МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ
Навчально-науковий інститут автоматики, кібернетики та обчислювальної техніки

"До захисту допущена"
Зав. кафедри комп'ютерних наук та прикладної математики
д.т.н., проф. Турбал Ю.В.
« » 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

Розробка інтернет-магазину косметики, біжутерії та аксесуарів
(назва теми роботи)

Виконав: Горбатюк Владислав Вікторович _____
(прізвище, ім'я, по батькові) (підпис)
студент групи ІПЗ-41

Керівник: к.т.н., доц. Климюк Ю. Є. _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рівне – 2023

Національний університет водного господарства та природокористування
(повне найменування вищого навчального закладу)

Навчально-науковий інститут автоматики, кібернетики та обчислювальної техніки

Кафедра комп'ютерних наук та прикладної математики
Освітньо-кваліфікаційний рівень **бакалавр**
Галузь знань 12 Інформаційні технології
(шифр і назва)
Спеціальність 121 Інженерія програмного забезпечення
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
_____
«_____»_____20___ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

_____
(прізвище, ім'я, по батькові)

1. Тема роботи *"Розробка інтернет-магазину косметики, біжутерії та аксесуарів"*
керівник роботи *Климюк Юрій Євгенійович*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада)
затверджені наказом по університету від "*19"квітня 2023* року С № 449

2. Термін подання роботи студентом *01.06.23*

3. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)*У першому розділі розглянуто важливість веб-сайтів та їх вплив на залучення нових клієнтів,досліджено засоби реалізації web-додатків. У другому розділі виставляються вимоги до проєкту та розглядається реалізація. У третьому розділі описано реалізований web-застосунок*

4. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Графічні зображення вигляду програмного продукту; мультимедійна презентація.

5. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ 1	*доцент Климюк Ю.Є.*	*01.04.23*	*17.04.23*
Розділ 2	*доцент Климюк Ю.Є.*	*18.04.23*	*20.05.23*
Розділ 3	*доцент Климюк Ю.Є.*	*21.05.23*	*27.05.23*

6. Дата видачі завдання *01.10.22*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Аналіз наукової літератури з обраною тематикою*	*02.10.22-03.01.23*	*виконав*
2	*Формулювання постановки завдання*	*04.01.23-12.03.23*	*виконав*
3	*Розробка інформаційної моделі завдання*	*13.03.23-17.04.23*	*виконав*
4	*Розробка алгоритму проектування рішення*	*18.04.23-25.04.23*	*виконав*
5	*Реалізація програмного коду*	*26.04.23-20.05.23*	*виконав*
6	*Проведення тестування системи*	*21.05.23-23.05.23*	*виконав*
7	*Оформлення звітної документації*	*24.05.23-27.05.23*	*виконав*

Студент _____ Горбатюк В.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Климюк Ю.Є.
(підпис) (прізвище та ініціали)

ЗМІСТ

РЕФЕРАТ

Кваліфікаційна робота: 52 с., 34 рисунка, 6 джерел.

Актуальність теми: Розробка інтернет-магазину косметики, біжутерії та аксесуарів є особливо актуальною в контексті сучасних тенденцій електронної комерції та зростання популярності онлайн-покупок. В умовах швидкого розвитку технологій та зростаючого впливу Інтернету, споживачі все частіше звертаються до онлайн-магазинів для придбання товарів у зручний та ефективний спосіб. Косметика, біжутерія та аксесуари є популярними товарами, а їх продажі в онлайн-сегменті ринку постійно зростають. Тому розробка функціонального, привабливого та швидкого веб-додатку для інтернет-магазину в цій сфері є актуальним завданням, спрямованим на задоволення потреб споживачів та підвищення конкурентоспроможності бізнесу.

Мета кваліфікаційної роботи: Метою роботи є створення функціонального веб-додатку інтернет-магазину косметики, біжутерії та аксесуарів з метою підвищення продажів, залучення клієнтів та забезпечення швидкого обміну інформацією між користувачем та системою. Важливим аспектом є забезпечення зручності та швидкості взаємодії, щоб користувачі мали можливість легко шукати та вибирати товари, здійснювати покупки та отримувати оперативну інформацію про замовлення.

Об'єкт дослідження: Веб-сайт інтернет-магазину косметики, біжутерії та аксесуарів.

Предмет дослідження: Методи та засоби проектування веб-додатків для створення швидкого та ефективного обміну інформацією між користувачем та системою, програмна реалізація інформаційної системи, функціонал користувацької та адміністративної частин веб-додатку.

Методи дослідження: Аналіз існуючих інтернет-магазинів для виявлення найкращих практик швидкого обміну інформацією, використання

HTML, CSS, JavaScript та PHP для розробки фронтенду та бекенду, використання Apache у якості веб-серверу, тестування розробленого веб-додатку з фокусом на швидкість та ефективність взаємодії.

Ключові слова: інтернет-магазин, база даних, веб-додаток, веб-розробка, косметика, біжутерія, аксесуари, проектування бази даних, HTML, CSS, JavaScript, PHP.

ПЕРЕЛІК ОСНОВНИХ УМОВНИХ ПОЗНАЧЕНЬ

БД - база даних. Це структурована збірка даних, яка зберігається та управляється в електронному форматі. Бази даних використовуються для зберігання і організації певних обсягів інформації.

PHP - мова програмування, яка використовується для створення серверної частини веб-додатків, що взаємодіють з базами даних.

JavaScript - мова програмування, яка використовується для програмування клієнтської частини веб-додатків, що взаємодіють з користувачем у браузері.

HTML - HyperText Markup Language (мова гіпертекстової розмітки) - це стандартна мова розмітки для створення та оформлення веб-сторінок.

Backend - програмно-апаратна частина сервісу, що відповідає за обробку даних та логіку. Backend зазвичай включає сервери, бази даних, застосунки та інші компоненти, що виконують обробку даних на серверному рівні.

Frontend - клієнтська сторона сервісу, що відповідає за взаємодію з користувачем у браузері. Frontend включає в себе розробку інтерфейсу користувача, валідацію даних та взаємодію з сервером для отримання та відправки даних.

ВСТУП

Онлайн-торгівля стає невід'ємною складовою сучасного бізнесу і щороку набуває все більшої актуальності. В Україні спостерігається постійне зростання обсягів продажів через Інтернет, а кількість електронних магазинів швидко збільшується. Це свідчить про ефективність та потенціал онлайн-торгівлі як засобу залучення клієнтів та збільшення обсягів продажів.

Проте, деякі підприємці та компанії ще не розуміють повністю переваги, які може надати створення власного веб-додатку. Вони можуть вважати, що традиційні методи реклами, такі як телебачення, радіо та зовнішня реклама, є достатніми для просування їхнього бізнесу. Однак, економічні зміни та зміна споживчої поведінки ставлять перед продавцями вимогу активніше розвивати онлайн-торгівлю.

Створення легкого веб-додатку є важливим аспектом, оскільки забезпечує більш швидке завантаження сторінок, зменшує використання ресурсів та підвищує загальну продуктивність. Використання мінімальної кількості або взагалі відсутність бібліотек та фреймворків допомагає підтримувати простоту, легкість та швидкодію веб-додатку, що сприяє зручності користування та ефективному обміну інформацією між системою та користувачами.

Метою цієї кваліфікаційної роботи є розробка невеликого, динамічного інтернет-магазину, спеціалізованого на продажу косметики, біжутерії та аксесуарів. Особливу увагу приділяється створенню веб-додатку, який буде привабливим та функціональним, а також не буде навантажений зайвими бібліотеками та фреймворками. Це дозволить досягти швидкої взаємодії та ефективного обміну інформацією з користувачами.

Для успішної реалізації проекту необхідно виконати ряд завдань:

1. Вибрати та налаштувати необхідні інструменти для реалізації веб-додатку. Врахувати вимоги до швидкості та ефективності, вибрати, по можливості, мінімальний набір бібліотек та фреймворків, що не навантажуватимуть додаток.
2. Визначити структуру даних та спроектувати веб-сайт інтернет-магазину. Розробити зручну навігацію, категорії товарів та можливості фільтрації, яка допоможе користувачам знаходити потрібні товари швидко та зручно.
3. Реалізувати веб-додаток, забезпечивши його функціональність та відповідність вимогам. Врахувати можливість реєстрації та авторизації користувачів, додавання товарів у кошик, оформлення замовлень, а також адміністративні функції для керування товарами та замовленнями.

Об'єктом дослідження є веб-сайт інтернет-магазину косметики, біжутерії та аксесуарів.

Предметом дослідження є методи та засоби проектування веб-додатків для створення швидкого та ефективного обміну інформацією між користувачем та системою, програмна реалізація інформаційної системи, функціонал користувацької та адміністративної частин веб-додатку.

У результаті виконання цієї кваліфікаційної роботи очікується створення інтернет-магазину, який допоможе залучити нових клієнтів, розширити ринкову присутність та збільшити обсяги продажів. Розроблений додаток може служити прикладом для інших бізнесів, які бажають розвиватися в галузі електронної комерції та ефективного використання Інтернет-ресурсів.

РОЗДІЛ 1
ТЕОРЕТИЧНА ЧАСТИНА

1.1. Аналіз останніх публікацій та досліджень

На сьогоднішній день існує значна кількість досліджень та публікацій, які стосуються інтернет-торгівлі та створення веб-додатків для продажу товарів і послуг. Дослідники зосереджуються на різних аспектах інтернет-торгівлі, включаючи її розвиток, переваги та виклики.

Один із напрямків досліджень полягає у вивченні розвитку інтернет-торгівлі в Україні. Згідно з дослідженнями, обсяги продажів через Інтернет постійно збільшуються, а кількість електронних магазинів постійно зростає. Це свідчить про успішний розвиток інтернет-торгівлі в країні.

Також проводяться дослідження, що порівнюють ефективність різних засобів реклами, включаючи телебачення, радіо, ЗМІ, банери та флаєри, з ефективністю веб-додатків. Дослідження показують, що веб-додатки дозволяють швидко та ефективно розповсюджувати інформацію серед широкої аудиторії, поліпшують імідж магазину та сприяють залученню клієнтів.

Економічна криза та зміна споживчої поведінки також є предметом досліджень. Вони показують, що зростають витрати на традиційні продажі через високі орендні ставки та витрати на персонал, а також змінюється споживча поведінка, і все більше людей шукають товари за більш привабливими цінами через Інтернет. Це спонукає багатьох підприємців до розвитку онлайн-торгівлі.

Однак, необхідно враховувати певні виклики при впровадженні інтернет-магазину. Наприклад, візуальна оцінка товару та емоційні аспекти покупки можуть бути обмежені у випадку онлайн-торгівлі. Проблеми також можуть виникнути з доставкою товарів за низькими цінами.

Останні дослідження підтверджують зростання популярності та успішність інтернет-магазинів, особливо великих містах. Кількість інтернет-магазинів постійно збільшується, і це вигідно як для покупців, так і для

підприємств. Інтернет-магазини пропонують широкий асортимент товарів та послуг, забезпечують зручність та ефективність покупок, а також дозволяють розширити ринкову присутність.

Отже, на підставі проведеного огляду можна зробити висновок, що створення інтернет-магазину з продажу косметики, біжутерії та аксесуарів є актуальним та перспективним напрямком. Зростання аудиторії Інтернету, зростання продажів через Інтернет та популярність онлайн-шопінгу створюють сприятливі умови для розвитку такого бізнесу.

1.2. Дослідження засобів реалізації web-додатку

На сучасному етапі розвитку веб-розробки перед фахівцями стоїть багато різноманітних завдань. Вони можуть створювати як інтерактивні сайти для розваг, так і серйозні бізнес-проекти, які вимагають високої надійності та захищеності від несанкціонованого доступу. Для успішної реалізації цих завдань важливо правильно підібрати мови програмування, фреймворки або системи управління контентом, які найкраще відповідатимуть поставленим вимогам. Вибір конкретної мови програмування залежить від рівня знань програміста та його можливостей у реалізації проекту.

Мови веб-програмування можна розділити на клієнтські та серверні.

Клієнтські мови використовуються для написання програм, які працюють у веб-браузерах. Однією з найпопулярніших мов клієнтської веб-розробки є JavaScript. Вона використовується для управління динамічними елементами веб-сторінок, забезпечує зміну вмісту та вигляду сторінки без перезавантаження. Разом з JavaScript широко використовуються мови розмітки HTML та CSS, які визначають структуру та вигляд веб-сторінок відповідно.

У сфері серверної веб-розробки широко використовується мова програмування SQL, яка дозволяє отримувати дані з баз даних та працювати з великими та складними наборами інформації. Крім того, Python є іншою

популярною мовою для серверної веб-розробки, відомою своєю простотою вивчення та широким спектром застосувань. Java також є однією з найбільш популярних мов програмування, спрямованою на об'єктно-орієнтовану розробку і використовується на різних платформах.

Крім того, є інші мови програмування, такі як C#, PHP та інші, які також знайшли своє застосування в веб-розробці. PHP є однією з найпопулярніших мов для серверної розробки веб-додатків. Вона має безкоштовну ліцензію, простий синтаксис та велике співтовариство розробників. PHP є динамічною мовою програмування, яка дозволяє швидко створювати динамічні веб-сторінки.

Кожна мова програмування має свої переваги та особливості, і вибір конкретної мови залежить від вимог проекту, володіння програмістом та контексту завдання. Розробники веб-програм можуть поєднувати різні мови та інструменти для досягнення найкращих результатів.

1.2.1 Мова розмітки – HTML

HTML (Hypertext Markup Language) - це мова розмітки, що використовується для створення структури та вигляду веб-сторінок. HTML визначає різні елементи (теги), які вказують браузеру, як відображати вміст сторінки.

Кожен HTML-документ складається з набору тегів, які вкладаються один в одного і визначають структуру сторінки. Наприклад, тег **<html>** визначає початок і кінець HTML-документа, **<head>** містить метаінформацію про сторінку, а **<body>** містить вміст, який буде відображено у веб-браузері.

HTML-теги також використовуються для форматування тексту, вставки зображень, створення посилань, таблиць, списків і багато іншого. Крім того, HTML підтримує можливість вбудовувати скрипти JavaScript для додавання інтерактивності до сторінки.

Веб-браузери, такі як Microsoft Internet Explorer, Mozilla Firefox, Google Chrome, Safari та інші, інтерпретують HTML-код і відображають сторінку на екрані користувача.

HTML є основою для розробки веб-сайтів і веб-додатків. Вона є стандартом для створення веб-контенту і використовується всіма веб-розробниками для створення доступних інформаційних ресурсів у Інтернеті.

1.2.2. Мова стилів CSS

CSS (Cascading Style Sheets) - це мова стилів, що використовується для опису вигляду і форматування елементів на веб-сторінках. За допомогою CSS можна задати кольори, шрифти, розміри, відступи, межі, фонові зображення і багато іншого.

Основна мета CSS полягає у відокремленні презентації (стилів) від вмісту (структури) веб-сторінки. Це дозволяє зберігати код сторінки більш чистим і організованим, спрощує зміни вигляду сторінки та полегшує її підтримку і розширення. Застосування CSS дозволяє змінювати стиль всіх елементів одночасно, використовувати одні й ті ж стилі на кількох сторінках і забезпечує більшу контрольованість над виглядом сторінки.

CSS використовується разом з HTML-кодом, де стилі визначаються за допомогою селекторів і оголошень властивостей. Селектори вказують, які елементи на сторінці будуть застосовуватися стилі, а оголошення властивостей визначають, як саме ці елементи будуть відображені.

CSS також використовує концепцію каскадування, де стилі можуть бути визначені на різних рівнях і мають пріоритети в залежності від специфічності селекторів і порядку їх визначення. Це дозволяє точно контролювати, які стилі будуть застосовуватися до конкретних елементів на сторінці.

CSS є стандартом, який підтримується всіма сучасними веб-браузерами і використовується веб-розробниками по всьому світу. Він дозволяє

створювати привабливі, сучасні та респонсивні веб-дизайни, що працюють на різних пристроях і розмірах екранів.

1.2.3. Мова програмування JavaScript

JavaScript є однією з найпопулярніших мов програмування, особливо в контексті веб-розробки. Основні переваги використання JavaScript на стороні клієнта включають:

1. Взаємодія з користувачем: JavaScript дозволяє створювати веб-сторінки, які реагують на дії користувача. Ви можете обробляти події, такі як кліки, наведення курсора миші, введення даних у форми тощо. Це дає можливість створювати більш інтерактивні та зручні для користувача веб-додатки.
2. Динамічність сторінок: JavaScript дозволяє динамічно змінювати вміст і структуру веб-сторінок після їх завантаження. Ви можете маніпулювати елементами HTML, змінювати їх властивості, додавати або видаляти елементи зі сторінки. Це дозволяє створювати живі, змінні відповіді на дії користувача.
3. Валідація форм: JavaScript може використовуватися для перевірки правильності введених даних у формах перед їх відправкою на сервер. Ви можете перевіряти, чи поля заповнені коректно, вимагати певні типи даних або виконувати інші перевірки, щоб забезпечити валідність даних перед їх обробкою.
4. Маніпуляція з DOM: JavaScript надає доступ до об'єктів Document Object Model (DOM), які представляють структуру і вміст веб-сторінки. Ви можете змінювати вміст елементів, створювати нові елементи, змінювати стилі, додавати анімацію та багато іншого. Це дозволяє повністю контролювати вигляд і поведінку сторінки.
5. Робота з кукісами та локальним сховищем: JavaScript дозволяє зберігати дані на боці клієнта за допомогою куків (cookies) або локального сховища (local storage). Ви можете зберігати стан

користувача, налаштування, тимчасові дані та іншу інформацію без необхідності постійної взаємодії з сервером.

Незважаючи на переваги, JavaScript також має деякі обмеження:

1. Безпека: Оскільки JavaScript код виконується на боці клієнта, він може бути доступний для перегляду та модифікації користувачами. Це може призводити до потенційних проблем з безпекою, таких як вразливості та зловмисний код.
2. Сумісність браузерів: JavaScript інтерпретується по-різному в різних браузерах, що може призводити до неузгодженості у функціональності та вигляді. Це вимагає від розробників виконання додаткових перевірок та забезпечення сумісності з різними браузерами.
3. Обмежені можливості: JavaScript не може виконувати деякі завдання, які потребують більшого рівня доступу до операційної системи або мережі. Наприклад, вона не може читати або записувати файли на локальному комп'ютері без дозволу користувача.
4. Залежність від браузера: JavaScript залежить від підтримки браузером інтерпретатора JavaScript. Якщо JavaScript вимкнений у браузері, код не виконається, що може призвести до проблем з функціональністю веб-сайту або додатку.

Хоча JavaScript має свої обмеження, вона є потужним і важливим інструментом для розробки динамічних інтерактивних веб-сторінок та додатків. Її переваги переважають недоліки, і вона продовжує бути однією з найпопулярніших мов програмування у веб-розробці.

1.2.4. Середовище Visual Studio Code

Visual Studio Code (VS Code) є потужним та розширюваним редактором вихідного коду, який надає зручні інструменти для розробки різних мов програмування. Особливості мови у VS Code залежать від встановлених

розширень і налаштувань редактора. Деякі загальні особливості мови, які можна знайти у VS Code, включають:

1. Підсвічування синтаксису: VS Code надає кольорове підсвічування синтаксису для різних мов програмування, що полегшує сприйняття коду і допомагає виділити ключові елементи.
2. Автодоповнення (IntelliSense): Редактор надає функцію автоматичного доповнення коду, що допомагає швидко завершувати рядки коду, функції та змінні. Це спрощує процес написання коду і допомагає уникнути помилок.
3. Лінтінг: VS Code підтримує лінтінг, що дозволяє виявляти помилки, недоліки та стилістичні проблеми в коді під час його написання. Він може надавати підказки та пропозиції щодо виправлення проблем.
4. Навігація по коду: Редактор дозволяє швидко переміщатися до визначення функцій, знаходити всі посилання на певні елементи коду та використовувати різні команди навігації для полегшення роботи з великими проектами.
5. Налагодження: VS Code надає можливість налагодження коду з можливістю встановлення точок зупинки, виконання крок за кроком, перегляду значень змінних та стеку викликів. Це допомагає знайти й виправити помилки та аналізувати виконання коду.
6. Рефакторинг: Редактор може надавати деякі функції рефакторингу, такі як перейменування змінних та функцій, витягування фрагментів коду в окремі функції або класи, знаходження та виправлення дублікатів коду. Це допомагає поліпшити структуру та ефективність коду.
7. Підтримка різних мов: VS Code дозволяє працювати з різними мовами програмування в одному проекті. Ви можете змінювати мову для окремих файлів або встановлювати розширення для підтримки

конкретних мов. Це забезпечує зручне переключення між мовами та роботу з різноманітними проектами.

Загалом, VS Code є потужним і гнучким редактором, який надає багато функціональних можливостей для розробки різних мов програмування. Його легка структура, розширюваність та широкий спектр підтримуваних мов роблять його чудовим вибором для розробки продукту.

1.3. Постановка задачі

Кваліфікаційний проєкт має на меті створення динамічного веб-додатка, який дозволить користувачам легко вибрати та замовити косметику, біжутерію або аксесуари без виходу з дому. Сайт складатиметься з взаємозалежних частин, кожна з яких матиме чітко визначені функції.

Основні можливості додатка будуть наступними:

1. Пошук та фільтрування товарів: користувачі зможуть здійснювати пошук товарів за всіма категоріями, конкретними категоріями, всіма підкатегоріями в межах певної категорії, конкретними підкатегоріями або за діапазоном цін.
2. Кошик: користувачі зможуть додавати товари до кошика, переглядати його зміст, змінювати кількість товарів, видаляти товари та робити остаточне оформлення замовлення.
3. Оформлення замовлення: користувачі зможуть заповнити необхідну інформацію для доставки товарів і підтвердити замовлення.
4. Авторизація та реєстрація: користувачі матимуть можливість створити обліковий запис, авторизуватись на сайті і використовувати особистий кабінет зі збереженою інформацією.
5. Адміністративна панель: буде розроблена окрема частина сайту для адміністратора, яка дозволить додавати, редагувати та видаляти наповнення інтернет-магазину.

Електронний магазин складатиметься з двох частин: адміністративної та користувацької. Адміністративна частина надасть можливість адміністратору маніпулювати категоріями, товарами та замовленнями.

Користувацький інтерфейс буде інтуїтивно зрозумілим, з легкою навігацією на всіх сторінках. Правильна структура інформації дозволить користувачам легко орієнтуватись на сайті та зручно переходити між розділами та сторінками.

Для успішної реалізації проекту необхідно виконати наступні завдання:

1. Вивчити процеси організації інтернет-торгівлі.
2. Обрати та налаштувати необхідні інструменти для реалізації додатка.
3. Визначити структуру даних та спроектувати веб-сайт.
4. Реалізувати додаток, враховуючи всі вимоги та функціональність.

Загальний успіх кваліфікаційної роботи залежатиме від точності виконання цих завдань та забезпечення зручного та функціонального користувацького досвіду.

РОЗДІЛ 2
РОЗРОБКА ТА РЕАЛІЗАЦІЯ ПРОЄКТУ

2.1. Постановка вимог.

Ідеєю інформаційної системи є створення динамічного веб-сайту зі зрозумілим та привабливим дизайном, який дозволить зацікавленим клієнтам замовляти косметику, біжутерію та аксесуари зручно та швидко, не виходячи з дому. Адміністратор сайту матиме можливість зручно переглядати замовлення, керувати ними та наповнювати товарами інтернет-магазин.

Враховуючи це, можна сформулювати постановку вимог наступним чином:

1. Анімована динамічна клієнтська сторінка: розробка привабливого дизайну з використанням анімацій, що привертають увагу клієнтів до товарів та послуг, пропонованих в інтернет-магазині.
2. Збір інформації про клієнта: створення форм та інших засобів, які дозволить клієнтам вказувати свої контактні дані та іншу необхідну інформацію для замовлення товарів. Ці дані будуть відправлятися в базу даних для подальшої обробки.
3. Динамічна сторінка для зручного адміністрування: створення інтерфейсу, який дозволить адміністратору легко переглядати та керувати замовленнями, змінювати їх статуси, взаємодіяти з клієнтами та оновлювати інформацію про доступні товари.
4. Перевірка запитів на легітимність за допомогою ключів доступу: використання ключів доступу або інших механізмів для перевірки легітимності запитів, що надходять в систему, щоб забезпечити безпеку даних клієнтів та захистити сайт від несанкціонованого доступу.
5. Використання тригерів та представлень у базі даних: використання функціоналу тригерів та представлень в базі даних для оптимізації роботи з даними, швидкого виконання запитів та забезпечення ефективного управління базою даних.

Така інформаційна система дозволить клієнтам зручно замовляти товари, а адміністратору легко керувати замовленнями та підтримувати актуальну інформацію на сайті.

2.2. Конфігурація та налаштування обладнання

Створення інформаційної системи передбачає важливий етап налаштування обладнання. В даному контексті ми використовуватимемо віртуальний сервер з операційною системою Linux Ubuntu 22 для бекенд-розробки.

2.2.1. Конфігурація Apache

На першому етапі, після авторизації в системі, важливо оновити її, використовуючи команди apt update і apt upgrade. Це дозволяє забезпечити стабільність системи, отримавши найновіші оновлення і усунувши можливі проблеми.

Рис.2.1. Процес оновлення системи Linux Ubuntu

Наступним кроком є встановлення веб-сервера Apache. Це можна зробити, використовуючи команду apt install apache2 apache2-utils. Apache є

одним з найпопулярніших веб-серверів, який забезпечує обробку веб-запитів та відправку веб-сторінок до кінцевого користувача.

Після встановлення сервера, важливо перевірити його налаштування, введенням IP-адреси сервера в браузер. Якщо на екрані з'явилася стандартна сторінка Apache, це свідчить про коректне встановлення і налаштування сервера.

Рис.2.2. Стандартна веб-сторінка Apache

Отже, ми успішно встановили та налаштували веб-сервер Apache на нашому віртуальному сервері з Linux Ubuntu 22, готовому до наступних кроків розробки інформаційної системи.

2.2.2. Налаштування бази даних (MariaDB)

Використання баз даних є критично важливим для нашого додатка, тому ми встановлюємо сервер баз даних MariaDB, використовуючи команду **sudo apt install -y mariadb-server mariadb-client**.

Ця команда встановлює сервер MariaDB і клієнтські компоненти, необхідні для роботи з базою даних. Налаштування безпеки MariaDB може бути здійснено за допомогою команди mysql_secure_installation.

2.2.3. Встановлення та налаштування PHP

Для налаштування PHP на нашому віртуальному сервері, який вже має встановлений веб-сервер Apache і базу даних MariaDB, використовується команда: sudo apt install -y php php-mysql libapache2-mod-php.

Ця команда встановить PHP, розширення для роботи з MySQL/MariaDB, а також модуль PHP для Apache. Після встановлення PHP, Apache повинен бути перезавантажений, щоб забезпечити коректну роботу всіх компонентів. Це можна зробити за допомогою команди: systemctl restart apache2.

2.2.4. Налаштування phpMyAdmin

Після завершення вищезазначених етапів, ми можемо приступити до встановлення phpMyAdmin, інструменту, який дозволить нам легко керувати нашою базою даних. Ми можемо встановити цей інструмент за допомогою команди: **apt-install -y phpmyadmin**.

Під час процесу встановлення, слід відповісти на кілька питань, які з'являються під час інсталяції phpMyAdmin. Після завершення встановлення, Apache має бути знову перезавантажений.

Нарешті, ми створимо базу даних для нашого інтернет-магазину і користувача, який матиме повний доступ до цієї бази даних в MariaDB:

CREATE DATABASE shop;

GRANT ALL ON shop.* TO 'admin'@'localhost' IDENTIFIED BY '11111112v' WITH GRANT OPTION; FLUSH PRIVILEGES;

З завершенням цих кроків, наш віртуальний сервер повністю налаштований і готовий до подальшої розробки інформаційної системи.

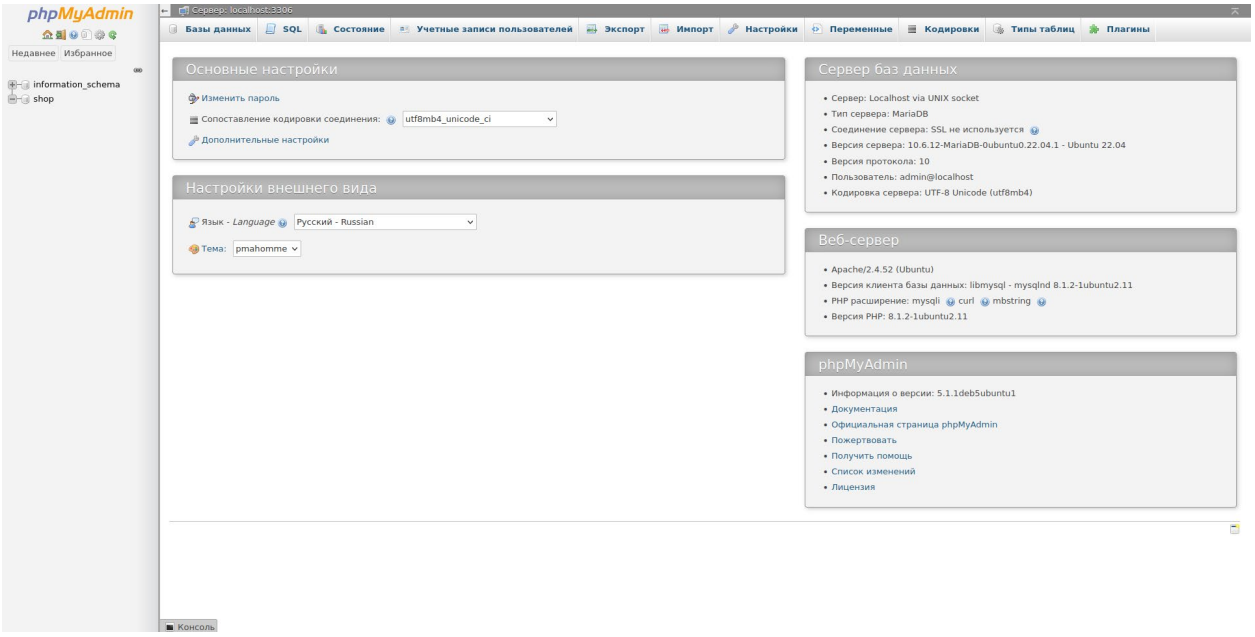

Рис. 2.3. Інтерфейс phpMyAdmin після авторизації

2.3. Проектування та створення структури бази даних

Створення структури бази даних для інформаційної системи включає проектування і визначення таблиць, які відображатимуть необхідні сутності та відношення між ними. Для нашого проекту інтернет-магазину буде створено шість основних таблиць:

admins — ця таблиця міститиме список адміністраторів системи з їх паролями та рівнем доступу.

categories — ця таблиця буде зберігати список доступних категорій та підкатегорій продуктів.

orders — ця таблиця буде використовуватися для зберігання інформації про замовлення, зроблені користувачами.

products — ця таблиця буде містити список всіх доступних товарів в інтернет-магазині.

product_list — ця таблиця зберігатиме інформацію про товари, які входять до конкретного замовлення.

users — кінцева таблиця, яка буде використовуватися для зберігання списку користувачів та їх даних.

Тепер розглянемо детальніше процес створення кожної з цих таблиць:

admins:

```
CREATE TABLE `admins` (
  `id` int(3) NOT NULL,
  `login` varchar(32) NOT NULL,
  `password` varchar(32) NOT NULL,
  `last_ip` varchar(32) NOT NULL,
  `access` int(32) DEFAULT 0,
  `token` varchar(32) DEFAULT NULL
);
```

В цій таблиці **id** слугує унікальним ідентифікатором адміністратора, **login** і **password** використовуються для входу в систему, **last_ip** зберігає IP-адресу останнього входу в систему, **access** визначає рівень доступу до системи, а **token** використовується для перевірки легітимності запитів.

categories:

```
CREATE TABLE `categories` (
  `id` int(4) NOT NULL,
  `category_name` varchar(32) NOT NULL,
  `pre_category` int(4) DEFAULT NULL
);
```

В цій таблиці **id** слугує унікальним ідентифікатором категорії, **category_name** відображає назву категорії, а **pre_category** використовується для зберігання інформації про батьківську категорію, якщо ця категорія є підкатегорією.

orders:

```
CREATE TABLE `orders` (
  `id` int(12) NOT NULL,
  `user` int(12) NOT NULL,
  `date` varchar(20) DEFAULT current_timestamp(),
  `status` int(1) DEFAULT 1,
  `full_address` varchar(200) NOT NULL,
  `full_name` varchar(60) NOT NULL,
  `phone` varchar(15) NOT NULL
);
```

В цій таблиці **id** є унікальним ідентифікатором замовлення, **user** зберігає ідентифікатор користувача, який зробив замовлення, **date** зберігає дату та час замовлення, **status** відображає поточний статус замовлення, а **full_address**, **full_name** та **phone** зберігають відповідну інформацію про отримувача замовлення.

products:

```
CREATE TABLE `products` (
  `id` int(6) NOT NULL,
  `name` varchar(100) NOT NULL,
```

`count` int(5) NOT NULL DEFAULT 0,

`category` int(4) NOT NULL,

`subcategory` int(4) DEFAULT NULL,

`price` int(6) NOT NULL,

`reserved` int(5) DEFAULT 0,

`image` varchar(45) NOT NULL,

`owner` int(3) NOT NULL

);

В цій таблиці **id** є унікальним ідентифікатором продукту, **name** відображає назву продукту, **count** показує наявну кількість продукту, **category** та **subcategory** використовуються для зберігання інформації про категорію та підкатегорію продукту відповідно, **price** зберігає ціну продукту, **reserved** зберігає кількість продукту, що в даний час є в бронюванні, **image** відображає назву файлу зображення продукту, а **owner** використовується для зберігання ідентифікатора адміністратора, який додав цей продукт.

product_list:

CREATE TABLE `product_list` (

`id` int(12) NOT NULL,

`order` int(12) NOT NULL,

`product` int(6) NOT NULL,

`count` int(4) NOT NULL

);

В цій таблиці **id** є унікальним ідентифікатором, **order** зберігає ідентифікатор замовлення, до якого прикріплений товар, **product** зберігає

ідентифікатор товару, який прикріплений до замовлення, а **count** показує кількість цього товару в замовленні.

users:

```
CREATE TABLE `users` (
  `id` int(12) NOT NULL,
  `username` varchar(16) NOT NULL,
  `password` varchar(32) NOT NULL,
  `last_ip` varchar(32) NOT NULL,
  `token` varchar(32) DEFAULT NULL,
  `full_name` varchar(60) NOT NULL,
  `full_address` varchar(200) NOT NULL,
  `phone` varchar(15) NOT NULL
);
```

В цій таблиці **id** є унікальним ідентифікатором користувача, **username** та **password** відповідно зберігають ім'я користувача та його пароль, **last_ip** використовується для зберігання IP-адреси останнього входу користувача, **token** використовується для зберігання ключа аутентифікації, **full_name**, **full_address** та **phone** зберігають відповідну інформацію про користувача.

Далі займемося тригерами. Тригери важливі для підтримки цілісності даних. Вони автоматично активуються після або перед виконанням певних операцій з базою даних. У даному контексті використовуються чотири тригери:

1. **Тригер видалення категорії.** Цей тригер автоматично видаляє всі продукти, що належать до категорії, коли ця категорія видаляється.

```
DELIMITER $$
CREATE TRIGGER `After_Delete_Category`
```

```
AFTER DELETE ON `categories`
FOR EACH ROW
DELETE FROM products WHERE products.subcategory= OLD.id OR
products.category= OLD.id ;
$$
DELIMITER ;
```

2. Тригер видалення продукту. Цей тригер автоматично видаляє продукт зі всіх замовлень, коли цей продукт видаляється.

```
DELIMITER $$
CREATE TRIGGER `After_Delete_Product`
AFTER DELETE ON `products`
FOR EACH ROW
DELETE FROM product_list WHERE product_list.product = OLD.id;
$$
DELIMITER ;
```

3. Тригер видалення замовлення. Цей тригер автоматично видаляє всі продукти зі списку прикріплених до замовлення продуктів, коли видаляється замовлення.

```
DELIMITER $$
CREATE TRIGGER `After_Delete_Order`
AFTER DELETE ON `orders`
FOR EACH ROW
DELETE FROM product_list WHERE product_list.order = OLD.id;
$$
DELIMITER ;
```

4. **Тригер видалення елементу замовлення.** Цей тригер автоматично видаляє всі замовлення користувача, коли цей користувач видаляється.

```
DELIMITER $$
CREATE TRIGGER `delete_product_list`
AFTER DELETE ON product_list
FOR EACH ROW
BEGIN
   DECLARE count_rows INT;
   SELECT COUNT(*) INTO count_rows FROM product_list WHERE `order` =
OLD.order;
   IF count_rows = 0 THEN
      DELETE FROM `orders` WHERE `id` = OLD.`order` LIMIT 1;
   END IF;
END $$
DELIMITER ;
```

Далі, для зручності, створимо пред'явлення з узагальненим списком замовлень, де поєднуються зв'язки між товарами і замовленням:

```
CREATE VIEW `order_list` AS
SELECT
   p.id,
   p.name,
   pl.order,
   pl.count,
   p.price
FROM
   product_list pl
JOIN
```

products p ON pl.product = p.id;

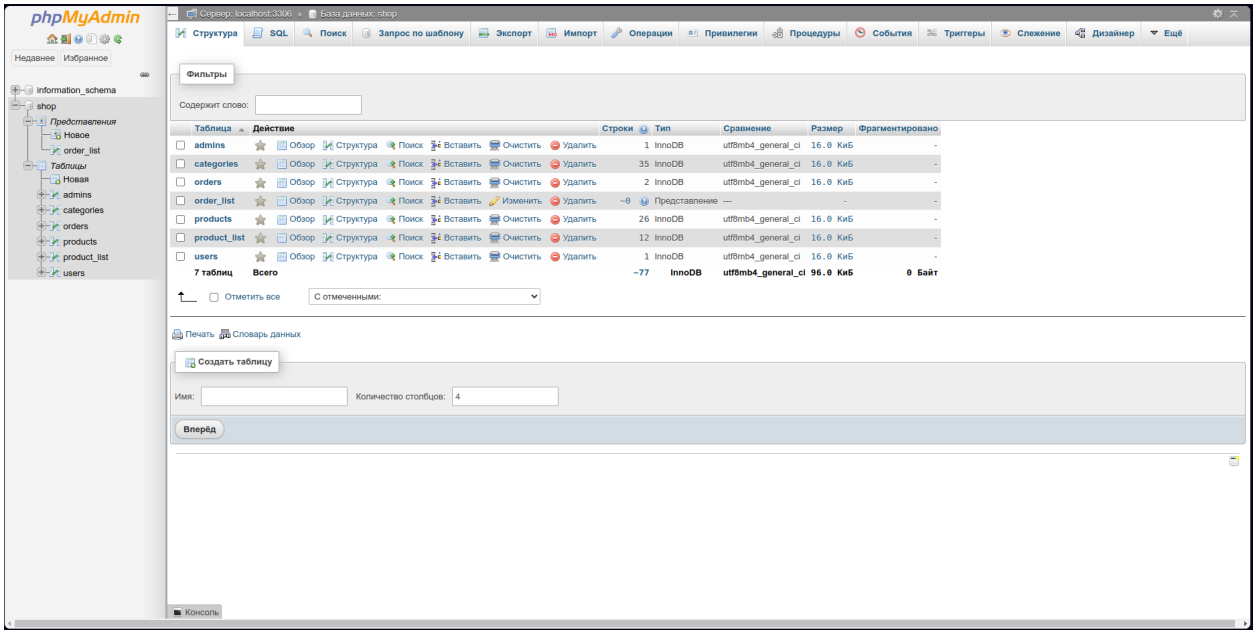

Рис 2.4. Кінцевий вигляд бази даних онлайн-магазину.

2.4. Створення бекенд частини

Система складається з двох основних компонентів: адміністративної панелі (admin.php) та клієнтської панелі (client.php). Ці два модулі мають декілька спільних частин, серед яких процедура підключення до бази даних та обробка відповідей від сервера.

Подробиці підключення до бази даних включають ініціалізацію параметрів, таких як ім'я сервера, ім'я користувача, пароль і назва бази даних. Нижче наведено відповідний код:

```
$servername = "localhost";

$username = "admin";

$password = "11111112v";

$dbname = "shop";

$connection = new mysqli($servername, $username, $password, $dbname);
```

Відповідь сервера готується заздалегідь як об'єкт PHP і має поле для зберігання помилок, яке ініціалізується порожнім рядком:

```
$response = (object)array();
```

$response->errors="";

Об'єкт **$response** відправляється у якості відповіді після обробки результатів на запити клієнтів у форматі JSON.

Заголовки встановлюються таким чином, щоб браузер клієнта відповідно обробляв інформацію:

header("Content-Type: application/json; charset=UTF-8");

*header("Access-Control-Allow-Origin: *");*

Дві додаткові функції, get_ip() та auth_key(), є важливими для системи. Перша дозволяє отримати IP-адресу користувача, а друга генерує серверний токен-ключ на основі клієнтського токен-ключа. Ці функції використовуються для валідації користувачів, забезпечуючи, що лише відповідні особи мають доступ до даних.

```
function get_ip()
{
        $value = '';
        if (!empty($_SERVER['HTTP_CLIENT_IP'])) {
                $value = $_SERVER['HTTP_CLIENT_IP'];
        } elseif (!empty($_SERVER['HTTP_X_FORWARDED_FOR'])) {
                $value = $_SERVER['HTTP_X_FORWARDED_FOR'];
        } elseif (!empty($_SERVER['REMOTE_ADDR'])) {
                $value = $_SERVER['REMOTE_ADDR'];
        }

        return $value;
}
function auth_key($auth){
        $ip = get_ip();
        return md5($auth.md5(md5($ip.$ip)));
}
```

Рис.2.5. Функції get_ip() та auth_key()

Це надасть можливість в подальшому перевіряти, чи дійсно користувач робить запит (для підтвердження користувач буде надсилати клієнтський token-ключ, на основі якого знову буде створюватися серверний та буде перевірятися на відповідність раніше записаному серверному ключу в базі

даних). Принцип використання зрозумілий на прикладі функції отримання прав доступу адміністратора в admin.php:

```
function getAdmin(){
        global $connection;
        if(!isset($_GET['token'])){
                return 0;
        }
        $key = auth_key($_GET['token']);
        $request = $connection->prepare("SELECT `access` FROM `admins` WHERE `token`=? LIMIT 1");
        $request->bind_param("s", $key);
        $access = 0;
        if($request->execute() === true)
        {
                $request->bind_result($access);
                $request->fetch();
        }
        $request->close();
        return $access;
}
```

Рис.2.6. Код функції, де використовується токен для підтвердження валідності адміністратора

В користувацькій частині також присутня подібна функція. Вона отримує id користувача.

Варто підмітити, що дана реалізація валідації користувача або адміністратора за допомогою токен-ключа дозволяє відслідковувати певні зміни. Оскільки під час хешування серверного ключа адміністратора використовується поточна IP адреса, то при зміні адреси без оновлення сторінки (наприклад, увімкнення vpn) заблокує будь-яку діяльність адміністратора, поки він не переавторизується, або не змінить IP адресу на попередню.

```
$request->execute();
$result =$request->get_result();
$request->close();
if(!$result->num_rows){
        $response->errors = "Невірний логін чи пароль!";
        die(json_encode($response));
}
$result = $result->fetch_assoc();
$auth = md5(md5($result['id'].$login.$password.$result['id'])); //генерація первинного ключа для клієнта
$auth_key = auth_key($auth); //генерація вторинного ключа на основі первинного для бд
$ip = get_ip();
$request = $connection->prepare("Update `admins` set `last_ip`=?, `token`=? WHERE `id`=? LIMIT 1"); // Запис в бд останнього IP та первинного токену
$request->bind_param("ssd",$ip, $auth_key, $result["id"]);
if($request->execute() === true)
{
        $response->authKey = $auth;
        $response->access = $result['access'];
        die(json_encode($response));
```

Рис.2.7. Фрагмент коду, на якому видно генерацію серверного токен-ключа при авторизації адміністратора

Процес взаємодії клієнта і сервера діє таким чином: сервер отримує запит від клієнта (GET або POST), оброблює його і заповнює об'єкт відповіді. Потім ця відповідь надсилається клієнтові у форматі JSON, що містить інформацію про результат взаємодії з базою даних або помилки, якщо вони присутні.

2.5. Створення фронтенд частини

В системі фронтенду, використовується модульна архітектура, яка сприяє організації коду та полегшує розробку та підтримку. Модульність дозволяє розділити функціональність на окремі класи та компоненти, що спрощує керування та збільшує зрозумілість коду.

У модулі client.js, який служить одним з прикладів (оскільки admin.js має таку ж структуру), реалізовано три основні класи: tURL, USER та MAIN, а також декілька допоміжних класів, які відповідають за окремі сторінки в додатку.

Клас tURL є спеціалізованим компонентом, який відповідає за керування URL-адресою в контексті веб-додатку. Він зберігає інформацію про поточний URL та надає можливість динамічно змінювати параметри URL без перезавантаження сторінки. Це особливо корисно при зміні контенту на сторінці, коли потрібно зберегти стан додатку та його параметри.

Клас USER представляє інформацію про поточного користувача в системі. Він виконує функції збереження та керування даними користувача. USER визначає статус авторизації користувача шляхом перевірки токена на сервері та контролює доступ користувача до виконання певних дій на клієнтській стороні.

Головним контролером модуля є клас MAIN. Він об'єднує функціональність класів URL та USER, забезпечуючи взаємодію між ними та керуючи їх роботою. MAIN відповідає за генерацію контенту веб-додатку та керування сторінками. Він ініціалізує допоміжні класи сторінок, кожен з яких відповідає за окрему сторінку.

Модульність дає змогу створити для кожної окремої сторінки свій окремий клас. Кожен з цих класів має свій унікальний набір змінних та функцій, необхідних для правильної роботи конкретної сторінки. Це дозволяє організувати логіку та функціонал кожної сторінки в окремому класі, спрощуючи розробку та підтримку системи.

Загальний принцип модульної архітектури у фронтенді полягає у створенні окремих компонентів та класів, які взаємодіють між собою, щоб забезпечити функціональність та розширюваність системи. Цей підхід полегшує розробку, утримання та розширення кодової бази веб-додатку.

```
class User{
    constructor(parent){
        this.id = 0;
        this.parent = parent;
    };
    isAuth()
    {
      let key = null;
      if(this.authKey) key = this.authKey;
      else key = window.localStorage.getItem("token");
      if(!key) return 0;
      let d = GetS_Request(site + "client.php?check_login&token="+key);
      if (d.errors.length) {
        this.authKey = null;
        window.localStorage.removeItem("token");
        return 0;
      } else {
        this.authKey = key;
        this.id = d.id;
        return 1;
      }
      return 0;
    };
    login(key,access){
      this.access = access;
          this.authKey = key;
      window.localStorage.setItem("token",key);
    };

};
```

Рис.2.8. Код класу User

```
class tUrl{
    constructor(parent){
      this.parent = parent;
        this.url = new URL(window.location.href);
        this.attributes = [];
        this.attributes = this.url.search.split('&');
        this.attributes.splice(0,1);
        this.page = this.url.searchParams.get('p');
    };
    setPage(page,attributes=null){
      if(page) window.history.pushState({lastpage:this.page},"",this.url.pathname + "?p="+page+((attributes && attributes.length)?("&"+attributes.join('&')):(""));
      else window.history.pushState({lastpage:this.page},"",this.url.pathname+((attributes && attributes.length)?("?"+attributes.join('&')):(""));
    };
    getAttr(attr=""){
      if(!attr.length) return 0;
      else return this.url.searchParams.get(attr);
    }
};
```

Рис.2.8. Код класу tURL

```
class Main{
  constructor()
  {
    this.user = new User(this);
    this.url = new tUrl(this);
    this.menu = null;
    this.content = document.querySelector("body");
    this.page = null;
    this.showPage(this.url.page,this.url.attributes);
  };
  showPage(page = null,a=null){
    this.url.setPage(page,a);
    if(page)    document.querySelector("body header").id = "min";
    else document.querySelector("body header").id = "";
    switch(page){
        case null:{
            if(!this.menu) this.showMenu();
            if(this.page) delete this.page;
            this.content = document.querySelector("section");
            this.content.innerHTML = "<center>"+page+"</center>";
            this.page = new MPage(this);
            break;
        }
      case 'login':{
        if(!this.menu) this.showMenu();
            if(this.page) delete this.page;
            this.content = document.querySelector("section");
            //this.content.innerHTML = "";
            this.page = new Login(this);
        break;
      }
```

Рис.2.9. Фрагмент коду класу Main

```
/* ==============[Сторінки]============ */
class Cart{
   constructor(parent){
       this.parent = parent;
       this.section = document.querySelector('section');
       this.section.id = "featured-products";
       this.section.innerHTML = loading;
       this.cart = JSON.parse(window.localStorage.getItem('cart'));
       if(!this.cart) this.cart = [];
       this.products = {};
       if(window.products && (new Date().getTime()-window.lastload)<600000){
           this.products = window.products;
       }else{
           let info = GetS_Request(site + 'client.php?getProducts');
           if(info.errors != 'no-data'){
               this.products = info.data;
               window.products = this.products;
               window.lastload = new Date().getTime();
           }
       }
       this.section.innerHTML = `<center><h2>~ Кошик ${(!this.cart.length)?"порожній ":""}~ </h2></center>`;
       if(this.cart.length) this.section.innerHTML = '<div class="b-filter">Створити замовлення</div>';
       this.products.forEach((i)=>{
           let index = this.cart.find(e => e.id == i.id);
           if(index){
               let image = document.createElement('img');
               image.src = site+'images/'+i.image;
               let elem = document.createElement('div');
               elem.classList.add('product-list');
               let imgc = document.createElement('div');
               imgc.classList.add('image');
               elem.id=i.id;
               let c = (i.count-i.reserved)>=0?(i.count-i.reserved):0;
               let tc = (c<(index.count))?c:index.count;
               elem.innerHTML += `\
               <h3 class="product-title">${i.name}</h3>\
               <p>${i.price} грн</p>\
               <p>Код товару: ${i.id}</p>\
               <p><input name='count' type='number' value='${tc}' min='0' max='${c>=0?c:0}' class='input' style='padding:5px;max-width:50%;background-color:#eee;'></p>\
               <button class="delete">Прибрати з кошика</button>`;
               this.section.appendChild(elem);
               image.onload = ()=>{
```

Рис.2.10. Приклад другорядного з фрагменту Cart(Кошик)

РОЗДІЛ 3
ОПИС РЕАЛІЗОВАНОГО WEB-ДОДАТКУ

3.1. Клієнтська частина web-додатку

Для того, щоб клієнт або адміністратор міг скористатися сайтом з продажу косметики, біжутерії та аксесуарів, потрібно мати встановлений браузер на вашому пристрої та доступ до Інтернету.

Якщо ваш пристрій відповідає цим вимогам, тоді ви можете відвідати web-ресурс. На головній сторінці переглянути список товарів у продажу, сформувати кошик товарів та перейти у інші розділи сайту, такі як “Про нас”, “Контакти” та “Часті запитання”.

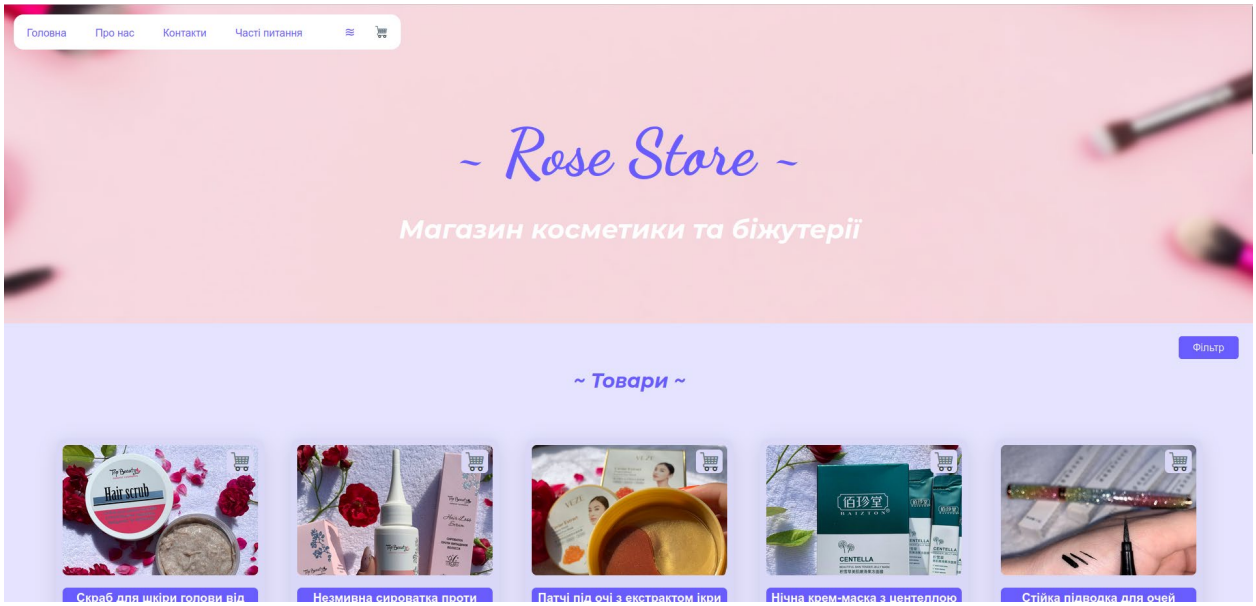

Рис. 3.1. Головна сторінка

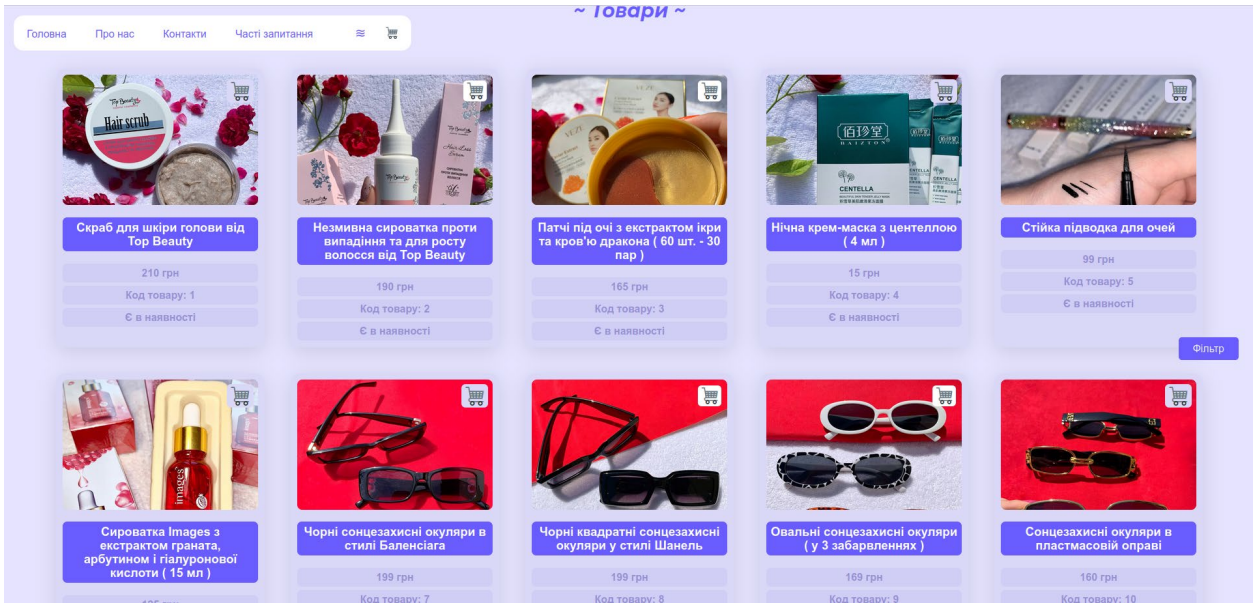

Рис. 3.2. Список доступних товарів на головній сторінці

В розділі «Про нас» можна перевірити контактну інформацію, графік відправки товару та іншу додаткову інформацію.

Рис. 3.3. Розділ “Про нас”

Для того, щоб сформувати список товарів, на основі якого буде оформлене замовлення, необхідно спочату на головній сторінці обрати товари до власного кошику та перейти у розділ “Кошик”.

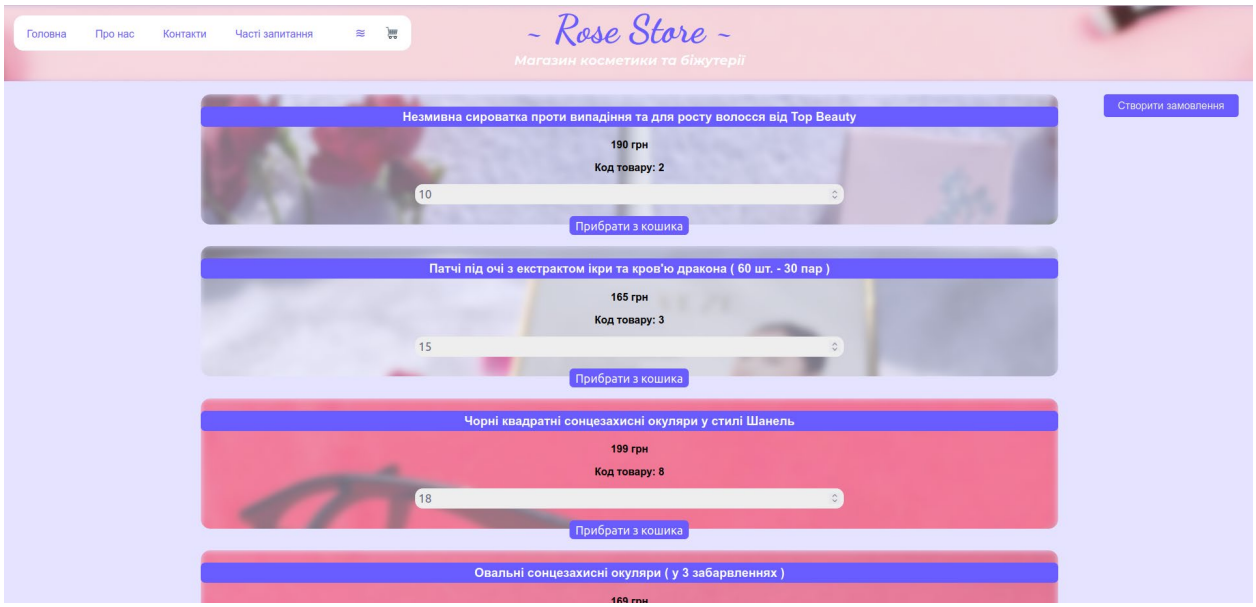

Рис.3.4. Розділ “Кошик”

Якщо натиснути “Створити замовлення” у розділі “Кошик” буде сформовано замовлення серверною частиною та буде відображено розділ “Замовлення”, якщо користувач авторизований.

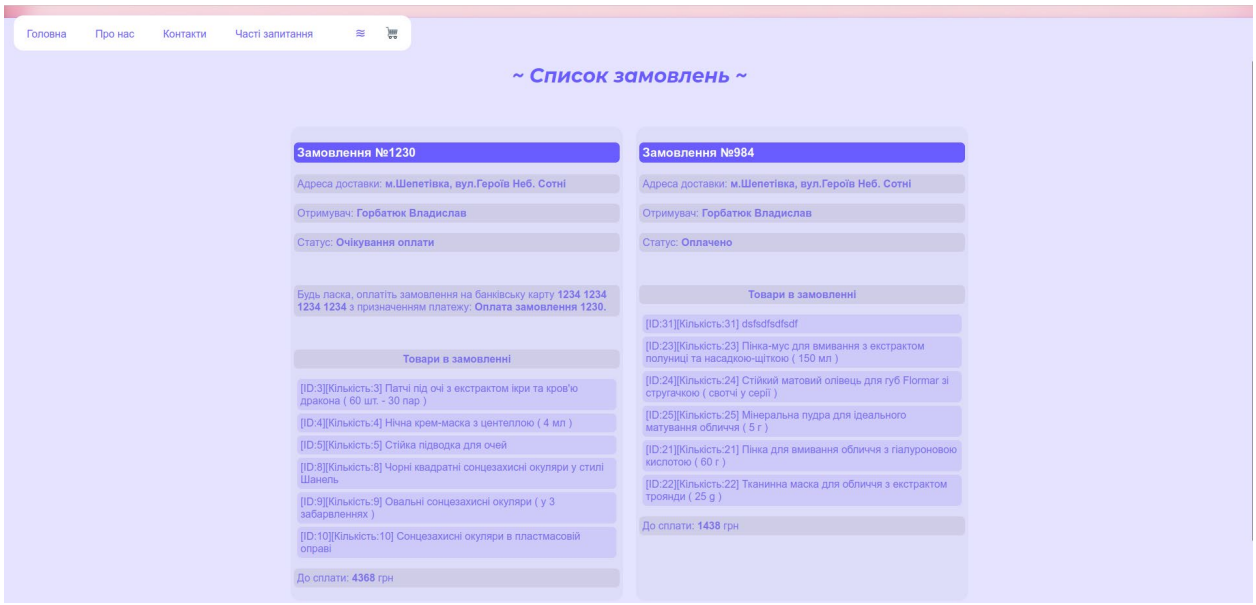

Рис. 3.5. Розділ “Замовлення”

У випадку, якщо користувач не авторизований, йому буде відображена сторінка авторизації.

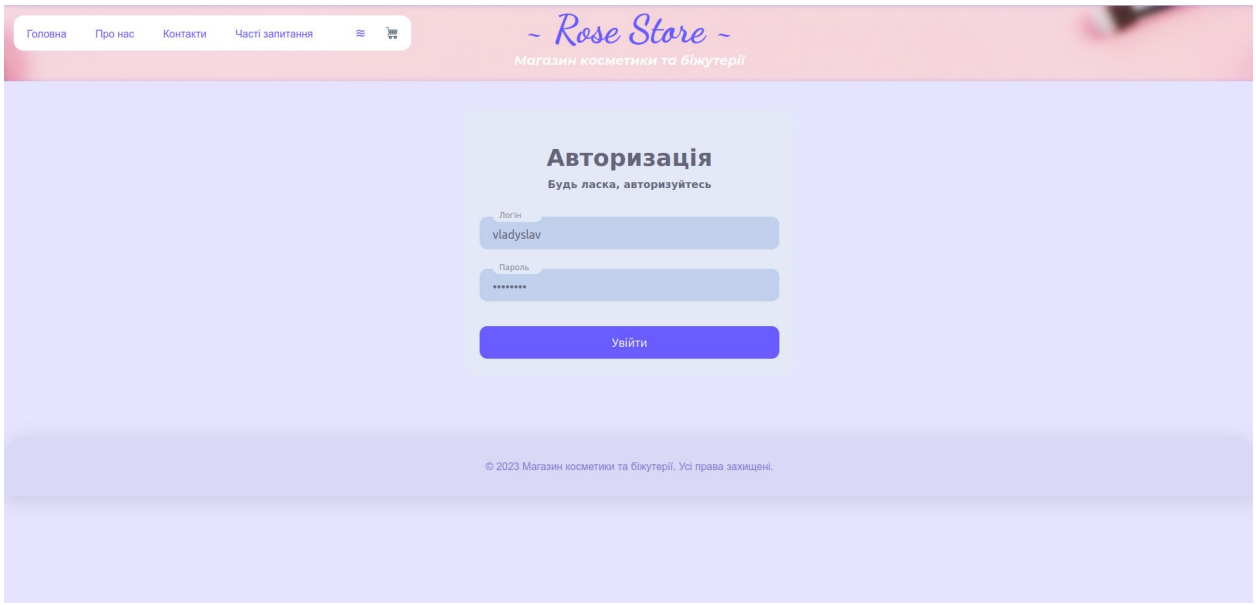

Рис.3.6. Авторизація користувача

В web-додатку реалізована перевірка на правильність введених даних як на серверній стороні, так і на клієнтській, тому у разі некоректного вводу, користувачу буде повідомлено про це.

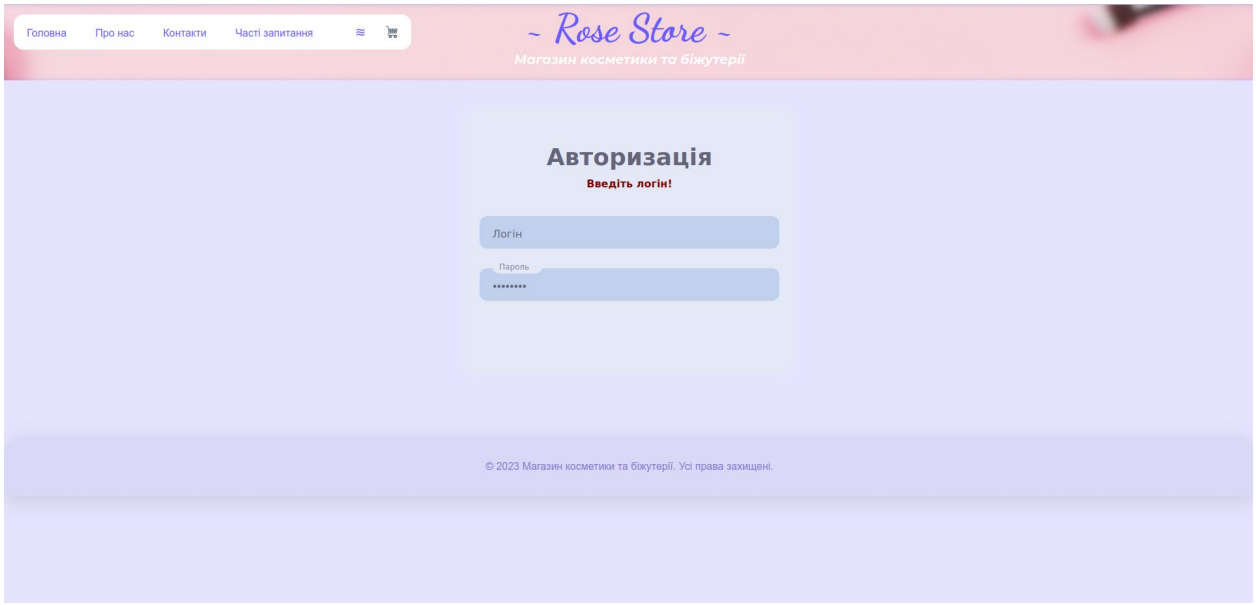

Рис.3.7. Відображення помилки користувачу в разі неправильних даних

Якщо у користувача немає акаунту, він може його створити, увівши бажані логін та пароль, а також інформацію, необхідну адміністратору для відправки поштою подальших замовлень користувача.

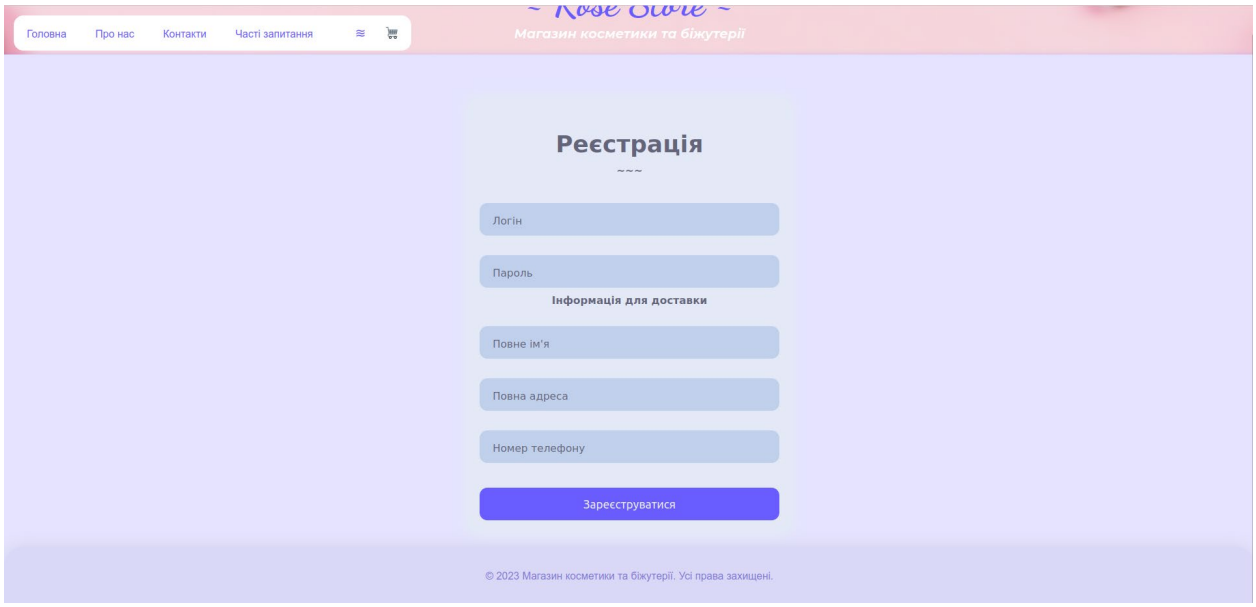

Рис.3.8. Створення нового користувача

Додатково для зручності підбору потрібних товарів на головній сторінці, було створено фільтр, за допомогою якого юзер може відсортувати доступні товари за категорією, підкатегорією та ціною.

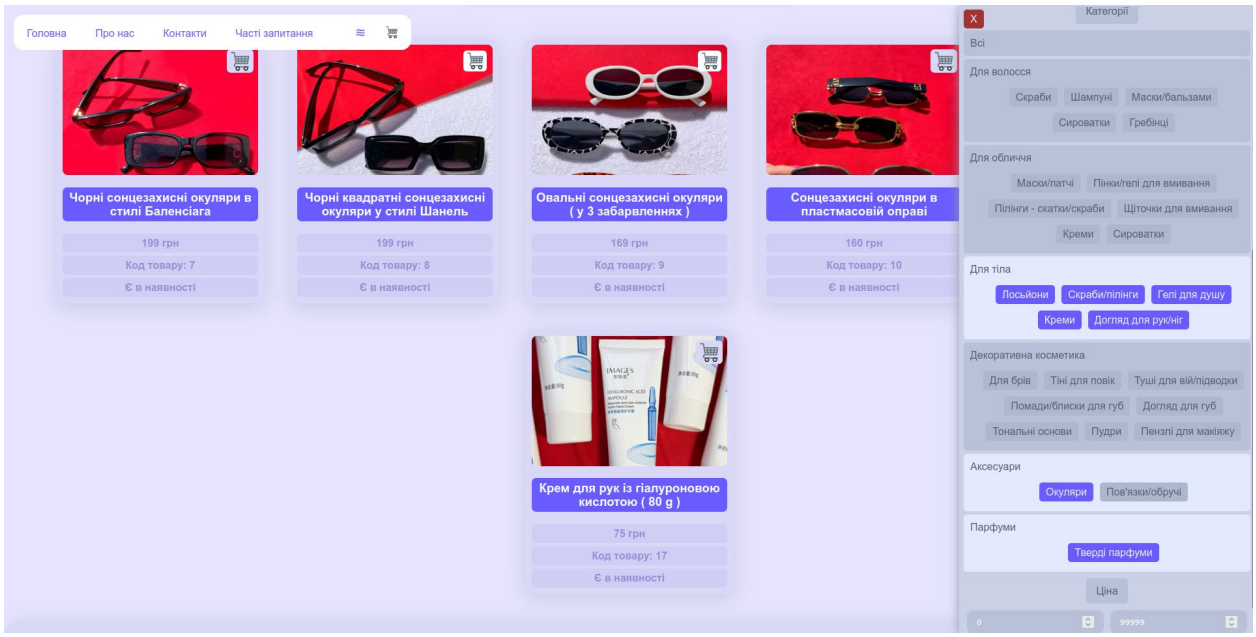

Рис.3.9. Фільтр

Також у додатку передбачена зміна інформації про покупця, для цього необхідно викликати контекстне підменю в головному та перейти у профіль.

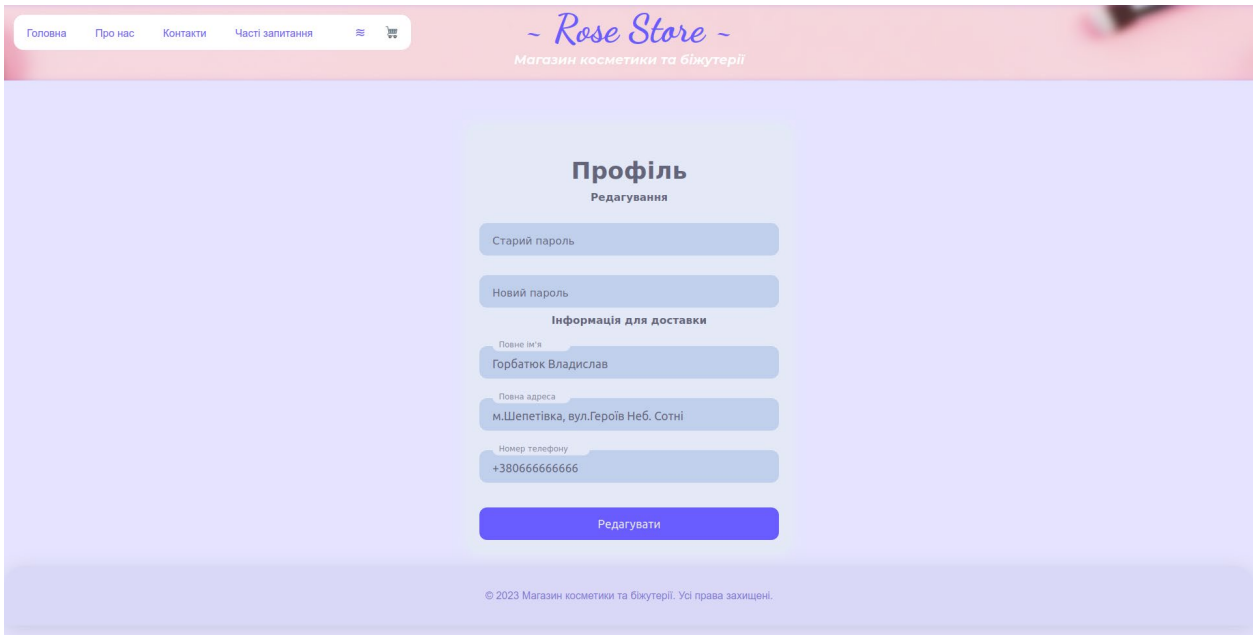

Рис.3.10. Зміна інформації про покупця

Для підтвердження, що зміна інформації ініційована власником акаунту, потрібно ввести поточний пароль, інакше зміни не будуть збережені.

3.2. Адміністраторська частина web-додатку

Тепер розглянемо частину web-додатку для адміністраторів. Щоб до неї перейти, необхідно до основної адреси сайту додати “/admin/”, після чого йому буде відображена авторизація в адміністративній панелі у випадку, якщо у браузері не збережений клієнтський ключ-токен доступу адміністратора. Цей ключ містить захешовану інформацію про адміністратора, його адміністративні права. Ключ повертається з серверу адміністратору після успішної авторизації. При авторизації на сервері генерується серверний ключ-токен на основі клієнтського та записується в базу даних для подальшої перевірки легітимності дій.

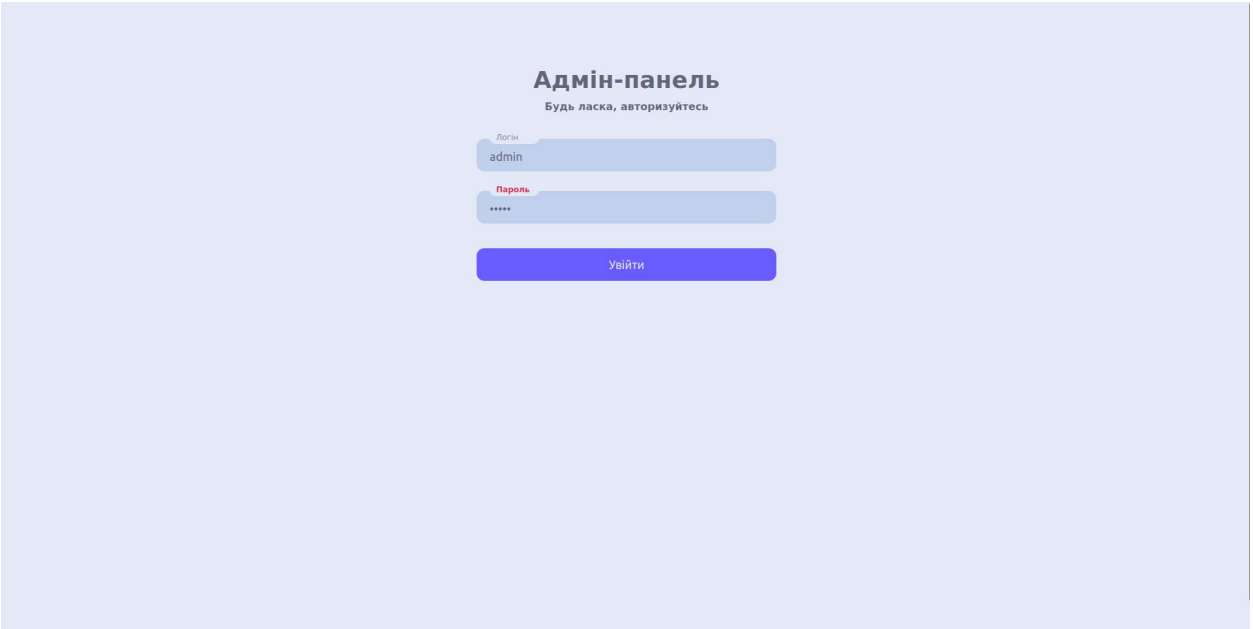

Рис.3.11. Авторизація адміністратора

Після авторизації адміністратора йому буде відображена адміністраторська панель керування, якщо точніше — стартовою сторінкою буде розділ “Категорії”, в якому адміністратор може створити/видалити категорію або субкатегорію.

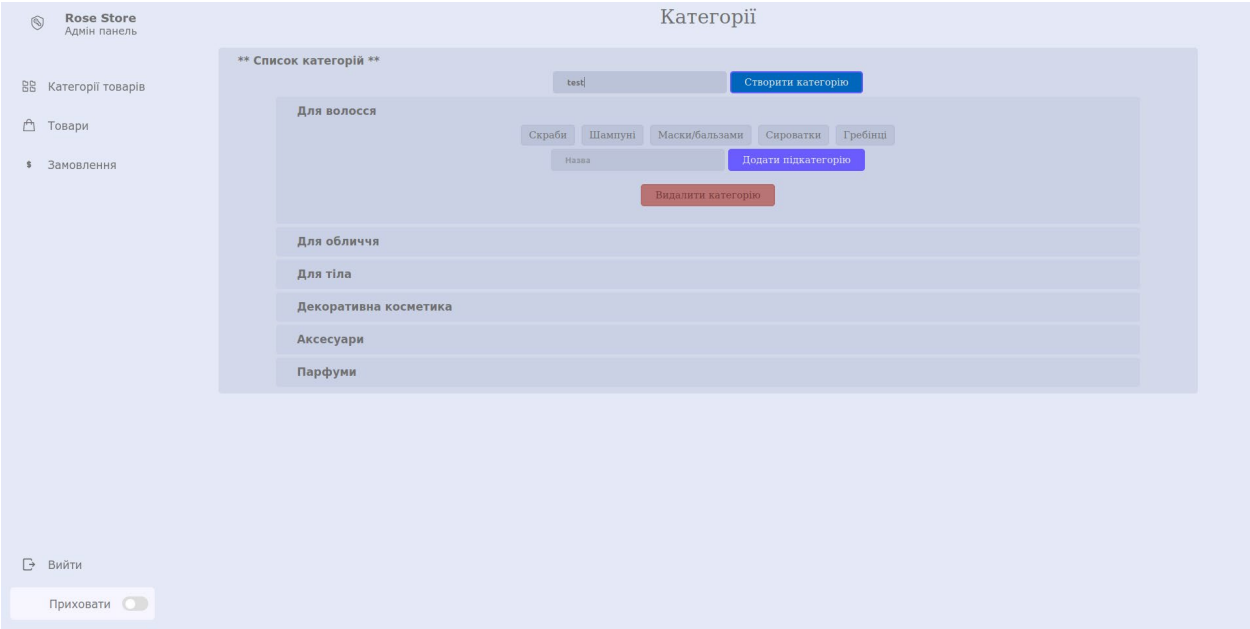

Рис. 3.12. Розділ “Категорії”

У випадку, якщо адміністратор хоче створити нову категорію/підкатегорію, йому необхідно ввести в відповідне поле назву

категорії чи підкатегорії та натиснути “Створити категорію” чи “Створити підкатегорію” відповідно.

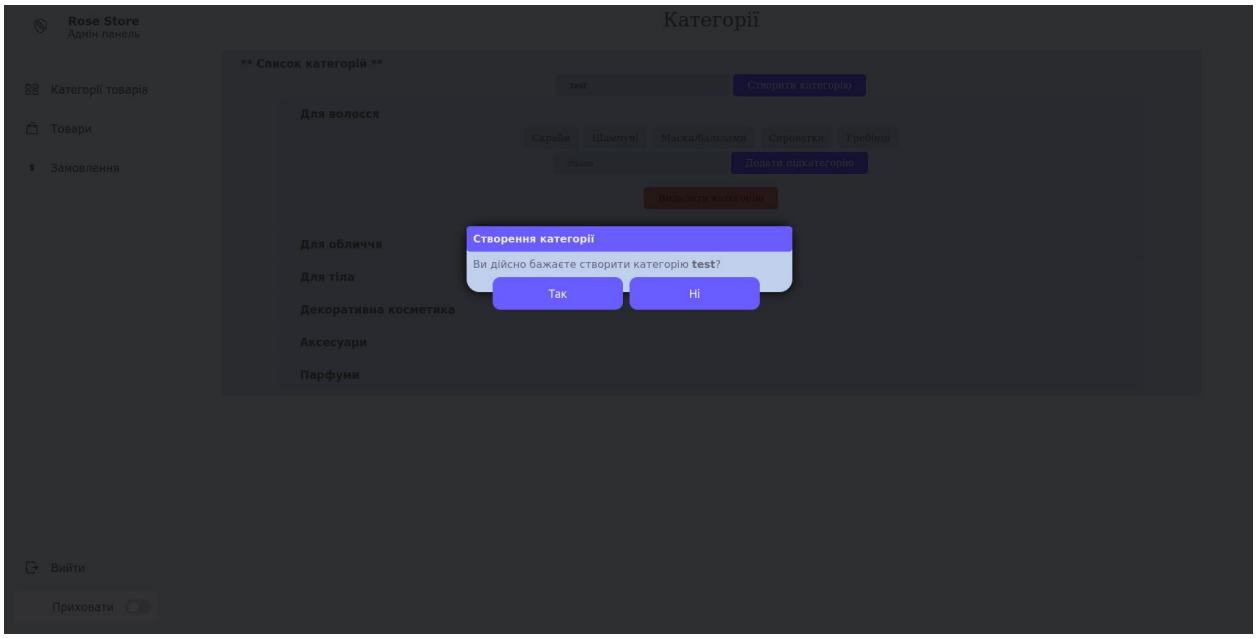

Рис.3.13. Створення нової категорії

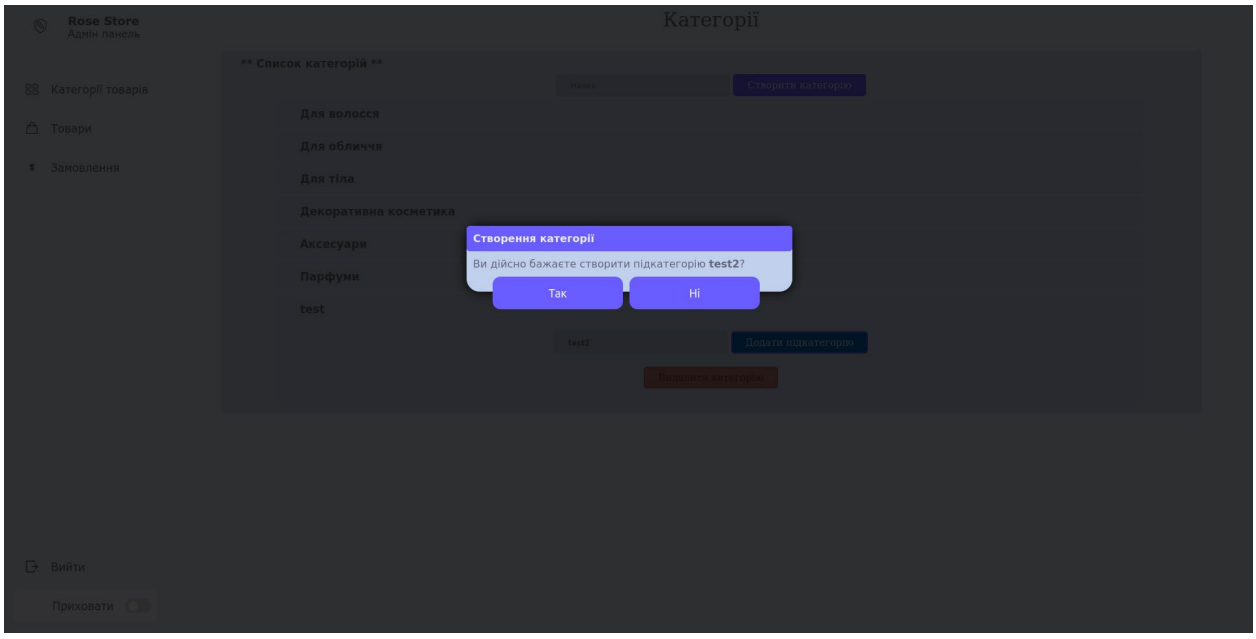

Рис.3.14. Створення підкатегорії

Рис.3.15. Результат створення нової категорії та підкатегорії в цій категорії

При видалені категорії всі підкатегорії, пов’язані з нею, автоматично видаляються. Після видалення певної підкатегорії, всі товари, пов’язані з нею, теж автоматично видаляються.

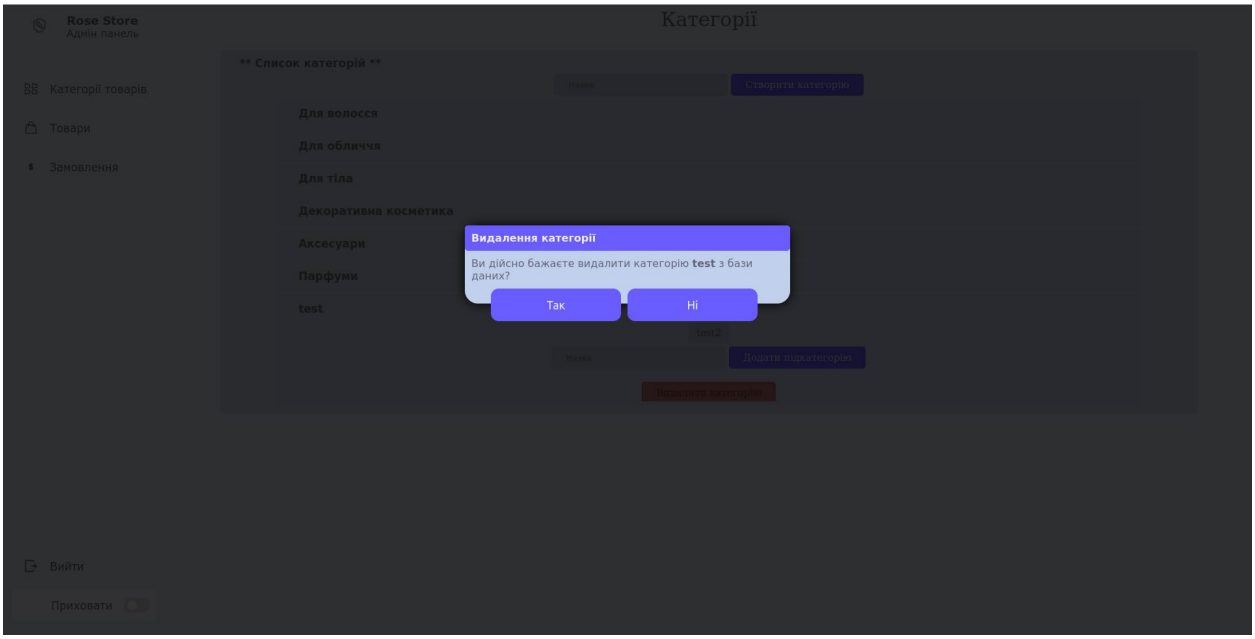

Рис.3.16. Видалення категорії

В меню адміністративної панелі можна ще перейти в розділи “Товари”, “Замовлення” або вийти з акаунту, натиснувши кнопку “Вихід”.

Передбачена можливість мінімізувати бокове меню, натиснувши на перемикач в самому низу цього меню, для зручності відображення інформації на пристроях невеликими розмірами екрану.

Рис.3.17. Меню після увімкнення перемикача “Приховати”

У розділі “Товари” адміністратор може переглянути список товарів та відфільтрувати їх, як і в користувацькій частині, товари за категорією, підкатегорією та ціною для подальших дій з необхідними товарами.

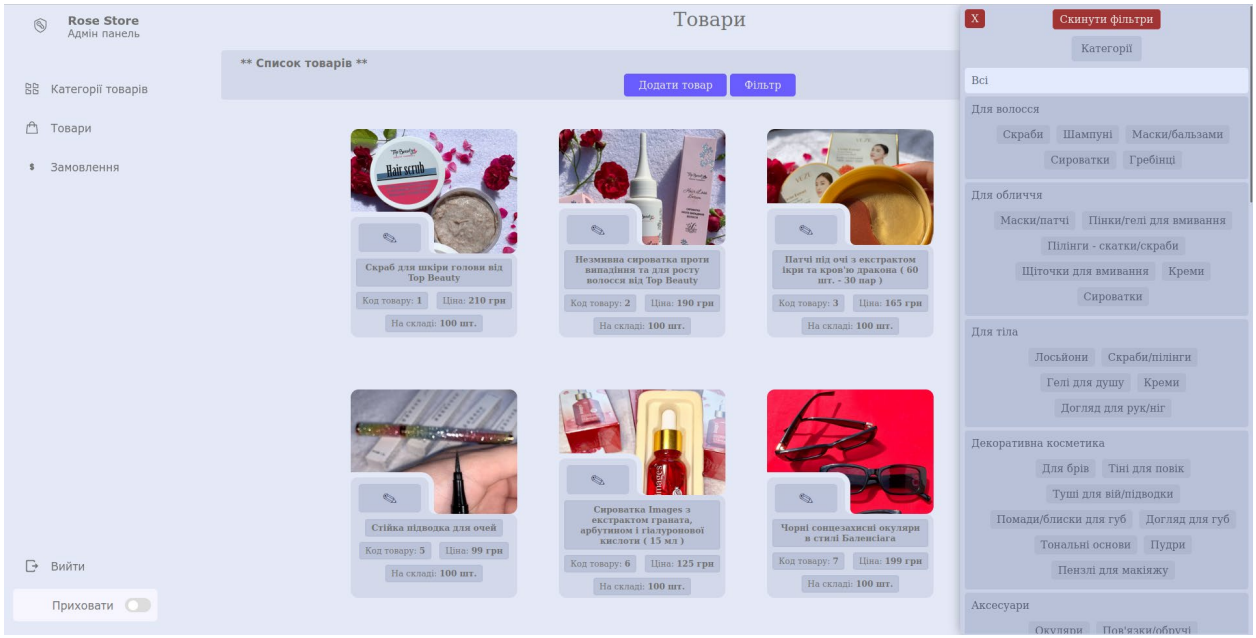

Рис.3.18. Розділ “Товари” з увімкненим фільтром

При натисканні кнопки “Додати товар” буде відображено форму для створення нового товару.

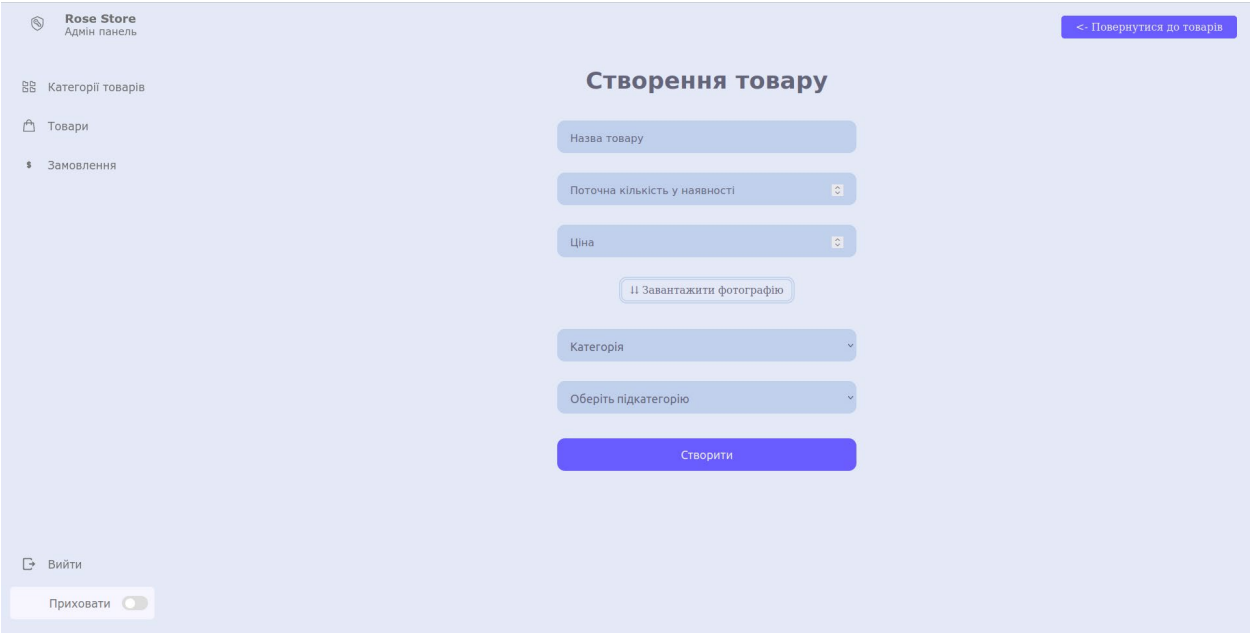

Рис.3.19. Сторінка створення нового товару

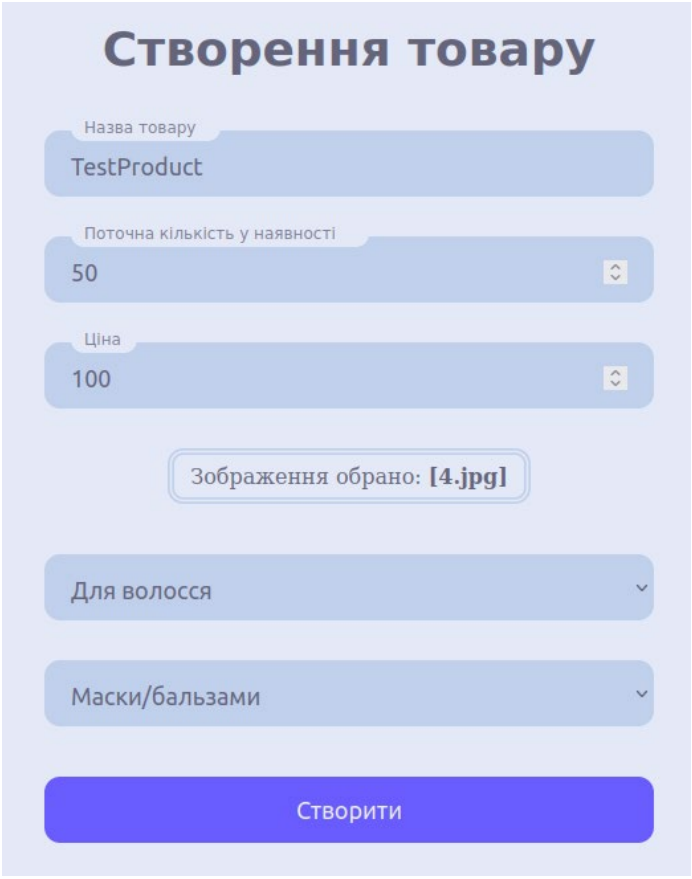

Рис.3.20. Заповнена форма створення товару

У цій формі необхідно ввести назву товару, його кількість у наявності, ціну. Після чого завантажити фотографію товару та обрати категорію, підкатегорію. Як тільки інформація буде заповнена коректно та буде натиснуто “Створити”, дані про товар перевіряться на валідність та, у випадку правильності інформації, в базі даних буде додано запис з відповідним товаром.

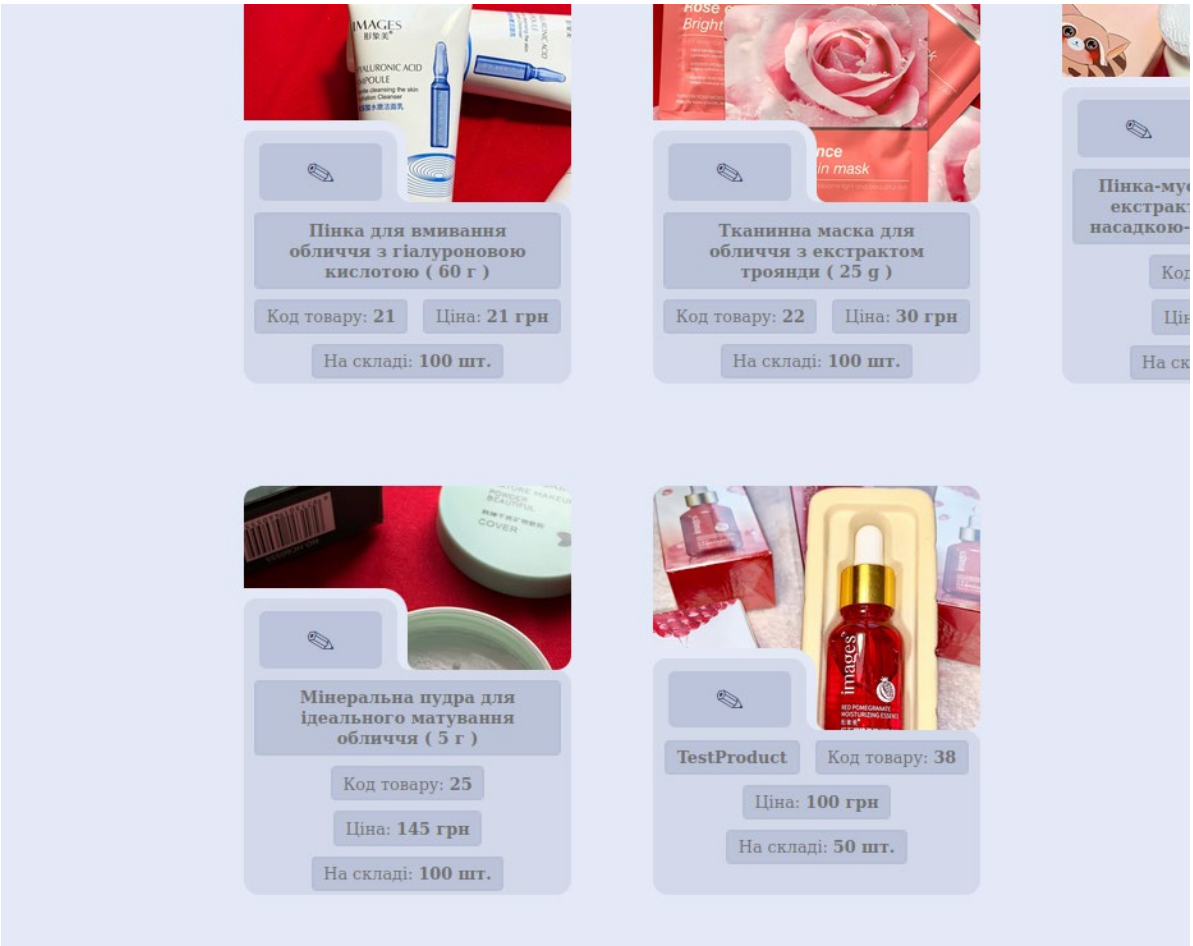

Рис.3.21. Результат створення товару

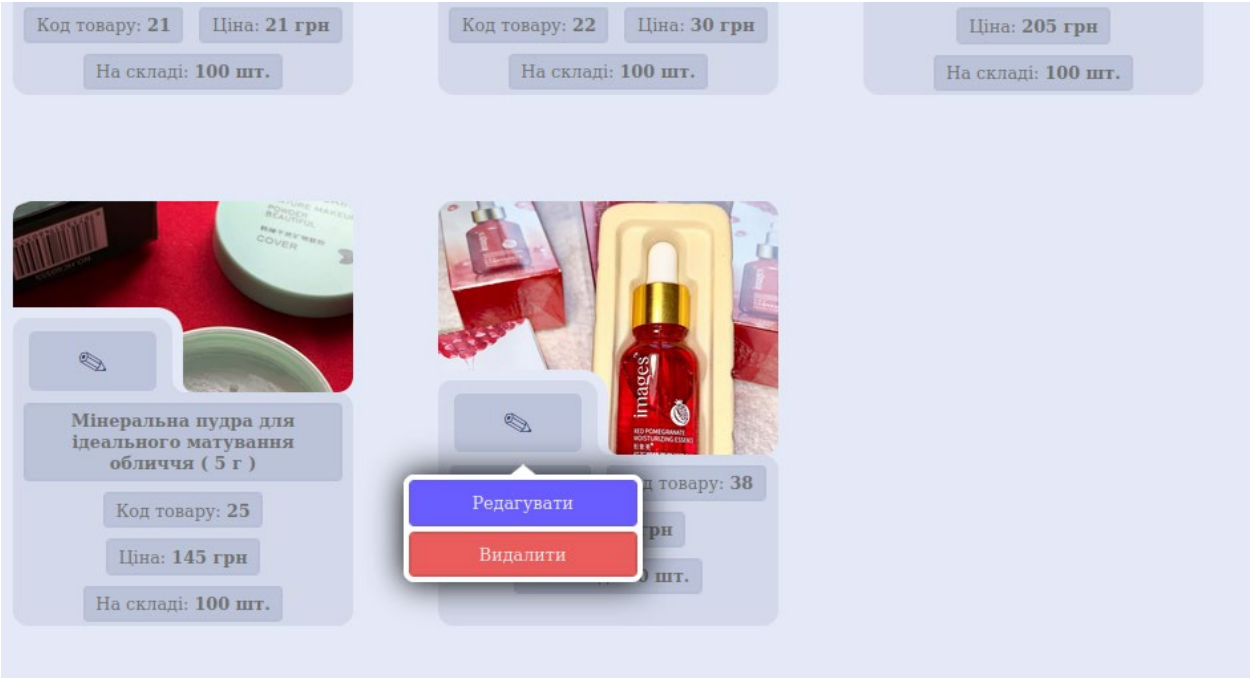

Рис.3.22. Контекстне меню товару

Повернувшись на сторінку з товарами, адміністратор може видалити товар або редагувати, викликавши контекстне меню товару.

Якщо натиснути “Редагувати”, з’явиться спливаюча форма “Редагування товару” для зміни інформації про товар.

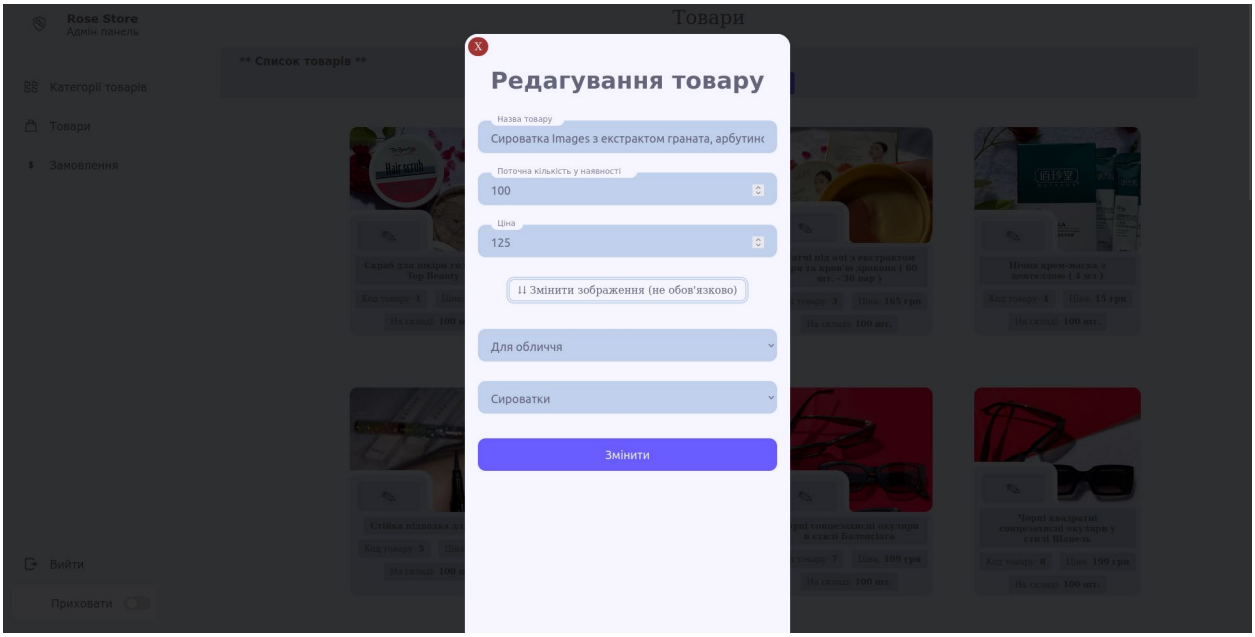

Рис.3.23. Спливаюча форма “Редагування товару”

У цій формі адміністратор може змінити лише необхідну до зміни інформацію або вибрати нову фотографію, якщо необхідно її змінити. Для підтвердження необхідно натиснути “Змінити”. Інформація про товар змінюється в базі даних та автоматично на сторінці з товарами.
Далі розглянемо розділ із замовленнями.

Рис.3.24. Розділ “Замовлення”

У цьому розділі містяться всі замовлення, які не є на стадії “Отримано замовником”. Вибравши пункт в випадаючому списку, автоматично зміниться статус замовлення в базі даних. У випадку, якщо обрано “Отримано”, замовлення перестане відображатися у поточних замовленнях. Це зроблено, щоб не було надлишкової інформації на сторінці у випадку великої кількості виконаних замовлень.
У адміністратора є можливість відфільтрувати замовлення по номеру (необхідний для призначення платежу).

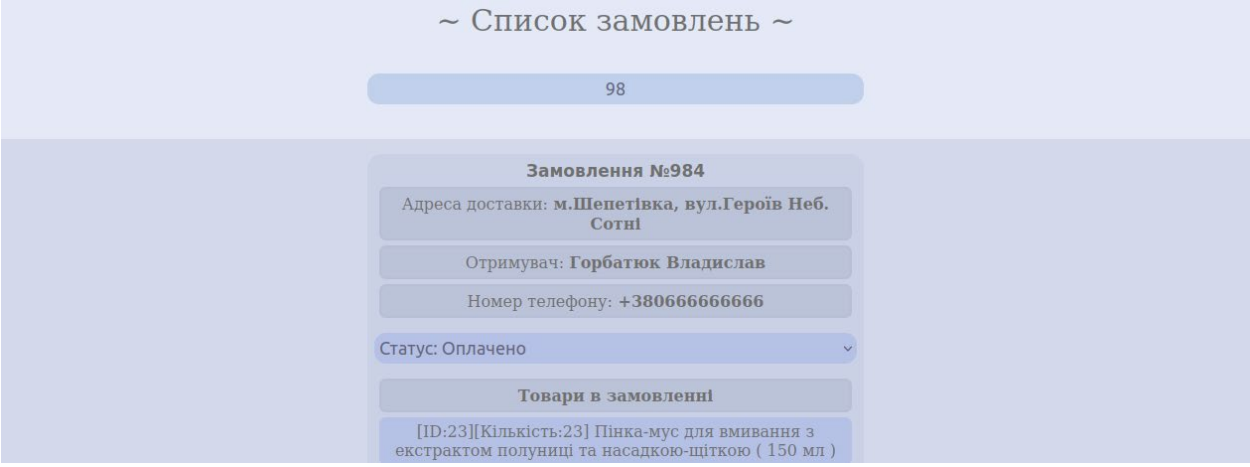

Рис.3.25. Фільтрація замовлень по номеру

3.3 Опис використання веб-сайту

Процес використання веб-сайту заснований на таких кроках:

1) Користувач відвідує користувацьку частину сайту, переглядає певну інформацію про магазин, на головній сторінці додає товари до кошику. Після цього переходить до кошика, переглядає додані товари, у разі необхідності видаляє товари, які передумав замовляти, виставляє кожному товару бажану кількість до замовлення та створює замовлення. Далі користувач очікує зворотний зв’язок для підтвердження.

2) Адміністратор(продавець) відвідує адміністративну частину сайту для наповнення онлайн-магазину товарами та переглядає, змінює або підтверджує статус замовлення і, згадно інформації про отримувача із замовлення, може поштою відправити товар покупцю, якщо той оплатив замовлення.

ВИСНОВКИ

У рамках даної кваліфікаційної роботи було проведено дослідження та розроблено інтернет-магазин косметики, біжутерії та аксесуарів. Метою проекту було створення функціонального та привабливого веб-додатку, який забезпечує швидкий та ефективний обмін інформацією між користувачем та системою.

У процесі реалізації проекту було враховано процеси та особливості організації інтернет-торгівлі в сфері косметики, біжутерії та аксесуарів, а також тенденції ринку та поведінку споживачів. Були прийняті до уваги вимоги з приводу швидкості та ефективності та реалізовано додаток без використання «тяжких» зайвих бібліотек для оптимальної роботи додатку.

Структура даних та веб-сайт магазину були розроблені з урахуванням зручної навігації, категорій товарів та можливостей фільтрації, що допомагають користувачам знаходити потрібні товари швидко та зручно.

У результаті було реалізовано функціональний веб-додаток, який дозволяє користувачам реєструватися, авторизовуватися, додавати товари у кошик, оформлювати замовлення, а продавцю здійснювати адміністративні функції для керування товарами та замовленнями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. W3Schools Online Web Tutorials [Електронний ресурс] – Режим доступу: https://www.w3schools.com/ (дата звернення: 13.03.2023)
2. MDN Web Docs [Електронний ресурс] - Режим доступу: https://developer.mozilla.org/ (дата звернення: 5.03.2023)
3. A List Apart – For people who make websites [Електронний ресурс] - Режим доступу: https://alistapart.com/ (дата зверення: 08.03.2023)
4. HubSpot Blog | Marketing, Sales, Agency, and Customer Success Content [Електронний ресурс] — Режим доступу: https://blog.hubspot.com/website?hubs_content=blog.hubspot.com%2Fmarketing%2Fweb-design-html-css-javascript&hubs_content-cta=null&hubs_post-cta=blognavcard-website (дата звернення: 10.05.2023)
5. Український веб-довідник [Електронний ресурс] — Режим доступу: https://css.in.ua/ (дата звернення: 20.05.2023)
6. The Modern JavaScript Tutorial [Електронний ресурс] - Режим доступу: https://javascript.info/ (дата звернення: 6.03.2023)