT);
$result->setFetchMode(PDO::FETCH_ASSOC);
$result->execute();
```

Виклик методу prepare() змушує систему управління базами даних заздалегідь скомпілювати внутрішню структуру SQL-шаблону, тоді як метод bindParam() із вбудованим прапорцем PDO::PARAM_INT беззаперечно гарантує, що замість текстової мітки :count до запиту буде передано суворо очищене ціле число. Таким чином, будь-які спроби впровадження стороннього шкідливого коду через цей параметр блокуються на найнижчому рівні драйвера.

Третьою інженерною особливістю розробленого додатка є алгоритм нелінійної трансформації та збереження складних структур даних. Традиційний реляційний підхід вимагає створення додаткової проміжної таблиці зв'язку для зберігання переліку обраних товарів у конкретному замовленні. Проте для підвищення загальної швидкодії системи та максимального спрощення транзакцій у класі Order (Додаток В) було реалізовано алгоритм серіалізації

асоціативного масиву кошика покупок у текстовий формат JSON безпосередньо перед його записом у базу даних.

```
// Конвертація об'єкта кошика покупця у JSON-рядок перед збереженням
$products = json_encode($products);
$result->bindParam(":products", $products, PDO::PARAM_STR);

// Зворотний процес десеріалізації при зчитуванні в адмін-панелі
$productsJson = $order['products'];
$productsObject = json_decode($productsJson);
```

Даний обчислювальний підхід дозволяє надійно зберегти сукупність пар «ідентифікатор товару — кількість» у межах лише одного текстового поля таблиці `product_order`. Це, своєю чергою, суттєво зменшує навантаження на дискову підсистему вебсервера, оскільки повністю виключає потребу виконання ресурсомістких операцій JOIN над великими таблицями під час зчитування історії покупок клієнта чи моніторингу статусу замовлень в адміністративній панелі [31].

Підсумовуючи, розроблена програмна реалізація демонструє збалансоване поєднання класичних об'єктно-орієнтованих архітектурних рішень із сучасними оптимізаційними технологіями, що у комплексі забезпечує стабільну, швидку та безпечну роботу інтернет-магазину навіть за умов обмежених апаратних ресурсів хостингового сервера.

Висновки до другого розділу

У другому розділі кваліфікаційної роботи виконано повний цикл системного проєктування, алгоритмізації та програмної реалізації архітектури, інформаційного забезпечення та ключової бізнес-логіки вебзастосунку інтернет-магазину цифрової техніки. За результатами виконання проєктно-конструкторського етапу розробки було сформовано низку підсумкових висновків.

Зокрема, на основі методології об'єктно-орієнтованого аналізу та проєктування (ООАП) розроблено статичну та динамічну структуру вебзастосунку за архітектурним патерном MVC. За допомогою мови

моделювання UML 2.0 побудовано діаграму варіантів використання (Use Case Diagram) та діаграму класів (Class Diagram), що дозволило чітко розмежувати зони відповідальності та права доступу трьох основних акторів (гість, покупець, адміністратор), формалізувати інтерфейси сутностей системи та встановити суворі вимоги до часових характеристик реакції інтерфейсу (TTFB до 250 мс, AJAX-запити до 150 мс).

Разом із тим, спроектовано та реалізовано даталогічну структуру реляційної бази даних shop під управлінням СУБД MySQL з використанням транзакційного рушія InnoDB. Фізична схема бази даних була приведена до третьої нормальної форми (3НФ) та складається з чотирьох взаємопов'язаних таблиць (category, product, user, product_order). Для оптимізації швидкодії, мінімізації навантаження на дискову підсистему сервера та уникнення необхідності створення надлишкових таблиць зв'язку «багато-до-багатьох» було успішно впроваджено інженерне рішення щодо збереження складу кошика покупок у вигляді серіалізованого рядка формату JSON.

Важливим етапом роботи стало те, що було детально описано та алгоритмічно обґрунтовано ключові процеси життєдіяльності ядра платформи. У межах цього завдання створено індивідуальний модуль маршрутизації (Router) на основі концепції Front Controller та механізму перехоплення URI за допомогою регулярних виразів. Крім того, реалізовано алгоритм сесійного управління кошиком покупця без проміжних звернень до бази даних, а також транзакційний алгоритм оформлення замовлення (Checkout) із валідацією полів форми та інтегрованим поштовим сповіщенням адміністратора.

Підсумовуючи, здійснено практичну реалізацію програмного комплексу мовою PHP 8.x із дотриманням суворої типізації методів та ізоляцією модулів у простори імен (namespaces), які підпорядковані міжнародному стандарту автозавантаження PSR-4 через сучасний менеджер залежностей Composer. З метою забезпечення високого рівня безпеки та абсолютного захисту від атак типу SQL-ін'єкцій усю взаємодію із базою даних побудовано на основі

об'єктно-орієнтованого інтерфейсу PDO із застосуванням виключно підготовлених запитів (prepare, bindParam) та суворого режиму обробки винятків (PDO::ERRMODE_EXCEPTION).

					ІПЗ.КР.Б - 121 - 26 - ПЗ	Арк.
						46

РОЗДІЛ 3. ІНТЕРФЕЙС, ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ ІНТЕРНЕТ-МАГАЗИНУ ЦИФРОВОЇ ТЕХНІКИ

3.1. Порядок встановлення та налаштування параметрів системи

Процес розгортання розробленого вебзастосунку інтернет-магазину розпочинається з підготовки апаратного та програмного середовища. Оскільки архітектура системи базується на використанні мови програмування PHP та реляційної системи управління базами даних MySQL, для забезпечення її повноцінного локального функціонування було обрано кросплатформний пакет ХАМРР. Даний програмний комплекс надає готове середовище з інтегрованим HTTP-сервером Apache, ядром PHP та інструментарієм для роботи з базами даних, що мінімізує ризики несумісності версій програмного забезпечення (рис. 3.1).

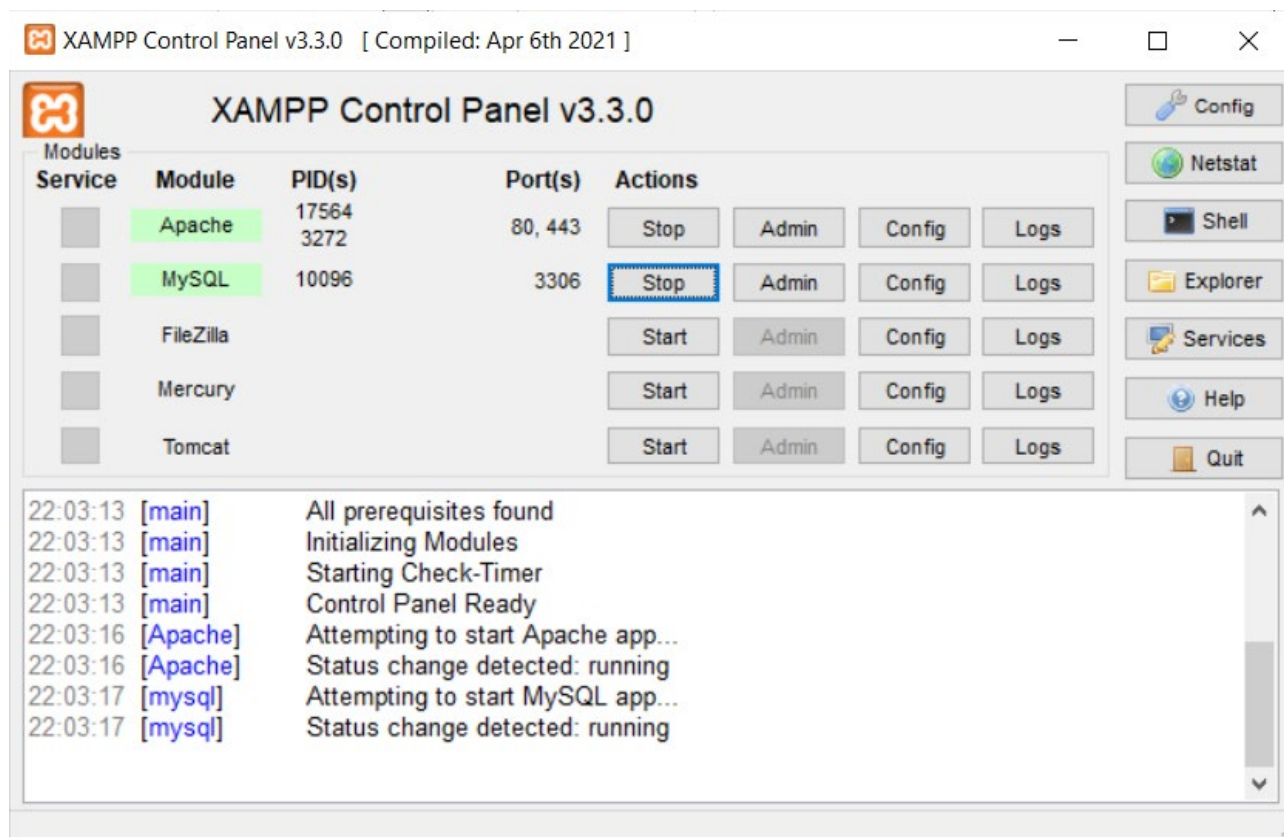


Рис. 3.1 – Панель керування серверним середовищем ХАМРР

Першочерговим практичним кроком для отримання актуальної версії вихідного коду є клонування репозиторію проєкту із системи контролю версій

GitLab. Завантаження коду здійснюється безпосередньо до робочої директорії локального вебсервера, якою в межах інфраструктури XAMPP виступає системна папка `htdocs`. Після успішного клонування утворюється ієрархічна структура проєкту, де ядро системи, моделі, представлення та контролери ізольовані в директорії `app`, а публічний доступ клієнтів дозволено виключно до кореневого каталогу. Таке архітектурне рішення суттєво підвищує загальний рівень безпеки вебзастосунку, унеможливлюючи прямий доступ до системних файлів ззовні.

Наступний етап розгортання передбачає ініціалізацію інформаційної моделі системи. За допомогою веб-інтерфейсу утиліти `phpMyAdmin` створюється порожня база даних із обов'язковим встановленням кодування UTF-8 (`utf8mb4`). Цей параметр є критично важливим, оскільки він гарантує коректне збереження текстової інформації, зокрема розширених описів цифрової техніки та коментарів користувачів українською мовою. Далі виконується імпорт фізичної структури бази даних із заздалегідь підготовленого SQL-дампа, який постачається разом із вихідним кодом у кореневій директорії проєкту. Ця процедура автоматично генерує всі необхідні реляційні таблиці для зберігання інформації про ієрархію категорій, асортимент продукції, облікові записи покупців та деталі оформлених замовлень.

Після успішного створення структури бази даних необхідно налаштувати зв'язок між нею та програмним кодом інтернет-магазину. Цей процес реалізується шляхом редагування спеціалізованого конфігураційного файлу `db_params.php`, розташованого у захищеній директорії системи. У цьому файлі адміністратор визначає асоціативний масив з актуальними параметрами доступу: вказує адресу хоста сервера баз даних, точну назву новоствореної бази, а також логін та пароль користувача MySQL, що володіє відповідними правами на виконання маніпуляцій із даними (рис. 3.2).

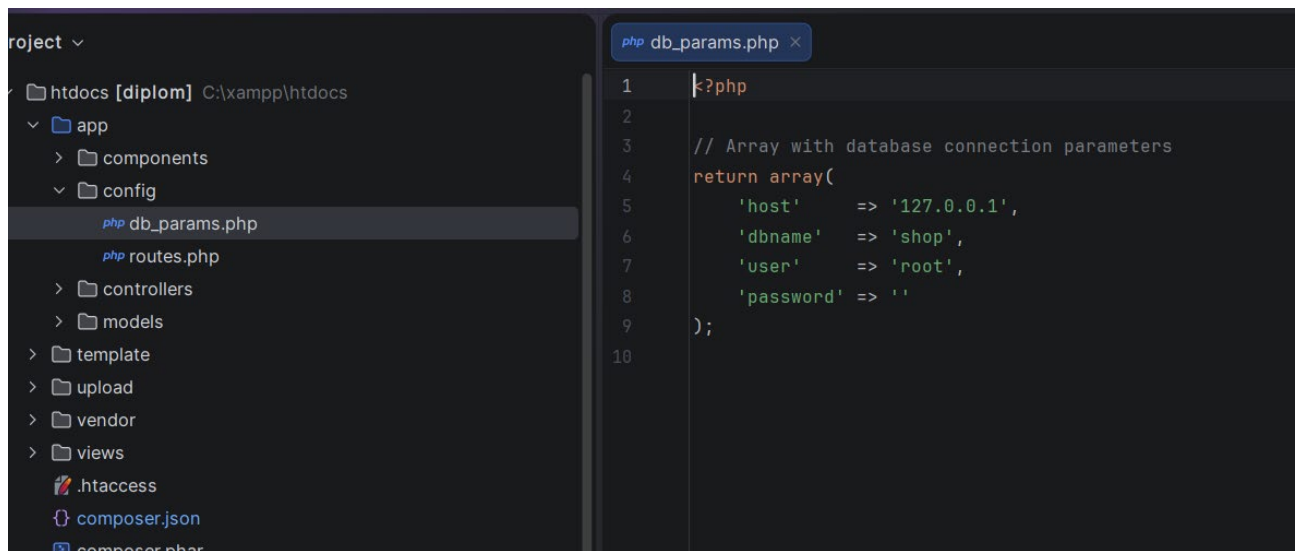


Рис. 3.2 – Конфігурація параметрів підключення

Важливою складовою налаштування є ініціалізація залежностей та механізму автозавантаження класів. Для цього через інтерфейс командного рядка в кореневій директорії проекту виконується команда `composer install`. Пакетний менеджер Composer аналізує файл конфігурації, створює системну директорію `vendor` і генерує файл `autoload.php`. Це дозволяє автоматизувати процес підключення PHP-класів за стандартом PSR-4, позбавляючи програміста необхідності використання ручних викликів функцій `require` чи `include` для кожного окремого компонента системи.

Завершальною стадією конфігурації є активація механізму єдиної точки входу (Front Controller). За допомогою конфігураційного файлу `.htaccess` для сервера Apache вмикається модуль `mod_rewrite`, який здійснює перенаправлення всіх клієнтських HTTP-запитів, що не є прямими зверненнями до існуючих файлів чи папок, на головний файл `index.php`. Саме в цьому файлі ініціалізується глобальна користувачська сесія, підключається згенерований автозавантажувач класів та створюється екземпляр системного класу `Router`, що бере на себе подальший розбір URL-адреси та управління бізнес-логікою.

Після успішного тестування вебзастосунку в локальному середовищі XAMPP його було перенесено на хмарний хостинг InfinityFree для забезпечення цілодобового публічного доступу. Процес міграції розпочався з реєстрації облікового запису, створення доменного імені та завантаження робочих файлів

системи до кореневої публічної папки htdocs на сервері. Паралельно через утиліту phpMyAdmin було створено віддалену базу даних, куди успішно імпортовано локальний SQL-дамп магазину.

Окремої уваги вимагала адаптація системи під специфіку реального Linux-сервера. У файлі app/config/db_params.php локальні реквізити було замінено на актуальні облікові дані віддаленого сервера. Для коректної роботи єдиної точки входу до файлу .htaccess додано директиви [L,QSA], а в системному класі Router метод отримання URI-рядка вдосконалено функцією parse_url. Це дозволило відсікати системні параметри хостингу та уникнути помилок маршрутизації. Такий підхід із чітким розділенням середовищ розробки та експлуатації відповідає сучасним стандартам інженерії програмного забезпечення і гарантує стабільну роботу інтернет-магазину в глобальній мережі (рис. 3.3).

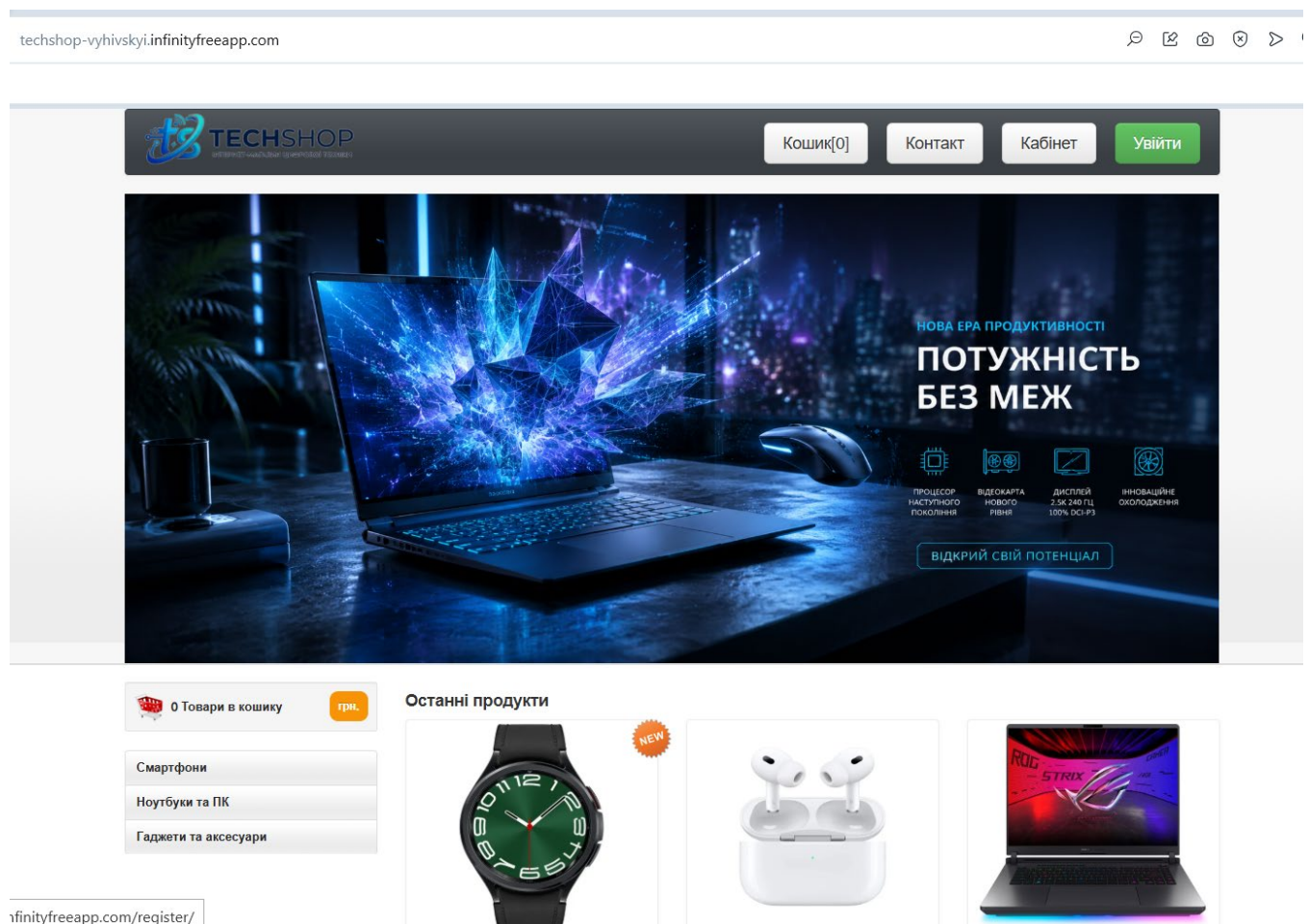


Рис. 3.3 – Головна сторінка вебзастосунку, розгорнутого на віддаленому сервері

3.2. Опис структури інтерфейсу користувача та адміністратора

Візуальне представлення розробленого інтернет-магазину спроектовано з урахуванням сучасних індустріальних вимог до ергономіки та принципів адаптивного вебдизайну. Графічний інтерфейс програмної системи логічно поділяється на дві цілком ізольовані зони: клієнтську частину (Front-office), яка призначена для безпосередньої взаємодії з покупцями, та закриту адміністративну панель (Back-office), що використовується для централізованого управління всіма бізнес-процесами платформи електронної комерції.

Головна сторінка вебзастосунку слугує ключовою точкою входу та базовим вузлом навігації. У верхній частині екрана розміщено глобальне меню з посиланнями на основні розділи та динамічним лічильником віртуального кошика, який завдяки використанню серверних сесій оновлюється без перезавантаження сторінки. Своєю чергою, основна робоча область містить бічне навігаційне меню з ієрархічним переліком категорій цифрової техніки та centralний блок вітрини, де відображаються останні надходження товарного асортименту (рис. 3.4). Для підвищення інформативності кожен товарний елемент вітрини оформлено у вигляді окремої структурованої картки, що візуалізує графічне зображення техніки, її комерційну назву, поточну вартість та інтерактивну кнопку швидкого додавання до кошика. Адаптивне розведення блоків цієї сітки реалізовано на базі гнучких медіазапитів, що гарантує коректне позиціонування та автоматичне масштабування навігаційних елементів на екранах мобільних гаджетів, планшетів і персональних комп'ютерів, повністю виключаючи потребу в горизонтальному прокручуванні сторінки користувачем.

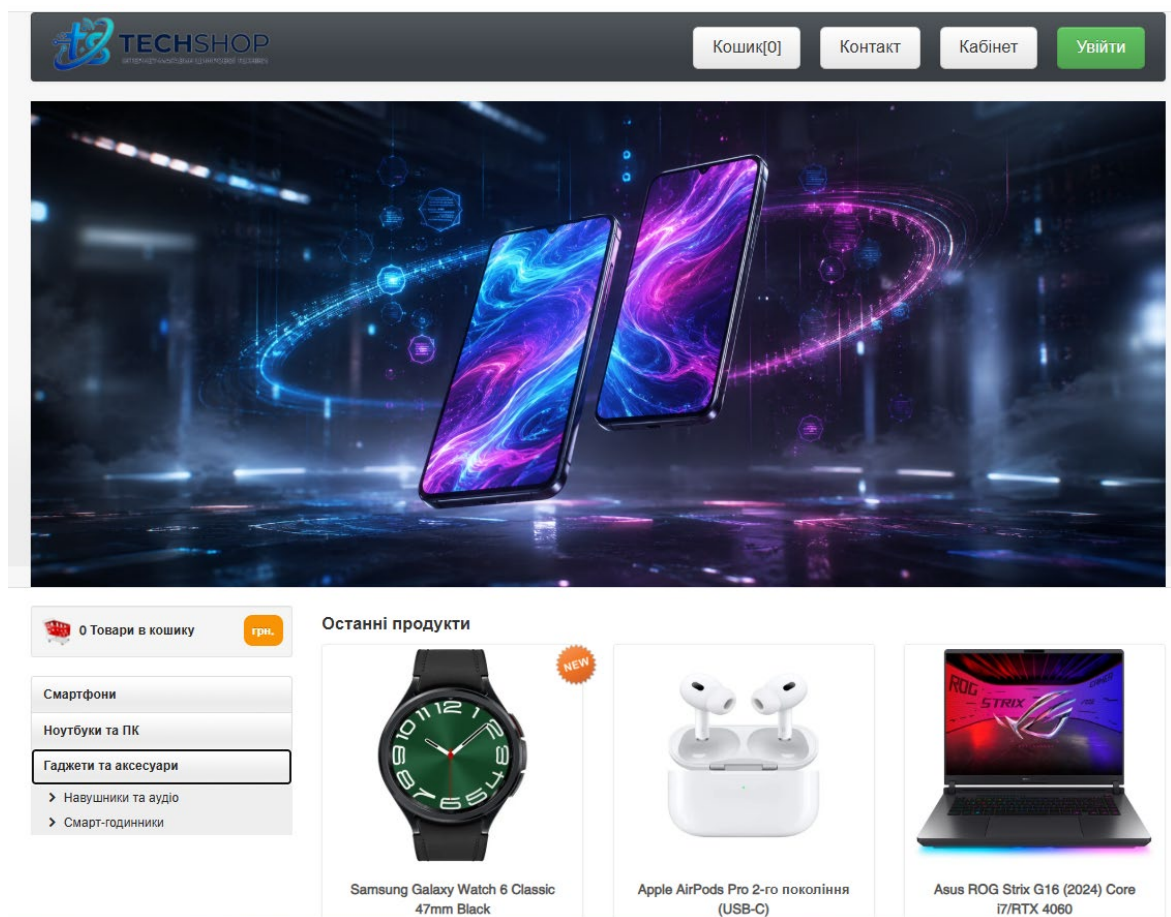


Рис. 3.4 – Головна сторінка вебзастосунку інтернет-магазину

Для детального ознайомлення з технічними характеристиками обраної моделі техніки користувач здійснює перехід на сторінку картки конкретного товару. Даний інформаційний модуль інтерфейсу консолідує всю необхідну комерційну інформацію, зокрема фотографію продукту, розширений технічний опис, дані про поточну наявність на складі та актуальну роздрібну ціну. Безпосередньо з цієї сторінки ініціюється процес покупки шляхом натискання відповідної цільової кнопки, що автоматично додає ідентифікатор обраного товару до глобального масиву користувацької сесії (рис. 3.5). Додатково для покращення навігаційних властивостей інтерфейсу у верхній частині модуля інтегровано навігаційний ланцюжок («хлібні крихти»), який відображає повний шлях від головної сторінки до поточної категорії цифрової техніки. Це дозволяє покупцеві миттєво повернутися до будь-якого попереднього рівня вкладеності каталогу, зберігаючи логіку взаємодії із системою та суттєво оптимізуючи показники користувацького досвіду (UX).

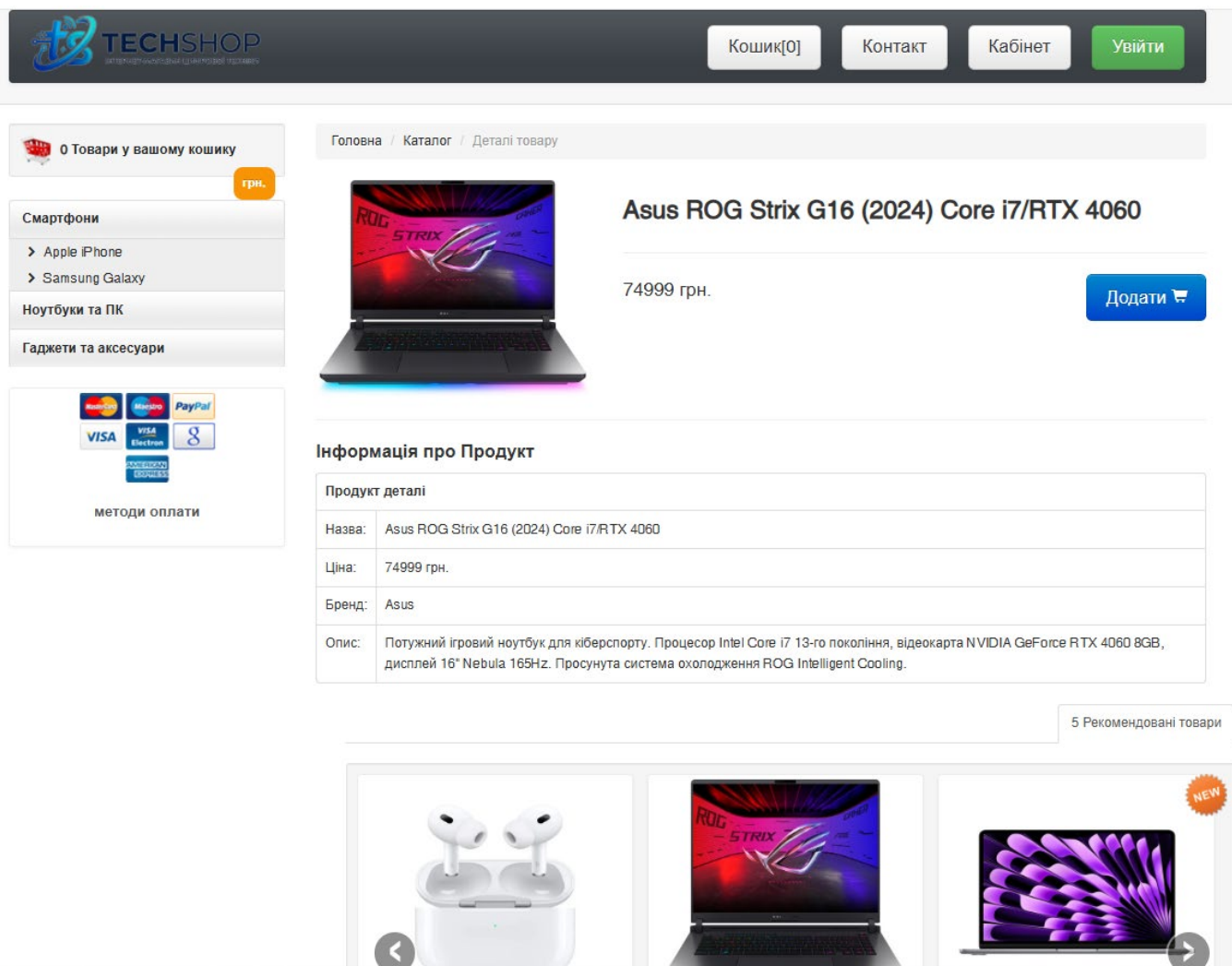


Рис. 3.5 – Інтерфейс сторінки картки товару

Процес оформлення замовлення реалізовано у вигляді окремої мінімалістичної екранної форми, що дозволяє максимально мінімізувати кількість кроків клієнта до успішного завершення покупки. Безпосередньо на сторінці кошика покупець має змогу переглянути повний перелік обраних товарних позицій, динамічно змінити їхню кількість, дізнатися підсумкову вартість сформованого замовлення та заповнити базові контактні дані, необхідні для служби доставки (рис. 3.6). Для запобігання надсиланню хибних або неповних заявок інтерфейс екранної форми оснащено модулем оперативного сповіщення про помилки валідації обов'язкових полів введення. Таке інженерне рішення мінімізує ризик потрапляння некоректних контактних даних покупця до реляційної бази даних та дозволяє користувачеві оперативно

виправити формат літерних чи цифрових значень без перезавантаження сторінки.

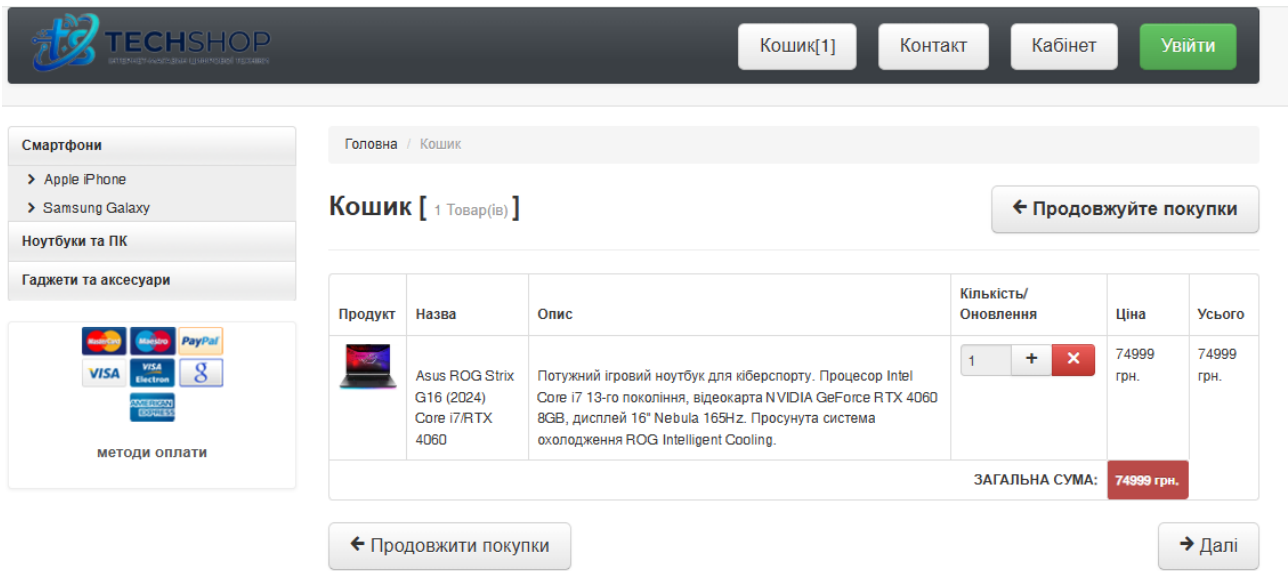


Рис. 3.6 – Екранна форма віртуального кошика та оформлення замовлення

Повноцінне управління контентом інтернет-магазину та обробка клієнтських заявок здійснюється через закриту адміністративну панель, доступ до якої надається виключно після проходження процедури безпечної автентифікації з відповідними правами доступу. Графічний інтерфейс Back-office побудовано у вигляді зручних табличних списків, що дозволяє адміністратору платформи максимально оперативно редагувати каталог товарів, змінювати статус виконання актуальних замовлень та гнучко модерувати ієрархію категорій (рис. 3.7).

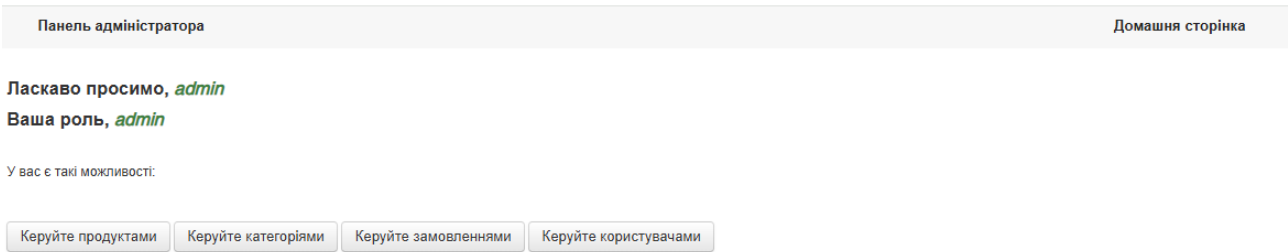


Рис. 3.7 – Головний інтерфейс панелі управління (Back-office)

Підсумовуючи, можна стверджувати, що запропонована структура користувацького та адміністративного інтерфейсів забезпечує інтуїтивно зрозумілу навігацію для кінцевого споживача, а також надає вичерпний і технологічний набір інструментів для персоналу, що безпосередньо обслуговує даний торговельний майданчик.

3.3. Тестування системи та аналіз результатів

Для забезпечення високої якості, стабільності та надійності розробленого програмного забезпечення було проведено комплексне тестування вебзастосунку. Головною метою цього етапу стало підтвердження повної відповідності реалізованого інтернет-магазину висунутим на етапі проектування технічним вимогам, а також гарантування безперебійного виконання ключових бізнес-процесів за різних умов експлуатації. Процес перевірки охоплював кілька взаємопов'язаних рівнів, зокрема функціональне тестування, перевірку механізмів валідації та оцінку коректності роботи з реляційною базою даних. Тестове середовище базувалося на використанні локального сервера XAMPP, а взаємодія з інтерфейсом здійснювалася через сучасні веббраузери з використанням інструментів розробника для моніторингу мережесих запитів.

Під час перевірки клієнтської частини (Front-office) основна увага приділялася сценарію вибору цифрової техніки та проходженню повного циклу оформлення замовлення. У ході функціонального тестування перевірявся механізм сесійного зберігання даних віртуального кошика. Система продемонструвала абсолютну точність при динамічному підрахунку загальної вартості замовлення та кількості обраних товарів. Додавання та видалення позицій з кошика відбувається коректно, без втрати даних під час навігації між сторінками каталогу.

Одним із найважливіших етапів стала перевірка механізмів жорсткої валідації вхідних даних на формах реєстрації, авторизації користувачів та безпосереднього оформлення замовлення. Архітектура контролерів була

протестована на здатність відхиляти помилкові запити. У ході тестування імітувалися спроби відправки порожніх форм, введення номерів телефонів недостатньої довжини та використання некоректного формату адреси електронної пошти. Перевірка показала, що бекенд-частина застосунку надійно блокує збереження таких даних, формує масив помилок і своєчасно виводить користувачеві відповідні попередження на екран, не допускаючи порушення цілісності бази даних (рис. 3.8).

The screenshot shows the login page of a website called TECHSHOP. At the top, there is a navigation bar with links for 'Кошик[0]', 'Контакт', 'Кабінет', and 'Увійти'. Below this, a header area shows '0 Товари у вашому кошику' and a 'Головна / Login' breadcrumb. The main content area is titled 'Login' and contains a login form. The form has three error messages in red text: '- Неправильно вказано електронну адресу', '- Пароль повинен містити щонайменше 6 символів', and '- Неправильні дані для входу'. The form fields are 'Email' and 'Password', both with input boxes. Below the fields are buttons for 'Увійти' and 'Забув пароль?'. On the left side of the page, there are sections for 'Смартфони' (listing Apple iPhone and Samsung Galaxy), 'Ноутбуки та ПК', 'Гаджети та аксесуари', and 'методи оплати' (listing various payment methods like Visa, MasterCard, PayPal, etc.).

Рис. 3.8 – Сторінка авторизації з повідомленнями про помилки валідації

Тестування адміністративної панелі (Back-office) підтвердило надійність механізму автентифікації та повну ізоляцію закритих маршрутів. Спроби прямого переходу за URL-адресою до панелі керування без наявності активної сесії адміністратора успішно блокуються системою маршрутизації з автоматичним перенаправленням на сторінку входу. У самій панелі було перевірено всі базові операції управління контентом (CRUD): створення нових карток товарів, редагування існуючих описів та цін, а також управління ієрархією категорій. Всі зміни миттєво та коректно відображаються у публічній частині магазину.

Фінальним етапом стало наскрізне тестування (End-to-End) процесу обслуговування клієнта. Сценарій передбачав повний шлях: від додавання товару до кошика та заповнення валідної контактної інформації до відправки форми. Тестування підтвердило, що після успішного оформлення замовлення кошик автоматично очищується, користувач отримує системне повідомлення про успішну операцію, а всі деталі замовлення (ідентифікатор клієнта, контактні дані, перелік товарів) коректно фіксуються в таблиці бази даних і стають доступними для обробки в панелі адміністратора (рис. 3.9).

Панель адміністратора

Домашня сторінка

Панель адміністратора Керуйте замовленнями

Список замовлень

ID замовлення:	Ім'я	Пошта	Дата	Статус			
18	Boris	borismarcenko173@gmail.com	2026-05-21 20:17:26	Нове замовлення	Переглянути	Редагувати	Видалити
17	Руслан	ruslannkor@gmail.com	2026-05-21 20:13:24	В обробці	Переглянути	Редагувати	Видалити
16	Andrii	andryimakarov@gmail.com	2026-05-21 20:03:17	Доставлено	Переглянути	Редагувати	Видалити

Рис. 3.9 – Панель адміністратора зі списком замовлень

За результатами проведеного розширеного комплексу випробувань можна з упевненістю констатувати, що розроблений вебзастосунок інтернет-магазину цифрової техніки функціонує стабільно. Система успішно обробляє виняткові ситуації, надійно захищає базу даних від запису некоректної інформації та є повністю готовою до впровадження у реальне бізнес-середовище.

З метою систематизації процесу перевірки та забезпечення повного покриття розробленого функціоналу було сформовано набір тестових сценаріїв, що охоплюють як позитивні, так і негативні випадки використання системи. Такий підхід дозволяє виявити вразливі місця в логіці обробки запитів на ранніх етапах та гарантувати стабільну роботу всіх модулів вебзастосунку в умовах реальної експлуатації. Детальний опис кроків тестування, очікуваних та фактичних результатів виконання кожного сценарію наведено у відповідній таблиці (табл. 3.1).

Таблиця 3.1

Сценарії та результати функціонального тестування вебзастосунку

ID.	Назва та кроки тестового сценарію	Очікуваний результат	Фактичний результат	Статус
TC-01	Авторизація з невірними даними: перейти на сторінку входу, ввести існуючий E-mail та навмисно хибний пароль, натиснути кнопку входу.	Система повинна відхилити запит, не ініціалізувати сесію та вивести повідомлення про помилку автентифікації.	Запит відхилено. Виведено повідомлення про помилку. Доступ заборонено.	Успішно
TC-02	Додавання товару до віртуального кошика: відкрити каталог товарів, обрати товар та натиснути кнопку додавання, перейти на сторінку кошика.	Товар має успішно додатися до масиву сесії. У шапці сайту лічильник повинен оновитися. У кошику має відображатися обрана позиція.	Товар додано у сесію. Лічильник товарів оновлено асинхронно. Товар та його ціна коректно відображаються.	Успішно
TC-03	Зміна кількості товарів у кошику: у кошику з наявним товаром збільшити кількість до двох одиниць, перевірити перерахунок вартості.	Система повинна оновити кількість товару в сесії та миттєво перерахувати загальну суму замовлення.	Кількість змінено. Загальна сума замовлення перерахована математично точно без перезавантаження сторінки.	Успішно
TC-04	Валідація форми оформлення замовлення: перейти до оформлення покупки, ввести ім'я, але залишити поле телефону порожнім, натиснути підтвердження.	Замовлення не повинно бути збережене у БД. Система має зупинити процес та повернути попередження.	Процес зупинено. Відображено текстове попередження біля поля. Дані у БД не відправлено.	Успішно

продовж. табл. 3.1

ID.	Назва та кроки тестового сценарію	Очікуваний результат	Фактичний результат	Статус
ТС-05	Управління каталогом через панель адміністратора: авторизуватися з правами доступу, перейти до списку товарів, заповнити форму додавання нової позиції (назва, ціна, категорія) та зберегти зміни.	Система повинна зберегти новий товар у базі даних, додати його до загального списку в панелі керування та миттєво відобразити у публічному каталозі магазину.	Товар успішно збережено. Дані коректно відображаються у списку товарів адміністративної панелі та на вітрині для клієнтів.	Успішно

Висновки до третього розділу

У третьому розділі кваліфікаційної роботи було здійснено практичне розгортання розробленого вебзастосунку інтернет-магазину цифрової техніки та проведено його комплексне тестування. Зокрема, детально описано процедуру ініціалізації локального серверного середовища за допомогою програмного комплексу XAMPP, алгоритм імпорту реляційної бази даних та конфігурацію параметрів підключення об'єктно-орієнтованого ядра системи, а також визначено методологію перенесення проєкту на віддалений продуктивний вебсервер із чітким розділенням середовищ розробки (Development) та експлуатації (Production). Крім того, наочно продемонстровано логічну структуру інтерфейсу користувача, який концептуально розділено на публічну клієнтську частину (Front-office) та ізольовану адміністративну панель (Back-office). Візуальну складову Front-office було оптимізовано завдяки впровадженню адаптивної сітки карткових елементів на базі медіазапитів для коректного масштабування на мобільних пристроях, а також інтеграції навігаційного ланцюжка («хлібних крихт») задля оптимізації користувацького досвіду (UX).

За результатами проведеного багаторівневого та наскрізного (End-to-End) тестування підтверджено безперебійну роботу механізмів формування

віртуального кошика, надійність системи жорсткої валідації вхідних даних із виведенням інтерактивних сповіщень про помилки на екранних формах, а також коректність виконання базових CRUD-операцій управління контентом в адміністративній панелі. Підсумовуючи, можна стверджувати, що створений програмний продукт повною мірою відповідає висунутим на етапі проектування технічним вимогам, характеризується високою стабільністю роботи, надійно обробляє виняткові ситуації та є повністю готовим до розгортання та комерційної діяльності.

					ІПЗ.КР.Б - 121 - 26 - ПЗ	Арк.
						60

ВИСНОВКИ

У випускній кваліфікаційній роботі на основі проведених теоретичних та експериментальних досліджень вирішено актуальне науково-практичне завдання з автоматизації процесів електронної комерції у сегменті роздрібної торгівлі цифровою технікою. Результати виконаного інженерного проєкту підтверджують повне досягнення поставленої мети та успішне розв'язання всіх визначених на етапі проєктування завдань.

Розроблений програмний продукт практично доводить технічну та економічну доцільність відмови від використання універсальних, важковагових систем управління контентом (CMS) на користь створення індивідуального програмного каркаса. Такий підхід дозволив спроектувати спеціалізовану архітектуру, яка повністю позбавлена надлишкових інфраструктурних абстракцій і орієнтована суто на специфіку обліку та реалізації складних ієрархічних груп товарів. Створена модель вебзастосунку продемонструвала високу ефективність декомпозиції логіки за принципами слабкої зв'язності. Завдяки чіткому розподілу обов'язків між системним ядром, модулями обробки даних та інтерфейсними представленнями у межах патерну MVC, програмний комплекс набув високого ступеня масштабованості та стійкості до модифікацій.

Важливим інженерним здобутком роботи є розроблений комбінований підхід до збереження інформації. Поєднання класичної реляційної структури бази даних для управління товарним каталогом і користувачами із нереляційним форматом збереження транзакційних складів кошиків у вигляді JSON-документів дозволило досягти оптимального балансу обчислювальної складності. Це архітектурне рішення нівелювало потребу в ресурсомістких операціях об'єднання масивів таблиць (JOIN) при високих навантаженнях, суттєво знизивши вимоги до оперативної пам'яті та дискової підсистеми хостингу.

Програмна реалізація вебресурсу виконана із суворим дотриманням сучасних індустріальних стандартів, зокрема з використанням чистої об'єктно-

орієнтованої мови PHP 8.x, суворой типізації та менеджменту залежностей за специфікацією автозавантаження PSR-4. Впроваджені алгоритми обробки запитів на базі динамічної маршрутизації та превентивний захист на рівні драйвера PDO забезпечили безперебійну відмовостійкість системи та її надійний захист від поширених векторів вебатак.

На етапі практичного впровадження здійснено успішне розгортання вебзастосунку в локальному серверному середовищі з налаштуванням захищеного з'єднання та ініціалізацією інформаційної бази. Спроектовано та реалізовано ергономічний користувацький інтерфейс, який логічно розділяє систему на публічну вітрину (Front-office) для зручної взаємодії з покупцями та ізольовану панель управління (Back-office) для адміністрування каталогу й оперативного опрацювання клієнтських заявок.

Проведений комплекс багаторівневого тестування повністю підтвердив заявлені технічні характеристики системи. Наскрізне (End-to-End) тестування повного циклу покупки, перевірка механізмів жорсткої валідації вхідних даних на формах реєстрації та оформлення замовлень, а також тестування операцій управління даними (CRUD) довели абсолютну стабільність роботи бізнес-логіки. Система коректно обробляє виняткові ситуації та надійно блокує збереження некоректної інформації до бази даних.

Практична цінність отриманих результатів підтверджується створенням повнофункціонального, протестованого та готового до розгортання вебзастосунку. Спроектоване легковагове ядро може бути успішно інтегроване в реальну діяльність підприємств малого та середнього бізнесу як надійна платформа для розширення каналів збуту, дозволяючи мінімізувати їхні витрати на комерційне обслуговування та подальший технічний супровід програмного коду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ ISO/IEC 12207:2016. Інженерія систем і програмного забезпечення. Процеси життєвого циклу програмного забезпечення. – Київ : ДП «УкрНДНЦ», 2016.
2. Якобсен А. Об'єктно-орієнтований аналіз та проєктування / А. Якобсен. – Київ : Вільямс, 2021. – 416 с.
3. Дейт К. Дж. Вступ до систем баз даних / К. Дж. Дейт. – Київ : Діалектика, 2019. – 1328 с.
4. Шварц Б. MySQL. Оптимізація продуктивності / Б. Шварц, П. Зайцев. – Київ : Фабула, 2018. – 832 с.
5. Apache HTTP Server Documentation [Електронний ресурс]. – Режим доступу: <https://httpd.apache.org/docs/> (дата звернення: 16.02.2026).
6. Bootstrap Documentation [Електронний ресурс]. – Режим доступу: <https://getbootstrap.com/docs/5.3/> (дата звернення: 12.02.2026).
7. Куліков С. Тестування програмного забезпечення. Базовий курс / С. Куліков. – Київ : ЕПАМ Системз, 2020. – 312 с.
8. ISTQB Foundation Level Syllabus [Електронний ресурс]. – Режим доступу: <https://www.istqb.org/> (дата звернення: 18.02.2026).
9. Гаррет Дж. Елементи досвіду користувача (User Experience) / Дж. Гаррет. – Київ : Символ-Плюс, 2020. – 192 с.
10. ДСТУ EN ISO 9241-210:2021. Ергономіка взаємодії людина-система. Частина 210. Людино-орієнтоване проєктування інтерактивних систем. – Київ : ДП «УкрНДНЦ», 2022. – 44 с.
11. Stack Overflow Developer Survey 2025 [Електронний ресурс]. – Режим доступу: <https://insights.stackoverflow.com/> (дата звернення: 12.02.2026).
12. Google Search Central. SEO Starter Guide [Електронний ресурс]. – Режим доступу: <https://developers.google.com/search/> (дата звернення: 12.02.2026).

13. Фаулер М. Архітектура корпоративних програмних застосунків / М. Фаулер. – Київ : Вільямс, 2018. – 544 с.
14. W3C HTML5 Specification [Електронний ресурс]. – Режим доступу: <https://www.w3.org/TR/html52/> (дата звернення: 12.02.2026).
15. Веллінг Л. Розробка вебдодатків з допомогою PHP і MySQL / Л. Веллінг, Л. Томсон. – Київ : Діалектика, 2019. – 848 с.
16. MDN Web Docs. MVC Architecture [Електронний ресурс]. – Режим доступу: <https://developer.mozilla.org/> (дата звернення: 12.01.2026).
17. Фрімен Е. Патерни проектування / Е. Фрімен, Е. Фрімен. – Київ : Фабула, 2022. – 680 с.
18. Мартін Р. Чиста архітектура: мистецтво розробки програмного забезпечення / Р. Мартін. – Харків : Фабула, 2019. – 368 с.
19. Зандстра М. PHP: об'єкти, шаблони та методики програмування / М. Зандстра. – Київ : Вільямс, 2020. – 576 с.
20. PHP The Right Way [Електронний ресурс]. – Режим доступу: <https://phptherightway.com/> (дата звернення: 12.02.2026).
21. Скляр Д. PHP 7. Керівництво / Д. Скляр. – Київ : Діалектика, 2018. – 450 с.
22. Apache Friends. XAMPP Installers and Downloads [Електронний ресурс]. – Режим доступу: <https://www.apachefriends.org/> (дата звернення: 16.02.2026).
23. PHP Manual [Електронний ресурс]. – Режим доступу: <https://www.php.net/manual/en/> (дата звернення: 10.02.2026).
24. OWASP Top 10 Web Application Security Risks [Електронний ресурс]. – Режим доступу: <https://owasp.org/www-project-top-ten/> (дата звернення: 15.02.2026).
25. Composer Documentation [Електронний ресурс]. – Режим доступу: <https://getcomposer.org/doc/> (дата звернення: 12.01.2026).

26. Макконнелл С. Досконалий код. Майстер-клас / С. Макконнелл. – Київ : Видавництво, 2019. – 896 с.
27. Фаулер М. UML. Основи / М. Фаулер. – Київ : Символ-Плюс, 2020. – 192 с.
28. PHP-FIG. PSR-4: Autoloader [Електронний ресурс]. – Режим доступу: <https://www.php-fig.org/psr/psr-4/> (дата звернення: 11.01.2026).
29. Ніксон Р. Створюємо динамічні вебсайти за допомогою PHP, MySQL, JavaScript, CSS та HTML5 / Р. Ніксон. – Київ : Діалектика, 2019. – 496 с.
30. MySQL Reference Manual [Електронний ресурс]. – Режим доступу: <https://dev.mysql.com/doc/refman/8.0/en/> (дата звернення: 10.02.2026).
31. JSON (JavaScript Object Notation) Standard [Електронний ресурс]. – Режим доступу: <https://www.json.org/> (дата звернення: 16.02.2026).

ДОДАТКИ

					ІПЗ.КР.Б - 121 - 26 - ПЗ	Арк.
						66

```

<?php
namespace App\components;

class Router {
    private $routes;

    public function __construct() {
        $routesPath = ROOT . '/app/config/routes.php';
        $this->routes = include($routesPath);
    }

    private function getURI() {
        if (!empty($_SERVER['REQUEST_URI'])) {
            return trim($_SERVER['REQUEST_URI'], '/');
        }
    }

    public function run() {
        $uri = $this->getURI();

        foreach ($this->routes as $uriPattern => $path) {
            if (preg_match("~$uriPattern~", $uri)) {
                $internalRoute = preg_replace("~$uriPattern~", $path,
$uri);

                $segments = explode('/', $internalRoute);

                $controllerName = array_shift($segments).'Controller';
                $controllerName = ucfirst($controllerName);
                $actionName = 'action'.ucfirst(array_shift($segments));
                $parameters = $segments;

                $controllerFile =
ROOT.'/app/controllers/'.$controllerName.'.php';
                if (is_file($controllerFile)) {
                    $controllerName = sprintf("\App\controllers\%s",
$controllerName);

                    $controllerObject = new $controllerName;
                }

                $result = call_user_func_array(array($controllerObject,
$actionName), $parameters);

                if ($result != null) {
                    break;
                }
            }
        }
    }
}

```

```

<?php
namespace App\models;
use App\components\Db;
use PDO;

class Product {
    const SHOW_BY_DEFAULT = 6;

    public static function getLatestProducts($count =
self::SHOW_BY_DEFAULT) {
        $db = Db::getConnection();

        $sql = 'SELECT id, name, price, is_new FROM product WHERE
status="1" ORDER BY id DESC LIMIT :count';

        $result = $db->prepare($sql);
        $result->bindParam(':count', $count, PDO::PARAM_INT);
        $result->setFetchMode(PDO::FETCH_ASSOC);
        $result->execute();

        $productsList = array();
        $i = 0;
        while ($row = $result->fetch()) {
            $productsList[$i]['id'] = $row['id'];
            $productsList[$i]['name'] = $row['name'];
            $productsList[$i]['price'] = $row['price'];
            $productsList[$i]['is_new'] = $row['is_new'];
            $i++;
        }
        return $productsList;
    }
}

```

```

<?php
namespace App\models;
use App\components\Db;
use PDO;

class Order {
    public static function save($userName, $userEmail, $userPhone,
$userComment, $userId, $products) {
        $db = Db::getConnection();

        $sql = "INSERT INTO product_order (user_name, user_email,
user_phone, user_comment, user_id, products) "
            . "VALUES (:user_name, :user_email, :user_phone,
:user_comment, :user_id, :products)";

        $products = json_encode($products);

        $result = $db->prepare($sql);
        $result->bindParam(":user_name", $userName, PDO::PARAM_STR);
        $result->bindParam(":user_email", $userEmail, PDO::PARAM_STR);
        $result->bindParam(":user_phone", $userPhone, PDO::PARAM_STR);
        $result->bindParam(":user_comment", $userComment, PDO::PARAM_STR);
        $result->bindParam(":user_id", $userId, PDO::PARAM_INT);
        $result->bindParam(":products", $products, PDO::PARAM_STR);

        return $result->execute();
    }
}

```