

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 з продажу чоловічого одягу»

Виконав студент групи ПЗ-19-1
ДРАК Тарас Сергійович

Керівник роботи:
БАКАЛЮК Тетяна Анатоліївна

Рецензент:
ЧИЖМОТРЯ Олексій Володимирович

Житомир – 2023

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

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«02» серпня 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ДРАК Тарас Сергійович**

Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**

Тема роботи: «Інтернет магазин з продажу чоловічого одягу»,

затверджена Наказом закладу вищої освіти від **«02» серпня 2023 р., №326с**

Вихідні дані для роботи: розробка інтернет магазину з продажу чоловічого одягу

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т.А.	02.08.2023р.	02.08.2023р.
2	Вакалюк Т.А.	11.08.2023р.	11.08.2023р.
3	Вакалюк Т.А.	10.08.2023р.	19.08.2023р.

РЕФЕРАТ

Випускна робота бакалавра складається із двох основних частин: комплексної програмної реалізації інтернет-магазину та пояснювальної записки. Пояснювальна записка до цієї кваліфікаційної роботи містить в собі 67 сторінок, 32 ілюстрації, 7 таблиць, 3 додатка та посилається на 20 літературних джерел.

Головною метою даної роботи є розробка інтернет магазину з продажу чоловічого одягу. Мета полягає в створенні ефективного та зручного інструменту, який забезпечує взаємодію між адміністратором та базою даних.

У роботі були сформульовані основні завдання для розробки системи, проведено аналіз методів вирішення цих завдань, розглянуто методи обробки та аналізу даних. Також надано алгоритм роботи програмного комплексу, описано архітектуру та взаємодію основних його компонентів.

КЛЮЧОВІ СЛОВА: ОНЛАЙН-СИСТЕМА, PHP, MYSQL СЕРВЕР, БД, ВЕБ-ДОДАТОК, СИСТЕМА.

					ІПЗ.КР.Б – 121 – 23 – ПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата	Інтернет магазин з продажу чоловічого одягу Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Т.С. Драк						
Керівник		Т.А. Вакалюк					3	67
						Житомирська політехніка, група ІПЗ-19-1		
Рецензент		О.В. Чижмотря						
Зав. каф.		А.В. Морозов						

ABSTRACT

The bachelor's thesis consists of two main parts: a comprehensive software implementation of an online store and an explanatory note. The explanatory note for this qualification work comprises 67 pages, 32 illustrations, 7 tables, 3 appendices, and references to 20 literary sources.

The main goal of this work is the development of an online store for the sale of men's clothing. The objective is to create an efficient and user-friendly tool that facilitates interaction between the administrator and the database.

In the paper, the main tasks for system development were formulated, methods for solving these tasks were analyzed, and data processing and analysis methods were discussed. Additionally, an algorithm of the software complex operation was provided, and the architecture and interaction of its main components were described.

KEYWORDS: ONLINE SYSTEM, PHP, MYSQL SERVER, DATABASE, WEB APPLICATION, SYSTEM.

					ИПЗ.КР.Б – 121 – 23 – ИПЗ	
						4

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	9
1.1 ПОСТАНОВКА ЗАДАЧІ	9
1.2 АНАЛІЗ АНАЛОГІВ ПРОГРАМНОГО ПРОДУКТУ	10
1.3 ВИБІР АРХІТЕКТУРИ ІНТЕРНЕТ МАГАЗИНУ	14
1.4 ОБҐРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ТА ВИМОГ ДО АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ	166
Висновки до першого розділу	18
РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНТЕРНЕТ МАГАЗИНУ З ПРОДАЖУ ЧОЛОВІЧОГО ОДЯГУ	19
2.1 ВИЗНАЧЕННЯ ВАРІАНТІВ ВИКОРИСТАННЯ ТА СТРУКТУРИ ДОДАТКУ	19
2.2 РОЗРОБКА БАЗИ ДАНИХ СИСТЕМИ	25
2.3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ АЛГОРИТМІВ РОБОТИ ДОДАТКУ	28
2.4 РЕАЛІЗАЦІЯ ФУНКЦІОНАЛУ ІНТЕРНЕТ МАГАЗИНУ З ПРОДАЖУ ЧОЛОВІЧОГО ОДЯГУ	32
Висновки до другого розділу	37
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНТЕРНЕТ МАГАЗИНОМ	38
3.1 ПОРЯДОК ВСТАНОВЛЕННЯ ТА НАЛАШТУВАННЯ СИСТЕМИ	38
3.2 ІНТЕРФЕЙС МАГАЗИНУ ТА ПОРЯДОК РОБОТИ	40
3.3 ТЕСТУВАННЯ РОБОТИ ВЕБ ДОДАТКУ	49
Висновки до третього розділу	52
ВИСНОВКИ	53
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	54
ДОДАТКИ	56

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – Програмне забезпечення;

API – Прикладний програмний інтерфейс;

SQL – Structured Query Language;

БД –База даних;

ІС – Інформаційна система;

ПК – Персональний комп'ютер

ORM – Object-Relational Mapping;

OSP – Open Server Panel

MVC - Model-View-Controller, це архітектурний шаблон, що використовується для розробки програмного забезпечення;

ВСТУП

Актуальність теми. Чоловічий гардероб – це відображення індивідуальності, стилю та смаку кожного представника сильної половини людства. Одяг для чоловіків завжди був не тільки засобом захисту від зовнішніх факторів, але і способом виразити себе, підкреслити свою унікальність та дотримуватися модних тенденцій. Сьогодні, у часи постійного розвитку та стрімких змін, інтернет-магазини чоловічого одягу стають необхідними для сучасного чоловіка, який цінує зручність, стиль та якість свого одягу.

У магазинах чоловічого одягу онлайн представлений безмежний асортимент стилів та виробників, що враховує різноманітні потреби і смаки. Тут можна знайти все, починаючи від елегантного вдягання для офісу та важливих подій, і закінчуючи спортивним одягом для активного відпочинку. Асортимент включає в себе різні типи костюмів, сорочок, светрів, джинсів, футболок, курток та багато інших елементів гардеробу, які допоможуть чоловікові виглядати завжди на висоті.

Зазначимо, що в інтернет-магазинах чоловічого одягу особлива увага приділяється якості товарів. Тут представлені відомі бренди, відомі своєю якістю та стильним дизайном. Це дозволяє кожному чоловікові бути впевненим, що його гардероб складається з надійних та тривалих речей.

Окрім великого вибору і якості, інтернет-магазини чоловічого одягу створюють зручність для своїх клієнтів. Вони пропонують можливість замовлення товарів онлайн, швидку доставку та зручну систему повернення товару, якщо щось не підійшло за розміром чи стилем. Також вони надають можливість отримати консультацію від модних експертів, які допоможуть з вибором найкращих рішень для створення стильного образу.

Загалом, інтернет-магазини чоловічого одягу – це незамінне місце для чоловіків, які прагнуть виглядати завжди стильно та дбати про свій гардероб. Вони пропонують безмежні можливості для виразу особистості через одяг, поєднуючи в собі широкий вибір, якість та зручність, і створюють ідеальний простір для втілення свого стилю в реальному житті.

Метою даної кваліфікаційної роботи є створення інтернет-магазину чоловічого одягу, використовуючи мову програмування PHP. Головною метою проекту є розробка функціонального та зручного онлайн-магазину, який дозволить чоловікам легко вибирати та придбавати високоякісний одяг та взуття для своїх дітей.

Крім того, веб-платформа буде використовувати PHP для впровадження системи управління запасами, що дозволить магазину ефективно контролювати наявність товарів, автоматично оновлювати кількість товарів після кожного замовлення і надавати користувачам реальний часовий доступ до інформації про доступність товарів.

Встановлена мета обумовлює наступні завдання:

- аналіз задач, пов'язаних з функціональністю і особливостями інтернет-магазину чоловічого одягу;
- визначення архітектури, узагальненої структури, ключових компонентів додатку;
- обґрунтування та вибір засобів реалізації продукту;
- проектування структурних складових та алгоритмів роботи додатку;
- провадження програмної реалізації;

Об'єктом дослідження є процес користування інтернет-магазином чоловічого одягу.

Предметом дослідження є сам процес розробки та використання інтернет-магазину чоловічого одягу з використанням мови програмування PHP. Розроблений інтернет-магазин може включати основні функції, які зазвичай присутні у подібних платформах, наприклад, перегляд каталогу товарів, додавання товарів у кошик, оформлення замовлення, керування товарами та інші.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1 Постановка задачі

Основною метою розробки інтернет-магазину чоловічого одягу на мові програмування PHP [16] є надання користувачам можливості зручного перегляду, вибору і покупки чоловічого одягу через веб-інтерфейс.

Для реалізації поставленого завдання, основними етапами є:

1. Аналіз проекту та збір вимог:
 - 1.1. Проведіть детальний аналіз вимог до функціональності вашого інтернет-магазину, включаючи можливості перегляду товарів, пошуку, додавання до кошика та оформлення замовлення;
 - 1.2. Визначити основні категорії товарів, їх атрибути (розмір, колір, матеріал тощо) і можливості фільтрації товарів;
 - 1.3. Розглянути вимоги щодо адміністративної панелі для керування товарами, замовленнями, клієнтами тощо;
2. Розробка бази даних:
 - 2.1. Спланувати структуру бази даних, яка включатиме таблиці для товарів, категорій, замовлень, користувачів тощо;
 - 2.2. Реалізувати механізм зв'язків між таблицями, наприклад, зв'язок один до багатьох між категоріями товарів і самими товарами;
 - 2.3. Забезпечте безпеку бази даних, використовуючи підготовлені запити або ORM;
3. Розробка функціональності:
 - 3.1. Реалізувати можливості перегляду списку товарів за категоріями, пошуку за різними параметрами, перегляду деталей конкретного товару;
 - 3.2. Розробити механізм додавання товарів до кошика, керування вмістом кошика і оформлення замовлення;
 - 3.3. Реалізувати систему авторизації та реєстрації користувачів для можливості входу, збереження персональних даних та перегляду історії.

3.4.Розробити адміністративну панель для управління товарами, замовленнями та іншими аспектами магазину

4. Дизайн та інтерфейс:

5.1 Створіть зручний та естетичний дизайн інтерфейсу магазину;

5.2 Реалізуйте адаптивний дизайн, щоб ваш магазин був добре відображений на різних пристроях (комп'ютерах, планшетах, смартфонах);

5.3 Забезпечити зручну навігацію та інтуїтивно зрозумілі елементи керування.

5. Тестування та забезпечення якості:

5.1.Провести ретельне тестування вашого інтернет-магазину для виявлення та виправлення помилок;

5.2.Переконайтеся, що всі функції, включаючи додавання товарів до кошик працюють без помилок;

Для досягнення поставленого завдання проект зосередиться на вищезазначених етапах [2], [3].

Успішна розробка та впровадження цього програмного забезпечення розширить можливості користувачів, надаючи їм доступ до широкого асортименту товарів і забезпечивши зручність в їхньому перегляді та покупці. Програмне забезпечення магазину забезпечить швидкий та безпроблемний пошук необхідних товарів, а також надасть можливість застосовувати фільтри для спрощення пошуку та вибору продуктів. Крім цього, інтернет-магазин буде обладнаний інструментами для замовлення, оплати та доставки, що робить процес покупок зручним і ефективним для клієнтів. Це рішення допоможе залучати більше клієнтів до вашого інтернет-магазину, гарантувати їхнє задоволення від покупок та сприяти росту вашого бізнесу.

1.2 Аналіз аналогів програмного продукту

Існує багато інтернет магазинів, які мають свої унікальні особливості та функціональні можливості. У цьому розділі ми проведемо аналіз трьох популярних магазинів в Україні: Kasta, le Boutique, Next.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						10

"Kasta" [13] - це інтернет-магазин, який має репутацію лідера у сфері чоловічого одягу. Тут ви знайдете широкий асортимент моделей, включаючи костюми для офісу, стильний вуличний одяг та аксесуари. "Kasta" завжди готовий задовольнити потреби сучасного чоловіка, надаючи якість і стиль в кожному виборі (рис. 1.1).

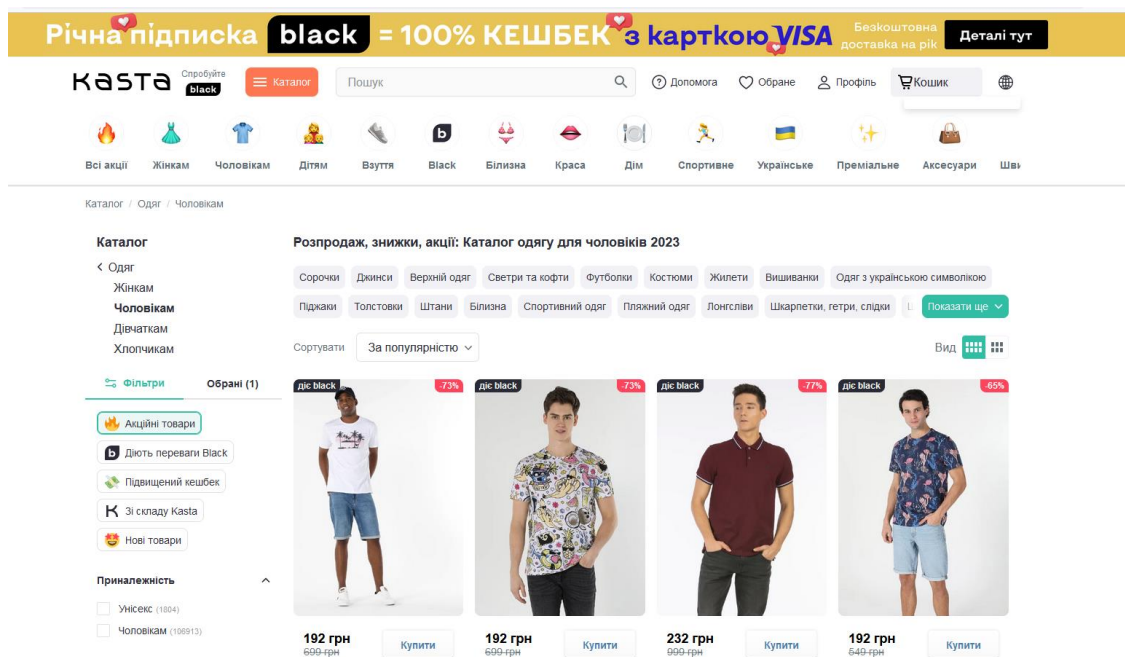


Рисунок 1.1 Інтерфейс магазину Kasta.

Переваги:

- Інтуїтивно зрозумілий і легкий у використанні інтерфейс;
- Широкий асортимент товарів для дорослих і дітей, задовольняючи різноманітні потреби клієнтів;
- Якісна функція реєстрації, яка надає можливість створити обліковий запис та зберігати особисту інформацію;
- Наявність номерів телефону для зв'язку з клієнтами і надання додаткової інформації;
- Особистий кабінет для зберігання даних, перегляду історії замовлень та налаштувань;

Недоліки:

- Застарілий дизайн, який може впливати на сприйняття інтерфейсу користувачами і виглядати менш привабливо в порівнянні з сучасними тенденціями дизайну;
- Існують баги з розмірами та локалізацією, що може викликати проблеми з правильним відображенням розмірів товарів або з адаптацією інтерфейсу до різних мов або регіональних налаштувань;
- Неякісне керування кошиком, що може призводити до складнощів при додаванні або видаленні товарів, редагуванні кількості тощо;

"Le Boutique" [11] - це вишуканий інтернет-магазин, спеціалізований на вищій моді для чоловіків. Тут ви зможете насолодитися шикарним вибором елегантних костюмів, модних сорочок, якісних аксесуарів та багато іншого. "Le Boutique" - це місце для справжніх цінителів стилю та якості (рис. 1.2).

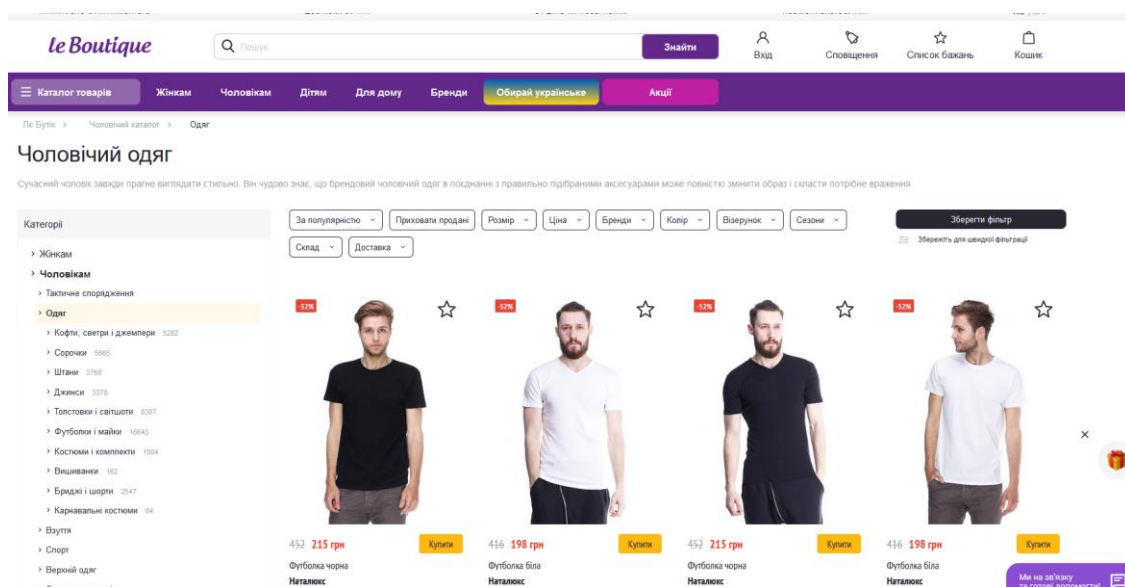


Рисунок 1.2 Інтерфейс магазину Le Boutique.

Переваги:

- Модний та сучасний дизайн, який відповідає актуальним тенденціям;
- Постійне оновлення асортименту з огляду на останні новинки;
- Легке додавання та видалення товарів у кошику, що забезпечує зручність покупок;

Недоліки:

- Недостатня адаптація до мобільних пристроїв може призвести до проблем зі зручністю перегляду та покупок на смартфонах та планшетах;
- Завантаження сторінок може бути повільним через велику кількість даних та зображень, що впливає на користувацький досвід;

"Next" [12] - це інтернет-магазин чоловічого одягу, який відомий своєю модністю та трендовими колекціями. Тут вас чекають останні модні новинки, включаючи молодіжний одяг, кросівки та аксесуари. "Next" допоможе вам підтримувати сучасний образ і виглядати завжди на висоті (рис.1.3).

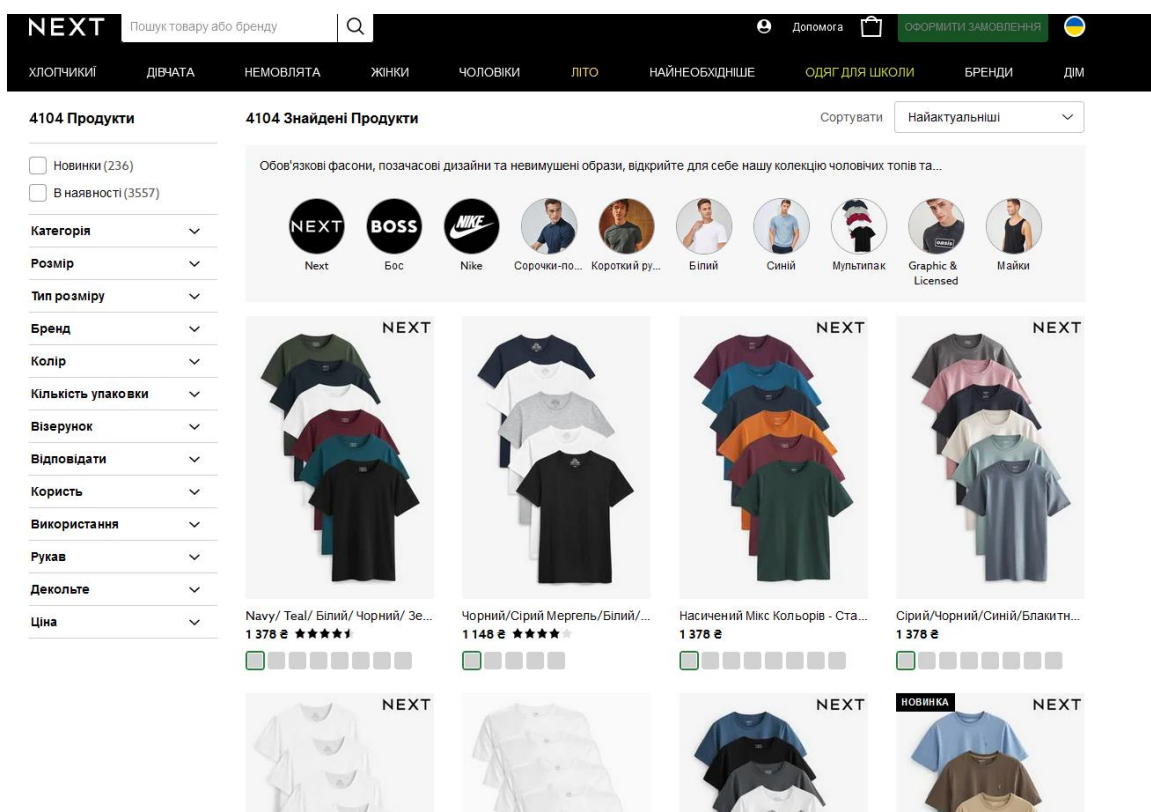


Рисунок 1.3 Інтерфейс Next.

Переваги:

- Вишуканий та елегантний дизайн, який надає інтерфейсу естетичний вигляд і приваблює користувачів;
- Великий асортимент вищої моди та стильного одягу для чоловіків;
- Професійна функція реєстрації, що дозволяє створювати облікові записи з багатьма особистими налаштуваннями;

Недоліки:

- Зважаючи на велику кількість транзакцій та обміну фінансовою інформацією, інтернет-магазин може бути більш вразливим до кібератак;
- З метою забезпечення однакового користувацького досвіду на різних платформах може вимагатися значний обсяг робіт з адаптації і оптимізації;

На основні аналізу аналогів програмного продукту було складено порівняльну таблицю:

Таблиця 1.1

Порівняння існуючих інтернет магазинів

Характеристика	Kasta	Le Boutique	Next
Дизайн	Простий	Сучасний	Сучасний
Навігація	Зручна	Зручна	Зручна
Асортимент	Великий	Широкий	Великий
Адаптивність	Присутня	Присутня	Присутня
Можливість реєстрації та авторизації	Присутня	Присутня	Присутня

Беручи до уваги всі аспекти, ми можемо ретельно проаналізувати та включити найкращі функції та можливості у наш власний продукт. Такий інтегрований підхід гарантує, що наш продукт не лише задовольняє різноманітні потреби користувачів, але й використовує сильні сторони цих інтернет магазинів.

Завдяки комплексному підходу наш продукт гарантує користувачам доступ до гарного та зрозумілого інтерфейсу, що надаватиме широкий спектр товарів, їх сортування та зручну адмін панель для адміністраторів.

1.3 Вибір архітектури інтернет магазину

Вибір архітектурного підходу для створення онлайн-системи для продажу будівельних матеріалів є критичним етапом у проектуванні. Один із найпоширеніших підходів - це архітектура MVC [1] (Model-View-Controller). Вона дозволяє розділити логіку магазину на три компоненти: модель, представлення та контролер, що забезпечує чітку структуру проекту, спрощує

тестування і підтримку проекту. Крім того, такий підхід сприяє масштабованості проекту та його розвитку.

Однією з основних переваг архітектури MVC є її гнучкість. Розділення логіки на три компоненти дозволяє внесення змін в одному компоненті без впливу на інші. Це полегшує ефективний розвиток і підтримку проекту у майбутньому. Крім того, такий підхід сприяє спрощенню коду та зменшенню залежностей між компонентами, що сприяє полегшенню розробки та тестування проекту.

Хоча існують інші архітектурні підходи, такі як MVP [1] (Model-View-Presenter) [2] та MVVM (Model-View-ViewModel) [3], використання архітектури MVC має свої переваги. Наприклад, MVC дозволяє збільшити продуктивність розробки завдяки можливості паралельної розробки окремих компонентів проекту. Крім того, MVC є досить простим у використанні, зі зрозумілим розподілом обов'язків між різними компонентами, що допомагає розробникам швидко орієнтуватися в коді (рис.1.4.).

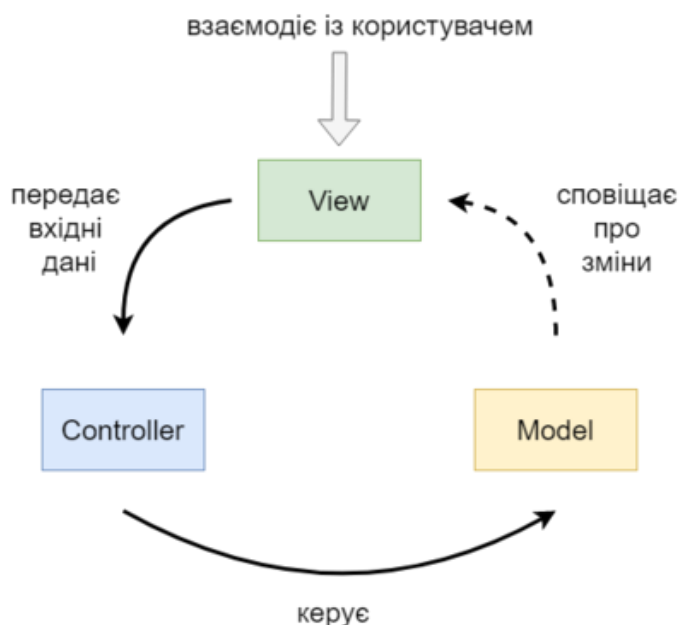


Рисунок 1.4 Архітектура MVC.

Систему керування базами даних було вибрано MySQL [3], оскільки вона є потужною, широко підтримується та має багатий функціонал.

Для адміністрування базами даних ми використовуємо PHPMyAdmin [4]. Обґрунтування цього вибору базується на декількох факторах:

1. Легкість використання: PHPMyAdmin пропонує користувачам зручний інтерфейс, що робить керування базами даних максимально простим. Ви можете створювати, редагувати та видаляти таблиці, запити та інші складові бази даних, не потребуючи великого досвіду в написанні складних SQL-запитів.
2. Функціональність: PHPMyAdmin оснащений широким спектром функцій, таких як імпорт та експорт даних, зміна структури таблиць, виконання складних запитів і багато інших. Це надає можливість зручно працювати з базами даних та виконувати різноманітні завдання безпосередньо з веб-інтерфейсу.
3. Підтримка MySQL: PHPMyAdmin розроблений спеціально для роботи з MySQL, однією з найпопулярніших систем управління базами даних. Це означає, що ви зможете ефективно керувати своїми базами даних MySQL та виконувати всі необхідні операції, пов'язані з ними, з використанням PHPMyAdmin.

Основна перевага такого підходу полягає в тому, що весь процес малювання відбувається в одному конвеєрі, що спрощує внесення нових функцій та забезпечує реалізацію за замовчуванням для функцій, які не підтримуються.

Таким чином була обрана комбінація Apache2 як веб-сервера та MySQL разом із PHPMyAdmin для управління базами даних, оскільки ці інструменти є надійними, широко використовуються та мають потужний функціонал.

1.4 Обґрунтування вибору інструментальних засобів та вимог до апаратного забезпечення

Для розробки серверної частини веб-застосунку був обраний набір інструментів, що включає наступні компоненти: веб-сервер Apache2, мову програмування PHP, систему керування базами даних MySQL та інструмент для адміністрування базами даних PHPMyAdmin.

Apache2 обраний в якості веб-сервера з метою забезпечення надійності та широкої підтримки. Цей сервер підтримує HTTP і HTTPS [2] протоколи і

забезпечує надійну доставку веб-сторінок клієнтам. Apache2 є відомим та довіреним рішенням для хостингу веб-додатків, і воно зарекомендувало себе як надійний веб-сервер з плінними можливостями.

CSS [15] - це файли, які містять описи стилів і використовуються для визначення вигляду та макету веб-сторінок. Використання CSS дозволяє відокремити стиль від контенту, що дає можливість змінювати зовнішній вигляд сторінки без впливу на її структуру та контент. CSS дозволяє встановлювати параметри, такі як кольори, шрифти, розміри, відступи, рамки та інші стилізаційні властивості для елементів на веб-сторінці.

JetBrains PhpStorm [16] є універсальним середовищем розробки, з основним акцентом на розробку на мові PHP. Це середовище надає розробникам інструменти для зручної роботи з кодом, включаючи інтелектуальний редактор, можливості рефакторингу, підтримку генерування коду та покращення синтаксису. PhpStorm також підтримує різні фреймворки на мові PHP і надає можливість гнучко налаштовувати плагіни. Ця програма дозволяє користувачам візуально налаштовувати інтерфейс, включаючи редактор для мови розмітки (HTML [14], CSS [15]), слідування за стилями коду (PSR1/PSR2) [18], потужні інструменти для роботи з базами даних.

Мова програмування PHP є широко використовуваною для розробки веб-додатків і має велику спільноту розробників. PHP дозволяє ефективно створювати веб-сторінки та працювати з базами даних, а також надає розширені можливості для розробки функціональних веб-застосунків.

MySQL використовується як система керування базами даних з великим набором функцій і широкими можливостями. Вона є потужною та надійною і забезпечує ефективне зберігання та обробку даних. MySQL є популярним вибором для багатьох веб-застосунків і користується великою спільнотою користувачів.

PHPMyAdmin є інструментом для управління базами даних MySQL. Він надає зручний веб-інтерфейс, що дозволяє створювати, редагувати та управляти базами даних без необхідності написання складних SQL-запитів. PHPMyAdmin

має широкий набір функцій, таких як імпорт та експорт даних, зміна структури таблиць та виконання складних запитів, що спрощує роботу з базами даних.

OpenServer [5] - це інтегроване розробницьке середовище, яке містить у собі Apache2 та додаткові компоненти для розробки веб-додатків.

Під час вибору апаратного забезпечення для сервера, на якому буде розміщений веб-застосунок, важливо враховувати вимоги, які поставлені обраними інструментами до серверного середовища. Зазвичай рекомендується мати достатньо оперативної пам'яті та процесорних ресурсів, щоб забезпечити оптимальну продуктивність і надійність веб-застосунку, який використовує Apache2, PHP, MySQL та PHPMyAdmin. Проте конкретні потреби в апаратному забезпеченні можуть відрізнятися в залежності від розміру проекту, передбачуваного навантаження та інших факторів. Тому рекомендується провести аналіз та оцінку навантаження для визначення оптимальних параметрів апаратного середовища.

Висновки до першого розділу

У даному розділі було проведено аналіз використання інформаційних технологій у системах дистанційного навчання, визначено завдання та виконано порівняльний аналіз наявних аналогів програмного продукту. На підставі цього аналізу була обрана архітектура для системи дистанційного навчання.

Під час аналізу аналогів було виявлено декілька схожих систем, які використовуються для дистанційного навчання. Цей аналіз допоміг з'ясувати переваги та недоліки цих систем і врахувати їх при розробці нової системи.

При виборі архітектури системи для дистанційного навчання враховувалися основні вимоги проекту. Обрана архітектура має забезпечити ефективну організацію навчального процесу та зручний інтерфейс для користувачів. Для розробки системи були обрані відповідні інструментальні засоби, які відповідають потребам проекту.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНТЕРНЕТ МАГАЗИНУ З ПРОДАЖУ ЧОЛОВІЧОГО ОДЯГУ

2.1 Визначення варіантів використання та структури додатку

У цьому розділі подано аналіз вимог до програмного продукту, з основним фокусом на виявленні функціональних та нефункціональних вимог. Також були визначені вимоги до часових характеристик відгуку додатку на дії користувача, і розглянуті різні сценарії використання.

Аналіз вимог передбачає ретельне дослідження потреб та очікувань користувачів щодо функціональності та характеристик веб-додатку. Основною метою аналізу вимог є зрозуміти, які функції та можливості необхідно впровадити в програмне забезпечення для задоволення потреб користувачів.

Функціональні вимоги:

- можливість авторизації користувачів;
- валідація введених даних;
- пошук і фільтрація товарів;
- адміністративний функціонал: окремий доступ для адміністраторів, що дозволяє керувати каталогом товарів;
- можливість збереження інформації: система повинна зберігати потрібну інформацію в базу даних;

Нефункціональні вимоги:

- забезпечення зручного оформлення замовлення та способів оплати;
- час відповіді на запити має бути мінімальним, зокрема для сторінок;
- підтримка різних веб-браузерів та пристроїв, забезпечення адаптивного дизайну для зручного відображення на різних розмірах екранів
- система повинна забезпечувати час відповіді на звичайні запити не більше 5 секунд, а на більш складні запити - не більше 10 секунд;
- інтерфейс повинен бути інтуїтивно зручним для користувача та не вимагати додаткової підготовки;
- забезпечення зручного оформлення замовлення та способів оплати;

- швидке отримання результатів при здійсненні функцій програми;

Системні вимоги:

1. Вимоги до середовища виконання:

Система повинна задовольняти зазначеним вимогам на комп'ютері або смартфоні в наступній мінімальній комплектації:

- підтримка браузера Google Chrome версії 45+;
- підтримка браузера Opera версії 30+;
- підтримка браузера Mozilla Firefox версії 38+;
- підтримка браузера Microsoft Edge версії 12+;
- підтримка браузера Safari версії 9+;

2. Вимоги до СУБД та доступу до даних:

У ядрі системи повинна бути використана СУБД реляційного типу. Таким чином, після визначення всіх вимог, можливостей і обмежень системи, можна створити діаграму варіантів використання. Ця діаграма візуалізує взаємодію між акторами (користувачами) та системою в межах конкретних функціональних сценаріїв або використання системи. Вона дозволяє ідентифікувати основні функції системи та розкрити деталі взаємодії між користувачами та системою у формі взаємодій (use cases).

Діаграма варіантів використання допомагає уточнити вимоги до системи, ілюструє, як система буде використовуватися в реальних сценаріях, і є важливим інструментом для розуміння функціональних вимог та взаємодії між компонентами системи (рис 2.1).

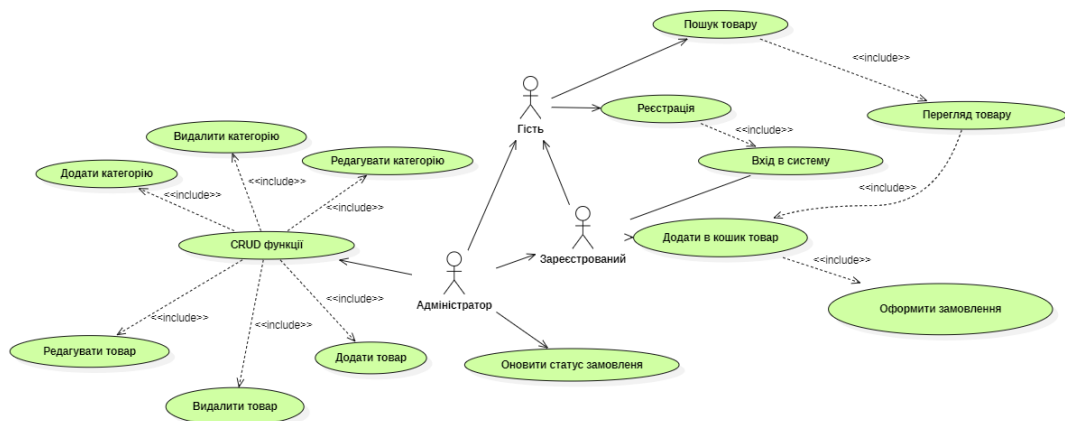


Рисунок 2.1 Діаграма варіантів використання.

Користувач:

- пошук товару в каталогі, по категоріям або по панелі;
- перегляд товару;
- реєстрація та авторизація;
- додати в кошик товар;
- оформити замовлення;

Адміністратор:

- CRUD функції товарів та категорій
- Оновлення статусу замовлення

Отже, система включає три основних акторів: адміністратора, гостя та користувача, кожен з яких має свої унікальні функції та можливості в системі.

Кожен актор має свої унікальні функції та можливості в системі.

Гість - це користувач, який ще не автентифікувався в системі. Його можливості обмежені, і він має доступ лише до функцій автентифікації та перегляду товарів. Іншими словами, гість може виконувати процедуру входу в систему для отримання ідентифікаційних прав і перегляду товарів, але він не може здійснювати покупки. Після успішного входу в систему, користувач отримує доступ до функції додавання товарів у кошик та оформлення замовлень.

Адміністратор має повний контроль над системою. Його функції включають створення та редагування товарів і категорій товарів в системі. Крім цього, адміністратор може переглядати замовлення та підтверджувати їх.

Після аналізу вимог до системи та створення діаграми використання, наступним кроком є побудова діаграми класів [17], яка буде розроблена за допомогою інструменту StarUML [18]. Діаграма класів є важливим етапом, оскільки вона допомагає візуалізувати структуру системи, основні сутності та зв'язки між ними.

При побудові діаграми класів визначаються класи системи, їх атрибути, методи та взаємозв'язки. Кожен клас представляє сутність або компонент системи з власними характеристиками та функціональністю. Елементами

діаграми класів є класи, зв'язки між класами (такі як асоціації, композиції та агрегації) та методи, які визначають операції, доступні для об'єктів цих класів.

Тому нижче наведена діаграма (рис 2.2) надає загальний огляд структури нашого проекту та відображає основні класи, їх атрибути та зв'язки між ними. Ця діаграма допомагає зрозуміти організацію даних в системі та їх взаємодію.

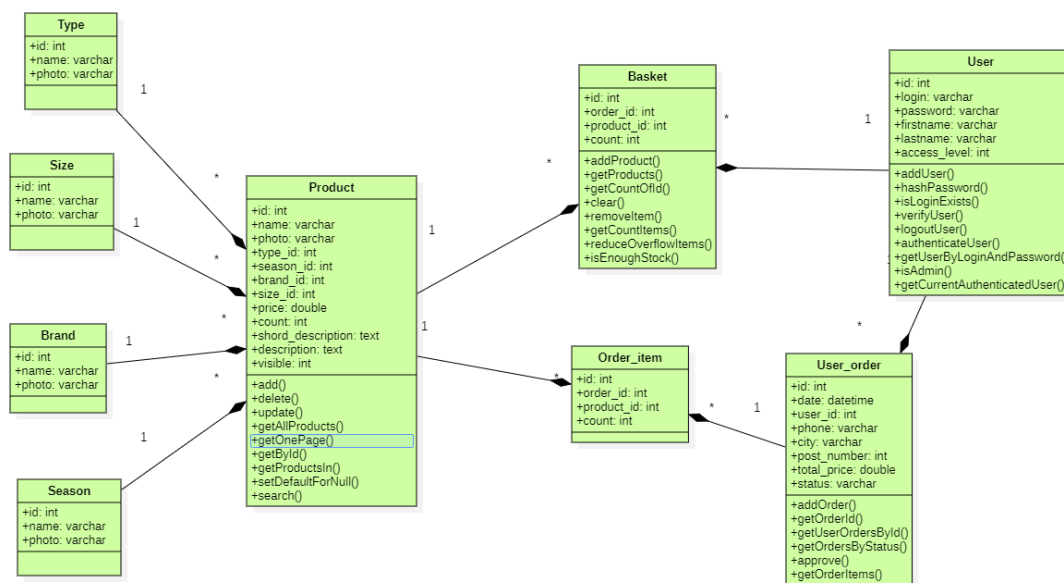


Рисунок 2.2 – Діаграма класів інтернет магазину

У шаблоні проектування MVC (Model-View-Controller), модель відіграє роль компонента, який керує логікою додатку і взаємодіє з базою даних або зовнішніми сервісами. Її функціональність охоплює обробку даних та бізнес-логіку, що визначають, як дані зберігаються та обробляються. Модель може включати в себе класи та функції, призначені для виконання операцій з даними, таких як отримання, оновлення, видалення та створення нових записів.

В додаток до цього, модель може вміщувати методи для виконання розрахунків, перевірки даних, валідації, обробки помилок і так далі. Також, модель може включати логіку для спілкування з іншими компонентами в рамках додатку, відправлення та отримання повідомлень, обробки подій або викликів функцій. Модель використовується контролером для взаємодії з нею та представленням. Контролер викликає методи моделі для отримання або зміни даних і потім передає ці дані представленню для їх відображення.

Клас **User** містить інформацію про зареєстрованого користувача, таку як його ідентифікатор, логін, ім'я, прізвище, пароль і призначену роль. Клас **User** також надає ряд методів для взаємодії з користувачами у системі. Основні методи класу **User** включають:

- addUser() дозволяє реєструвати нових користувачів у системі, які ще не мають облікового запису;
- hashPassword() використовується для хешування паролів користувачів, щоб забезпечити безпеку їх зберігання;
- verifyUser() перевіряє, чи правильно введені логін і пароль користувача.
- logoutUser() надає можливість авторизованому користувачу вийти з свого облікового запису;
- isAdmin() перевіряє чи має поточний користувач доступ до адміністративної панелі;

Клас **Product** зберігає інформацію про товари, таку як їх ідентифікатор, назва, фото, ціна, опис, кількість, видимість та категорії. Основні методи класу **Product** включають:

- add() додає новий продукт до бази даних, використовуючи передані дані та шлях до фотографії продукту;
- delete() видаляє продукт з бази даних за його ідентифікатором;
- update() оновлює існуючі дані продукту, використовуючи нові дані, які передаються у вигляді асоціативного масиву та ідентифікатора продукту;
- count() повертає загальну кількість продуктів, які знаходяться у базі даних;
- search() виконує пошук продуктів за заданою рядковою фразою;
- getOnePage() повертає сторінку зі списком продуктів на основі поточної сторінки та кількості продуктів, які відображаються на сторінці;

Клас **User_order** містить інформацію про замовлення товарів, включаючи ідентифікатор, назву товару, фотографію, ціну, опис, кількість, видимість та категорії. Основні методи класу **User_order** включають:

- `addOrder()` додає нове замовлення до бази даних, зберігаючи всю необхідну інформацію про товари в цьому замовленні;
- `getOrderId()` повертає ідентифікатор замовлення на основі заданих значень ідентифікатора користувача та дати;
- `getUserOrdersById()` повертає всі замовлення користувача, використовуючи його ідентифікатор;
- `getOrdersByStatus()` повертає всі замовлення з вказаним статусом;
- `approve()` змінює статус замовлення на "approved" за його ідентифікатором;
- `getOrderItems()` повертає всі елементи замовлення для заданого ідентифікатора замовлення, забезпечуючи детальну інформацію про товари, включаючи їх назву, фотографію, ціну, опис і кількість;

Клас **Basket** зберігає інформацію про товари, які розміщені у кошику користувача. Основні методи класу **Basket** включають:

- `addProduct()` додає продукт до кошика користувача;
- `getProducts()` повертає всі продукти, які знаходяться у кошику користувача;
- `getCountOfId()` повертає кількість конкретного продукту за його ідентифікатором у кошику користувача;
- `clear()` видаляє всі записи про продукти у кошику користувача, очищуючи кошик;
- `removeItem()` видаляє запис про конкретний продукт з кошика користувача за його ідентифікатором;
- `getCountItems()` повертає загальну кількість продуктів у кошику користувача;
- `reduceOverflowItems()` зменшує кількість продуктів у кошику користувача, якщо кількість товару в кошику перевищує доступну кількість на складі;
- `isEnoughStock()` перевіряє, чи є достатня кількість товару на складі для замовлення, допомагаючи уникнути надмірного замовлення товарів;

2.2 Розробка бази даних системи

Для забезпечення ефективної роботи інформаційної системи важливо мати добре структуровану базу даних, яка буде зберігати інформацію про користувачів, класи, предмети, тести та інші важливі дані. Необхідно правильно спроектувати структуру цієї бази даних, визначити відповідні типи даних для кожного атрибута та встановити зв'язки між таблицями для забезпечення цілісності даних і оптимальної продуктивності системи.

Оскільки розроблювана платформа має велику кількість компонентів, вона також включає в себе різні таблиці в базі даних. У наведеній таблиці та на рисунку представлені основні таблиці, які складають нашу базу даних. Ці таблиці відображають важливі аспекти системи і забезпечують зберігання, організацію та доступ до даних (рис 2.3).

Схема бази даних містить наступні таблиці:

- Brand
- Basket
- Season
- Size
- Order_item
- Product
- Type
- User
- User_order

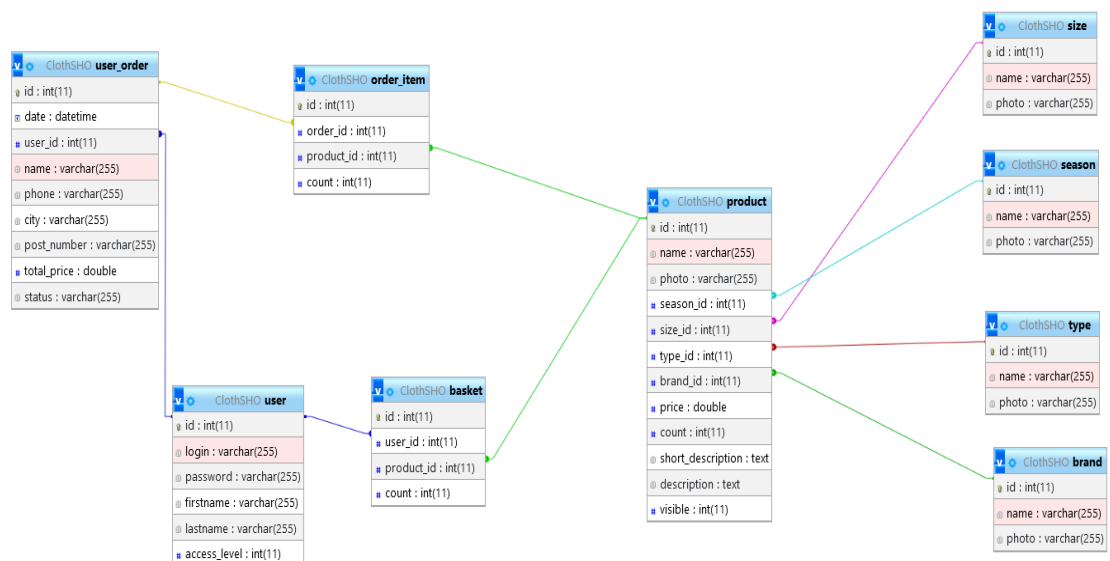


Рисунок 2.3 Діаграма бази даних

Нижче наведений опис кожної з наведених таблиць бази даних:
Для зберігання інформації про користувачів, була створена таблиця під назвою "Users". Ця таблиця містить дані, що стосуються користувачів системи.

Структура таблиці наведена нижче:

Структура таблиці "Users"

Таблиця 2.1

Назва	Тип	ППК	ЗЗК	Опис
id	int	+	-	ID у базі
login	varchar(50)	-	-	Логін для авторизації у системі
pass	Varchar(50)	-	-	Пароль аккаунту у системі
access	int	-	+	Рівень доступу
firstname	varchar(20)	-	-	Ім'я користувача
lastname	varchar(20)	-	-	Прізвище користувача

Для збереження товару в корзині, було створено табличку basket. В цій таблиці зберігається інформація про товари для подальшого придбання.

Структура таблиці наведена нижче:

Структура таблиці " basket "

Таблиця 2.2

Назва	Тип	ППК	ЗЗК	Опис
id	int	+	-	ID у базі
User_id	int	-	+	ID юзера
Product_id	int	-	+	ID товару
quantity	int	-	+	Кількість

Для збереження інформації про замовлення, які зробив користувач, була створена таблиця з назвою user_order. У цій таблиці міститься інформація про окремі замовлення, включаючи їх деталі, статус, інформацію про користувача та дату замовлення.

Структура таблиці наведена нижче:

Структура таблиці "user_order "

Таблиця 2.3

Назва	Тип	ППК	ЗЗК	Опис
id	int	+	-	ID у базі
date	datetime	-	-	Дата
User_id	int	-	+	ID користувача
name	int	-	-	Ім'я користувача
phone	varchar(255)	-	-	Номер телефону
city	varchar(255)	-	-	Місто
Post_number	Varchar(255)	-	-	Поштовий номер
Total_price	double	-	+	Повна ціна
status	Varchar(255)	-	-	Статус товару

Для збереження зв'язку між замовленнями та їх складовими елементами у базі даних була створена таблиця з назвою order_item. Ця таблиця містить інформацію про товари, які були замовлені в рамках певного замовлення.

Структура таблиці наведена нижче

Структура таблиці "order_item "

Таблиця 2.4

Назва	Тип	ППК	ЗЗК	Опис
id	int	+	-	ID у базі
order_id	int	-	+	Дата
product_id	int	-	+	ID користувача
count	int	-	+	Ім'я користувача

Сортуючи таблиці які сортують товари по категоріям.

Структура таблиці наведена нижче:

Структури таблиць "size/brand/type/season"

Таблиця 2.5

Назва	Тип	ППК	ЗЗК	Опис
id	int	+	-	ID у базі
name	Varchar(255)	-	-	Дата
photo	Varchar(255)	-	-	ID користувача

Для збереження інформації про товари була створена таблиця з назвою product. Ця таблиця містить всі дані про товари та надає можливість їх відображення на веб-сайті:

Структура таблиці "product "

Таблиця 2.6

Назва	Тип	ПП К	ЗЗК	Опис
id	int	+	-	ID у базі
name	datetime	-	-	Назва товару
photo	int	-	-	Фото товару
Type_id	int	-	+	Тип товару
Gender_id	varchar(255)	-	+	Стать до якої підходить товар
Material_id	varchar(255)	-	+	Матеріал товару
Brand_id	Varchar(255)	-	+	Бренд товару
price	double	-	+	Ціна товару
count	Varchar(255)	-	+	Кількість товару
Short_description	text	-	-	Короткий опис товару
description	text	-	-	Повний опис товару
visible	int	-	+	Видимість товару

2.3 Проектування та реалізація алгоритмів роботи додатку

Розробка та реалізація системних алгоритмів відповідають за виконання різних обчислювальних завдань, обробку даних і прийняття рішень в системі. Для досягнення більшої ясності і кращого розуміння послідовності подій в системі, було створено низку діаграм активності. Ці діаграми надають зручний спосіб візуалізації як послідовних, так і одночасних дій, які відбуваються під час виконання процесів.

Розглянемо приклад діаграми активності для процесу купівлі товару (рис.2.4). з урахуванням різних ролей, таких як "Адміністратор", "Гість" і "Авторизований користувач":

На цій діаграмі ми можемо виділити наступні ключові елементи та послідовність подій:

- 1. **Головна сторінка:** Процес починається з вибору користувачем опцій на головній сторінці.
- 2. **Роль "Гість":** Гість може переглядати товари і вибирати їх, але не може їх купити без авторизації. Якщо гість намагається здійснити покупку, система вимагає від нього ввійти або зареєструватися.
- 3. **Роль "Авторизований користувач":** Після авторизації гість стає авторизованим користувачем і отримує доступ до додавання товарів до кошика та оформлення замовлення.
- 4. **Роль "Адміністратор":** Адміністратор має розширені права, включаючи можливість додавання, редагування, видалення та керування товарами. Він також може приймати та обробляти замовлення від користувачів.

Ця діаграма активності допомагає зрозуміти, як взаємодіють різні ролі у процесі купівлі товарів та які можливості та обмеження є для кожної з них.

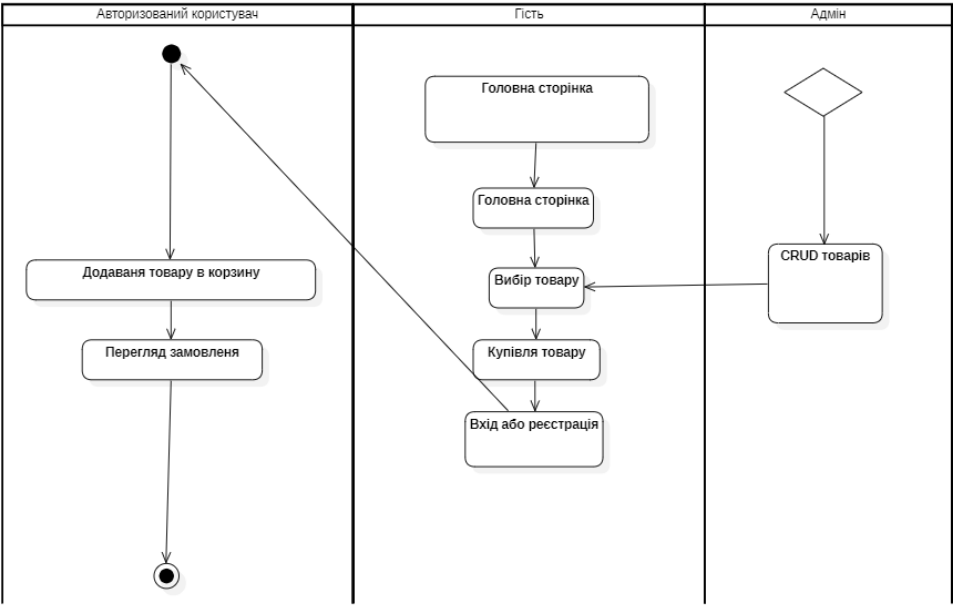


Рисунок 2.4 Діаграма активності для купівлі товару.

- 1. **Початок:** Початок виконання діаграми.
- 2. **Роль "Адміністратор":** Адміністратор має можливість виконувати CRUD-операції (створення, читання, оновлення та видалення) для товарів

та категорій. Це включає в себе створення нових елементів, перегляд і редагування існуючих, а також видалення.

3. **Роль "Авторизований користувач"**: Авторизований користувач може вибирати товари та переглядати категорії. Це може включати вибір конкретного товару, перегляд категорій товарів та вибір товару за категорією.
4. **Товари та Категорії**: Ці об'єкти представляють конкретні товари та категорії, які доступні у системі.
5. **CRUD Операції**: Ці активності відображають CRUD-операції, які можна виконати для товарів та категорій. Наприклад, створення нового товару, оновлення інформації про товар, видалення товару, тощо.
6. **Вибір Категорій та Товарів**: Ці активності відображають вибір категорій та товарів за допомогою інтерфейсу.

Наступна діаграма (рис. 2.5) допомагає візуалізувати, як адміністратор може управляти товарами та категоріями, а авторизовані користувачі можуть вибирати їх.

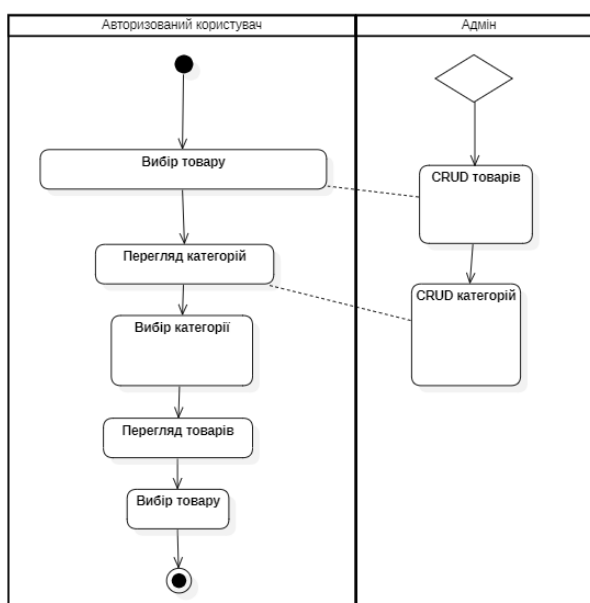


Рисунок 2.5 Діаграма активності для вибору товару.

Ці діаграми активності та таблиці специфікації грають важливу роль у візуалізації та уявленні послідовності дій та взаємодії користувача з інтернет-магазином.

Діаграма послідовності в магазині допомагає показати, як різні об'єкти та компоненти системи взаємодіють під час виконання певних функцій. Вона відображає, як інформація та дії передаються між об'єктами системи і як вони спільно працюють для досягнення певних цілей.

Діаграма послідовності, яка описує дії користувача під час купівлі товару, надає зрозумілий ілюстративний засіб для розуміння того, як користувач взаємодіє з системою під час цього процесу. Вона допомагає відобразити послідовність кроків, включаючи вибір товару, можливу авторизацію або реєстрацію, додавання товару до корзини та завершення оплати. Ця діаграма спрощує відображення та розуміння важливих аспектів користувацького досвіду в магазині (рис.2.6).

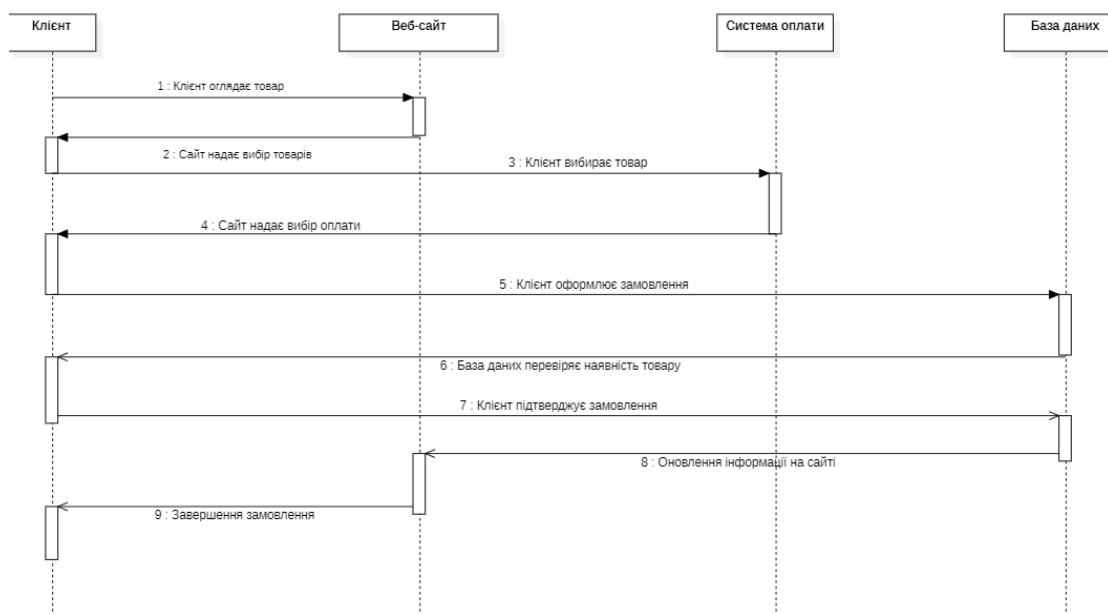


Рисунок 2.6 Діаграма послідовності купівлі товару.

Загалом, діаграми послідовності надають лаконічне і зрозуміле відображення динаміки взаємодії та потоку повідомлень у системі. Вони відіграють важливу роль у розумінні послідовності подій та обміну інформацією, допомагаючи краще аналізувати та розуміти поведінку програми.

Ефективний процес розробки та впровадження цих системних алгоритмів дозволяє забезпечити безперебійну роботу системи. Це включає в себе забезпечення користувачам ефективного управління завданнями, надійну

комунікацію, інтеграцію з зовнішніми інструментами та здатність генерувати докладні звіти.

2.4 Реалізація функціоналу інтернет магазину з продажу чоловічого одягу

У цьому розділі ми обговоримо етапи процесу впровадження та висвітлимо ключові аспекти розробки додатку.

По-перше, ми визначаємо схему валідації, яка описує правила та обмеження для певних полів форми, а також повідомлення про помилку, яке має відображатися у випадку невідповідності.

Далі поданий код перевіряє наявність користувача з вказаним логіном в базі даних. Він використовує статичний метод **isLoginExists()** класу **User**.

Код отримує доступ до об'єкта бази даних і виконує запит до таблиці користувачів, використовуючи переданий логін як умову пошуку. Результат запиту зберігається у змінній.

Після цього перевіряється, чи не є **User** порожнім. Якщо результат не є порожнім, це означає, що користувач з вказаним логіном вже існує у базі даних, тому метод повертає **true**. У іншому випадку, коли результат порожній, це означає, що користувача з таким логіном немає у базі даних, тому метод повертає **false**.

```
public static function isLoginExists($login) {  
    $user = \core\Core::getInstance()->db->select(self::$tableName, '*', [  
        'login' => $login  
    ]);  
    return !empty($user);  
}
```

В якості методу для обробки паролів ми використовуємо функцію (**md5**) яка широко використовується для збереження хеш-сум паролів у базі даних. Замість зберігання паролів у відкритому вигляді, ми обчислюємо їх хеш-суми за допомогою **md5** і зберігаємо ці хеш-суми. При перевірці введеного пароля, він

також хешується за допомогою **md5**, і отримана хеш-сума порівнюється зі збереженою хеш-сумою в базі даних.

Цей підхід забезпечує підвищений рівень безпеки, оскільки паролі не зберігаються у відкритому вигляді, і їхні хеш-суми мають складну структуру, яка важко піддається розшифруванню. Такий метод допомагає захистити конфіденційні дані користувачів у випадку, якщо база даних стане доступною для несанкціонованого доступу

```
public static function hashPassword($password) {  
  
    return md5($password);  
}
```

Цей код відповідає за перевірку логіну та паролю, автентифікацію користувача, якщо вони є вірними, та відображення повідомлення про помилку, якщо логін або пароль недійсні.

```
public function loginAction() {  
    if(User::isUserAuthenticated())  
        $this->redirect('/');  
    if (Core::getInstance()->requestMethod === 'POST') {  
        $user = User::getUserByLoginAndPassword($_POST['login'], $_POST['password']);  
        $error = null;  
        if(empty($user)) {  
            $error = 'Incorrect login or password';  
        } else {  
            User::authenticateUser($user);  
            $this->redirect('/');  
        }  
    }  
    return $this->render(null, [  
        'error' => $error  
    ]);  
}
```

Цей метод використовується для пошуку даних в таблиці за певним рядком. Створюється змінна яка є масивом умов для пошуку.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						33

```

public static function search($searchString, $all) {
    $conditionsArray = [
        'name' => "%{$searchString}%"
    ];
    if(!$all)
        $conditionsArray += ['visible' => 1];
    $rows = Core::getInstance()->db->select(self::$tableName, '*', $conditionsArray,
'LIKE');
    return array_reverse($rows);
}

```

Сортування відбувається наступним чином в функції **getSortedItem**

Цей метод повертає всі записи з таблиці, відсортовані за зростанням значень стовпця "name", з урахуванням того, що рядок зі значенням "Default" буде розташований на початку списку.

```

public static function getSortedItems($tableName) {
    $rows = self::getItems($tableName);
    $name = array_column($rows, 'name');
    array_multisort($name, SORT_ASC, $rows);
    // Move 'Default' item to the first place
    $i = 0;
    foreach ($rows as $key => $value) {
        if($value['name'] === 'Default') {
            $default = $rows[$i];
            unset($rows[$key]);
            $rows = array($default) + $rows;
            break;
        }
        $i++;
    }
    return $rows;
}

```

Наступний метод перевіряє наявність товару у корзині і додає його або оновлює кількість, якщо він вже є. Якщо товару немає у корзині, він додається з вказаною кількістю.

```
public static function addProduct($product_id, $count = 1) {
    $rowProduct = Product::getById($product_id);
    $user_id = User::getCurrentAuthenticatedUser()['id'];
    if(isset($rowProduct)) {
        $rowBasket = Core::getInstance()->db->select(self::$tableName, '*', [
            'user_id' => $user_id,
            'product_id' => $product_id
        ])[0];
        if(isset($rowBasket)) {
            Core::getInstance()->db->update(self::$tableName, [
                'count' => $rowBasket['count'] + $count
            ], [
                'id' => $rowBasket['id']
            ]);
        }
    }
    else {
        Core::getInstance()->db->insert(self::$tableName, [
            'user_id' => $user_id,
            'product_id' => $rowProduct['id'],
            'count' => $count
        ]);
    }
}
```

Цей код виконує перетворення корзини в замовлення шляхом вставки відповідних рядків в таблиці **user_order** і **order_item**, а також оновлення кількості товарів.

```
public static function addOrder($row) {
    $products = $row['products'];
    $fieldsList = ['date', 'user_id', 'name',
```

```

        'phone', 'city', 'post_number', 'total_price'];
$rowOrder = Filter::filterArray($row, $fieldsList);
Core::getInstance()->db->insert(self::$tableName, $rowOrder);
$orderId = self::getOrderId($rowOrder['user_id'], $rowOrder['date']);
foreach ($products as $product) {
    if($product['product']['count'] != 0) {
        $product['product']['count'] -= $product['count'];
        Product::update($product['product']['id'], $product['product']);
        Core::getInstance()->db->insert('order_item', [
            'order_id' => $orderId,
            'product_id' => $product['product']['id'],
            'count' => $product['count']
        ]);
    }
}
}
}

```

Висновки до другого розділу

У цьому розділі ми глибоко розглянули процес розробки та реалізації інтернет-магазину на РНР. По-перше, ми визначили сценарії використання та створили об'єктно-орієнтовану структуру магазину. За допомогою ретельного аналізу ми визначили ключові функціональні можливості, необхідні для задоволення потреб різних користувачів.

Далі ми зосередилися на описі структури даних, з якими працює магазин. Ми розглянули класи, які використовуються для обробки товарів, корзини покупок, замовлень та користувачів.

Потім ми розпочали розробку та впровадження функціональності системи. Це включало створення ефективних алгоритмів для додавання товарів у корзину та оформлення замовлення. Ми також забезпечили можливість перевірки наявності товарів на складі перед оформленням замовлення.

Ми перетворили дизайн системи та функціональні алгоритми на повнофункціональний інтернет-магазин. Протягом усього процесу реалізації ми надавали пріоритет зручності використання та масштабованості, щоб створити надійне та зручне для користувача рішення.

Загалом, цей розділ був присвячений різним етапам розробки та впровадження інтернет-магазину, включаючи визначення сценаріїв використання, об'єктно-орієнтовану структуру та функціональні алгоритми. Використовуючи найкращі практики розробки веб-додатків та відповідні технології, ми створили повнофункціональний інтернет-магазин, який задовольняє потреби користувачів у зручному та безпечному придбанні товарів.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНТЕРНЕТ МАГАЗИНОМ

3.1 Порядок встановлення та налаштування системи

Розроблений веб-сайт є онлайн-платформою, яка забезпечує зручну можливість здійснення покупок віддалено та доступний через різні веб-браузери.

Однією з основних переваг цього інтернет-магазину є його безпроблемна доступність для користувачів без необхідності використання спеціалізованого обладнання або проведення складних налаштувань. Цей інтернет-магазин спроектований з урахуванням максимальної зручності для користувачів, що дозволяє їм здійснювати покупки без зайвих клопотів або необхідності встановлення спеціального програмного забезпечення.

Для налагодження та налаштування нашого магазину потрібно розгорнути всі компоненти системи та належним чином налаштувати як окремі елементи, так і систему в цілому. Для цього нам допомагатимуть діаграми розгортання.

Діаграма розгортання - це інструмент, що відображає конфігурацію вузлів обробки часу виконання та компонентів, які працюють на цих вузлах. Діаграми розгортання - це варіант структурної діаграми, яка використовується для моделювання фізичних аспектів об'єктно-орієнтованої системи. Зазвичай їх використовують для моделювання статичного аспекту системи та її фізичного розміщення (рис. 3.1).

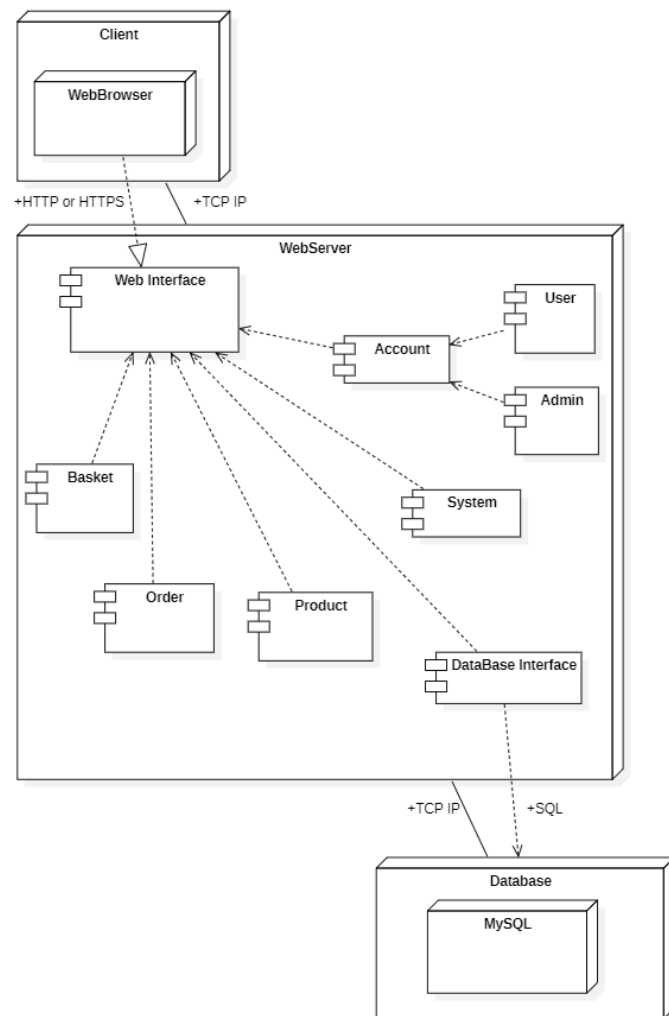


Рисунок 3.1. Діаграма розгортання.

Діаграма розгортання вказує, які компоненти системи функціонують на окремих обчислювальних пристроях під час виконання програми. Вона допомагає нам визначити, де знаходяться різні частини системи. У цьому проекті структура діаграми розгортання виглядає так:

- Web Interface - цей компонент включає в себе шаблони веб-сайту, що використовуються для взаємодії користувачів із системою через веб-браузер. Вона забезпечує графічний інтерфейс, через який користувачі можуть взаємодіяти з системою.
- DataBase Interface - цей компонент відповідає за взаємодію з базою даних системи. Вона забезпечує доступ до бази даних та обробку запитів, що дозволяє зберігати та отримувати необхідну інформацію.

- Log file - це файл, який використовується системним адміністратором для відстеження та налагодження роботи системи. Він містить інформацію про події та помилки, які виникають під час функціонування системи.
- Web Browser - це програмне забезпечення, встановлене на пристрої користувача, яке використовується для взаємодії з системою. Веб-браузер дозволяє користувачам отримувати доступ до функціоналу системи через веб-інтерфейс.

3.2 Інтерфейс магазину та порядок роботи

На головній сторінці (рис. 3.2) буде вітання щодо магазину, а також буде розміщено фотографію самого магазину. У верхній частині сторінки розташований рядок для пошуку за текстом, а також список сортувань за статтею, брендом, темою та типом. Крім цього, в правому верхньому куті будуть доступні посилання на сторінки для входу та реєстрації користувачів. У нижній частині сторінки буде розташований футер, який надасть можливість переглядати додаткову інформацію.

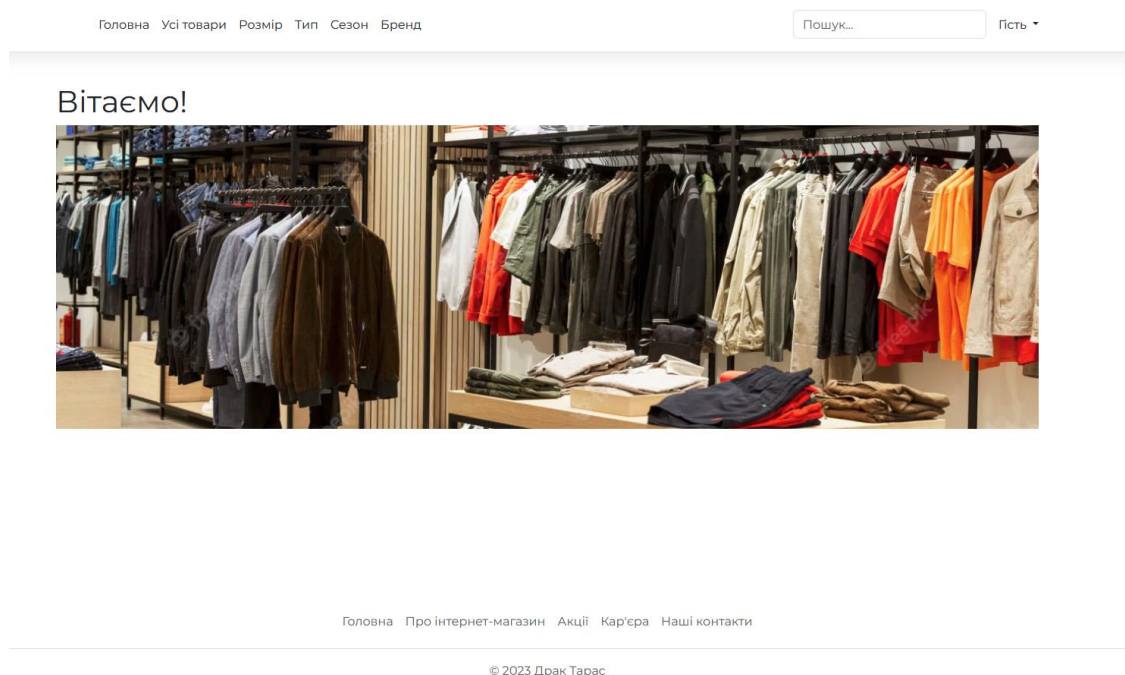


Рисунок 3.2. Головна сторінка.

При переході до модуля "Усі товари" будуть відображені всі товари, впорядковані за новизною поступлення (рис. 3.3.).

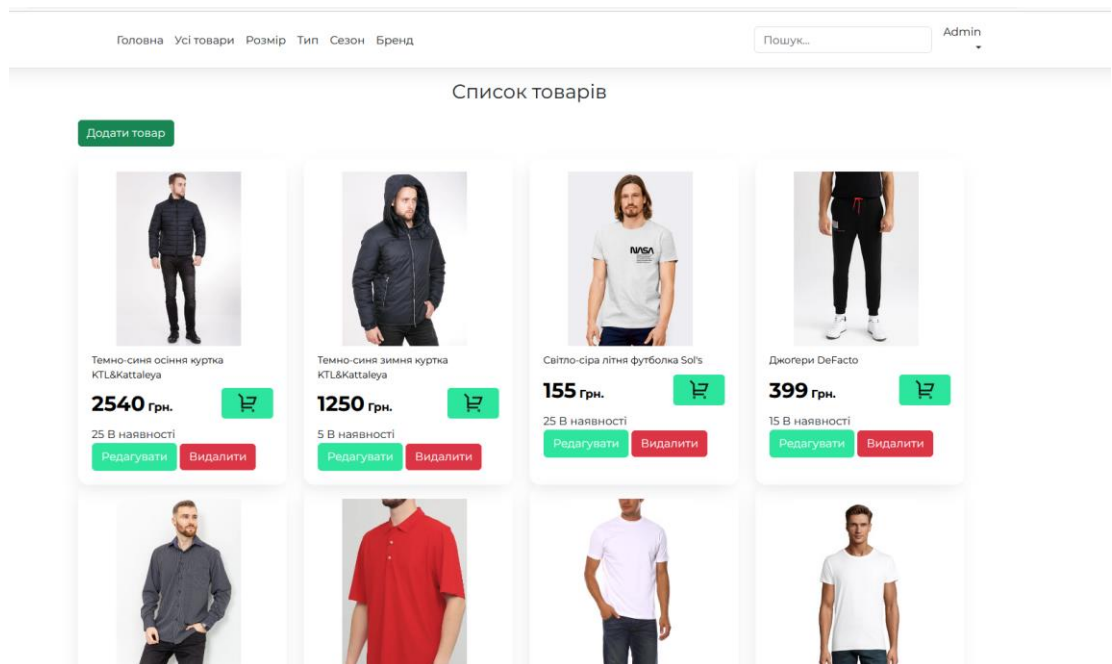


Рисунок 3.3. Модуль усіх товарів.

Після перегляду сторінки з переліком товарів, переходимо до розділу "Бренди", де будуть відображатися лише виробники товарів та кількість товарів від цього виробника (рис. 3.4).

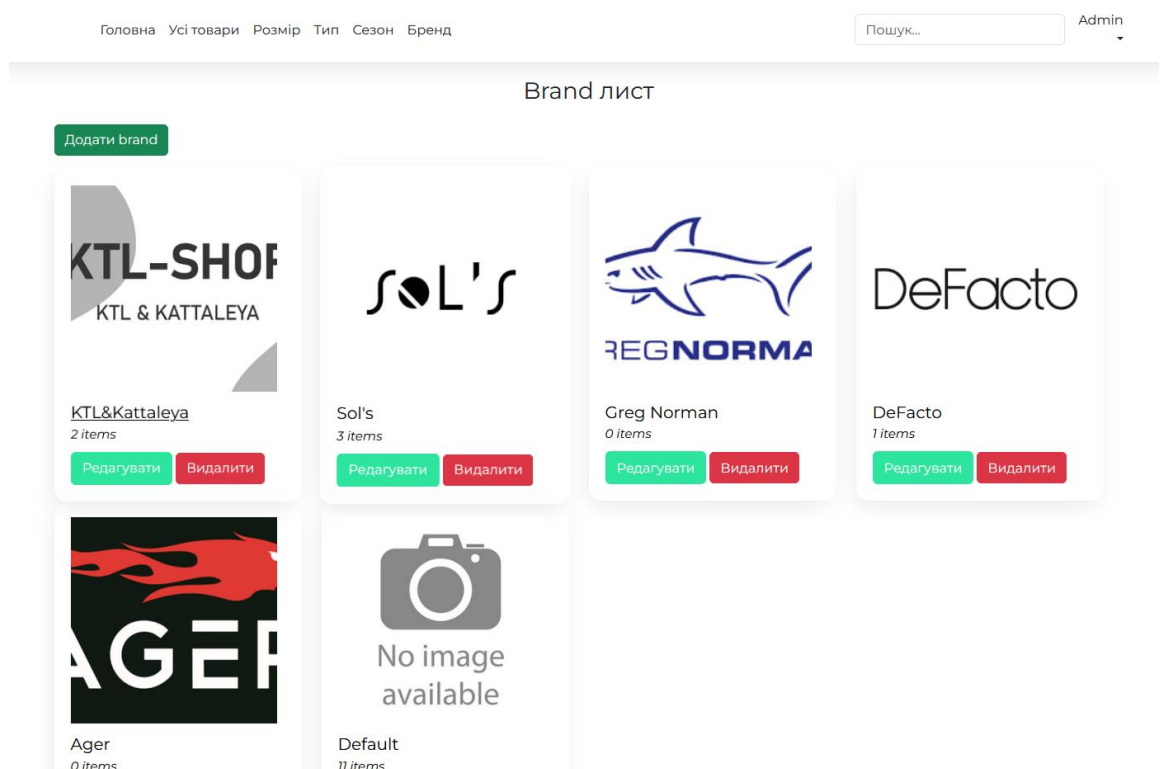


Рисунок 3.4 Сторінка брендів.

Обираємо компанію Lindo та переглянемо які товари є в наявності від цього виробника (рис 3.5).

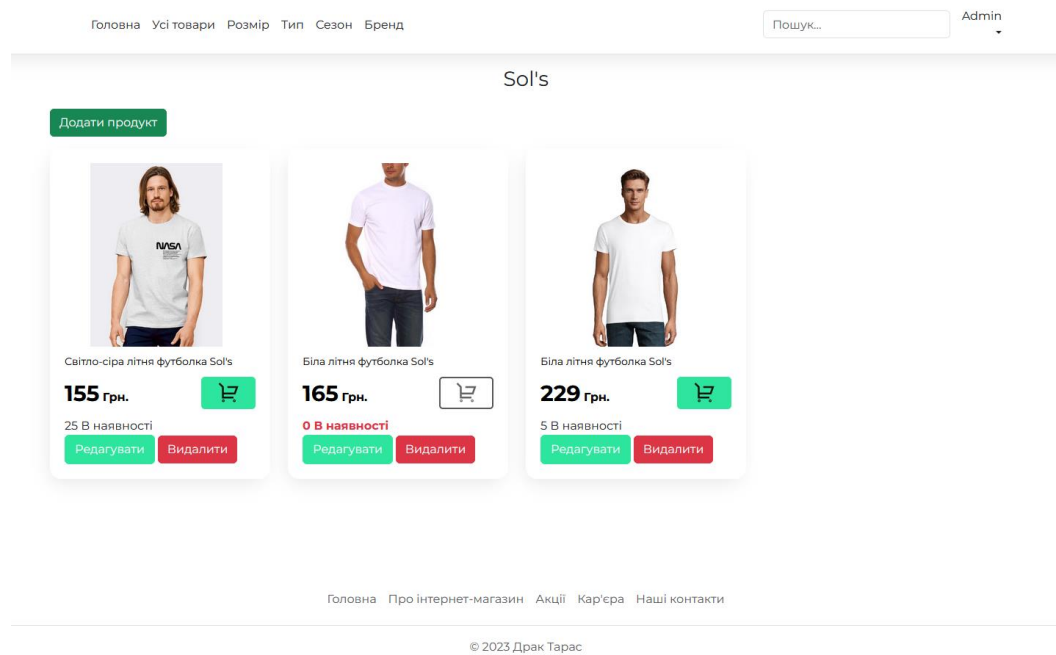


Рисунок 3.5 Сторінка товарів від виробника “Sol`s”.

Далі при перегляді товару ми можемо помітити категорії до яких належать ці чоботи, способи як можна доставити до вас товар, типи оплати, короткий та повний описи (рис 3.6).

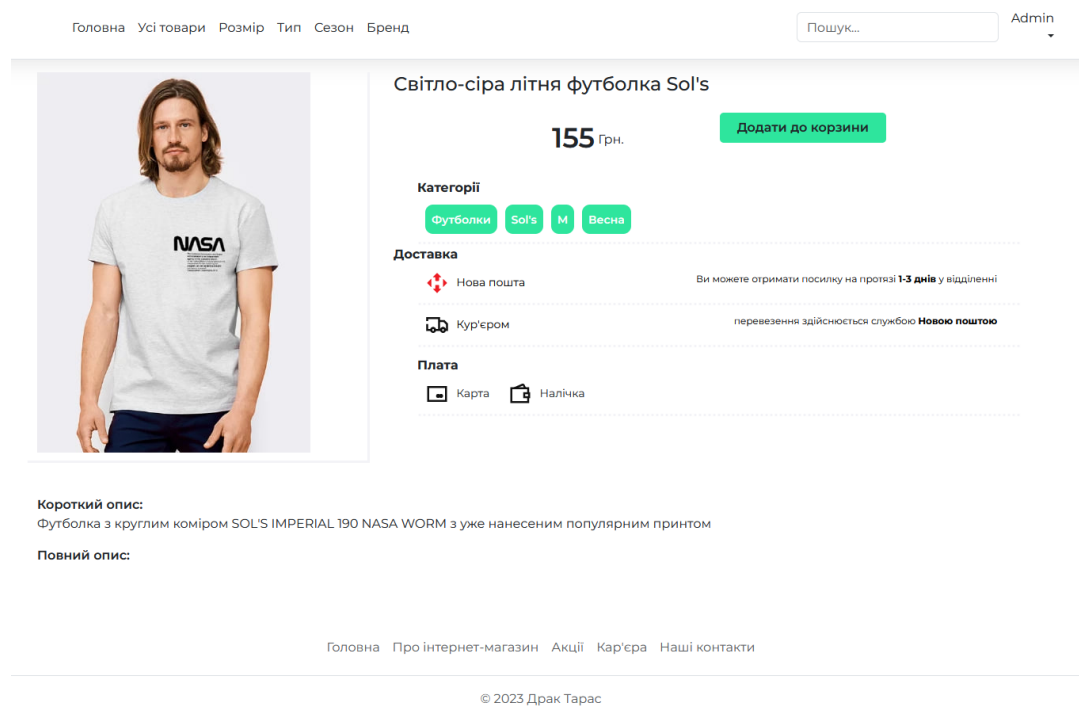
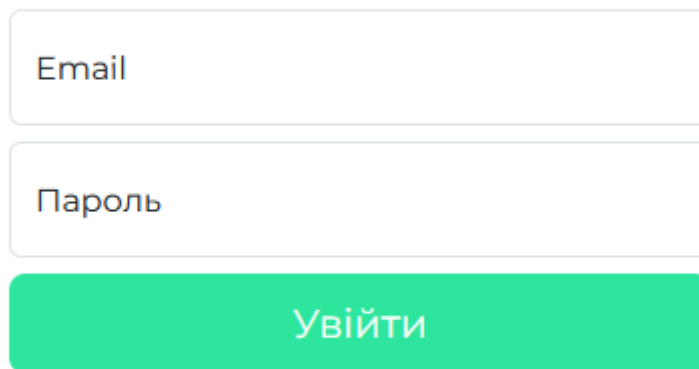


Рисунок 3.6 Інформація про товар “Футболка Sol`s”.

В цьому сайті незареєстровані користувачі не можуть здійснювати покупки, після натискання на кнопку "Додати до кошика" вас автоматично перенаправляє на сторінку авторизації (рис 3.7)

Увійти до акаунту



Email

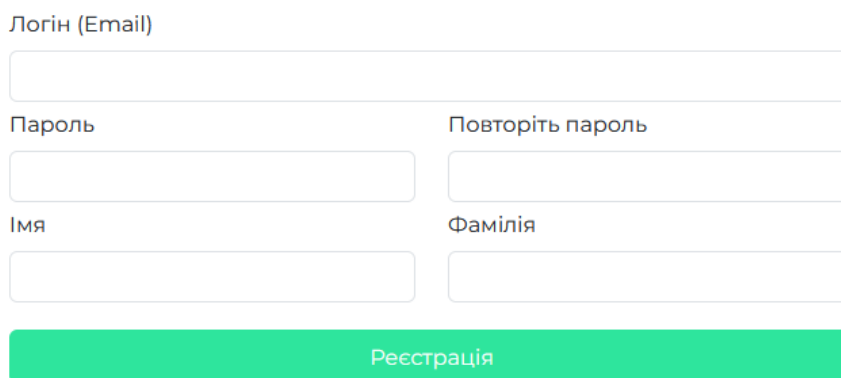
Пароль

Увійти

Рисунок 3.7 Спроба придбати товар не авторизованому користувачеві.

Оскільки в нас нема облікового запису тому перейдемо на вкладку реєстрації (рис 3.8). Після чого ми введемо дані для нового користувача.

Реєстрація нового користувача



Логін (Email)

Пароль

Повторіть пароль

Ім'я

Фамілія

Реєстрація

Рисунок 3.8 Реєстрація користувача.

При натиску якщо ви вводите вже існуючий mail тоді вас про це попередять та попросять зайти до свого акаунту тому зареєструємо користувача за іншим Email (рис 3.9.1). Та після вводу коректних даних ми отримуємо те що реєстрація успішна (рис 3.9.2).

Реєстрація нового користувача

Логін (Email)

Цей @mail вже зареєстрований Війти?

Пароль

Повторіть пароль

Імя

Фамілія

Реєстрація

Рисунок 3.9.1. Помилка при реєстрації.

Успішна реєстрація!

Залогінйтесь щоб продовжити

Рисунок 3.9.2. успішна реєстрація

Вводимо данні для входу до акаунту (рис 3.10.1).

Увійти до акаунту

Невірний логін чи пароль

Email
new1@gmail.com

Пароль



Увійти

Рисунок 3.10.1 Вхід до акаунту.

Тепер у правому кутку зверху освітлені ім'я користувача (рис 3.10.2) та можна проводити купівлю товарів.

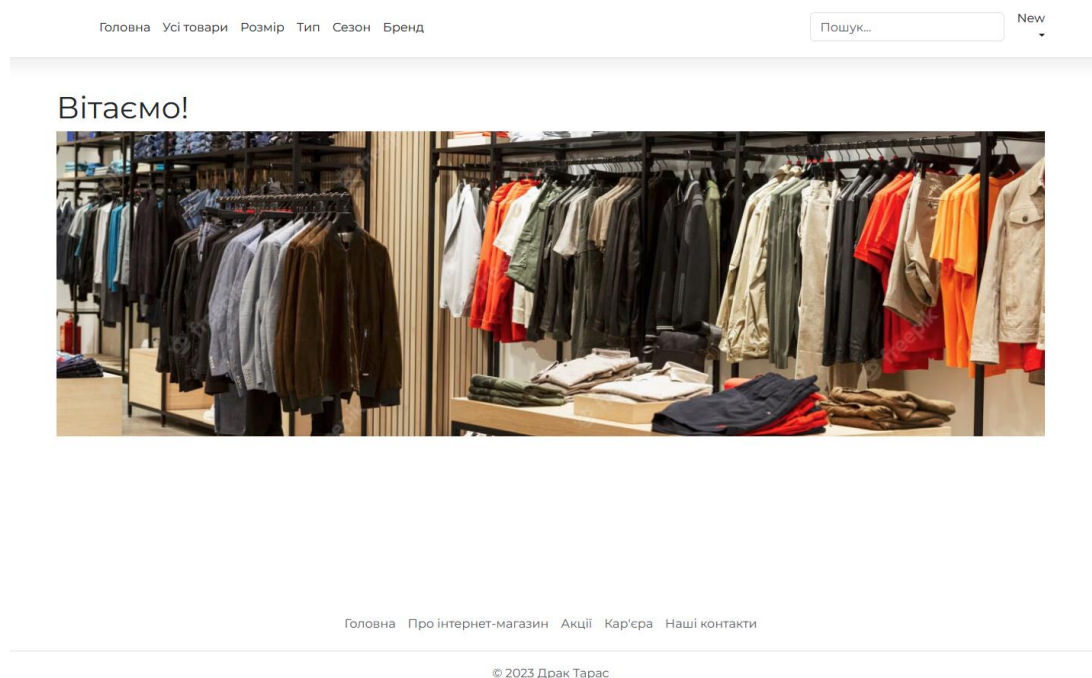


Рисунок 3.10.2 Сторінка авторизованого користувача.

Натиснувши на вкладку "Додати до кошика" (рис. 3.11), деякі товари будуть додані до вашої корзини. Зверху справа видно кількість доданих товарів, а також загальну вартість цих товарів.

Головна

Усі товари

Розмір

Тип

Сезон

Бренд

Пошук...

New

Корзина

	№	Фото	Ім'я	Ціна одиниці	Кількість	Ціна всіх	Наявність
X	1		Темно-синя осіння куртка KTL&Kattaleya	2540 грн.	1	2540 грн.	В наявності
X	2		Сорочка Ager	165 грн.	1	165 грн.	В наявності
Загальна ціна						2705 грн.	Замовлення

Головна

Про інтернет-магазин

Акції

Кар'єра

Наші контакти

© 2023 Драк Тарас

Рисунок 3.11 Кошик.

Після того, як ми вже додали товари до кошика, ми маємо можливість здійснити замовлення. Для цього необхідно ввести своє ім'я, номер телефону, місто та поштовий індекс (рис. 3.12).

Замовлення

Ім'я

New User

Номер телефону

+380984533312

Місто

Житомир

Поштовий номер

10001

Замовлення

Рисунок 3.12. Сторінка замовлення.

Після цього ми увійдемо до панелі адміністратора, де зможемо переглянути список замовлень. В цьому списку ми маємо можливість підтвердити замовлення від покупців (рис. 3.13).

Замовлення

№	Дата	Ім'я	Телефон	Місто	Поштовий номер	Товар	Статус
1	2023-08-26 19:29:50	New User	+380984533312	Житомир	10001	1 x Темно-синя осінн... 1 x Сорочка Ager...	In processing

Рисунок 3.13 Сторінка замовлення в очікуванні.

Після цього ми перейдемо на вкладку з усіма товарами, де зможемо помітити різницю в порівнянні зі сторінкою для звичайних користувачів та гостей. Тут доступні функції редагування, видалення та додавання товарів (рис. 3.14).

[Головна](#)
[Усі товари](#)
[Розмір](#)
[Тип](#)
[Сезон](#)
[Бренд](#)

Admin

Список товарів

Додати товар



Темно-синя осіння куртка KTL&Kattaleya

2540 грн.

24 В наявності

Редагувати Видалити



Темно-синя зимня куртка KTL&Kattaleya

1250 грн.

5 В наявності

Редагувати Видалити



Світло-сіра літня футболка Sol's

155 грн.

25 В наявності

Редагувати Видалити



Джогери DeFacto

399 грн.

15 В наявності

Редагувати Видалити

Рисунок 3.14 Сторінка з усіма товарами(Адмін).

При додаванні товару ми маємо можливість вибрати його назву, тип, сезон, розмір, бренд, ціну, кількість, а також надати короткий та повний опис. Ми також можемо прикріпити фотографію товару та визначити, чи має бути цей товар показаний (рис. 3.15).

3.3 Тестування роботи веб додатку

Один з ключових етапів у процесі розробки програмного забезпечення - це процес тестування та виправлення помилок. Початковий етап перевірок включає процедуру авторизації на сторінці, де користувач повинен ввести свій логін та пароль (рис. 3.17). Усі поля, необхідні для авторизації або реєстрації, є обов'язковими, і система виводитиме попередження у випадку їх відсутності.

Реєстрація нового користувача

Email

Login не заповнено

Пароль

Password не заповнено

Повторіть пароль

Password2 не заповнено

Ім'я

Firstname не заповнено

Фамілія

Lastname не заповнено

Зареєструватися

Рисунок 3.17 Помилка при надсиланні пустої форми при реєстрації

Якщо користувач намагається ввести існуючу електронну адресу під час реєстрації, система автоматично виявить цю помилку і запропонує користувачеві увійти в свій вже існуючий обліковий запис. У випадку, якщо користувач не введе електронну адресу або введе некоректні дані, які не містять поштовий домен (наприклад, "user" без додавання "@gmail.com"), система не дозволить продовжувати реєстрацію. Це забезпечує необхідність введення дійсної електронної адреси для успішного завершення процесу реєстрації. Крім того, при некоректному введенні повторного пароля також буде відображатися відповідне повідомлення про помилку (рис. 3.18).

Реєстрація нового користувача

Email

Цей @mail вже зареєстрований Війти?

Пароль

Повторіть пароль

Пароль не сходиться

Ім'я

Фамілія

Зареєструватися

Рисунок 3.18 Помилка паролів та email

Під час процесу реєстрації, якщо користувач введе некоректний пароль або електронну адресу, система надсилатиме повідомлення про помилку. Це допомагає нагадати користувачеві, що необхідно ввести правильні дані для успішного завершення реєстрації (рис. 3.19).

Увійти до акаунту

Невірний логін чи пароль

Email

Пароль

Увійти

Рисунок 3.19 Помилка невірного логіну чи пароля

Перевірка, як система реагує на введення адреси, яка фактично не існує, є важливим етапом тестування. Цей етап дозволяє визначити, як система обробляє некоректні або недоступні дані (рис. 3.20).

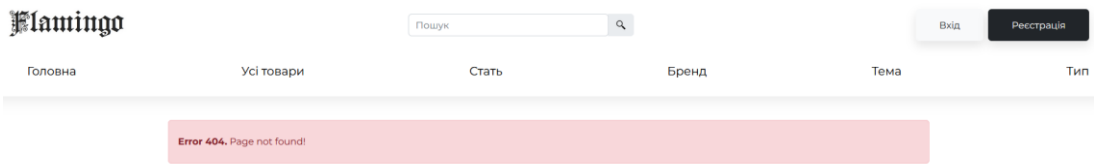


Рисунок 3.20 Помилка при відвідуванні неіснуючої сторінки

Висновки до третього розділу

У цьому розділі ми встановили мінімальні вимоги до користувачів, які бажають використовувати наш інтернет-магазин. Ці вимоги включають наявність стандартних компонентів, які, як правило, присутні на більшості сучасних операційних систем. Це спрощує роботу магазину та робить його доступним для широкого кола користувачів.

В даному розділі також була надана схема розгортання розробленої системи. Ця схема докладно вказує, як різні компоненти магазину взаємодіють між собою. Вона допомагає краще зрозуміти структуру системи та допомагає правильно розмістити компоненти для оптимальної роботи магазину.

Після встановлення та розгортання системи були проведені необхідні тести додатку. Ці тести успішно підтвердили коректну роботу магазину та відповідність його функціоналу вимогам. Це гарантує стабільну та ефективну роботу магазину, що задовольняє потреби користувачів.

ВИСНОВКИ

У ході виконання моєї бакалаврської випускної роботи був успішно створений та впроваджений інтернет-магазин, спеціалізований на продажі чоловічого одягу. Процес розробки включав важливі кроки та етапи, спрямовані на забезпечення ефективної роботи системи. У цьому розділі наведено основні висновки та досягнення цієї роботи.

Початковий етап включав у себе зрозуміння та формалізацію постановки завдання. Ми чітко сформулювали цілі та завдання проекту, а також провели докладний аналіз подібних програмних продуктів, розглянувши їхні особливості та відповідність вимогам підприємства. Особливу увагу було приділено вибору архітектури для розробки веб-додатку.

У другому розділі ми визначили різні варіанти використання, ідентифікували різні сценарії та функціональні можливості, які повинні були бути доступні веб-додатку. Здійснюючи цей аналіз, ми створили основу для подальшої розробки.

Третій розділ розглядав створення інтерфейсу та процедур роботи інтернет-магазину. Ми звернули особливу увагу на структуру інтерфейсу для забезпечення зручного та зрозумілого користувацького досвіду під час навігації по сайту.

Результатом роботи став функціональний, зручний та надійний веб-додаток, який повністю відповідає потребам користувачів у дистанційному навчанні. Майбутні перспективи розвитку включають розширення функціоналу, поліпшення інтерфейсу, розробку мобільного додатку, вдосконалення аналітики та звітності, а також інтеграцію з іншими системами. Це дозволить покращити функціональність та зручність використання програмного продукту.

У підсумку, випускна робота була успішною у розробці інтернет-магазину, який може задовольнити потреби користувачів та мати перспективи для подальшого розвитку та вдосконалення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розділяй та володарюй: що таке патерни MVC і MVP, та як їх використовувати [Електронний ресурс] – Режим доступу до ресурсу: <https://highload.today/uk/blogs/shho-take-mvc-ta-mvp-patterni/>
2. Офіційна документація HTTPS [Електронний ресурс] – Режим доступу до ресурсу: <http://surl.li/wxpq>
3. Посібник із MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mysql.com/>
4. Документація по використанню PHPMyAdmin [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.phpmyadmin.net/uk/latest/>
5. Встановлення OpenServer [Електронний ресурс] – 2020 – Режим доступу до ресурсу: <https://ospanel.io/docs/>
6. Побудова діаграми використання [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modelinglanguage/what-is-use-case-diagram/>
7. Побудова діаграми класів [Електронний ресурс] – Режим доступу до ресурсу: https://flexberry.github.io/en/gpg_class-diagram.html
8. StarUML documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.staruml.io/>
9. Побудова діаграми діяльності [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modelinglanguage/what-is-activity-diagram/>
10. Побудова діаграми розгортання [Електронний ресурс] – Режим доступу до ресурсу: <https://creately.com/guides/deployment-diagram-tutorial/>
11. Інтернет магазин одяжі “Kasta” [Електронний ресурс] – Режим доступу до ресурсу: <https://kasta.ua/uk/?main=men>
12. Інтернет магазин чоловічої одяжі “Next” [Електронний ресурс] – Режим доступу до ресурсу: <https://www.next.ua/uk>
13. Інтернет магазин брендової одяжі “le Boutique” [Електронний ресурс] – Режим доступу до ресурсу: <https://leboutique.com/uk/men>

					ІПЗ.КР.Б – 121 – 23 – ІЗ	
						54

14. “HTML Documentation”, MDN, 1998-2023. [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/HTML>
15. “CSS Documentation”, mozilla.org, 1998–2023. [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>
16. “PHP Documentation”, The PHP Group, 2001-2023. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.php.net/docs.php>
17. “IDE PHPStorm”, JetBrains s.r.o., 2000-2023. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.jetbrains.com/phpstorm/>
18. PSR-1: Basic Coding Standard, PHP Framework Interop Group, 2023, [Електронний ресурс] – Режим доступу до ресурсу: <https://www.php-fig.org/psr/psr-1/>
19. Офіційна документація Swiper [Електронний ресурс] – Режим доступу до ресурсу: <https://swiperjs.com/get-started>
20. Офіційна документація FancyBox [Електронний ресурс] – Режим доступу до ресурсу: <https://fancyapps.com/fancybox/>

ДОДАТКИ

					ПЗ.КР.Б – 121 – 23 – ПЗ	
						56

ProductController.php

```
<?php

namespace controllers;

use core\Controller;
use core\Core;
use models\DataTable;
use models\Product;
use models\User;
use utils\Photo;

class ProductController extends Controller
{
    public function indexAction() {
        $searchString = $_GET['search'];
        if(isset($_GET['page']))
            $page = intval($_GET['page']);
        else
            $page = 1;
        $countPerPage = 12;
        $rows = Product::search($searchString, User::isAdmin());
        $products = Product::getOnePage($rows, $page, $countPerPage);
        $totalNumber = count($rows);
        if(!isset($products))
            return $this->render('views/main/exception-no-content.php');
        if(($page - 1) * $countPerPage > $totalNumber || $page <= 0)
            return $this->error(404);
        return $this->render(null, [
            'products' => $products,
            'pagination' => [
                'page' => $page,
```

```

        'count' => $countPerPage,
        'totalNumber' => $totalNumber
    ],
    'searchString' => $searchString
]);
return $this->render();
}

public function getInputErrors() {
    $errors = [];
    $_POST['name'] = trim($_POST['name']);
    if (empty($_POST['name']))
        $errors['name'] = 'Не вказано назву продукту';
    if ($_POST['price'] <= 0)
        $errors['price'] = 'Неправильно вказана ціна';
    if ($_POST['count'] < 0)
        $errors['count'] = 'Неправильно вказана кількість продуктів';
    return $errors;
}

public function addAction($params) {
    $moduleName = $params[0];
    $module_id = intval($params[1]);
    if (empty($module_id))
        $module_id = null;

    $sizesList = DataTable::getSortedItems('size');
    $typesList = DataTable::getSortedItems('type');
    $seasonsList = DataTable::getSortedItems('season');
    $brandsList = DataTable::getSortedItems('brand');
    if (Core::getInstance()->requestMethod === 'POST') {
        $errors = $this->getInputErrors();

        if (empty($errors)) {
            $fileName = Photo::loadPhoto('product', $_FILES['file']['tmp_name']);
            Product::add($_POST, $fileName);
            $this->redirect('/product');
        }
    }
}

```

```

    } else {
        $model = $_POST;
        return $this->render(null, [
            'errors' => $errors,
            'model' => $model,
            'sizes' => $sizesList,
            'types' => $typesList,
            'seasons' => $seasonsList,
            'brands' => $brandsList,
            $moduleName.'_id' => $module_id
        ]);
    }
}

return $this->render(null, [
    'sizes' => $sizesList,
    'types' => $typesList,
    'seasons' => $seasonsList,
    'brands' => $brandsList,
    $moduleName.'_id' => $module_id
]);
}

public function editAction($params) {
    $id = $params[0];
    if(!User::isAdmin())
        return $this->error(403);
    if($id > 0) {
        $row = Product::getById($id);
        $sizesList = DataTable::getSortedItems('size');
        $typesList = DataTable::getSortedItems('type');
        $seasonsList = DataTable::getSortedItems('season');
        $brandsList = DataTable::getSortedItems('brand');
        if(Core::getInstance()->requestMethod === 'POST') {
            $errors = [];
            $_POST['name'] = trim($_POST['name']);
            $errors = $this->getInputErrors();
        }
    }
}

```

```

        if (empty($errors)) {
            Product::update($id, $_POST);
            if(!empty($_FILES['file']['tmp_name']))
                Photo::changePhoto('product', $id, $_FILES['file']['tmp_name']);
            $this->redirect("/product/view/{ $id }");
        } else {
            $model = $_POST;
            return $this->render(null, [
                'errors' => $errors,
                'model' => $model,
                'row' => $row,
                'sizes' => $sizesList,
                'types' => $typesList,
                'seasons' => $seasonsList,
                'brands' => $brandsList
            ]);
        }
    }

    return $this->render(null, [
        'row' => $row,
        'sizes' => $sizesList,
        'types' => $typesList,
        'seasons' => $seasonsList,
        'brands' => $brandsList
    ]);
} else
    return $this->error(403);
}

public function deleteAction($params) {
    $id = intval($params[0]);
    if(!User::isAdmin())
        return $this->error(403);
    if($id > 0) {
        Photo::deletePhoto('product', $id);
    }
}

```

```

        Product::delete($id);
        $this->redirect('/product/index');
    } else
        return $this->error(403);
    }

    public function viewAction($params) {
        $id = intval($params[0]);
        $product = Product::getById($id);
        return $this->render(null, [
            'product' => $product
        ]);
    }

    public function searchAction($params) {
        $searchString = $params[0];
        $rows = Product::search($searchString);
    }
}

```

DataController.php

```

<?php
namespace controllers;
use \core\Controller;
use core\Core;
use models\DataTable;
use models\Product;
use models\User;
use utils\Photo;

class DataController extends Controller
{
    public static function indexAction($controller, $moduleName) {
        $rows = DataTable::getItems($moduleName);
        if(!isset($rows))
            return $controller->render('views/main/exception-no-content.php');
    }
}

```

```

        foreach ($rows as &$row)
            $row += array('products_count' => count(self::getProductsIn($moduleName,
$row['id'])));

        return $controller->render(null, [
            'moduleName' => $moduleName,
            'rows' => $rows
        ]);
    }

    public static function addAction($controller, $moduleName)
    {
        if (!User::isAdmin())
            return $controller->error(403);

        if (Core::getInstance()->requestMethod === 'POST') {
            $errors = [];
            $_POST['name'] = trim($_POST['name']);
            if (empty($_POST['name']))
                $errors['name'] = "The name of the {$moduleName} is not specified";
            if (empty($errors)) {
                $fileName = Photo::loadPhoto($moduleName, $_FILES['file']['tmp_name']);
                DataTable::addItem($moduleName, $_POST['name'], $fileName);
                return $controller->redirect("/{ $moduleName }/index");
            } else {
                $model = $_POST;
                return $controller->render("views/$moduleName/add.php", [
                    'moduleName' => $moduleName,
                    'errors' => $errors,
                    'model' => $model
                ]);
            }
        }

        return $controller->render(null, [
            'moduleName' => $moduleName
        ]);
    }
}

```

```

public static function deleteAction($controller, $moduleName, $id, $answer) {
    if(!User::isAdmin())
        return $controller->error(403);
    if($id > 0) {
        $row = DataTable::getById($moduleName, $id);
        if($answer) {
            $filePath = "files/{$moduleName}/{ $row['photo']}";
            if(is_file($filePath)) {
                unlink($filePath);
            }
            $products = Product::getProductsIn($moduleName, $id);
            foreach ($products as $product) {
                Product::setDefaultForNull($moduleName, $product['id']);
            }
            Photo::deletePhoto($moduleName, $id);
            DataTable::deleteItem($moduleName, $id);
            return $controller->redirect("/{ $moduleName}/index");
        }
        return $controller->render(null, [
            'moduleName' => $moduleName,
            'row' => $row
        ]);
    } else
        return $controller->error(403);
}

public static function editAction($controller, $moduleName, $id) {
    if(!User::isAdmin())
        return $controller->error(403);
    if($id > 0) {
        $row = DataTable::getById($moduleName, $id);
        if(Core::getInstance()->requestMethod === 'POST') {
            $errors = [];
            $_POST['name'] = trim($_POST['name']);
            if (empty($_POST['name']))

```

```

        $errors['name'] = "The name of the $moduleName is not specified";

    if (empty($errors)) {
        DataTable::updateItem($moduleName, $id, $_POST['name']);
        if(!empty($_FILES['file']['tmp_name']))
            Photo::changePhoto($moduleName, $id, $_FILES['file']['tmp_name']);
        return $controller->redirect("/{ $moduleName }/index");
    } else {
        $model = $_POST;
        return $controller->render(null, [
            'moduleName' => $moduleName,
            'errors' => $errors,
            'model' => $model,
            'row' => $row
        ]);
    }
}

return $controller->render(null, [
    'moduleName' => $moduleName,
    'row' => $row
]);
} else
    return $controller->error(403);
}

public static function viewAction($controller, $moduleName, $id) {
    $row = DataTable::getById($moduleName, $id);
    $products = self::getProductsIn($moduleName, $id);
    return $controller->render(null, [
        'moduleName' => $moduleName,
        'row' => $row,
        'products' => $products
    ]);
}

public static function getProductsIn($moduleName, $id) {

```

```

        $className = 'Product';
        $methodName = 'getProductsIn';
        $products = call_user_func("\\models\\".$className.'::'.$methodName, $moduleName,
        $id);

        return $products;
    }
}

```

UserController.php

```

<?php
namespace controllers;
use core\Controller;
use core\Core;
use models\Order;
use models\User;

class UserController extends Controller
{
    public function registerAction() {
        if(User::isUserAuthenticated())
            $this->redirect('/');
        if (Core::getInstance()->requestMethod === 'POST') {
            $errors = [];

            if(!filter_var($_POST['login'], FILTER_VALIDATE_EMAIL))
                $errors['login'] = 'Помилка при введені @mail';
            if(User::isLoginExists($_POST['login']))
                $errors['login'] = 'Цей @mail вже зареєстрований <a
href="/user/login">Війти?</a>';

            if($_POST['password'] != $_POST['повторити пароль']) {
                $errors['password'] = 'Пароль не сходиться';
            }

            foreach ($_POST as $key => $value) {
                if(trim($value) == null) {

```

					ІПЗ.КР.Б – 121 – 23 – ІЗ	
						65

```

        $errors[$key] = ucfirst($key).' не заповнено';
    }
}

if(count($errors) > 0) {
    $model = $_POST;
    return $this->render(null, [
        'errors' => $errors,
        'model' => $model
    ]);
} else {
    User::addUser($_POST['login'], $_POST['password'], $_POST['firstname'],
$_POST['lastname']);
    return $this->renderView('register-success');
}
} else
    return $this->render();
}

public function loginAction() {
    if(User::isUserAuthenticated())
        $this->redirect('/');
    if (Core::getInstance()->requestMethod === 'POST') {
        $user = User::getUserByLoginAndPassword($_POST['login'], $_POST['password']);
        $error = null;
        if(empty($user)) {
            $error = 'Невірний логін чи пароль';
        } else {
            User::authenticateUser($user);
            $this->redirect('/');
        }
    }
    return $this->render(null, [
        'error' => $error
    ]);
}

```

```

    }

    public function logoutAction() {
        if(!User::isUserAuthenticated())
            $this->redirect('/user/login');

        User::logoutUser();
        $this->redirect('/');
    }

    public function indexAction() {
        if(!User::isUserAuthenticated())
            $this->redirect('/user/login');

        if(User::isAdmin())
            $orders = Order::getOrdersByStatus('in processing');
        else
            $orders = Order::getUserOrdersById(User::getCurrentAuthenticatedUser()['id']);

        return $this->render(null, [
            'orders' => $orders
        ]);
    }

    public function approveAction($params) {
        if (!User::isAdmin())
            return $this->error(403);
        $orderId = intval($params[0]);
        Order::approve($orderId);
        $this->redirect("/user");
    }
}

```