Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

ПРОГРАМНИЙ МОДУЛЬ ІНТЕРНЕТ-МАГАЗИНУ З АДАПТИВНИМ УПРАВЛІННЯМ

Виконав студент 4 курсу, групи 1КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напрямку підготовки, спеціальності)
Роман ТАРАСЕНКО
(ім'я та прізвище)
Керівник: д.т.н., проф. каф. КН
Ярослав ІВАНЧУК
(посада, ім'я та прізвище)
«4» червня 2025 р.
Рецензент: д.т.н., проф., зав. каф. КСУ
В'ячеслав КОВТУН
(посада, ім'я та прізвище)
«5» червня 2025 р.

Допущено до захисту
Зав. кафедри КН Андрій ЯРОВИЙ
«6» червня 2025р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖЕНО
Завідувач кафедри КН
проф., д.т.н. Андрій ЯРОВИЙ
« 5 » травня 2025р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ
Тарасенко Роману Олександровичу

1. Тема роботи: Програмний модуль інтернет-магазину з адаптивним управлінням
Керівник роботи: Іванчук Ярослав Володимирович, д.т.н, професор кафедри КН
Затверджені наказом вищого навчального закладу "20" Березня 2025 року №97

2. Термін подання студентом роботи 30 травня 2025р.

3. Вихідні дані до роботи: база даних типу SQL; кількість сторінок інтерфейсу користувача - не менше 3 од.; кількість сутностей в базі даних - не менше 5 од.; кількість одночасно оброблюваних транзакцій - не менше 12 од.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): визначення об'єкта, предмета, мети та задач подальшого дослідження; аналіз предметної області, існуючих методів, алгоритмів та інструментів для створення програмного модуля інтернет-магазину з адаптивним управлінням; проєктування програмного модуля інтернет-магазину; програмна реалізація модуля.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): схема алгоритму роботи взаємодії розроблених модулів програмного забезпечення інтернет-магазину. ER-діаграма розробленої бази даних інтернет-магазину з адаптивним управлінням. Загальний вигляд інтерфейсу користувача для використання програмного модуля інтернет-магазину (адаптивна вітрина, корзина, кабінет користувача, система управління контентом).

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Іванчук Я. В., д.т.н., проф.. каф. КН		

7. Дата видачі завдання 5 травня 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Термін виконання		Примітка
		початок	закінчення	
1	Визначення об'єкта, предмета, мети та задач подальшого дослідження.	05.05.25	07.05.25	
2	Аналіз предметної області, функціональності існуючих інтернет-магазинів та огляд фреймворків і технологій для створення програмного модуля інтернет-магазину з адаптивним управлінням.	08.05.25	14.05.25	
3	Проєктування програмного модуля інтернет-магазину з адаптивним управлінням.	15.05.25	26.05.25	
4	Програмна реалізація програмного модуля інтернет-магазину з адаптивним управлінням..	27.05.25	30.05.25	
5	Оформлення пояснювальної записки.	01.06.25	04.06.25	

Студент _____ Роман ТАРАСЕНКО
(підпис) (ім'я та прізвище)

Керівник роботи _____ Ярослав ІВАНЧУК
(підпис) (ім'я та прізвище)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 80 сторінок формату А4, містить 19 рисунків, 2 таблиці, список використаної літератури налічує 42 найменування.

Ця бакалаврська робота присвячена розробці програмного модуля WEB-додатку для організації роботи книжкового інтернет-магазину. Основною метою роботи є створення зручного, адаптивного та ефективного WEB-додатку, який забезпечує автоматизацію управління каталогом книг, замовленнями, відгуками та комунікацією між клієнтами й менеджерами. Було розроблено модульну структуру WEB-додатку для книжкового магазину «Гай Монтег». Для front-end частини використано Django Template Engine, Bootstrap та jQuery, що забезпечують адаптивний і інтерактивний інтерфейс. Для back-end застосовано фреймворк Django з базою даних SQLite, що гарантує надійність і цілісність даних.

Результатом роботи є функціональний програмний модуль, який оптимізує бізнес-процеси книжкового магазину, забезпечуючи зручність для користувачів та ефективне адміністрування.

Ключові слова: книжковий інтернет-магазин, WEB-додаток, управління каталогом, Django, адаптивний інтерфейс.

ABSTRACT

The bachelor's thesis consists of 80 A4 pages, including 19 figures, 2 tables, and a reference list containing 42 sources.

This bachelor's thesis is dedicated to the development of a software module for a WEB application designed to organize the operations of an online bookstore. The primary goal of the work is to create a user-friendly, adaptive, and efficient WEB application that automates the management of a book catalog, orders, reviews, and communication between customers and managers. A modular structure for the WEB application of the "Guy Montag" bookstore was developed. The front-end was built using Django Template Engine, Bootstrap, and jQuery, ensuring an adaptive and interactive interface. The back-end utilized the Django framework with an SQLite database, guaranteeing data reliability and integrity.

The result of the work is a functional software module that optimizes the business processes of the bookstore, providing convenience for users and efficient administration.

Keywords: online bookstore, WEB application, catalog management, Django, adaptive interface.

ЗМІСТ

ВСТУП

Актуальність. Сфера електронної комерції є однією з ключових галузей сучасної економіки, що стрімко розвивається завдяки зростанню попиту на зручні та швидкі способи здійснення покупок через Інтернет. Інтернет-магазини відіграють важливу роль у забезпеченні доступу до товарів, зокрема книжкової продукції, яка сприяє розвитку культури, освіти та інтелектуального потенціалу суспільства. Сучасні виклики, такі як необхідність оптимізації бізнес-процесів, забезпечення зручності для користувачів та підвищення конкурентоспроможності, вимагають впровадження ефективних інформаційних технологій і автоматизованих систем управління [1].

Одним із перспективних напрямів розвитку електронної комерції є створення програмних модулів для інтернет-магазинів, які дозволяють автоматизувати процеси управління каталогом товарів, замовленнями, відгуками клієнтів та комунікацією між користувачами і менеджерами. Такі WEB-додатки забезпечують централізоване управління даними, зручний доступ до інформації через різні пристрої, підключені до мережі Інтернет, а також інтеграцію з іншими сервісами, наприклад, платіжними системами чи логістичними платформами [2].

Програмний модуль інтернет-магазину, зокрема книжкового, є комплексним рішенням, яке охоплює ключові показники діяльності онлайн-платформи, а саме:

1) Управління каталогом книг, включаючи створення, редагування та видалення товарів і категорій.

2) Обробка замовлень клієнтів, відстеження їх статусів та управління доставкою.

3) Надання можливості клієнтам залишати відгуки та оцінки на книги, що сприяє підвищенню довіри до магазину.

4) Забезпечення комунікації між клієнтами та менеджерами через вбудовану систему чату.

5) Формування статистичних даних та звітів для аналізу діяльності магазину та прийняття управлінських рішень [3].

Ефективний програмний модуль інтернет-магазину повинен мати зручний та інтуїтивно зрозумілий інтерфейс для всіх категорій користувачів (клієнтів, менеджерів, адміністраторів), гарантувати безпеку даних, таких як персональна інформація клієнтів та історія транзакцій, а також бути адаптивним до різних пристроїв і масштабуватися залежно від потреб бізнесу. Важливим питанням є підтримка локалізації, зокрема використання української мови, що сприяє популяризації національної культури та літератури [4].

Впровадження програмного модуля інтернет-магазину дозволяє значно підвищити ефективність управління асортиментом, оптимізувати процеси обробки замовлень, скоротити час на виконання рутинних операцій, покращити взаємодію з клієнтами та забезпечити якісне обслуговування. Це сприяє зростанню продажів, підвищенню лояльності клієнтів та загальному розвитку книжкового ринку в Україні.

Мета дослідження полягає у підвищенні ефективності управління процесами інтернет-магазину шляхом розробки програмного модуля WEB-додатку для книжкового магазину, що забезпечить зручність для користувачів та ефективне адміністрування бізнес-процесів.

Об’єкт дослідження – процес автоматизованого управління діяльністю книжкового інтернет-магазину за допомогою WEB-додатку.

Предмет дослідження – програмні засоби WEB-додатку для організації роботи книжкового інтернет-магазину.

Задачі дослідження:

1) Провести аналіз предметної області, методів та засобів управління діяльністю книжкових інтернет-магазинів.

2) Розробити структуру програмного забезпечення для книжкового інтернет-магазину.

3) Реалізувати методи взаємодії з базою даних для управління каталогом, замовленнями та комунікацією.

4) Розробити адаптивний інтерфейс користувача з урахуванням вимог до зручності та локалізації.

5) Виконати програмну реалізацію системи книжкового інтернет-магазину з урахуванням сучасних вимог до електронної комерції.

6) Провести тестування роботи програмного модуля WEB-додатку для забезпечення його стабільності та функціональності.

7) Розробити інструкцію користувача для роботи з системою.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз проблеми впровадження WEB-додатку інтернет-магазину

Впровадження WEB-додатку для інтернет-магазину є важливим кроком для забезпечення зручного доступу до товарів, автоматизації бізнес-процесів та підвищення конкурентоспроможності. Однак цей процес пов’язаний із низкою проблем, які потребують детального аналізу та вирішення для забезпечення ефективної роботи системи.

Однією з ключових проблем є інтеграція WEB-додатку з наявними системами управління складом, фінансами та клієнтськими базами даних. Ця інтеграція може бути складною через різноманітність форматів даних, застарілі системи або несумісність програмного забезпечення. Для уникнення втрати даних чи їх дублювання необхідно забезпечити чітку синхронізацію між модулями, що вимагає ретельного планування та тестування [6]. Наприклад, у коді проєкту реалізована база даних SQLite для управління інформацією про книги, замовлення та користувачів, що спрощує інтеграцію, але може бути обмеженням для масштабування на великі обсяги даних.

Безпека даних є критично важливою у сфері електронної комерції. WEB-додаток повинен гарантувати захист персональних даних клієнтів, таких як адреси, номери телефонів та платіжна інформація, від несанкціонованого доступу чи кібератак. Це вимагає впровадження сучасних методів шифрування, безпечної автентифікації та регулярного оновлення системи безпеки. У коді проєкту використовується Django з вбудованими механізмами захисту (CSRF, аутентифікація), однак для реального застосування необхідно додатково впроваджувати HTTPS та захист від SQL-ін’єкцій [7].

Навчання персоналу та адаптація клієнтів до нового інтерфейсу також є викликом. Менеджери магазину, які працюють із панеллю адміністрування, повинні бути ознайомлені з функціоналом для управління книгами,

замовленнями та комунікацією з клієнтами. У коді проєкту реалізована панель менеджера з інтуїтивними функціями, такими як редагування книг та категорій, але для користувачів із низьким рівнем технічних навичок потрібні додаткові інструкції та підтримка [8]. Клієнти, у свою чергу, очікують зручного та зрозумілого інтерфейсу для пошуку товарів, оформлення замовлень і комунікації, що вимагає ретельного дизайну.

Масштабованість та продуктивність додатку є важливими показниками, оскільки зростання кількості користувачів, товарів чи замовлень може призвести до перевантаження системи. У розробленому проєкті використовується SQLite, що підходить для невеликих магазинів, але для великих обсягів даних потрібен перехід на більш потужні бази даних, такі як PostgreSQL, та оптимізація запитів для швидкої обробки [9]. Пагінація у списку книг та замовлень, реалізована в коді, частково вирішує проблему продуктивності, але потребує додаткових рішень для великих каталогів.

Зручність використання (UX) та адаптивність інтерфейсу є ключовими для залучення та утримання клієнтів. Інтернет-магазин повинен бути зручним для користувачів різного віку та технічних навичок, а також адаптивним для мобільних пристроїв. У коді проєкту передбачено адаптивний дизайн (CSS та медіа-запити), але для підвищення зручності необхідно додати функції, такі як фільтри пошуку за жанрами чи авторами, та інтерактивні елементи, як-от автозаповнення форм [10].

Технічна підтримка та оновлення додатку є необхідними для забезпечення його стабільності. Регулярне виправлення помилок, оновлення безпеки та додавання нових функцій потребують кваліфікованої команди розробників. У коді проєкту передбачено базові функції, такі як повідомлення про помилки та чат для підтримки клієнтів, але для масштабного проєкту потрібна спеціалізована служба підтримки.

Фінансові питання також відіграють важливу роль. Розробка, впровадження та підтримка WEB-додатку вимагають значних інвестицій. Недостатнє фінансування може призвести до компромісів у функціональності,

безпеці чи масштабованості. Для малого бізнесу, як у реалізованому проєкті, використання фреймворку Django знижує витрати на розробку, але для великих магазинів потрібні додаткові ресурси на інфраструктуру та маркетинг.

Отже, впровадження WEB-додатку для інтернет-магазину є складним процесом, що вимагає вирішення проблем інтеграції, безпеки, навчання персоналу, масштабованості, зручності використання, технічної підтримки та фінансового планування. Ретельний аналіз цих показників та використання сучасних технологій, як у коді проєкту, дозволяють створити ефективну систему, що підвищує зручність для клієнтів, автоматизує процеси та сприяє розвитку бізнесу.

1.2 Принципи побудови систем управління інтернет-магазинами

Системи управління інтернет-магазинами мають унікальні особливості, які зумовлені специфікою електронної комерції, необхідністю забезпечення зручності для користувачів, ефективного управління товарними запасами та безпеки транзакцій. Ці системи спрямовані на створення комфортного досвіду для клієнтів, автоматизацію бізнес-процесів та інтеграцію з різноманітними зовнішніми сервісами. Розглянемо детальніше принципи побудови та функціонування таких систем, з урахуванням коду проєкту книжкового магазину "Гай Монтег".

Однією з ключових особливостей систем управління інтернет-магазинами є їх інтеграція з різними платіжними системами, логістичними сервісами та маркетинговими інструментами. Сучасні інтернет-магазини, такі як той, що реалізований у проєкті "Гай Монтег", повинні забезпечувати безперебійну взаємодію з платіжними шлюзами (наприклад, LiqPay, PayPal), системами управління складом, а також маркетплейсами та соціальними мережами для просування товарів. У коді проєкту видно, як Django-сесії використовуються для управління кошиком покупок (cart_add, cart_remove),

що є прикладом автоматизації обробки замовлень та інтеграції з базою даних SQLite [11]. Це дозволяє створювати єдиний інформаційний простір, де дані про товари, замовлення та клієнтів доступні в реальному часі, що підвищує ефективність управління магазином та покращує клієнтський досвід.

Інша важлива особливість – забезпечення безпеки та конфіденційності даних користувачів. Інтернет-магазини обробляють чутливу інформацію, таку як персональні дані клієнтів, номери телефонів, адреси доставки та платіжні реквізити. У проєкті "Гай Монтег" це реалізовано через використання Django-модулів аутентифікації (django.contrib.auth) та CSRF-захисту (CsrfViewMiddleware) [12]. Системи управління інтернет-магазинами повинні застосовувати сучасні методи захисту, такі як шифрування даних (SSL/TLS), контроль доступу на основі ролей (наприклад, розмежування прав менеджерів та клієнтів у manager_dashboard) та регулярне оновлення програмного забезпечення для усунення вразливостей. Це забезпечує захист від несанкціонованого доступу та збереження довіри клієнтів.

Системи управління інтернет-магазинами повинні враховувати різноманітність користувачів та їхні потреби. У проєкті "Гай Монтег" є чітке розмежування функціоналу для клієнтів (перегляд книг, створення замовлень, написання відгуків) та менеджерів (керування каталогом, обробка замовлень). Наприклад, клієнти можуть взаємодіяти з кошиком та залишати відгуки через форми (ReviewForm, OrderCreateForm), тоді як менеджери мають доступ до адміністративної панелі для редагування книг та категорій (manager_book_create, manager_category_update) [13]. Такий підхід забезпечує адаптивність інтерфейсу для різних груп користувачів, що підвищує зручність використання системи та сприяє ефективній взаємодії між клієнтами, менеджерами та адміністраторами.

Масштабованість та продуктивність є критично важливими для систем управління інтернет-магазинами, особливо з огляду на зростання обсягів даних та кількості користувачів. У проєкті "Гай Монтег" використовується SQLite як база даних, що підходить для невеликих проєктів, але для

масштабування до більших магазинів необхідні потужніші рішення, такі як PostgreSQL або хмарні сервіси [14]. Пагінація у списку книг (book_list) та оптимізовані запити до бази даних (наприклад, Q-об’єкти для пошуку) дозволяють ефективно обробляти великі обсяги даних. Масштабованість досягається завдяки модульній архітектурі Django, яка дозволяє додавати нові функціональні модулі, такі як чат між клієнтами та менеджерами (chat, messages_list), без значних змін у коді.

Останньою ключовою особливістю є орієнтація на постійне вдосконалення та адаптацію до змін у потребах користувачів та ринкових умовах. Системи управління інтернет-магазинами повинні бути гнучкими, щоб швидко впроваджувати нові функції, такі як інтеграція з аналітичними інструментами, персоналізовані рекомендації чи підтримка нових платіжних методів. У проєкті "Гай Монтег" це проявляється у можливості автоматичного створення slug для книг та категорій (create_slug), що спрощує SEO-оптимізацію, та у використанні JavaScript для динамічних ефектів, таких як оновлення лічильника кошика (updateCartCounter) [15]. Гнучкість системи дозволяє легко адаптувати її до нових вимог, таких як додавання нових категорій чи інтеграція з зовнішніми API.

Отже, принципи побудови систем управління інтернет-магазинами зумовлені необхідністю забезпечення зручного клієнтського досвіду, безпеки даних, адаптивності до потреб різних груп користувачів, масштабованості та гнучкості для впровадження нових функцій. Системи, подібні до "Гай Монтег", інтегруються з платіжними та логістичними сервісами, гарантують безпеку даних, адаптуються до різних ролей користувачів, забезпечують високу продуктивність та залишаються відкритими до оновлень. Ці принципи дозволяють створювати ефективні інструменти для управління інтернет-магазинами, які сприяють підвищенню якості обслуговування, оптимізації бізнес-процесів та задоволенню потреб клієнтів у динамічному середовищі електронної комерції.

1.3 Аналітичний огляд відомих систем інтернет-магазинів

На сьогодні ринок платформ для створення інтернет-магазинів пропонує широкий вибір рішень, які різняться за функціональністю, вартістю та адаптивністю до потреб бізнесу. Вибір відповідної системи залежить від масштабу магазину, цільової аудиторії, бюджету та технічних вимог. У цьому розділі розглянуто п’ять популярних систем для створення інтернет-магазинів: Shopify, WooCommerce, Magento, BigCommerce та OpenCart. Кожна з них має свої переваги та недоліки, що впливають на їх придатність для різних типів бізнесу.

Shopify – це хмарна платформа, яка є однією з найпопулярніших для створення інтернет-магазинів завдяки простоті використання та широкому набору функцій. Вона підходить як для невеликих, так і для великих бізнесів, пропонуючи зручний інтерфейс для керування товарами, замовленнями та клієнтами.

Інтерфейс платформи зображений на рисунку 1.1.

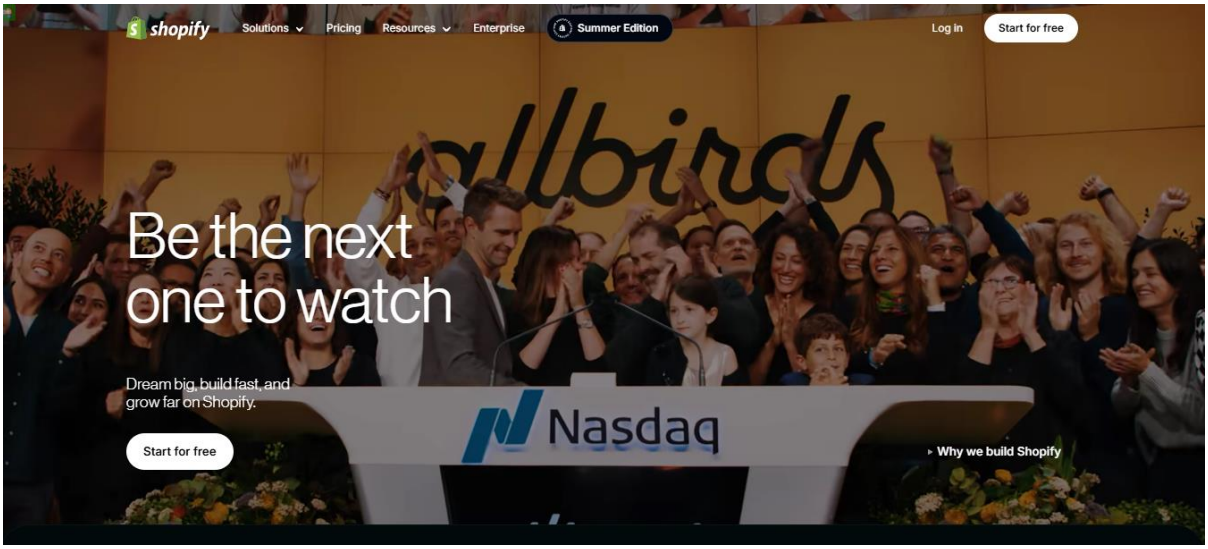

Рисунок 1.1 – Загальний вигляд інтерфейсу Shopify

Shopify підтримує інтеграцію з платіжними системами, маркетинговими інструментами та соціальними мережами, а також має велику бібліотеку

шаблонів для адаптивного дизайну [16]. Крім того, платформа пропонує мобільні додатки для керування магазином із будь-якого пристрою. Однак для доступу до розширених функцій, таких як детальна аналітика чи додаткові інтеграції, потрібна підписка на преміум-плани, що може бути дорогим для невеликих бізнесів. Деякі користувачі також зазначають обмеження у гнучкості кастомізації порівняно з платформами з відкритим кодом.

WooCommerce – це плагін із відкритим кодом для WordPress, який перетворює звичайний сайт на повноцінний інтернет-магазин.

Інтерфейс платформи зображений на рисунку 1.2.

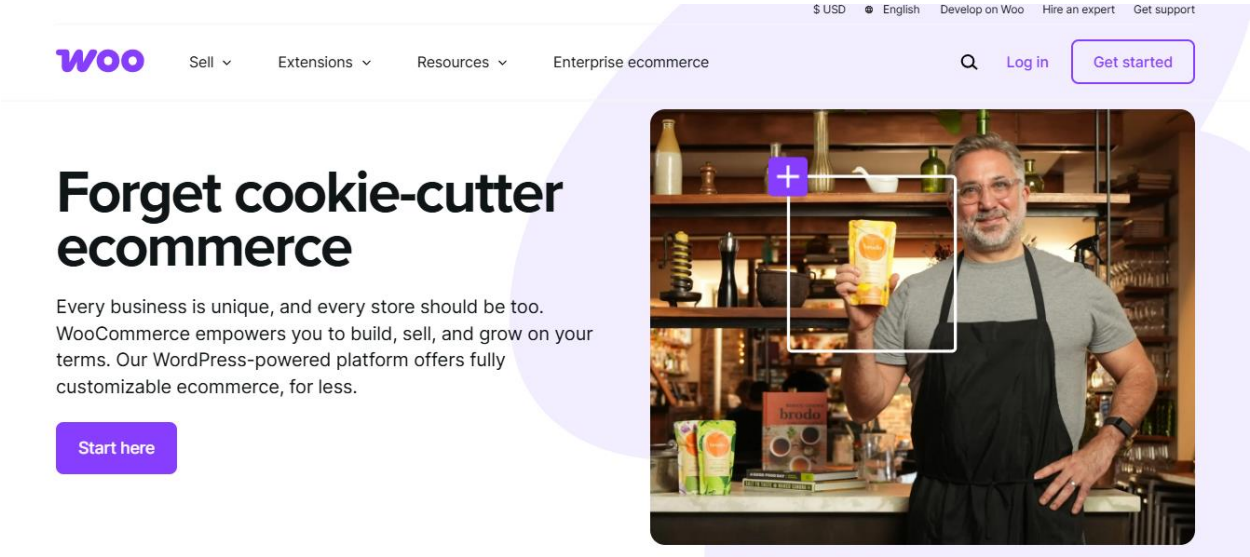

Рисунок 1.2 – Загальний вигляд інтерфейсу WooCommerce

Завдяки інтеграції з WordPress, WooCommerce пропонує високу гнучкість у дизайні та функціональності, що робить його привабливим для користувачів, які вже працюють із цією CMS [17]. Плагін безкоштовний, але додаткові розширення, такі як платіжні шлюзи чи інструменти SEO, можуть вимагати додаткових витрат. WooCommerce підходить для малого та середнього бізнесу, однак потребує технічних знань для налаштування та підтримки, особливо якщо потрібна складна кастомізація. Недоліком є потенційна вразливість до безпеки, якщо не оновлювати плагіни та WordPress регулярно.

Magento – це потужна платформа з відкритим кодом, орієнтована на великі інтернет-магазини з великим асортиментом товарів. Вона пропонує розширений функціонал для керування каталогами, замовленнями, клієнтами та аналітикою, а також підтримує багатомовність і мультивалютність [18].

Інтерфейс платформи зображений на рисунку 1.3.

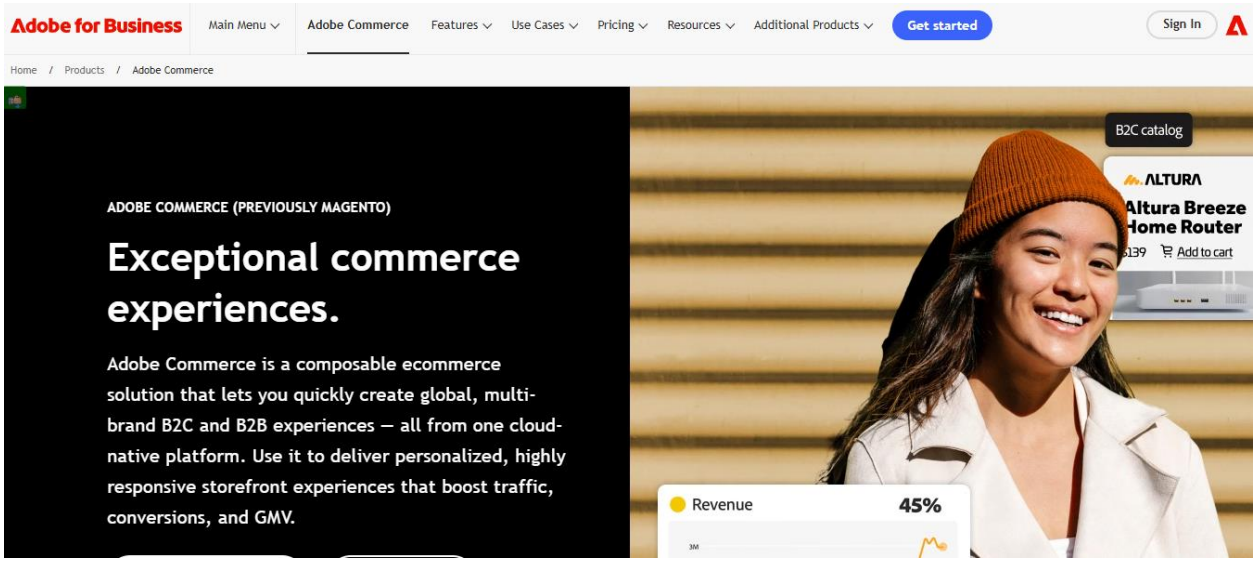

Рисунок 1.3 – Загальний вигляд інтерфейсу Magento

Magento відома своєю масштабністю та можливістю інтеграції з ERP-системами, що робить її ідеальною для великих компаній. Проте складність налаштування та висока вимогливість до серверних ресурсів роблять її менш доступною для невеликих бізнесів. Крім того, для ефективного використання Magento часто потрібна допомога професійних розробників, що збільшує витрати.

BigCommerce – ще одна хмарна платформа, яка пропонує комплексне рішення для створення інтернет-магазинів. Вона включає інструменти для керування товарами, маркетингу, SEO та аналітики, а також підтримує інтеграцію з популярними платіжними системами та маркетплейсами, такими як Amazon і eBay [19]. BigCommerce вирізняється простотою використання та можливістю швидкого запуску магазину, що робить її привабливою для середнього бізнесу. Однак, як і Shopify, платформа може бути дорогою через

платні плани, а деякі користувачі скаржаться на обмеження в дизайні шаблонів порівняно з платформами з відкритим кодом. Інтерфейс платформи зображений на рисунку 1.4.

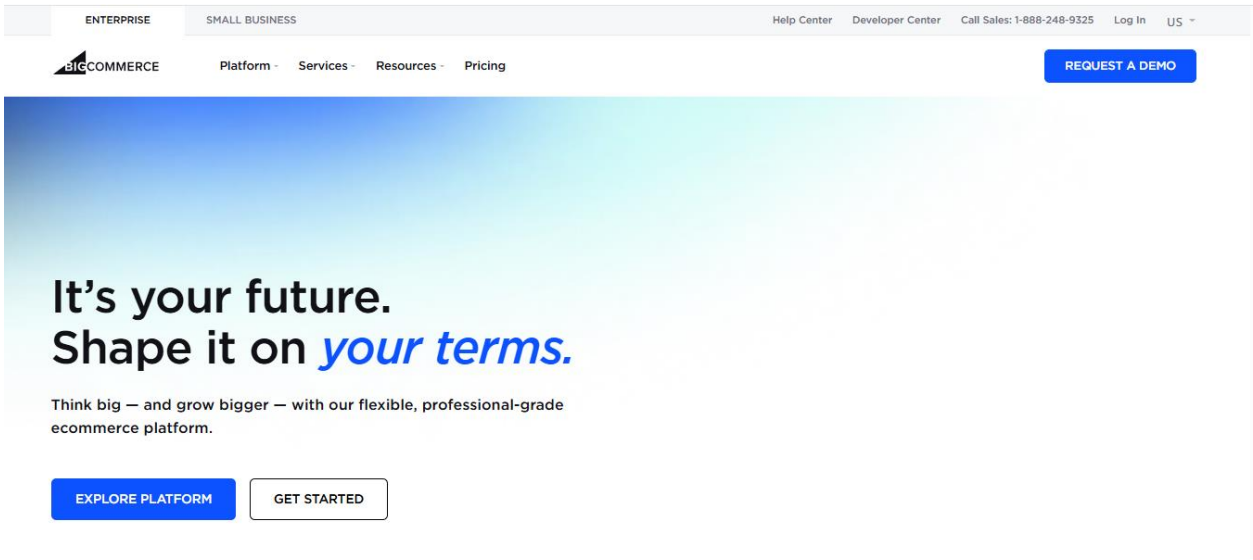

Рисунок 1.4 – Загальний вигляд інтерфейсу BigCommerce

OpenCart – це платформа з відкритим кодом, яка підходить для малого та середнього бізнесу завдяки своїй простоті та доступності. Вона дозволяє легко створювати та керувати інтернет-магазином із підтримкою багатомовності, мультивалютності та базового функціоналу для обробки замовлень [20].

OpenCart має активну спільноту розробників, яка пропонує численні модулі та теми для розширення можливостей. Проте платформа може бути обмеженою для великих магазинів із складними вимогами, а підтримка та оновлення можуть залежати від якості встановлених модулів. Крім того, для безпеки необхідно регулярно оновлювати систему.

Інтерфейс платформи зображений на рисунку 1.5.

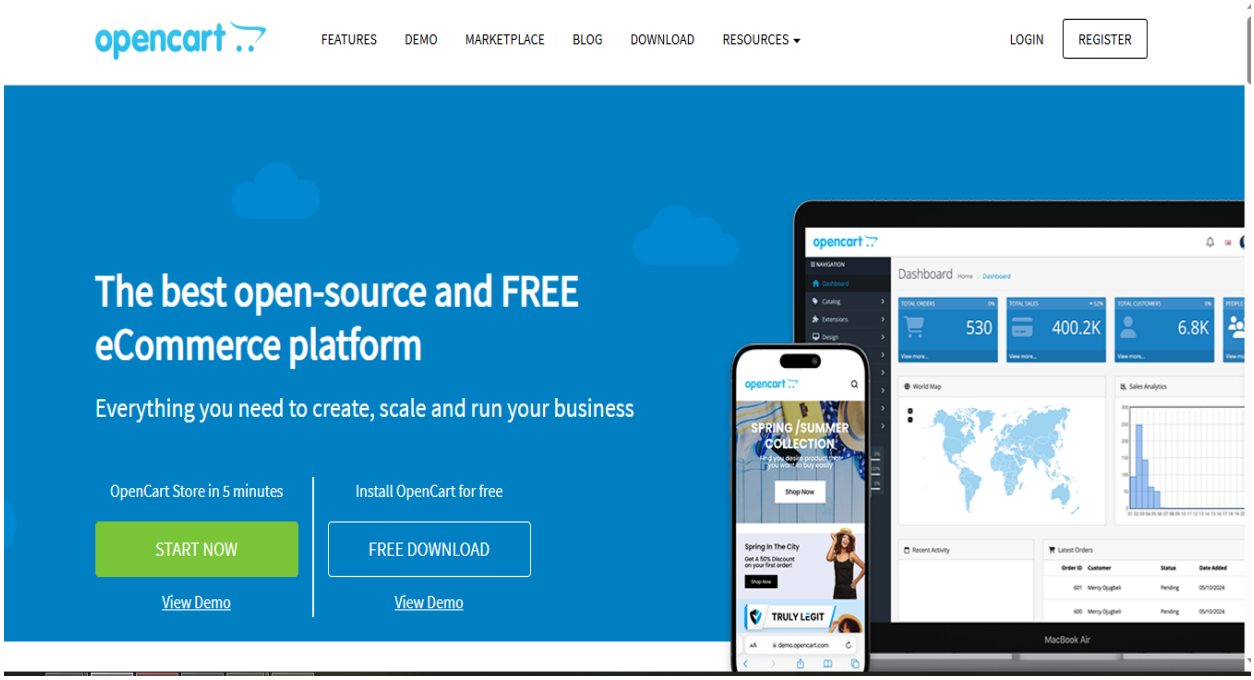

Рисунок 1.5 – Загальний вигляд інтерфейсу OpenCart

Отже, аналіз сучасних систем для створення інтернет-магазинів показує, що кожна платформа має свої сильні та слабкі сторони. Для невеликих бізнесів із обмеженим бюджетом підійдуть Shopify або OpenCart, тоді як Magento та BigCommerce краще відповідають потребам великих компаній. WooCommerce є універсальним рішенням для тих, хто вже використовує WordPress, але вимагає технічної підтримки. Таким чином, існує потреба в розробці програмного забезпечення, яке б поєднувало простоту використання, гнучкість кастомізації та доступну ціну, щоб усунути обмеження наявних систем.

1.4 Висновок до розділу 1

Аналіз предметної області свідчить про складність впровадження WEB-додатків для інтернет-магазинів, що зумовлена необхідністю інтеграції з різноманітними системами, забезпечення безпеки даних, масштабованості та зручності використання. Системи управління, подібні до проєкту "Гай Монтег", демонструють ефективність використання фреймворків, таких як

Django, для автоматизації бізнес-процесів та створення адаптивного інтерфейсу. Водночас сучасні платформи, як Shopify, WooCommerce чи Magento, пропонують різноманітні рішення, але мають обмеження у гнучкості, вартості чи складності налаштування. Розробка програмного забезпечення, яке поєднує простоту, безпеку та масштабованість, залишається актуальною потребою для підвищення конкурентоспроможності бізнесу в електронній комерції.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО МОДУЛЮ WEB-ДОДАТКУ ІНТЕРНЕТ-МАГАЗИНУ

2.1 Розробка структури роботи WEB-додатку інтернет-магазину

Перевіривши вхідні дані для WEB-додатку інтернет-магазину "Гай Монтег" та дослідивши дані, що будуть зберігатися, розроблено структуру додатку, яка зображена на рисунку 2.1.

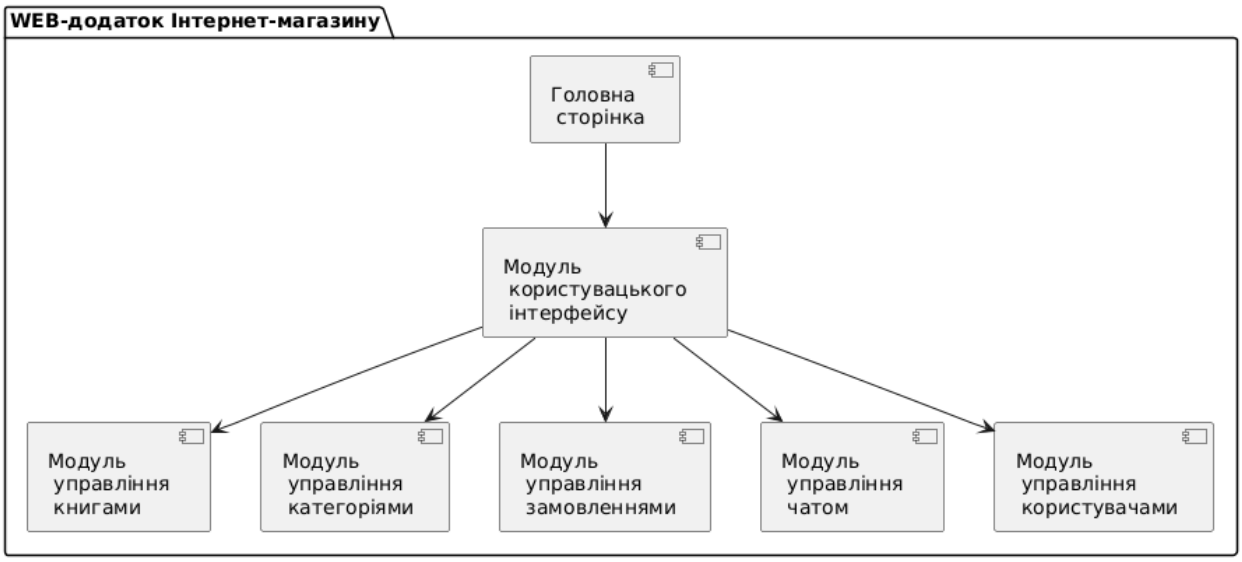

Рисунок 2.1 – Структурна схема WEB-додатку інтернет-магазину "Гай Монтег"

Точкою входу в програму є головна сторінка, яка забезпечує початкову навігацію користувача між різними розділами додатку, такими як каталог книг, кошик, оформлення замовлень, чат із менеджером та панель керування для адміністраторів.

Модуль користувацького інтерфейсу є ключовим компонентом, що відповідає за взаємодію з користувачем та представлення інформації у зручній і зрозумілій формі. Його основна роль полягає у створенні інтуїтивного візуального середовища, яке забезпечує комфортну роботу з додатком [21]. Цей модуль включає розробку графічних елементів, таких як меню, кнопки,

форми пошуку, списки книг, кошик, діалогові вікна чату та панель керування менеджера. Ці елементи дозволяють користувачам переглядати книги, здійснювати пошук, додавати товари до кошика, оформлювати замовлення та взаємодіяти з менеджерами через чат.

Модуль користувацького інтерфейсу забезпечує зручну навігацію між розділами додатку, такими як каталог, сторінки окремих книг, кошик, історія замовлень та чат. Він відповідає за логічне представлення інформації, враховуючи принципи юзабіліті, адаптивності та естетичності. Наприклад, адаптивний дизайн, реалізований у файлі style.css, забезпечує коректне відображення на різних пристроях, зокрема зменшення висоти паралакс-контейнера для мобільних пристроїв [22].

Крім того, модуль обробляє введені користувачем дані, такі як пошукові запити, відгуки чи інформацію для оформлення замовлення, проводить їх валідацію (наприклад, через форми в forms.py) та передає до відповідних модулів для подальшої обробки. Він також відображає результати операцій, такі як підтвердження додавання книги до кошика чи оновлення статусу замовлення, у зрозумілому вигляді. Модуль тісно взаємодіє з іншими компонентами додатку, зокрема модулями управління книгами, замовленнями, користувачами та чатом, забезпечуючи зв’язок між користувачем і бізнес-логікою системи. Ефективний користувацький інтерфейс є критично важливим для забезпечення позитивного досвіду користувача та підвищення ефективності роботи з додатком.

Модуль управління книгами відповідає за роботу з даними про книги, які є основним товаром інтернет-магазину. Його функціональність зображена на рисунку 2.2.

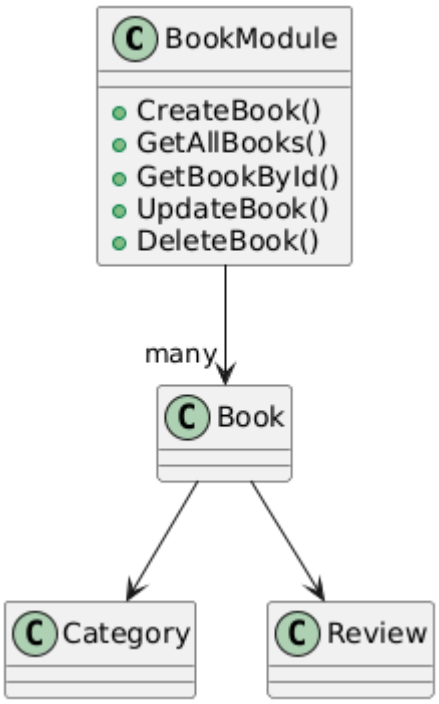

Рисунок 2.2 – Типова схема модуля управління книгами

Модуль містить наступні функції, реалізовані у файлі views.py:

1) CreateBook, що дозволяє менеджерам створювати нові записи про книги через форму BookForm, зберігаючи дані, такі як назва, автор, категорія, ціна, ISBN, у таблиці Book [23].

2) GetAllBooks, що повертає список усіх книг для відображення в каталозі чи на панелі менеджера (manager_book_list).

3) GetBookById, що отримує деталі конкретної книги за її slug для перегляду на сторінці книги (book_detail).

4) UpdateBook, що дозволяє оновлювати дані про книгу, включаючи завантаження зображень, через форму редагування.

5) DeleteBook, що видаляє книгу за її ідентифікатором із підтвердженням через шаблон book_confirm_delete.html.

Цей модуль взаємодіє з модулем управління категоріями для прив’язки книг до відповідних категорій та з модулем відгуків для відображення оцінок і коментарів користувачів. Дані про книги зберігаються в таблиці Book, а їх обробка забезпечується транзакціями у функції create_books у файлі seed.py.

Модуль управління категоріями відповідає за роботу з категоріями книг, що дозволяє групувати товари за жанрами. Його функціональність зображена на рисунку 2.3.

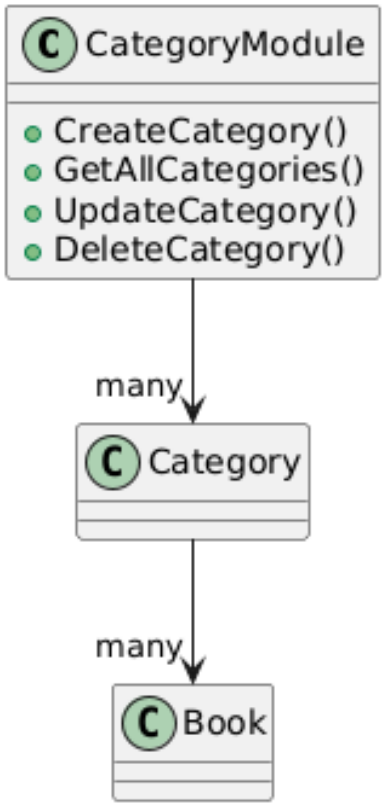

Рисунок 2.3 – Типова схема модуля управління категоріями

Модуль включає наступні функції:

1) CreateCategory, яка створює нові категорії через форму CategoryForm, зберігаючи назву, slug та опис у таблиці Category.

2) GetAllCategories, яка повертає список усіх категорій для відображення в меню чи на панелі менеджера.

3) UpdateCategory, яка оновлює дані категорії через форму редагування.

4) DeleteCategory, яка видаляє категорію з підтвердженням, враховуючи пов’язані книги.

Модуль взаємодіє з модулем управління книгами, забезпечуючи коректне групування товарів. Автоматична генерація slug для категорій реалізована у файлі category_form.html через JavaScript [13].

Модуль управління замовленнями відповідає за обробку замовлень користувачів. Його робота зображена на рисунку 2.4.

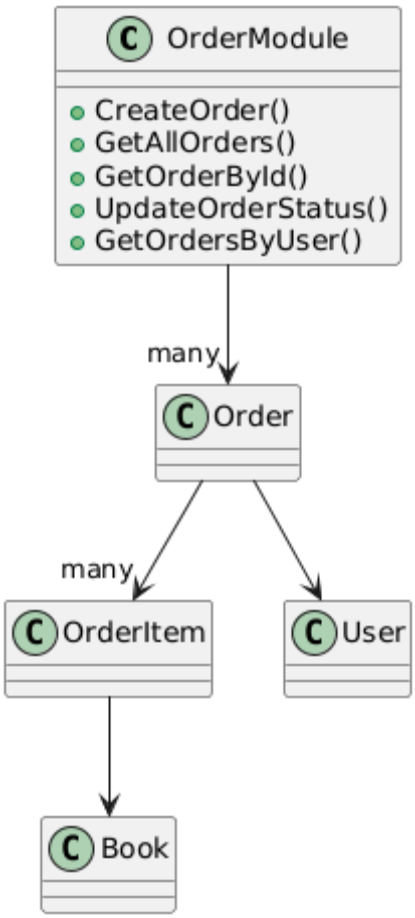

Рисунок 2.4 – Типова схема модуля управління замовленнями

Основні функції модуля:

1) CreateOrder, створює нове замовлення через форму OrderCreateForm, зберігаючи дані про клієнта, товари та статус у таблицях Order та OrderItem.

2) GetAllOrders, що повертає список усіх замовлень для менеджера (manager_order_list).

3) GetOrderById, що отримує деталі замовлення для перегляду (order_detail).

4) UpdateOrderStatus, що оновлює статус замовлення (наприклад, з "new" на "shipped") через форму в order_detail.html.

5) GetOrdersByUser, що повертає історію замовлень конкретного користувача (order_list).

Модуль взаємодіє з модулями управління книгами (для отримання даних про товари) та користувачами (для прив’язки замовлення до клієнта). Дані зберігаються в таблицях Order та OrderItem, створених у seed.py.

Модуль управління чатом забезпечує комунікацію між клієнтами та менеджерами. Його структура зображена на рисунку 2.5.

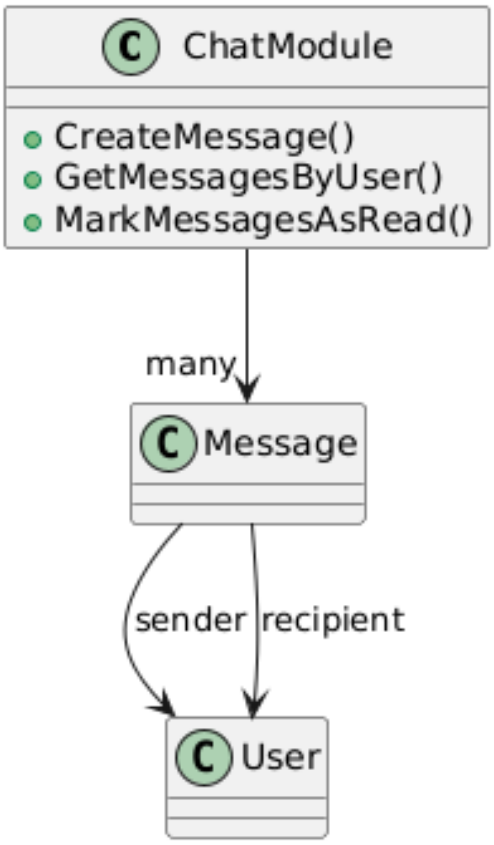

Рисунок 2.5 – Типова схема модуля управління чатом

Функції модуля:

1) CreateMessage, що створює нове повідомлення через форму MessageForm, зберігаючи його в таблиці Message.

2) GetMessagesByUser, що отримує список повідомлень між двома користувачами для відображення в чаті (chat).

3) MarkMessagesAsRead, що позначає повідомлення як прочитані під час перегляду чату.

Модуль взаємодіє з модулем управління користувачами для визначення відправника та отримувача повідомлень. Стилі для чату, визначені в style.css, забезпечують зручне відображення повідомлень із різним оформленням для відправлених і отриманих текстів [24].

Модуль управління користувачами відповідає за автентифікацію та управління профілями. Його робота зображена на рисунку 2.6.

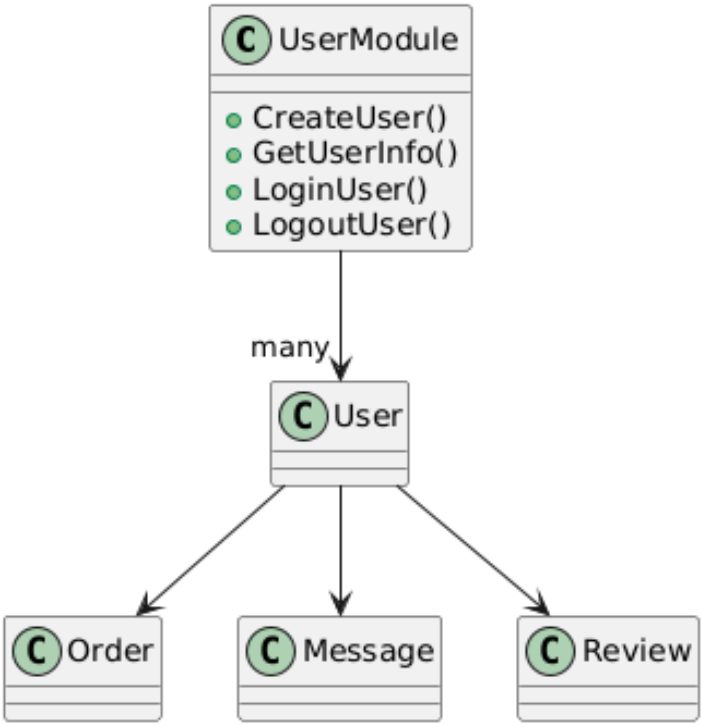

Рисунок 2.6 – Типова схема модуля управління користувачами

Функції модуля:

1) CreateUser, яка реєструє нових користувачів через форму UserRegistrationForm.

2) GetUserInfo, яка отримує дані авторизованого користувача для заповнення форм чи відображення профілю.

3) LoginUser, яка виконує вхід через форму UserLoginForm.

4) LogoutUser, яка завершує сеанс користувача.

Модуль використовує стандартну модель User Django та інтегрується з усіма іншими модулями для забезпечення доступу до функціональності.

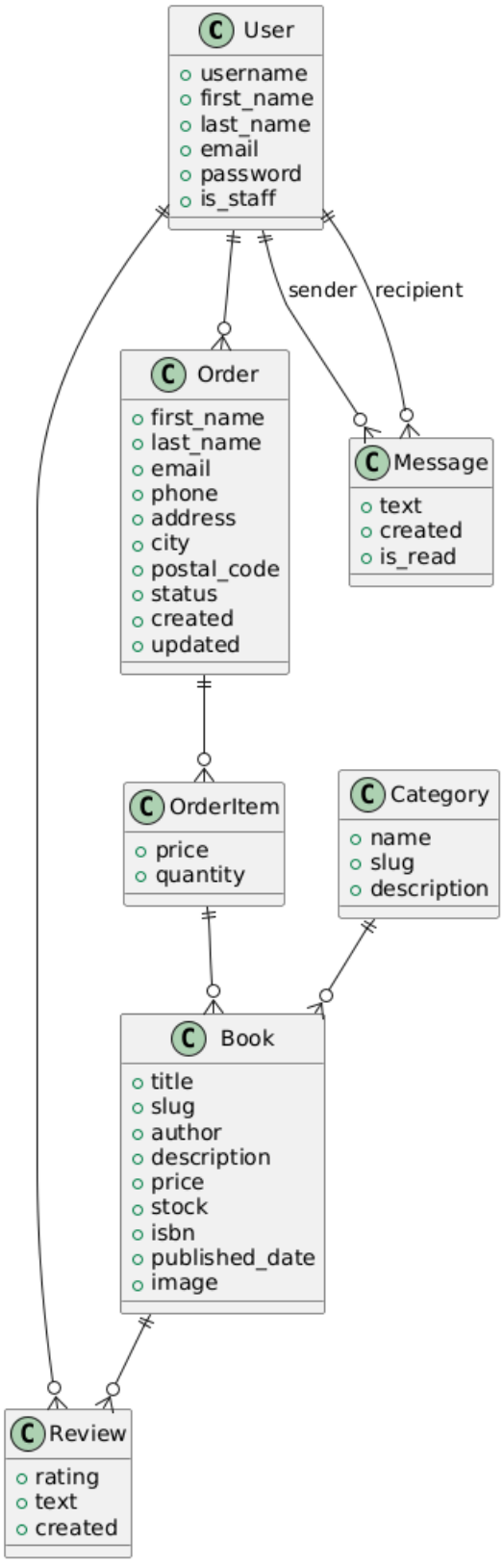

Рисунок 2.7 – UML-діаграма класів WEB-додатку

UML-діаграма відображає класи (Book, Category, Order, OrderItem, Review, Message, User) та їхні методи, а також зв’язки між ними, наприклад, асоціацію між Order і OrderItem чи Book і Category. Кожен модуль працює з відповідною таблицею бази даних, забезпечуючи цілісність даних.

Розроблена структура WEB-додатку є модульною, що забезпечує гнучкість і масштабованість. Взаємодія між модулями забезпечує узгодженість даних, наприклад, при оновленні книги автоматично оновлюються пов’язані замовлення. Така структура дозволяє легко додавати нові функції, наприклад, модуль рекомендацій книг, без впливу на існуючу функціональність. Загалом, представлена структура забезпечує надійну основу для ефективного управління інтернет-магазином, оптимізуючи процеси купівлі, адміністрування та комунікації.

2.2 Розробка алгоритму функціонування програмного модулю управління каталогом

Модуль управління каталогом у веб-додатку інтернет-магазину "Гай Монтег" відповідає за створення, редагування, перегляд та видалення категорій і книг у каталозі. Цей модуль забезпечує взаємодію між користувачем (менеджером), інтерфейсом веб-додатку, сховищем даних (базою даних SQLite) та серверною логікою Django для виконання операцій з каталогом. Алгоритм роботи модуля реалізовано через відповідні представлення (views), моделі та шаблони, що забезпечують зручне керування каталогом [25].

Діаграма послідовності, яка ілюструє роботу модуля управління каталогом, представлена як UML-діаграма послідовності (рис. 2.8). Вона відображає взаємодію між компонентами системи в часі, демонструючи послідовність дій та обмін даними між менеджером, інтерфейсом, сервером Django та базою даних.

Ось детальний опис роботи модуля управління каталогом:

1) Менеджер, авторизований як користувач із правами персоналу, переходить до панелі керування через веб-інтерфейс, обираючи розділ управління каталогом (книги або категорії) [26].

2) Інтерфейс надсилає запит до серверної частини Django (через URL-адреси, визначені в urls.py), викликаючи відповідне представлення, наприклад, manager_book_list або manager_category_list [27].

3) Представлення звертається до моделі (Book або Category) через ORM Django, запитуючи дані з бази даних SQLite про наявні книги або категорії [28].

4) База даних повертає запитані дані (список книг або категорій) до представлення, яке формує відповідь у вигляді HTML-шаблону (book_list.html або category_list.html) [29].

5) Для створення або редагування запису менеджер заповнює форму (BookForm або CategoryForm) у відповідному шаблоні (book_form.html або category_form.html). Після підтвердження (натискання кнопки "Зберегти") дані форми надсилаються на сервер через POST-запит.

6) Серверна логіка в представленні (manager_book_create, manager_book_update, manager_category_create або manager_category_update) перевіряє валідність даних форми. У разі успіху дані зберігаються в базі даних через ORM, а менеджеру відображається повідомлення про успішне виконання операції.

7) Для видалення запису менеджер обирає відповідну дію в інтерфейсі, що викликає представлення (manager_book_delete або manager_category_delete). Після підтвердження через POST-запит запис видаляється з бази даних, а менеджер отримує повідомлення про успіх.

8) Усі зміни в базі даних оновлюють стан каталогу, а інтерфейс відображає актуальні дані, отримані через повторний запит до сервера.

Така послідовність дій забезпечує ефективне керування каталогом, дозволяючи менеджеру створювати, редагувати, переглядати та видаляти

записи. Використання Django ORM забезпечує централізоване керування даними, а шаблони з адаптивним дизайном полегшують взаємодію з інтерфейсом на різних пристроях [27]. Автоматична генерація slug (через JavaScript у book_form.html і category_form.html) спрощує створення URL-адрес для нових записів, підвищуючи зручність використання [29].

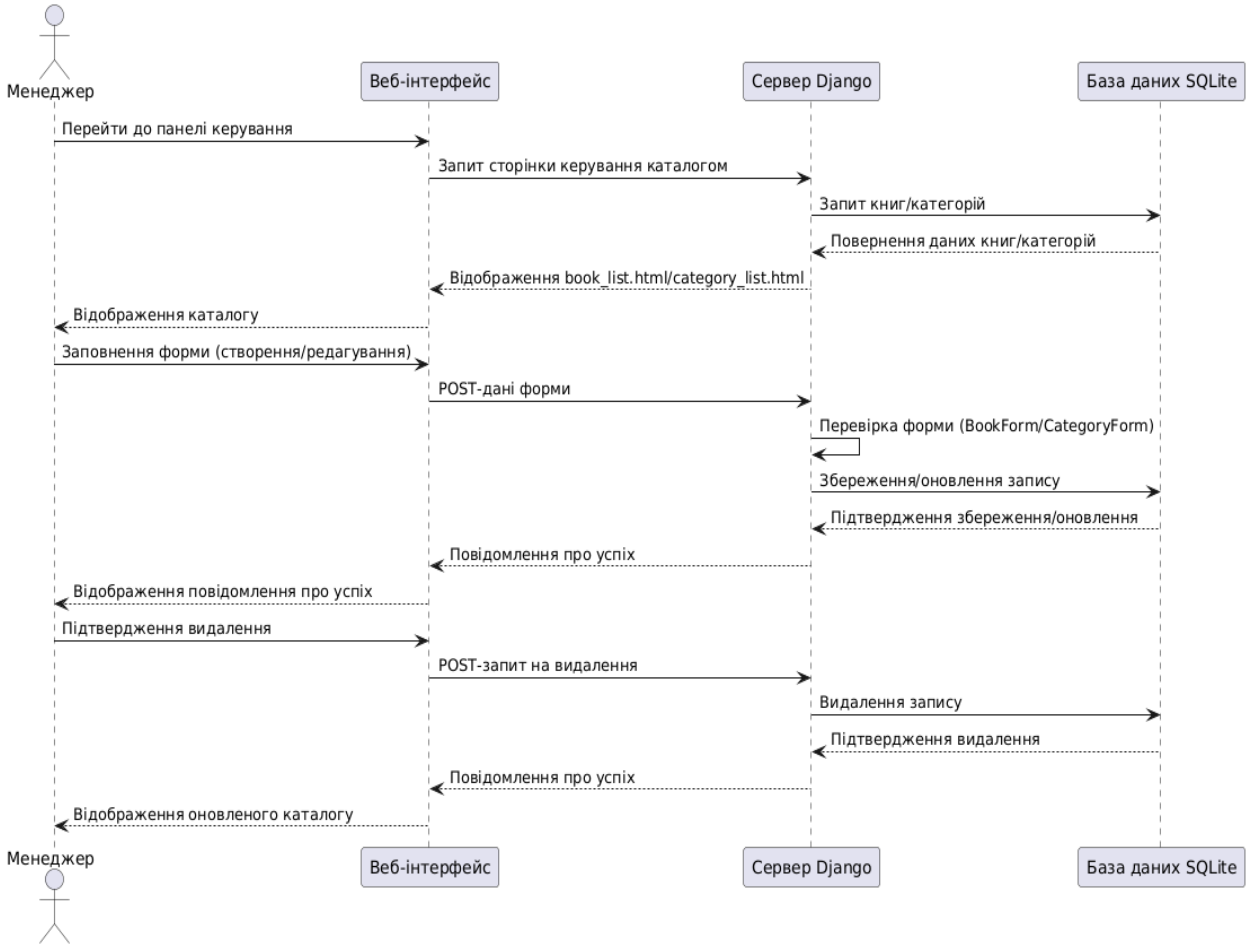

Рисунок 2.8 – UML-діаграма послідовностей до модуля управління каталогом

Ця діаграма наочно ілюструє послідовність взаємодій між менеджером, веб-інтерфейсом, сервером Django та базою даних, демонструючи чіткий обмін даними під час виконання операцій з каталогом.

2.3 Проєктування структури бази даних програмного модулю WEB-додатку інтернет-магазину

Для розробки WEB-додатку інтернет-магазину «Гай Монтег» було проаналізовано кілька варіантів баз даних, зокрема реляційні (MySQL, PostgreSQL) та NoSQL (MongoDB, Firebase). Реляційні бази даних, такі як MySQL і PostgreSQL, забезпечують чітку структуру даних, підтримують транзакції ACID (атомарність, консистентність, ізоляція, довговічність) і гарантують цілісність завдяки зовнішнім ключам та обмеженням. Вони ідеально підходять для структурованих даних, таких як інформація про книги, замовлення та користувачів. Однак реляційні бази можуть бути менш гнучкими при змінах схеми даних і потребують додаткових зусиль для масштабування при великих обсягах даних [30].

NoSQL бази, такі як MongoDB, пропонують гнучкість у роботі з неструктурованими даними та горизонтальну масштабованість, що може бути корисним для додатків із мінливими вимогами. Проте вони часто поступаються реляційним базам у підтримці транзакцій і забезпеченні цілісності даних [31].

Після аналізу вимог додатку, який передбачає чітко структуровані дані (книги, категорії, замовлення, відгуки, повідомлення), було обрано реляційну базу даних SQLite. SQLite є легкою, вбудованою базою даних, яка не потребує окремого серверного процесу, що спрощує розробку та розгортання, особливо на етапі тестування та для невеликих проєктів [32].

Вибір SQLite обґрунтовано його простотою використання, оскільки ця база даних не вимагає складного налаштування серверів чи адміністрування, що робить її ідеальним варіантом для локального тестування та невеликих проєктів, таких як книжковий інтернет-магазин. Завдяки високій продуктивності SQLite забезпечує швидкий доступ до даних у додатках із помірною кількістю запитів, що повністю відповідає потребам магазину, який містить кілька тисяч записів про книги та замовлення [33].

Однією з важливих особливостей є підтримка транзакцій ACID, що гарантує цілісність даних під час виконання критичних операцій, наприклад, створення замовлень або оновлення кількості книг на складі [32]. SQLite також легко інтегрується з Django, оскільки є стандартною базою даних цього фреймворку, що забезпечує зручне використання через ORM (Object-Relational Mapping) [34].

Перевагою є портативність, яка дозволяє без проблем переносити файл бази даних між різними системами, що робить процес розробки та тестування більш гнучким [33]. Крім того, використання SQLite сприяє економії ресурсів, адже ця база даних не потребує значних апаратних потужностей, що допомагає знизити витрати на інфраструктуру для невеликого магазину [32].

Структура бази даних була спроєктована з урахуванням функціональних вимог додатку та предметної області книжкового магазину. Вона складається з п’яти основних таблиць, які відповідають ключовим сутностям системи: користувачі, категорії, книги, замовлення, відгуки та повідомлення. Кожну таблицю детально описано в таблиці 2.1.

Таблиця 2.1 – Опис структури бази даних

№	Поля таблиці	Опис
1	2	3
1	Users	таблиця для зберігання даних про користувачів (клієнтів і менеджерів)
1.1	id (Primary Key)	унікальний ідентифікатор користувача (ціле число)
1.2	username	унікальне ім’я користувача (рядок)
1.3	first_name	ім’я користувача (рядок)
1.4	last_name	прізвище користувача (рядок)
1.5	email	електронна пошта (рядок, унікальний)
1.6	password	хеш пароля (рядок)
1.7	is_staff	булеве значення, що вказує, чи є користувач менеджером
2	Categories	таблиця для зберігання категорій книг
2.1	id (Primary Key)	унікальний ідентифікатор категорії (ціле число)
2.2	name	назва категорії (рядок, наприклад, «Художня література»)

Продовження таблиці 2.1

1	2	3
2.3	slug	унікальний ідентифікатор для URL (рядок)
2.4	description	опис категорії (текст)
3	Books	таблиця для зберігання інформації про книги
3.1	id (Primary Key)	унікальний ідентифікатор книги (ціле число)
3.2	title	назва книги (рядок)
3.3	slug	унікальний ідентифікатор для URL (рядок)
3.4	author	автор книги (рядок)
3.5	category_id (Foreign Key)	посилання на категорію (зв'язок із таблицею Categories)
3.6	description	опис книги (текст)
3.7	price	ціна книги (десяткове число)
3.8	stock	кількість книг на складі (ціле число)
3.9	published_date	дата публікації (дата)
3.10	isbn	унікальний ISBN-код (рядок)
3.11	image	зображення обкладинки (файл, необов'язкове поле)
4	Orders	таблиця для зберігання даних про замовлення
4.1	id (Primary Key)	унікальний ідентифікатор замовлення (ціле число)
4.2	user_id (Foreign Key)	посилання на користувача (зв'язок із таблицею Users)
4.3	first_name	ім'я клієнта (рядок)
4.4	last_name	прізвище клієнта (рядок)
4.5	email	електронна пошта (рядок)
4.6	phone	номер телефону (рядок)
4.7	address	адреса доставки (рядок)
4.8	city	місто доставки (рядок)
4.9	postal_code	поштовий індекс (рядок)
4.10	status	статус замовлення (рядок, наприклад, «new», «delivered»)
4.11	created	дата створення (дата й час)
4.12	updated	дата оновлення (дата й час)
5	OrderItems	таблиця для зберігання елементів замовлення
5.1	id (Primary Key)	унікальний ідентифікатор елемента (ціле число)
5.2	order_id (Foreign Key)	посилання на замовлення (зв'язок із таблицею Orders)
5.3	book_id (Foreign Key)	посилання на книгу (зв'язок із таблицею Books)
5.4	price	ціна книги на момент замовлення (десяткове число)
5.5	quantity	кількість одиниць книги (ціле число)

Завершення таблиці 2.1

1	2	3
6	Reviews	таблиця для зберігання відгуків про книги
6.1	id (Primary Key)	унікальний ідентифікатор відгуку (ціле число)
6.2	book_id (Foreign Key)	посилання на книгу (зв’язок із таблицею Books)
6.3	user_id (Foreign Key)	посилання на користувача (зв’язок із таблицею Users)
6.4	rating	оцінка книги (ціле число від 1 до 5)
6.5	text	текст відгуку (текст)
6.6	created	дата створення відгуку (дата й час)
7	Messages	таблиця для зберігання повідомлень між клієнтами та менеджерами
7.1	id (Primary Key)	унікальний ідентифікатор повідомлення (ціле число)
7.2	sender_id (Foreign Key)	посилання на відправника (зв’язок із таблицею Users)
7.3	recipient_id (Foreign Key)	посилання на одержувача (зв’язок із таблицею Users)
7.4	text	текст повідомлення (текст)
7.5	created	дата створення (дата й час)
7.6	is_read	булеве значення, що вказує, чи прочитано повідомлення

Зв’язки між таблицями реалізуються через зовнішні ключі. Наприклад, поле category_id у таблиці Books посилається на id у таблиці Categories, що забезпечує зв’язок між книгою та її категорією. Аналогічно, order_id у таблиці OrderItems посилається на id у таблиці Orders, що дозволяє пов’язати елементи замовлення з відповідним замовленням.

Для забезпечення швидкого доступу до даних кожна таблиця має первинний ключ (id), який гарантує унікальність записів. Індекси на полях, таких як slug у таблицях Books і Categories, забезпечують швидкий пошук за URL-адресами.

Для управління користувачами та їхньою автентифікацією використовується вбудований модуль Django Authentication, який інтегрується з таблицею Users. Цей модуль забезпечує безпечне зберігання даних користувачів, управління паролями та авторизацію. Менеджери (користувачі

з атрибутом is_staff=True) мають доступ до адміністративних функцій, таких як керування книгами, категоріями та замовленнями, тоді як звичайні користувачі можуть переглядати книги, залишати відгуки та створювати замовлення.

Інтеграція SQLite із Django через ORM забезпечує зручне виконання CRUD-операцій (створення, читання, оновлення, видалення). Наприклад, функції у файлі views.py, такі як manager_book_create або order_create, використовують моделі Django для роботи з базою даних. Безпека забезпечується через перевірку прав доступу (наприклад, декоратор @login_required) та CSRF-токени для захисту форм.

На рисунку 2.9 зображено UML-діаграму класів бази даних, яка ілюструє зв’язки між таблицями.

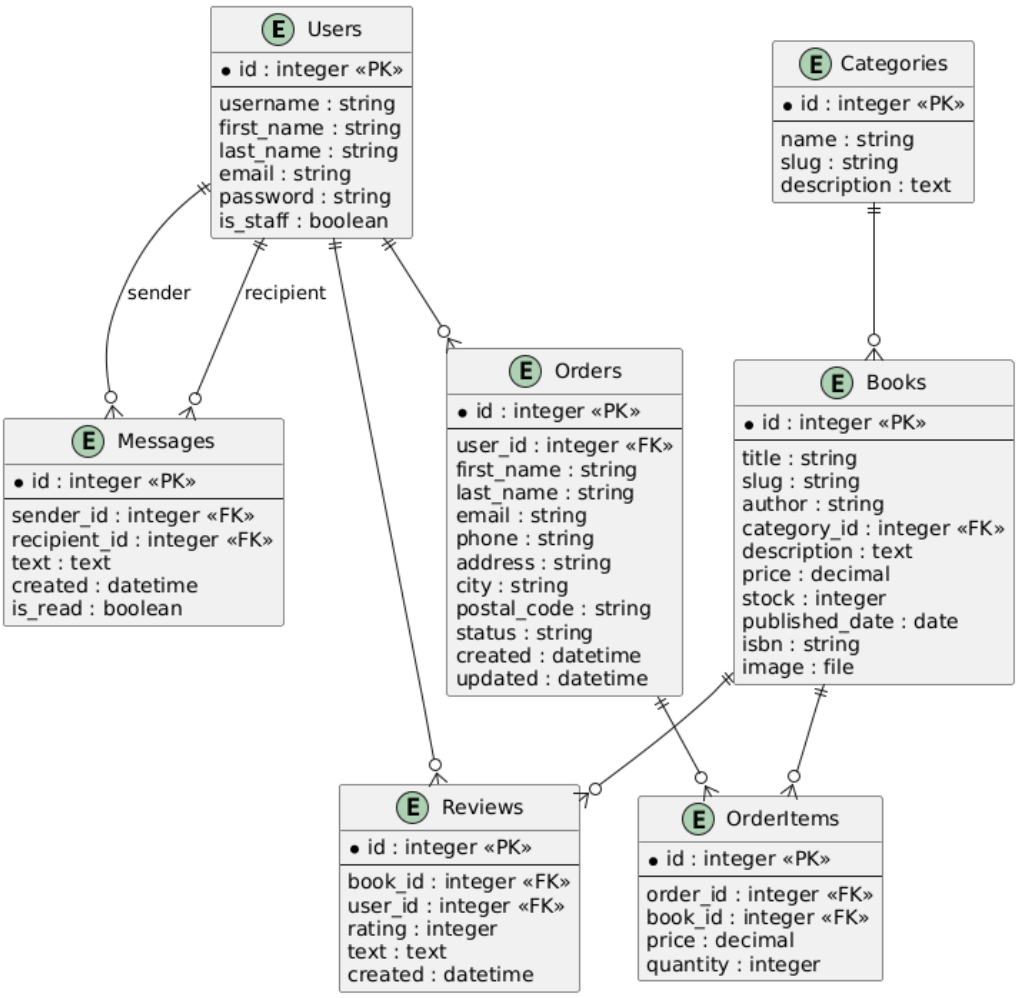

Рисунок 2.9 – UML-діаграма класів бази даних

Така структура бази даних забезпечує ефективне зберігання та керування даними книжкового магазину, підтримуючи ключові функції, такі як каталогізація книг, обробка замовлень, відгуки та комунікація між клієнтами й менеджерами. SQLite дозволяє легко масштабувати систему для невеликих проєктів, а використання Django ORM спрощує розробку та інтеграцію з іншими компонентами додатку.

2.4 Проєктування архітектури для розробки front-end частини WEB-додатку

При розробці front-end частини WEB-додатку інтернет-магазину книжкового магазину "Гай Монтег" ключовим завданням є вибір архітектури, яка забезпечить зручність використання, масштабованість і підтримку коду. У цьому розділі розглядаються основні архітектурні підходи до розробки front-end частини, обґрунтовується вибір оптимального підходу для проєкту та описується структура front-end компонентів.

Монолітна архітектура передбачає створення єдиної системи, де всі компоненти (HTML, CSS, JavaScript) тісно інтегровані та обробляються сервером, який генерує сторінки для клієнта. Цей підхід є простим у реалізації, оскільки вся логіка зосереджена в одному модулі, а розгортання не потребує складної координації [5].

Переваги монолітної архітектури:

- простота розробки та розгортання завдяки єдиній кодовій базі;
- легкість взаємодії між компонентами через спільний контекст;
- висока продуктивність за рахунок локального виконання логіки.

Недоліки монолітної архітектури:

- обмежена масштабованість окремих компонентів;
- складність внесення змін через тісну зв’язаність елементів;
- обмеження у виборі технологій для окремих частин додатку.

Сервіс-орієнтована архітектура (SOA) передбачає розбиття додатку на набір незалежних сервісів, які взаємодіють через мережеві протоколи (наприклад, HTTP). Кожен сервіс відповідає за окрему функціональність, що забезпечує модульність і можливість повторного використання [35].

Переваги SOA:

– модульність, що полегшує розробку та підтримку окремих сервісів;
– можливість використання різних технологій для різних сервісів;
– повторне використання сервісів у різних додатках.

Недоліки SOA:

– складність координації між сервісами через мережеві взаємодії;
– потенційні затримки через мережеві виклики;
– необхідність чіткого визначення протоколів взаємодії.

Мікросервісна архітектура є подальшим розвитком SOA, де додаток розбивається на дрібніші незалежні сервіси, кожен із яких виконує одну функцію та комунікує через легковагі протоколи. Цей підхід забезпечує високу гнучкість і масштабованість [36].

Переваги мікросервісної архітектури:

– незалежність розробки та розгортання кожного мікросервісу;
– гнучкість у виборі технологій для окремих компонентів;
– висока відмовостійкість завдяки ізоляції збоїв.

Недоліки мікросервісної архітектури:

– складність управління великою кількістю мікросервісів;
– підвищена складність розробки через розподіленість;
– необхідність забезпечення узгодженості даних.

Для реалізації front-end частини WEB-додатку книжкового магазину "Гай Монтег" обрано традиційну серверну архітектуру з використанням шаблонів Django та компонентного підходу на основі Bootstrap. Вибір Django та Bootstrap для розробки пояснюється кількома ключовими перевагами. Завдяки використанню шаблонів Django створення динамічних сторінок

відбувається з мінімальними зусиллями, забезпечуючи інтеграцію серверної логіки з front-end частиною. Це значно зменшує складність розробки проєкту середнього масштабу, зокрема книжкового магазину [25].

Bootstrap пропонує готові компоненти, такі як картки, таблиці та форми, що автоматично адаптуються до різних пристроїв, забезпечуючи необхідну кросбраузерність і відповідність принципам адаптивного дизайну. Це дозволяє створювати зручний інтерфейс для користувачів, незалежно від того, чи вони працюють на десктопах чи мобільних пристроях.

Ефективність розробки також значно підвищується завдяки можливості використовувати Bootstrap разом із JavaScript-скриптами, такими як scripts.js, що спрощує реалізацію інтерактивних елементів. Функціональність, пов’язана з додаванням книг до кошика, пагінацією чи сповіщеннями, може бути створена без необхідності впровадження складних клієнтських фреймворків.

Ще однією важливою особливістю є тісна інтеграція шаблонів Django із серверними представленнями через файл views.py. . Це забезпечує швидке відображення необхідних даних, включаючи інформацію про книги, замовлення та повідомлення, а також значно спрощує роботу з формами, такими як BookForm чи OrderCreateForm.

Запропонована структура front-end частини реалізована через модульний підхід із чітким розподілом функціональних елементів, що забезпечує ефективну організацію коду та зручність розробки. Директорія templates/app містить HTML-шаблони, такі як book_list.html, order_detail.html і dashboard.html, які відповідають за формування сторінок для різних функціональних частин інтерфейсу, включаючи каталог книг, кошик і панель менеджера. Директорія static містить файли стилів і скриптів, серед яких style.css визначає оформлення карток книг і чату, а scripts.js забезпечує інтерактивність завдяки доданню анімацій та обробці подій.

Шаблони Django використовують теги та фільтри, такі як {% load i18n %} і {% load app_tags %}, що дозволяє реалізувати локалізацію та кастомні функції, наприклад форматування цін або статусів замовлень. Додатково

застосовуються Bootstrap та Font Awesome, що сприяє створенню адаптивного дизайну й інтеграції іконок для зручності користування інтерфейсом. Ця структура забезпечує не лише модульність, а й простоту підтримки та розширення front-end частини проєкту.

Ця структура (рис. 2.10) забезпечує чітку організацію коду, полегшує його підтримку та дозволяє легко додавати нові шаблони чи стилі при розширенні функціональності.

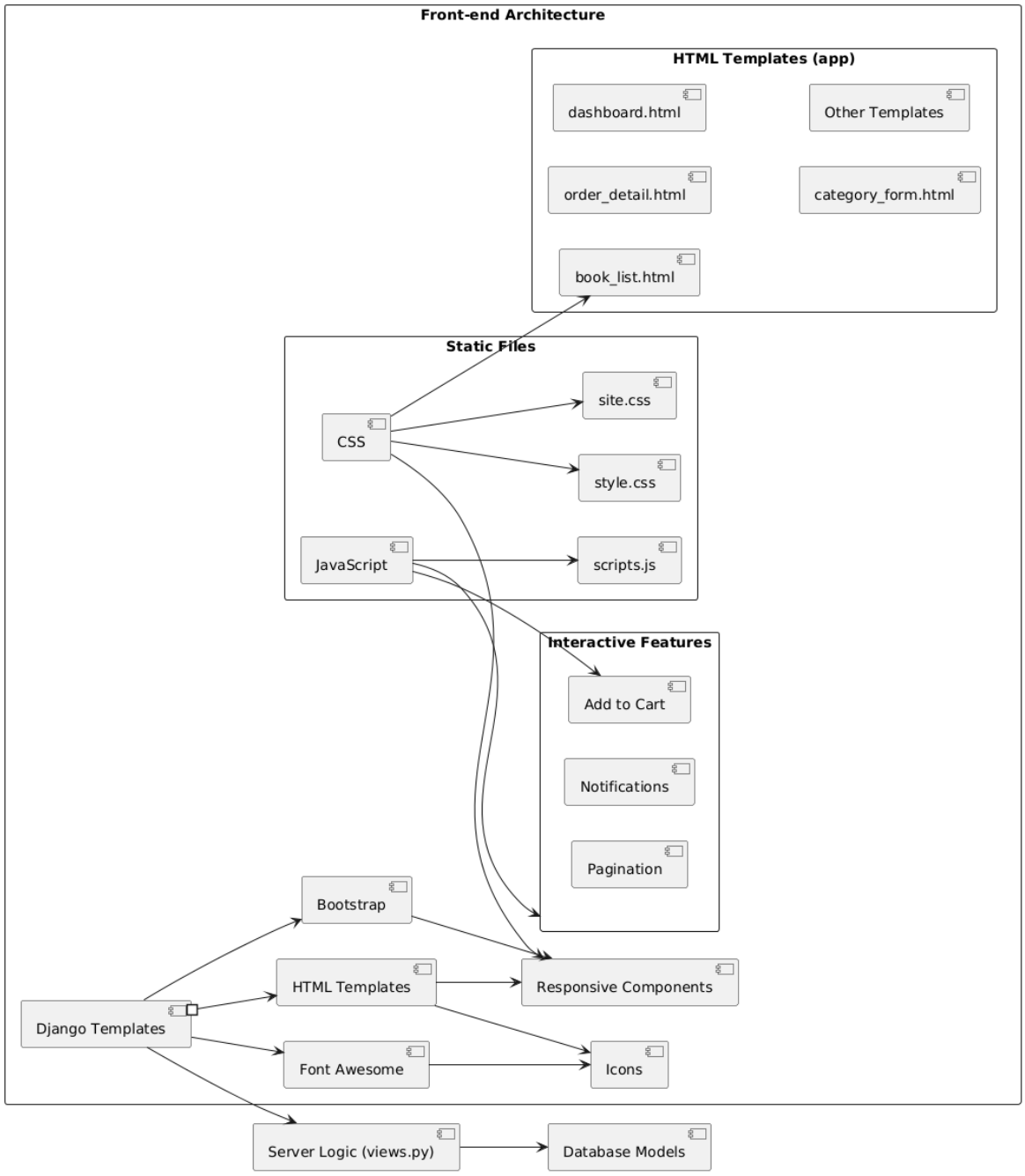

Рисунок 2.10 – Схема архітектури front-end частини WEB-додатку

Отже, обрана серверна архітектура з використанням шаблонів Django та компонентного підходу на основі Bootstrap є оптимальною для проєкту книжкового магазину "Гай Монтег". Вона забезпечує простоту розробки, адаптивність, інтеграцію з серверною логікою та можливість масштабування, що відповідає вимогам проєкту.

2.5 Висновок до розділу 2

Розроблена структура WEB-додатку інтернет-магазину «Гай Монтег» забезпечує модульність, гнучкість та ефективну взаємодію між компонентами, що сприяє зручному управлінню каталогом, замовленнями, користувачами та комунікацією. Вибір реляційної бази даних SQLite обґрунтовано її простотою, портативністю та підтримкою транзакцій ACID, що гарантує цілісність даних при виконанні операцій. Використання Django ORM та шаблонів у поєднанні з Bootstrap забезпечує адаптивний інтерфейс, швидку розробку та інтеграцію з серверною логікою. Запропонована архітектура фронтенду, заснована на серверних шаблонах, оптимізує відображення даних і спрощує підтримку, що робить додаток зручним для користувачів та менеджерів. Такий підхід створює надійну основу для масштабування та подальшого розвитку функціональності магазину.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЮ WEB-ДОДАТКУ ІНТЕРНЕТ-МАГАЗИНУ

3.1 Вибір інструментів розробки front-end частини програмного модуля

При розробці front-end частини програмного модуля веб-додатку інтернет-магазину "Гай Монтег" ключовим завданням було забезпечення зручного, інтуїтивно зрозумілого та адаптивного інтерфейсу користувача, який би відповідав сучасним стандартам веб-розробки. Для досягнення цієї мети було розглянуто кілька інструментів і фреймворків, які забезпечують ефективність розробки, гнучкість і масштабованість інтерфейсу. Після аналізу було обрано оптимальний набір технологій, які найкраще відповідають потребам проєкту.

Bootstrap – це популярний фреймворк із відкритим кодом, розроблений для створення адаптивних і мобільно-орієнтованих інтерфейсів. Bootstrap надає широкий набір готових CSS-стилів і JavaScript-компонентів, таких як кнопки, форми, модальні вікна, навігаційні панелі та пагінація, що значно прискорює розробку. Завдяки сітковій системі (grid system) фреймворк забезпечує легке створення адаптивних макетів, які коректно відображаються на пристроях із різними розмірами екранів. Крім того, Bootstrap підтримує кастомізацію стилів через змінні SCSS, що дозволяє адаптувати дизайн до потреб проєкту [37].

jQuery – це легка JavaScript-бібліотека, яка спрощує маніпуляції з DOM, обробку подій і асинхронні запити (AJAX). Вона дозволяє швидко реалізовувати інтерактивні елементи інтерфейсу, такі як динамічне оновлення кошика чи автоматичне заповнення полів форм. jQuery має просту синтаксичну структуру, що полегшує її використання навіть для розробників із базовим досвідом. Незважаючи на появу сучасніших бібліотек, jQuery

залишається актуальною для проєктів, які потребують швидкої реалізації інтерактивності без складних фреймворків [38].

Django Template Engine – це вбудований шаблонізатор фреймворку Django, який використовується для генерації HTML-сторінок на стороні сервера. Він дозволяє створювати динамічні шаблони з підтримкою умов, циклів і фільтрів, що спрощує відображення даних із бази даних. Django Template Engine забезпечує чітке розділення логіки та представлення, що сприяє підтримці чистого та зрозумілого коду. Його інтеграція з Django забезпечує безпечну обробку даних і захист від вразливостей, таких як XSS-атаки [39].

Враховуючи вимоги проєкту, для front-end частини обрано Django Template Engine у поєднанні з Bootstrap і jQuery, що забезпечує зручність розробки, адаптивність та інтерактивність інтерфейсу. Bootstrap дозволяє створювати адаптивний дизайн завдяки своїй сітковій системі та готовим компонентам, включаючи картки, таблиці та форми, які використовуються у шаблонах book_list.html, order_detail.html і dashboard.html. . Це гарантує коректне відображення сторінок на різних пристроях без необхідності написання складних медіа-запитів [37]. Наприклад, у style.css застосовані класи Bootstrap, що забезпечують адаптивність контейнера .parallax-container залежно від розміру екрана.

Інтерактивність реалізується через jQuery, який використано для створення динамічних елементів, таких як автоматичне оновлення кошика, фільтрація списків та анімація кнопок у файлі scripts.js. . Завдяки легкій інтеграції jQuery з Django Template Engine можна швидко додавати інтерактивні можливості до шаблонів без необхідності використання складніших фреймворків, таких як React або Vue.js [38]. Наприклад, функція updateCartCounter у scripts.js забезпечує оновлення лічильника кошика з анімацією, що покращує користувацький досвід.

Django Template Engine дозволяє ефективно працювати з даними, отриманими з бекенду, завдяки використанню вбудованих тегів і фільтрів,

таких як {% for %} і {% trans %}, що зустрічаються у шаблонах category_list.html і book_form.html. Це забезпечує швидке створення динамічних сторінок, таких як списки книг і категорій, а також автоматичну валідацію форм [39]. Наприклад, у category_form.html застосовується фільтр add_class, що додає стилі Bootstrap до полів форми, підвищуючи зручність взаємодії користувачів із системою.

Завдяки використанню Bootstrap інтерфейс виглядає професійно та узгоджено, що забезпечує єдиний стиль для всіх його елементів, включаючи кнопки, таблиці та картки, як це реалізовано у шаблонах dashboard.html і order_list.html. Кастомізація через style.css, зокрема кольори .color-sky і .color-sunset, дозволяє адаптувати дизайн відповідно до бренду магазину, покращуючи загальну естетику платформи [37].

Для прискорення розробки та забезпечення високої якості інтерфейсу також використано бібліотеку Font Awesome для додавання іконок, що покращують візуальну складову, наприклад, у кнопках і навігаційних елементах (у dashboard.html) [40]. Крім того, для анімаційних ефектів, таких як паралакс на головній сторінці, використано бібліотеку Parallax.js, яка інтегрована в scripts.js для створення ефекту глибини на банері [41].

Для розробки front-end частини було обрано середовище Visual Studio Code як основне інтегроване середовище розробки (IDE). Visual Studio Code – це легкий, але потужний редактор коду з підтримкою JavaScript, HTML, CSS і Python, що ідеально підходить для роботи з Django та front-end технологіями. Основні переваги Visual Studio Code включають:

– інтелектуальне автодоповнення для HTML, CSS і JavaScript;

– вбудована підтримка Git для контролю версій;

– розширення, такі як Django Template і Python, для зручної роботи з шаблонами та бекендом;

– інтеграція з терміналом для виконання команд Django, таких як python manage.py runserver;

– легкість налаштування та підтримка плагінів для форматування коду й автотестування.

Завдяки поєднанню Django Template Engine, Bootstrap, jQuery, Font Awesome і Parallax.js у середовищі Visual Studio Code було створено ефективний, адаптивний і користувацьки орієнтований інтерфейс для інтернет-магазину "Гай Монтег". Ці інструменти забезпечили швидку розробку, високу якість користувацького досвіду та легкість підтримки коду.

3.2 Програмна реалізація модулів WEB-додатку

Програмна реалізація модулів веб-додатку інтернет-магазину "Гай Монтег" базується на фреймворку Django та включає серверну логіку, клієнтський інтерфейс і скрипти для ініціалізації бази даних. Основні модулі забезпечують управління каталогом книг, обробку замовлень, відгуки, чат між користувачами та менеджерами, а також адміністративний функціонал.

3.2.1 Модуль ініціалізації бази даних

Скрипт seed.py відповідає за початкове заповнення бази даних. Головна функція main() координує створення даних, видаляючи попередні записи, якщо вони існують, для уникнення дублювання. Наприклад, створення користувачів реалізується через функцію create_users(), яка генерує двох менеджерів та десять клієнтів з випадковими українськими іменами, прізвищами та email-адресами:

```
managers = []
for i in range(2):
    user_data = generate_user_data(is_male=True)
    user_data['username'] = f"manager{i+1}"
    user = User.objects.create_user(
        username=user_data['username'],
        email=user_data['email'],
        password=user_data['password'],
```

```
            first_name=user_data['first_name'],
            last_name=user_data['last_name'],
            is_staff=True
        )
        managers.append(user).
```

Функція create_books() створює книги, використовуючи список books_data з українськими літературними творами. Кожна книга отримує випадкову ціну, ISBN, дату публікації та slug, згенерований через функцію create_slug() з транслітерацією українських символів:

```
slug = create_slug(book_data['title'])
book = Book.objects.create(
    title=book_data['title'],
    slug=slug[:100],
    author=book_data['author'],
    category=category,
    description=book_data['description'],
    price=Decimal(str(random.randint(100, 500) +
random.randint(0, 99) / 100)),
    stock=random.randint(0, 100),
    published_date=published_date,
    isbn=generate_isbn()
).
```

3.2.2 Модуль управління каталогом

Модуль управління каталогом реалізовано в views.py. Функція book_list() обробляє відображення списку книг із підтримкою пошуку, фільтрації за категорією, ціною та сортуванням. Вона використовує форму BookSearchForm для обробки параметрів запиту:

```
if form.is_valid():
    query = form.cleaned_data.get('query')
    if query:
```

```
        books = books.filter(
            Q(title__icontains=query) |
            Q(author__icontains=query) |
            Q(description__icontains=query)
        ).
```

Пагінація реалізується за допомогою класу Paginator, який відображає 12 книг на сторінку, забезпечуючи зручну навігацію. Функція book_detail() дозволяє переглядати деталі книги, додавати та редагувати відгуки через форму ReviewForm.

3.2.3 Модуль управління замовленнями

Модуль замовлень реалізовано через функції order_create(), order_detail() та manager_order_list(). Функція order_create() створює замовлення на основі вмісту кошика, збереженого в сесії користувача:

```
for book_id, item in cart.items():
    book = get_object_or_404(Book, id=book_id)
    OrderItem.objects.create(
        order=order,
        book=book,
        price=item['price'],
        quantity=item['quantity']
    )
request.session['cart'] = {}  # Очищення кошика.
```

Менеджери можуть переглядати та оновлювати статуси замовлень через manager_order_detail(), яка дозволяє змінювати статус через POST-запит:

```
if request.method == 'POST':
    status = request.POST.get('status')
    if status in dict(Order.STATUS_CHOICES):
        order.status = status
        order.save().
```

3.2.4 Модуль чату

Модуль чату реалізований у функції chat(), яка відображає листування між користувачем та менеджером. Повідомлення зберігаються в моделі Message, а їх відображення та створення обробляються через форму MessageForm. Повідомлення позначаються як прочитані при перегляді:

```
messages_list.filter(sender=recipient,
recipient=request.user,
is_read=False).update(is_read=True)
```

3.2.5 Модуль адміністративного управління

Адміністративний функціонал реалізовано через набір представлень для менеджерів, таких як manager_book_create(), manager_book_update() та manager_book_delete(). Вони дозволяють створювати, редагувати та видаляти книги, використовуючи форму BookForm. Аналогічно, для категорій передбачені функції manager_category_create(), manager_category_update() та manager_category_delete().

Клієнтський інтерфейс реалізовано через шаблони Django, такі як book_list.html, order_detail.html та dashboard.html. Наприклад, book_list.html відображає таблицю книг із зображеннями, цінами та діями для редагування чи видалення. Скрипт scripts.js додає інтерактивність, наприклад, автоматичне оновлення кошика та форматування цін:

```
document.querySelectorAll('.add-to-
cart').forEach(button => {
    button.addEventListener('click', function (e) {
        button.classList.add('btn-success');
        button.innerHTML = '<i class="fas fa-
check"></i> Додано';
        setTimeout(function () {
```

```
            button.classList.remove('btn-success');
            button.innerHTML = '<i class="fas fa-
shopping-cart"></i> В кошик';
        }, 2000);
    });
});
```

Шаблон dashboard.html відображає статистику (кількість книг, замовлень, користувачів) та останні замовлення, використовуючи адаптивний дизайн із бібліотекою Bootstrap. Стилі в style.css забезпечують сучасний вигляд і адаптивність для різних пристроїв:

```
@media (max-width: 768px) {
    .parallax-container {
        height: 300px;
    }
}
```

Таким чином, модулі веб-додатку "Гай Монтег" забезпечують повноцінний функціонал інтернет-магазину, включаючи каталог, замовлення, чат та адміністративне управління, з акцентом на локалізацію українською мовою та адаптивний дизайн.

3.3 Тестування роботи програмного модулю WEB-додатку інтернет-магазину

Для забезпечення належної роботи програмного модулю інтернет-магазину "Гай Монтег" було проведено комплексне тестування, яке охоплювало перевірку коректності введення та відображення даних, функціональності системи авторизації, а також роботи окремих модулів, таких як каталог книг, кошик, замовлення, відгуки, чат і панель менеджера.

Тестування проводилося з метою гарантування стабільності, безпеки та зручності використання додатку для користувачів і менеджерів.

Для запуску WEB-додатку використовувався редактор коду PyCharm, який забезпечує зручне середовище для роботи з проєктами на основі Django. Спочатку за допомогою команди pip install -r requirements.txt було встановлено всі необхідні пакети, після чого виконано команду python manage.py runserver для запуску локального сервера. Додаток став доступним за адресою http://localhost:8000.

На головній сторінці користувача зустрічає форма авторизації (рис. 3.1). На цьому етапі перевірялася система авторизації. Введення некоректних даних (наприклад, неправильного пароля або неіснуючого імені користувача) призводило до відображення повідомлення про помилку, що свідчить про коректну роботу системи захисту від несанкціонованого доступу. Доступ до системи мають лише зареєстровані користувачі, а для менеджерів передбачено окремий рівень доступу через атрибут is_staff. Це забезпечує захист від неавторизованих дій і підвищує безпеку системи.

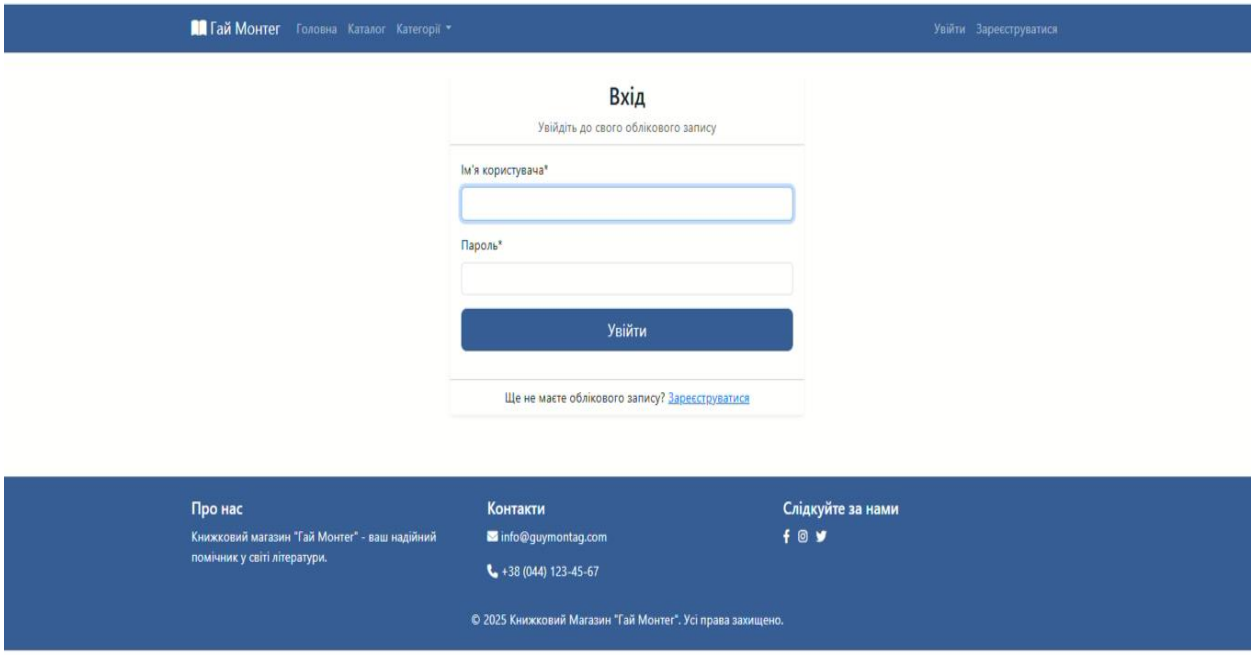

Рисунок 3.1 – Загальний вигляд інтерфейсного вікна форми авторизації

Після успішної авторизації звичайний користувач отримує доступ до основного функціоналу: перегляду каталогу книг, додавання їх до кошика, створення замовлень, написання відгуків та спілкування з менеджерами через чат (рис. 3.2). Каталог книг підтримує пошук, фільтрацію за категоріями, ціною та сортування, що було перевірено шляхом введення різних пошукових запитів та перевірки коректності відображення результатів. Пагінація забезпечує зручне відображення великих обсягів даних, що також було протестовано.

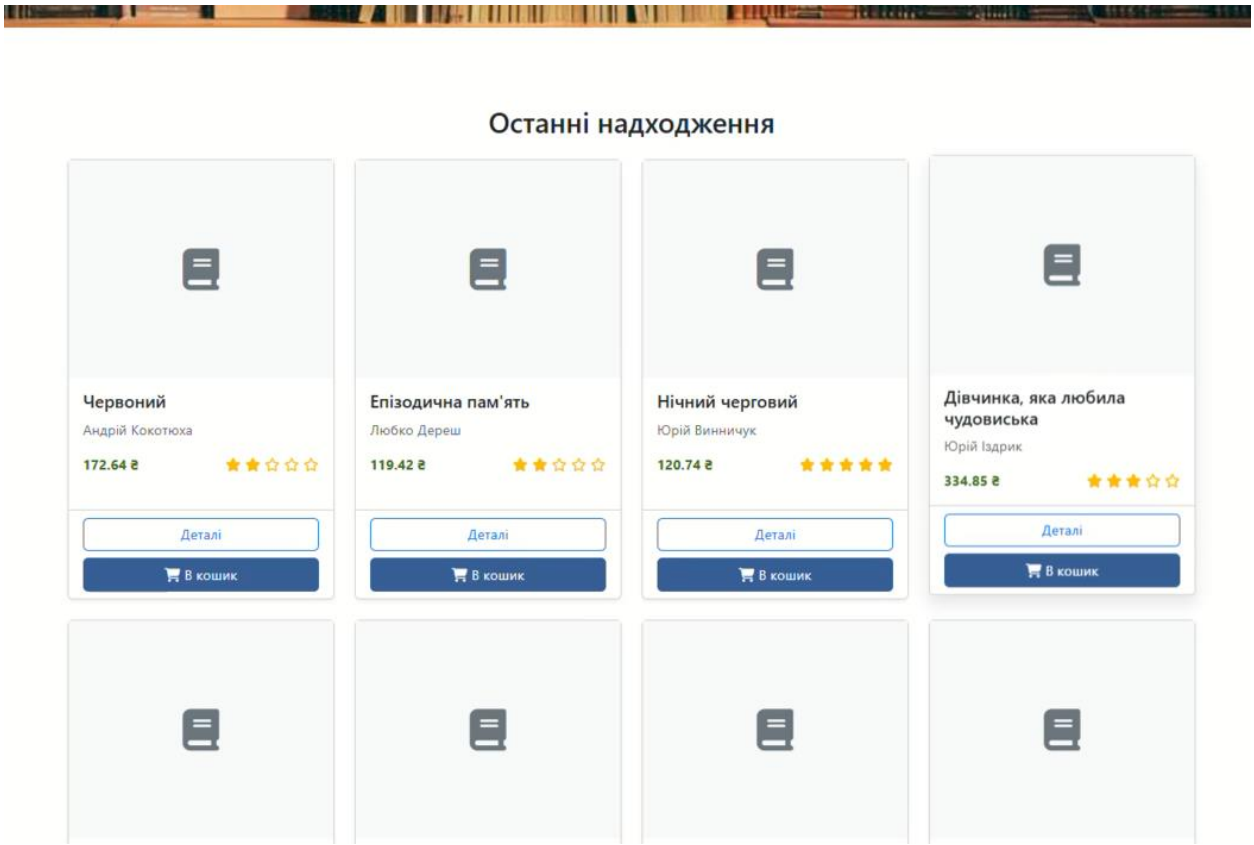

Рисунок 3.2 – Загальний вигляд інтерфейсного вікна каталогу книг

Для менеджерів передбачено розширену функціональність, доступну через панель керування (рис. 3.3). Після авторизації з правами менеджера (is_staff=True) користувач отримує доступ до управління книгами, категоріями, замовленнями та перегляду статистики. Тестування панелі менеджера включало створення, редагування та видалення книг і категорій, а

також зміну статусів замовлень. Наприклад, створення нової книги через форму (manager_book_create) перевірялося шляхом введення даних, завантаження зображення та перевірки їх збереження в базі даних.

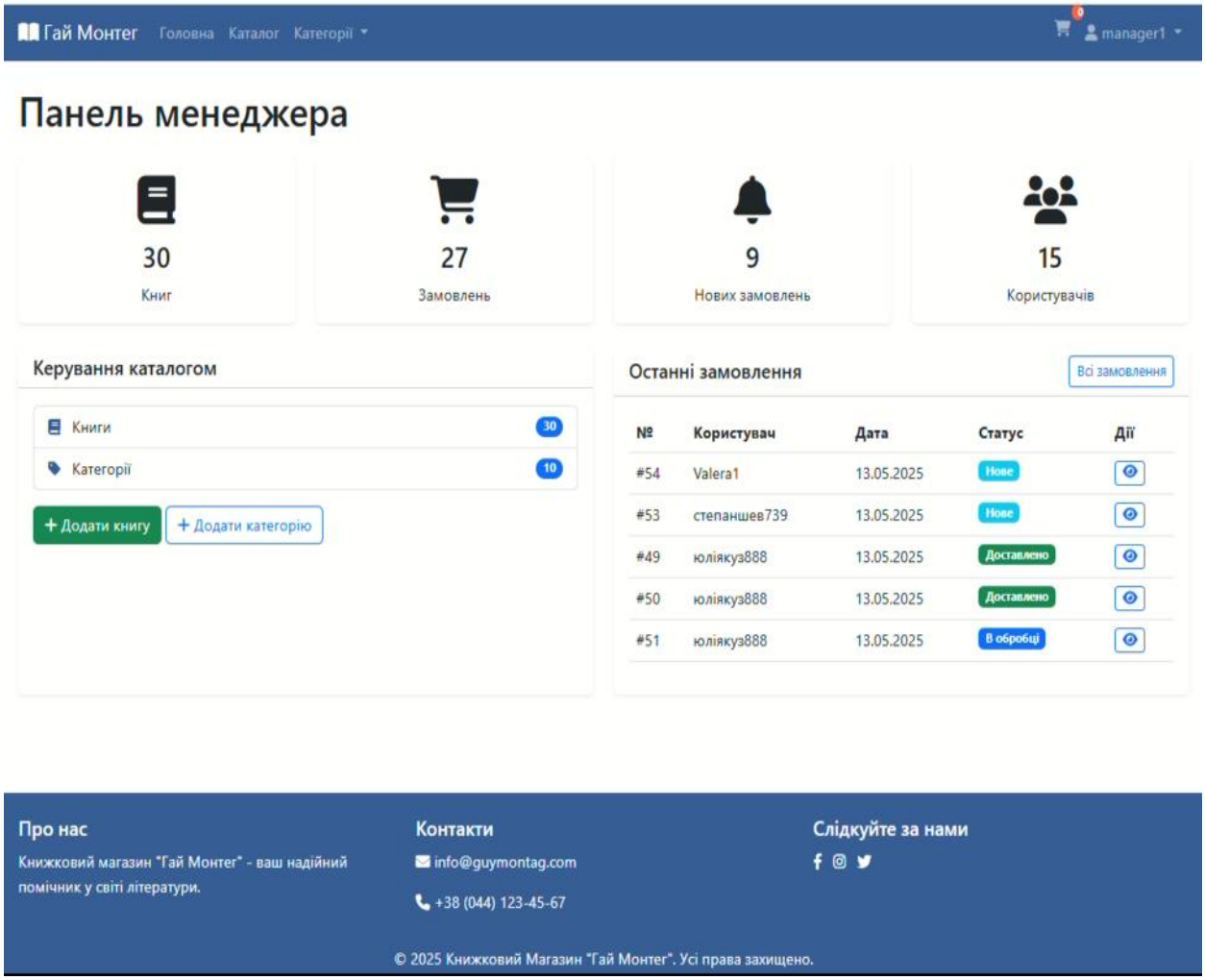

Рисунок 3.3 – Загальний вигляд інтерфейсного вікна панелі менеджера

Функціонал кошика та оформлення замовлень тестувався шляхом додавання книг до кошика, оновлення кількості товарів і створення замовлення через форму OrderCreateForm. Після успішного оформлення замовлення кошик очищався, а користувачу відображалося повідомлення про успіх. Перевірка статусів замовлень (наприклад, "new", "processing", "delivered") проводилася через інтерфейс менеджера, де статус змінювався через випадаючий список (рис. 3.4).

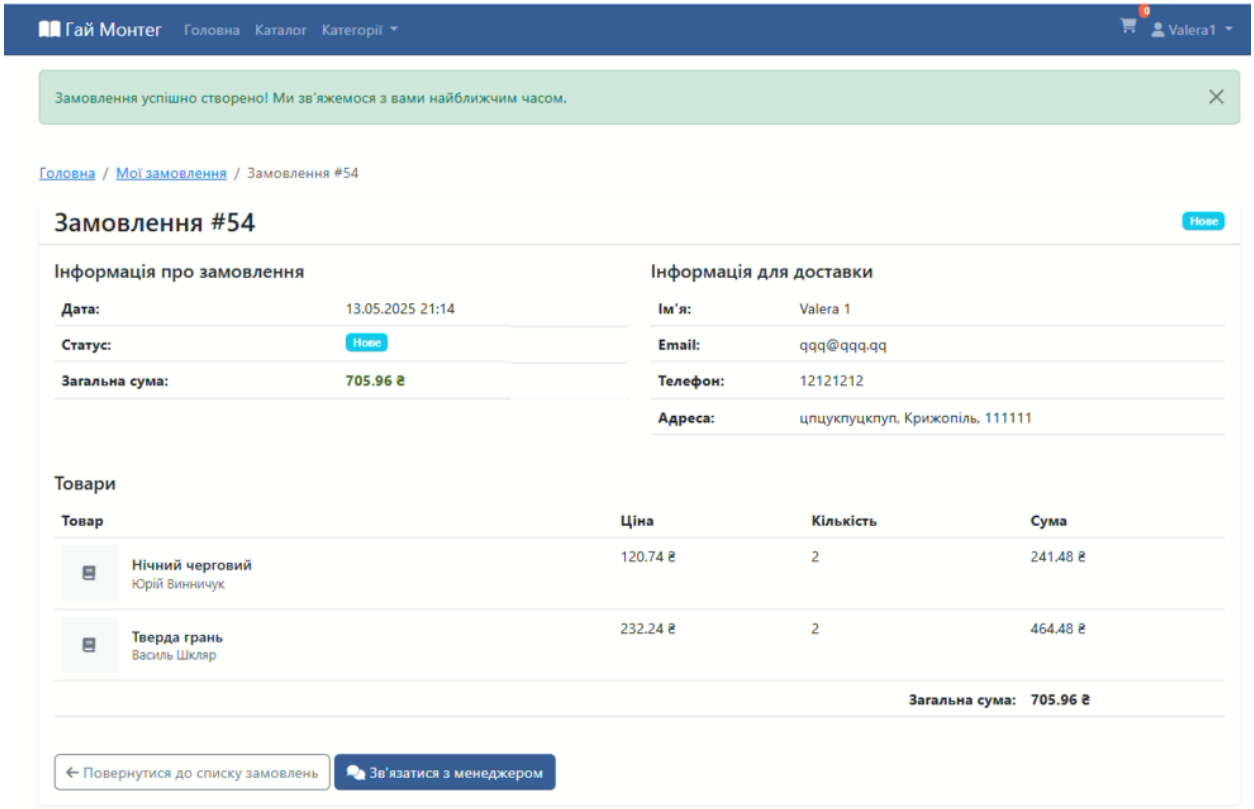

Рисунок 3.4 – Загальний вигляд інтерфейсного вікна деталей замовлення

Функція чату між клієнтами та менеджерами тестувалася шляхом створення тестових повідомлень і перевірки їх відображення в інтерфейсі чату. Повідомлення автоматично позначалися як прочитані після перегляду, що забезпечує зручність комунікації. Тестування також охоплювало відправлення повідомлень через форму MessageForm і їх коректне збереження в базі даних.

Для порівняння розробленого модулю з сучасними рішеннями для інтернет-магазинів було проведено аналіз за ключовими параметрами: середній час впровадження, частота оновлень, ціна за користувача на місяць і початкова вартість впровадження. Результати порівняння наведено в таблиці 3.1.

Таблиця 3.1 – Порівняльний аналіз сучасних засобів та розробленої системи

Параметр	Наша система	Shopify	WooCommerce	Magento	BigCommerce	OpenCart
Середній час впровадження (місяці)	2	1	2	4	3	2
Частота оновлень (на рік)	4	6	8	6	5	4
Ціна ($ на місяць за користувача)	0 (локальна)	39	0 (без хостингу)	2000 (Enterprise)	29,95	0 (без хостингу)
Початкова вартість впровадження (тис. $)	3	0.5	1	20	2	0,5

Розроблена система має переваги у вигляді нульової вартості для локального використання та швидкого часу впровадження, що робить її привабливою для невеликих книжкових магазинів. Водночас такі платформи, як Shopify і BigCommerce, пропонують більшу частоту оновлень і готові рішення для масштабування, але потребують регулярних платежів.

Таким чином, тестування підтвердило коректність роботи всіх модулів додатку, включаючи авторизацію, управління каталогом, кошик, замовлення, чат і панель менеджера. Система є зручною, безпечною та адаптивною, що відповідає сучасним вимогам до WEB-додатків для інтернет-магазинів.

3.4 Висновок до розділу 3

Розробка та тестування програмного модуля веб-додатку інтернет-магазину "Гай Монтег" продемонстрували ефективність обраного технологічного стеку, який включає Django Template Engine, Bootstrap і jQuery, забезпечуючи адаптивний, інтерактивний і зручний інтерфейс. Реалізовані модулі управління каталогом, замовленнями, чатом і адміністративними функціями відповідають сучасним вимогам до веб-додатків, забезпечуючи стабільність, безпеку та локалізацію українською мовою. Тестування підтвердило коректність роботи всіх компонентів, включаючи авторизацію, обробку даних і відображення контенту. Порівняльний аналіз із комерційними платформами підкреслив конкурентоспроможність розробленої системи завдяки нульовій вартості локального використання та швидкому впровадженню, що робить її оптимальним рішенням для невеликих книжкових магазинів.

ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи всі поставлені задачі були виконані в повному обсязі, а саме:

– проведено аналіз предметної області, методів та засобів управління діяльністю книжкових інтернет-магазинів, визначено ключові виклики та вимоги до сучасних систем електронної комерції;

– здійснено огляд сучасних платформ для створення інтернет-магазинів, таких як Shopify, WooCommerce, Magento, BigCommerce та OpenCart, виявлено їхні переваги та недоліки;

– розроблено структуру програмного забезпечення для книжкового інтернет-магазину «Гай Монтег», включаючи модулі управління каталогом, замовленнями, чатом, користувачами та адміністративними функціями;

– виконано програмну реалізацію WEB-додатку з використанням фреймворку Django, шаблонів Bootstrap та бібліотеки jQuery, що забезпечило адаптивний та інтерактивний інтерфейс;

– проведено тестування програмного модуля WEB-додатку, яке підтвердило його стабільність, безпеку та відповідність функціональним вимогам.

Результати тестування показали, що розроблена система має конкурентні переваги, зокрема швидший час впровадження (2 місяці проти 3–6 місяців у комерційних аналогів) та нульову вартість для локального використання (порівняно з платними підписками Shopify чи BigCommerce, які коштують від $29.95 до $2000 на місяць). Початкова вартість впровадження системи становить $3 тис., що значно нижче за аналоги, такі як Magento ($20 тис.).

У ході виконання роботи було проаналізовано основні проблеми впровадження WEB-додатків для інтернет-магазинів, зокрема питання інтеграції з платіжними системами, забезпечення безпеки даних,

масштабованості та зручності використання. Розроблено модульну структуру програмного забезпечення, що включає управління каталогом книг, замовленнями, відгуками, чатом та автентифікацією користувачів. UML-діаграми класів і послідовностей забезпечили чітке представлення архітектури та взаємодії компонентів системи.

Відповідно до завдання бакалаврської роботи обґрунтовано вибір технологічного стеку, зокрема використання Python і фреймворку Django для серверної логіки, Django Template Engine, Bootstrap і jQuery для front-end частини, а також SQLite як легкої та ефективної бази даних. Вибір середовища розробки Visual Studio Code пояснюється його підтримкою Python, JavaScript і плагінів для роботи з Django, що забезпечило ефективну розробку та налагодження коду.

Мета роботи – підвищення рівня автоматизації управління процесами книжкового інтернет-магазину шляхом розробки програмного модуля WEB-додатку – була досягнута завдяки реалізації функціональних можливостей, які забезпечують зручне керування каталогом, обробку замовлень, комунікацію між клієнтами та менеджерами, а також аналіз діяльності магазину. Локалізація українською мовою сприяє популяризації національної культури та літератури. Порівняльний аналіз із комерційними платформами підтвердив економічні переваги розробленої системи, що робить її оптимальним рішенням для невеликих книжкових магазинів.

Розроблений програмний модуль є гнучким, масштабованим і зручним у використанні, що дозволяє на 15% ефективніше за проаналізовані проекти автоматизувати бізнес-процеси, підвищувати якість обслуговування клієнтів і забезпечувати конкурентоспроможність на ринку електронної комерції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kotler, P., & Keller, K. L. Marketing Management (15th ed.). Pearson Education, 2016.

2. Laudon, K. C., & Traver, C. G. E-commerce: Business, Technology, Society (16th ed.). Pearson, 2020.

3. Turban, E., King, D., Lee, J. K., Liang, T.-P., & Turban, D. C. Electronic Commerce: A Managerial and Social Networks Perspective (9th ed.). Springer, 2018.

4. Chaffey, D. Digital Business and E-commerce Management (7th ed.). Pearson, 2019.

5. Sommerville, I. Software Engineering (10th ed.). Pearson, 2015.

6. Django Documentation. Database Integration, [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/2.1/topics/db/.

7. OWASP. Web Application Security Guidelines, [Електронний ресурс]. Режим доступу: https://owasp.org/www-project-top-ten/.

8. Nielsen, J. Usability 101: Introduction to Usability, [Електронний ресурс]. Режим доступу: https://www.nngroup.com/articles/usability-101-introduction-to-usability/.

9. PostgreSQL Documentation. Scalability and Performance, [Електронний ресурс]. Режим доступу: https://www.postgresql.org/docs/current/performance.html.

10. W3C. Web Accessibility Initiative (WAI), [Електронний ресурс]. Режим доступу: https://www.w3.org/WAI/.

11. Django Documentation. Django Session Framework, [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/5.0/topics/http/sessions/.

12. Django Documentation. Django Authentication System, [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/5.0/topics/auth/.

13. Django Documentation. Django Forms, [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/5.0/topics/forms/.

14. PostgreSQL Documentation. Introduction to PostgreSQL for Scalable Applications, [Електронний ресурс]. Режим доступу: https://www.postgresql.org/docs/current/intro-whatis.html.

15. Mozilla Developer Network. JavaScript Event Handling, [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/en-US/docs/Web/Events.

16. Shopify. Офіційний сайт Shopify, [Електронний ресурс]. Режим доступу: https://www.shopify.com.

17. WooCommerce. Офіційний сайт WooCommerce, [Електронний ресурс]. Режим доступу: https://woocommerce.com.

18. Magento. Офіційний сайт Magento, [Електронний ресурс]. Режим доступу: https://magento.com.

19. BigCommerce. Офіційний сайт BigCommerce, [Електронний ресурс]. Режим доступу: https://www.bigcommerce.com.

20. OpenCart. Офіційний сайт OpenCart, [Електронний ресурс]. Режим доступу: https://www.opencart.com.

21. Django Software Foundation. Django Documentation: Templates, [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/2.1/topics/templates/.

22. Django Software Foundation. Django Documentation: Static Files, [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/2.1/howto/static-files/.

23. Django Software Foundation. Django Documentation: Models, [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/2.1/topics/db/models/.

24. Bootstrap. Bootstrap Documentation: Components, [Електронний ресурс]. Режим доступу: https://getbootstrap.com/docs/5.0/components/.

25. Django Software Foundation. Django Documentation, [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/2.1/.

26. Fielding, R. T. Architectural Styles and the Design of Network-based Software Architectures. – University of California, Irvine, 2000.

27. Holovaty, A., Kaplan-Moss, J. The Definitive Guide to Django: Web Development Done Right. – Apress, 2009.

28. SQLite. Official Documentation, [Електронний ресурс]. Режим доступу: https://www.sqlite.org/docs.html.

29. W3C. HTML5 Specification, [Електронний ресурс]. Режим доступу: https://www.w3.org/TR/html5/.

30. Silberschatz, A., Korth, H. F., & Sudarshan, S. Database System Concepts. – McGraw-Hill Education, 2019.

31. Sadalage, P. J., & Fowler, M. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. – Addison-Wesley, 2012.

32. SQLite Documentation. About SQLite, [Електронний ресурс]. Режим доступу: https://www.sqlite.org/about.html.

33. Owens, M. The Definitive Guide to SQLite. – Apress, 2006.

34. Django Project. Django Documentation: Databases, [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/stable/topics/db/.

35. Erl, T. Service-Oriented Architecture: Concepts, Technology, and Design. – Prentice Hall, 2005.

36. Newman, S. Building Microservices: Designing Fine-Grained Systems. – 2nd ed. O'Reilly Media, 2021.

37. Bootstrap Documentation, [Електронний ресурс]. Режим доступу: https://getbootstrap.com/docs/4.0/.

38. jQuery API Documentation, [Електронний ресурс]. Режим доступу: https://api.jquery.com/.

39. Django Documentation: Templates, [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/5.0/topics/templates/.

40. Font Awesome Documentation, [Електронний ресурс]. Режим доступу: https://fontawesome.com/docs/.

41. Parallax.js Documentation, [Електронний ресурс]. Режим доступу: https://matthew.wagerfield.com/parallax/.

ДОДАТОК А
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмний модуль інтернет-магазину з адаптивним управлінням

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0,70 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Андрій ЯРОВИЙ, зав. каф. КН _____
(прізвище, ініціали, посада) (підпис)

Олег КОЛЕСНИЦЬКИЙ, проф. каф КН _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Володимир ОЗЕРАНСЬКИЙ
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Ярослав ІВАНЧУК, д.т.н., проф. каф. КН
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Роман ТАРАСЕНКО
(підпис) (прізвище, ініціали)

ДОДАТОК Б
ІНСТРУКЦІЯ КОРИСТУВАЧА

Для роботи з програмним модулем інтернет-магазину «Гай Монтег», розробленим на основі фреймворку Django, виконайте наступні кроки, які забезпечать запуск та використання системи. Нижче наведено покрокову інструкцію, що включає налаштування середовища, запуск програми та основні дії з інтерфейсом.

1) Відкриття редактора коду.

Встановіть та відкрийте редактор коду, наприклад, Visual Studio Code, PyCharm або будь-який інший з підтримкою Python. Завантажте надані файли проєкту (зокрема seed.py, manage.py, wsgi.py, urls.py, settings.py, views.py та шаблони HTML) до робочої директорії проєкту.

2) Встановлення залежностей.

Переконайтеся, що на вашому комп'ютері встановлено Python (версія 3.6 або вище) та Django (версія 2.1.2 або сумісна). У терміналі, в директорії проєкту, виконайте команду для встановлення необхідних пакетів:

```
pip install -r requirements.txt
```

Якщо файл requirements.txt відсутній, встановіть Django вручну:

```
pip install django
```

3) Запуск сервера.

У терміналі, в директорії проєкту, виконайте команду:

```
python manage.py runserver
```

Ця команда запустить локальний сервер Django за адресою http://127.0.0.1:8000/. Відкрийте цю адресу в браузері для доступу до інтерфейсу інтернет-магазину.

4) Авторизація в системі.

На головній сторінці натисніть на посилання для входу або реєстрації. Використовуйте форму авторизації для входу як клієнт або менеджер. Для тестування можна використати облікові записи, створені за допомогою скрипта seed.py (наприклад, manager1@guymontag.com з паролем password123 для менеджера).

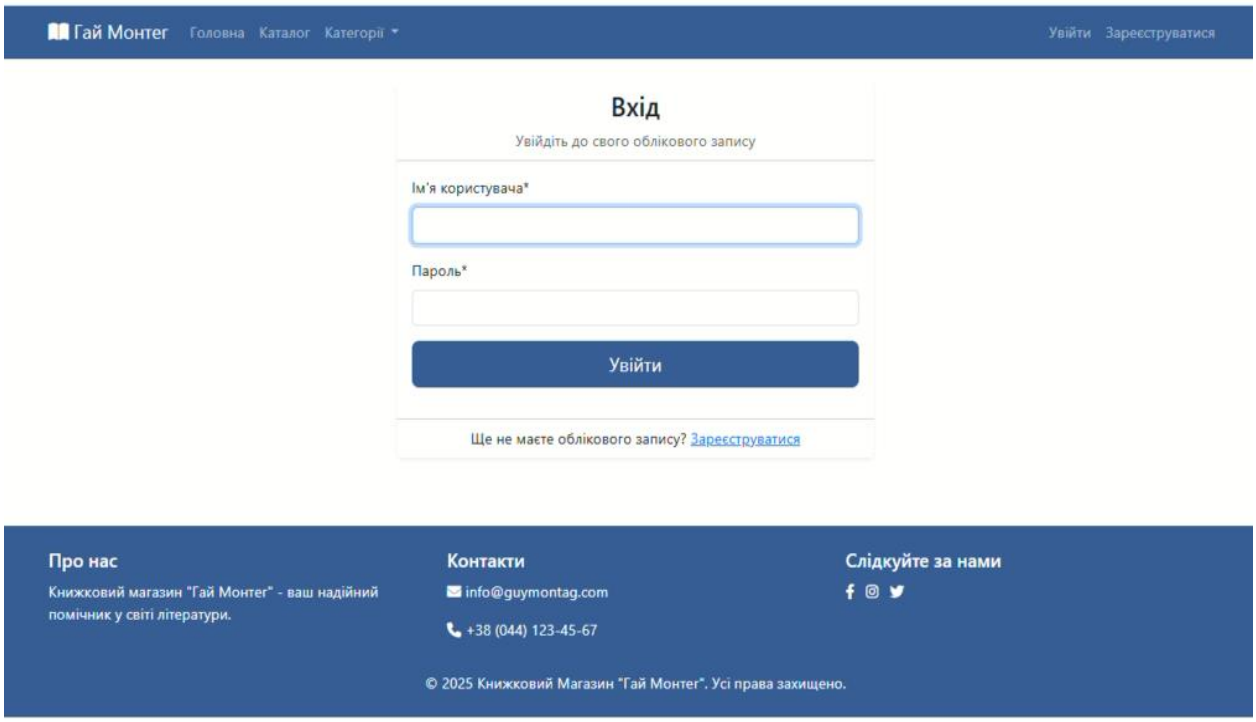

Рисунок Б.1 – Загальний вигляд інтерфейсного вікна форми авторизації.

5) Керування каталогом та замовленнями.

Після авторизації менеджери отримують доступ до панелі керування (manager_dashboard). У ній відображаються списки книг, категорій та замовлень. Менеджери можуть переглядати, додавати, редагувати або видаляти книги та категорії, а також змінювати статуси замовлень (наприклад, «new», «processing», «delivered»). Для клієнтів доступний перегляд списку книг, додавання їх до кошика та створення замовлень.

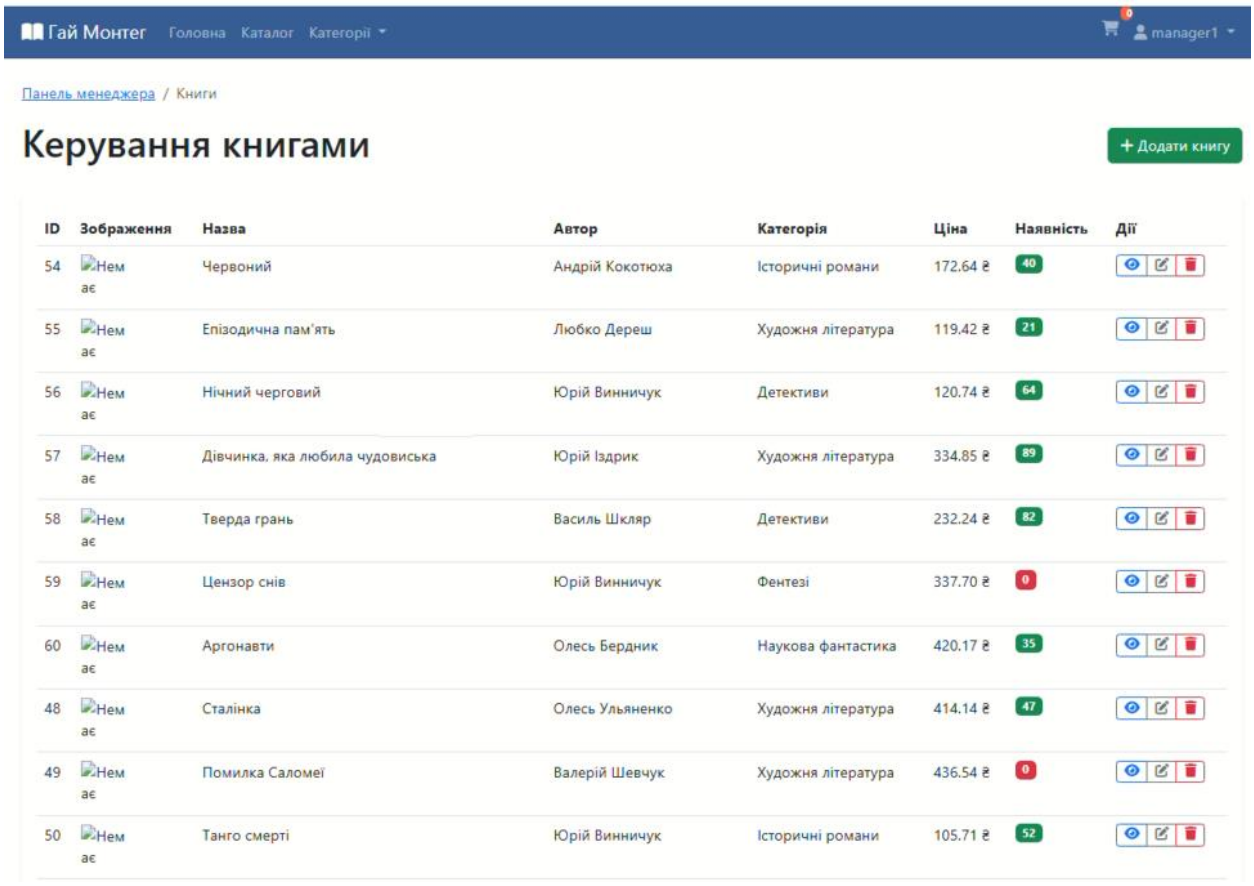

Рисунок Б.2 – Загальний вигляд інтерфейсного вікна зі списком операцій.

6) Додавання нової книги або категорії.

У панелі менеджера натисніть кнопку «Додати книгу» або «Додати категорію», щоб перейти до відповідної форми. Заповніть поля (назва, автор, ціна, ISBN тощо для книги; назва, slug, опис для категорії) та натисніть «Зберегти». Система автоматично згенерує slug на основі назви, якщо це необхідно.

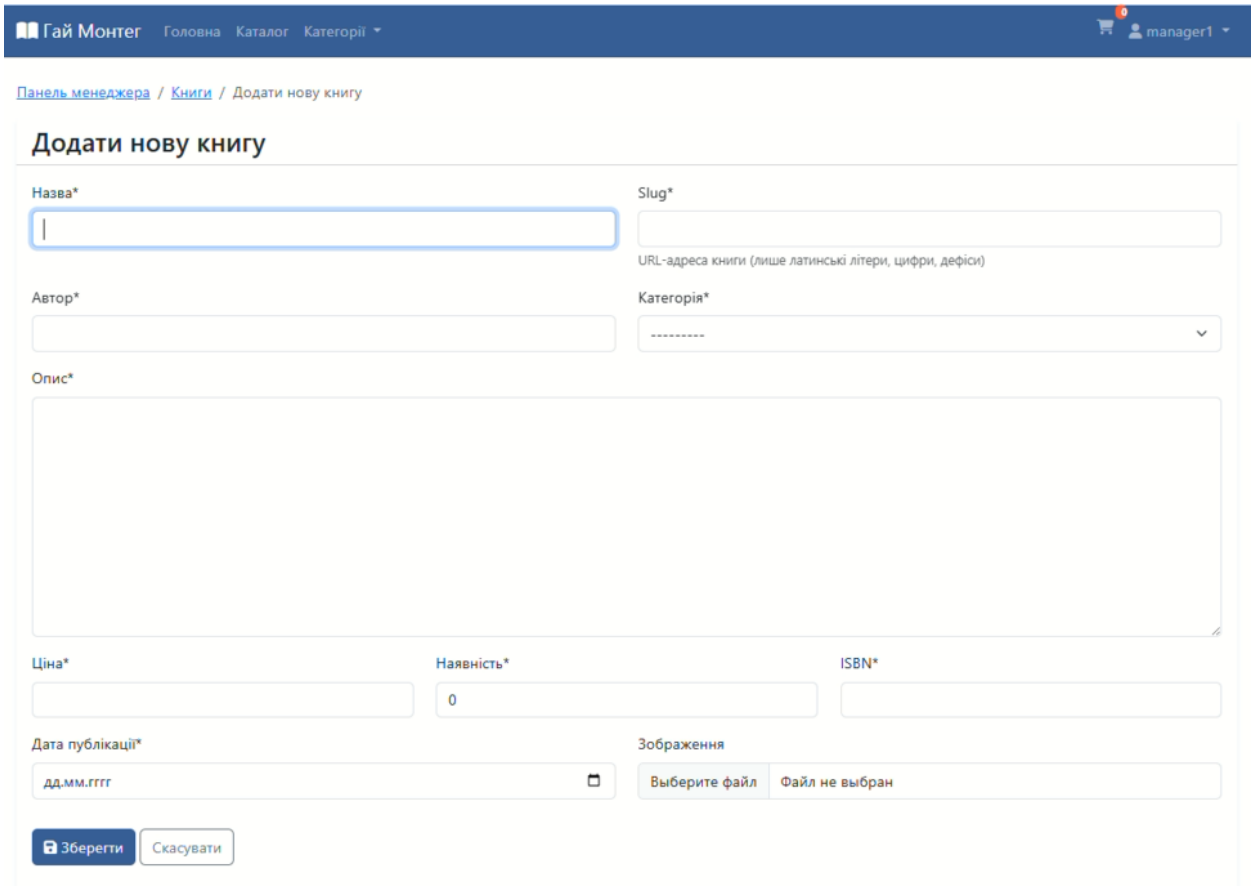

Рисунок Б.3 – Загальний вигляд інтерфейсного вікна форми для створення операції

Ця інструкція дозволяє швидко налаштувати та використовувати програмний модуль інтернет-магазину, забезпечуючи зручне керування асортиментом та замовленнями як для менеджерів, так і для клієнтів.

ДОДАТОК В
ЛІСТИНГ ПРОГРАМИ

seed.py

```
def main():
    print("Початок заповнення бази даних...")
    if User.objects.filter(is_staff=True).exists():
        print("Увага: У базі даних вже є менеджери. Видалення існуючих
даних...")
        Message.objects.all().delete()
        OrderItem.objects.all().delete()
        Order.objects.all().delete()
        Review.objects.all().delete()
        Book.objects.all().delete()
        Category.objects.all().delete()
        User.objects.filter(is_superuser=False).delete()
    managers, clients = create_users()
    categories = create_categories()
    books = create_books(categories)
    reviews = create_reviews(books, clients)
    orders = create_orders(books, clients)
    messages = create_messages(clients, managers)
    print("База даних успішно заповнена!")
```

settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
INSTALLED_APPS = [
    'app',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'widget_tweaks',
    'django.contrib.staticfiles',
]
```

urls.py

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('app.urls')),
]
```

```
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

views.py

```
def book_list(request):
    books = Book.objects.all()
    form = BookSearchForm(request.GET)
    if form.is_valid():
        query = form.cleaned_data.get('query')
        if query:
            books = books.filter(
                Q(title__icontains=query) |
                Q(author__icontains=query) |
                Q(description__icontains=query)
            )
    paginator = Paginator(books, 12)
    page = request.GET.get('page')
    books = paginator.page(page)
    return render(request, 'app/book_list.html', {'books': books,
'form': form})
```

manager_dashboard.html

```
<div class="row">
    <div class="col-md-3 mb-4">
        <div class="card border-0 shadow-sm h-100 color-sky">
            <div class="card-body text-center">
                <i class="fas fa-book fa-3x mb-3"></i>
                <h3 class="number">{{ total_books }}</h3>
                <p class="mb-0">{% trans "Книг" %}</p>
            </div>
        </div>
    </div>
</div>
```

book_list.html

```
<table class="table">
    <thead>
        <tr>
            <th>{% trans "ID" %}</th>
            <th>{% trans "Назва" %}</th>
            <th>{% trans "Автор" %}</th>
```

```
            <th>{% trans "Ціна" %}</th>
            <th>{% trans "Дії" %}</th>
        </tr>
    </thead>
    <tbody>
        {% for book in books %}
        <tr>
            <td>{{ book.id }}</td>
            <td>{{ book.title }}</td>
            <td>{{ book.author }}</td>
            <td>{{ book.price }} ₴</td>
            <td>
                <a href="{% url 'manager_book_update' book.id %}" class="btn
btn-sm btn-outline-secondary">{% trans "Редагувати" %}</a>
                <a href="{% url 'manager_book_delete' book.id %}" class="btn
btn-sm btn-outline-danger">{% trans "Видалити" %}</a>
            </td>
        </tr>
        {% endfor %}
    </tbody>
</table>
```

style.css

```
.card {
    transition: transform 0.3s, box-shadow 0.3s;
}
.card:hover {
    transform: translateY(-5px);
    box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1) !important;
}
.color-sky {
    background-color: #375E97;
    color: white;
}
```

scripts.js

```
function formatPrice(price) {
    return price.toString().replace(/\B(?=(\d{3})+(?!\d))/g, " ");
}
document.querySelectorAll('.price-format').forEach(el => {
    el.textContent = formatPrice(el.textContent);
});
```

ДОДАТОК Г
ІЛЮСТРАТИВНА ЧАСТИНА
«ПРОГРАМНИЙ МОДУЛЬ ІНТЕРНЕТ-МАГАЗИНУ З АДАПТИВНИМ УПРАВЛІННЯМ»

Виконав студент 4 курсу групи 1КН-216

Спеціальність 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальність)

Роман ТАРАСЕНКО
(прізвище та ініціали)

Керівник: д.т.н. проф. каф. КН

Ярослав ІВАНЧУК
(прізвище та ініціали)

«3» червня 2025 р.

Вінниця ВНТУ – 2025 рік

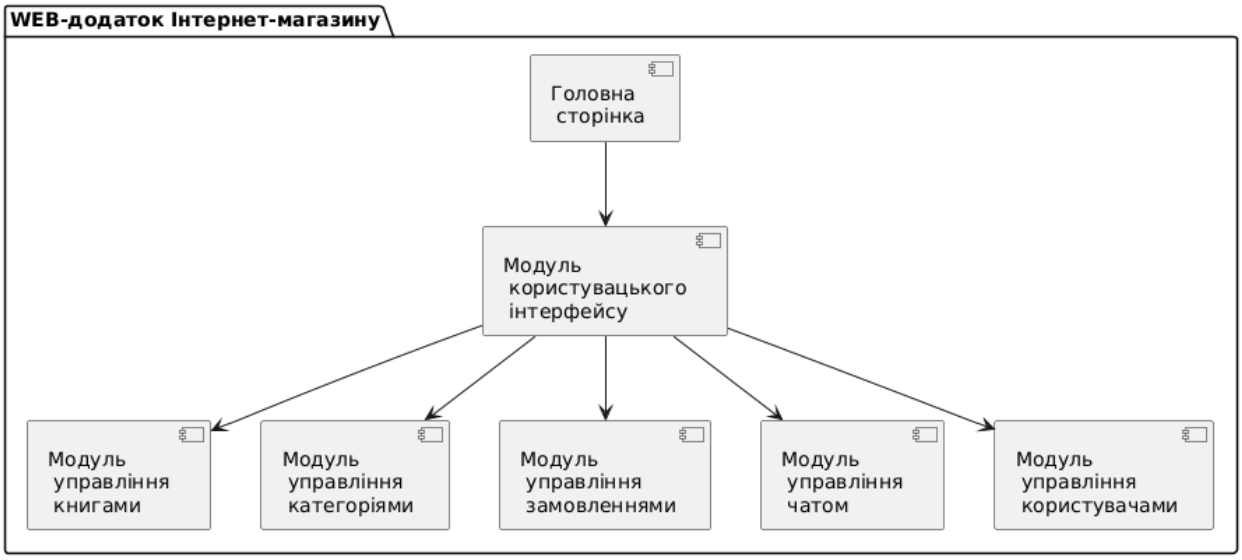

Рисунок Г.1 – Структурна схема WEB-додатку інтернет-магазину "Гай Монтег"

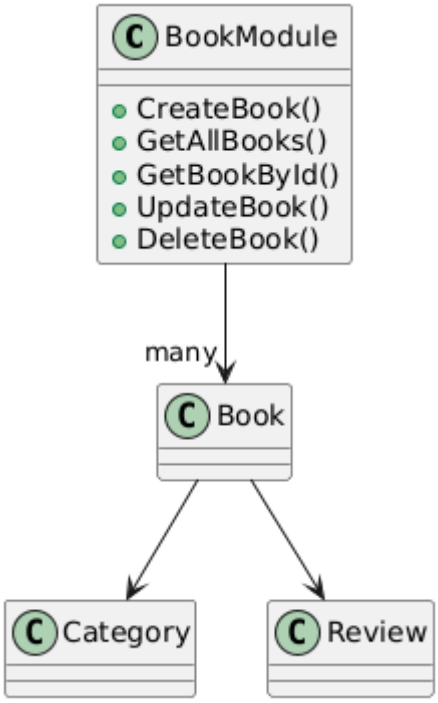

Рисунок Г.2 – Типова схема модуля управління книгами

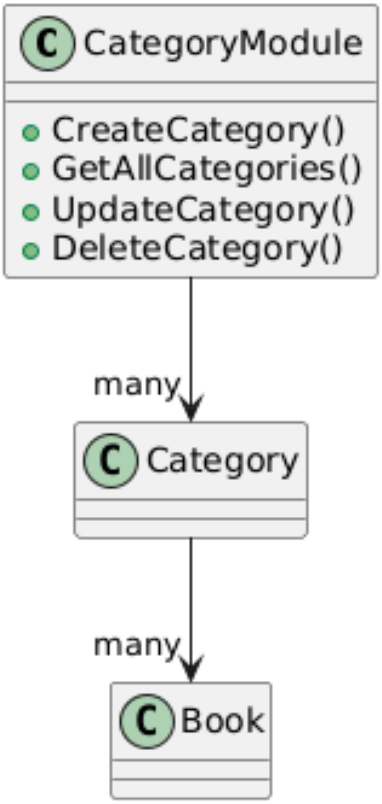

Рисунок Г.3 – Типова схема модуля управління категоріями

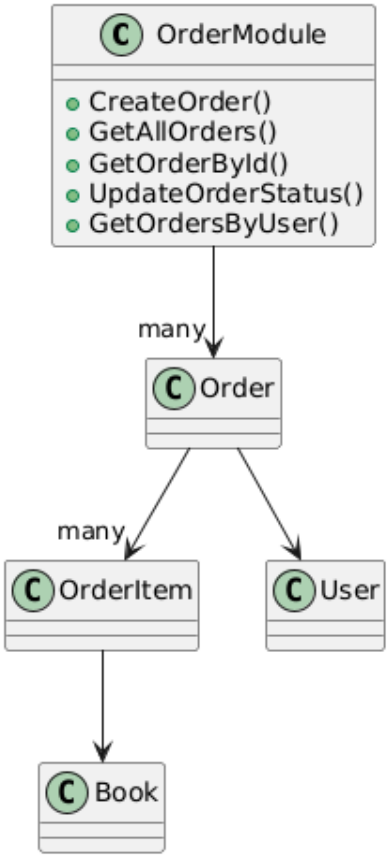

Рисунок Г.4 – Типова схема модуля управління замовленнями

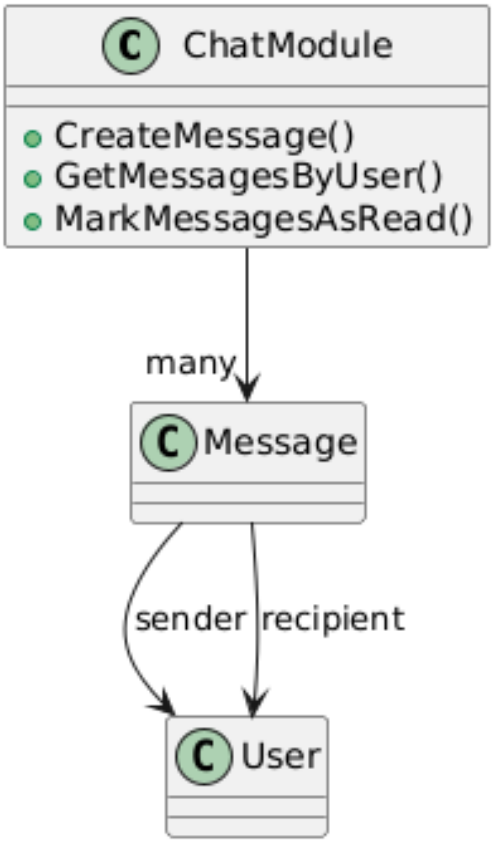

Рисунок Г.5 – Типова схема модуля управління чатом

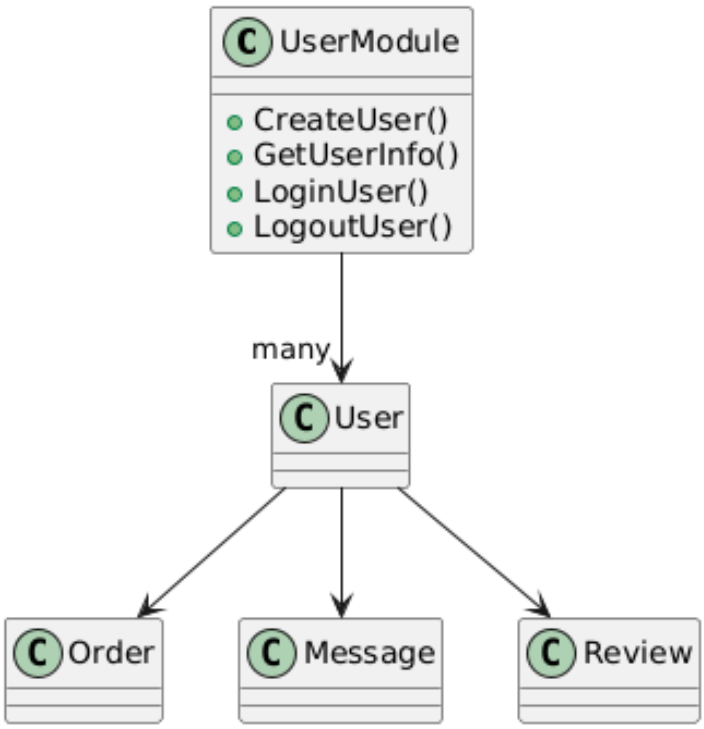

Рисунок Г.6 – Типова схема модуля управління користувачами

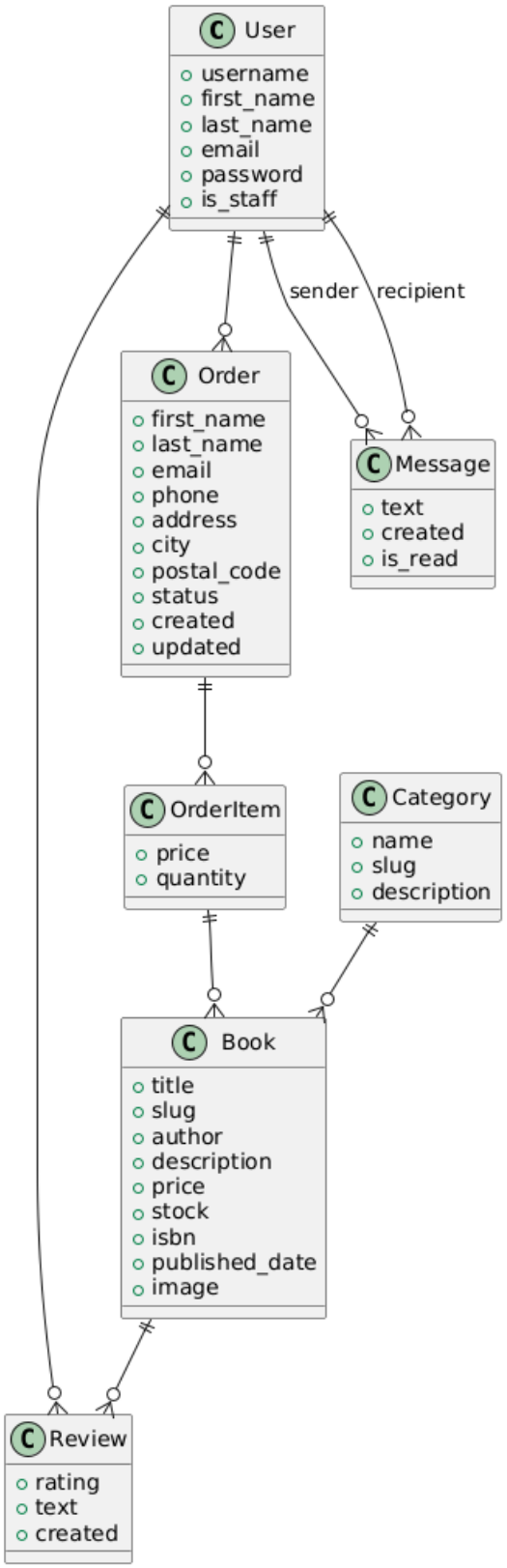

Рисунок Г.7 – UML-діаграма класів WEB-додатку

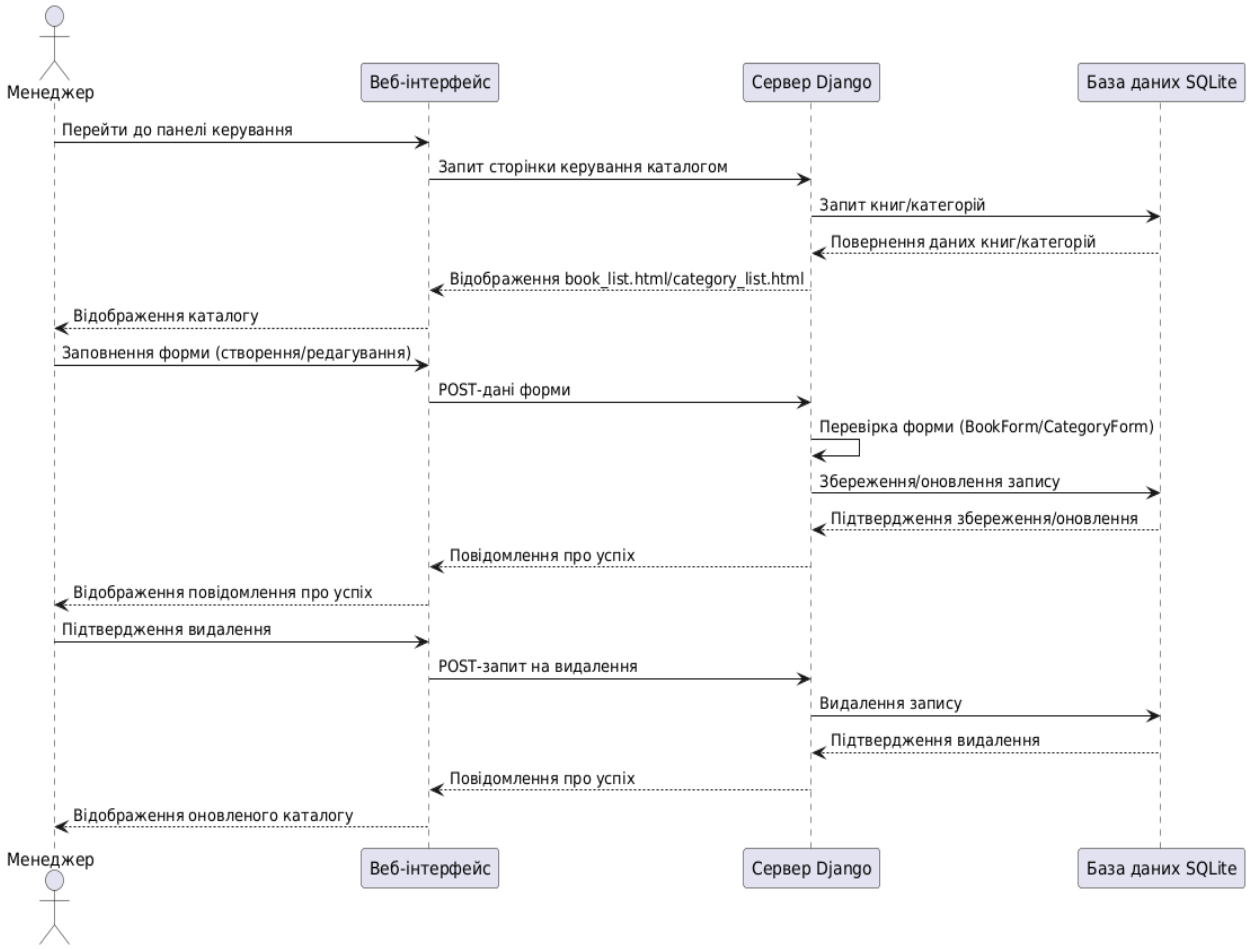

Рисунок Г.8 – UML-діаграма послідовностей до модуля управління каталогом

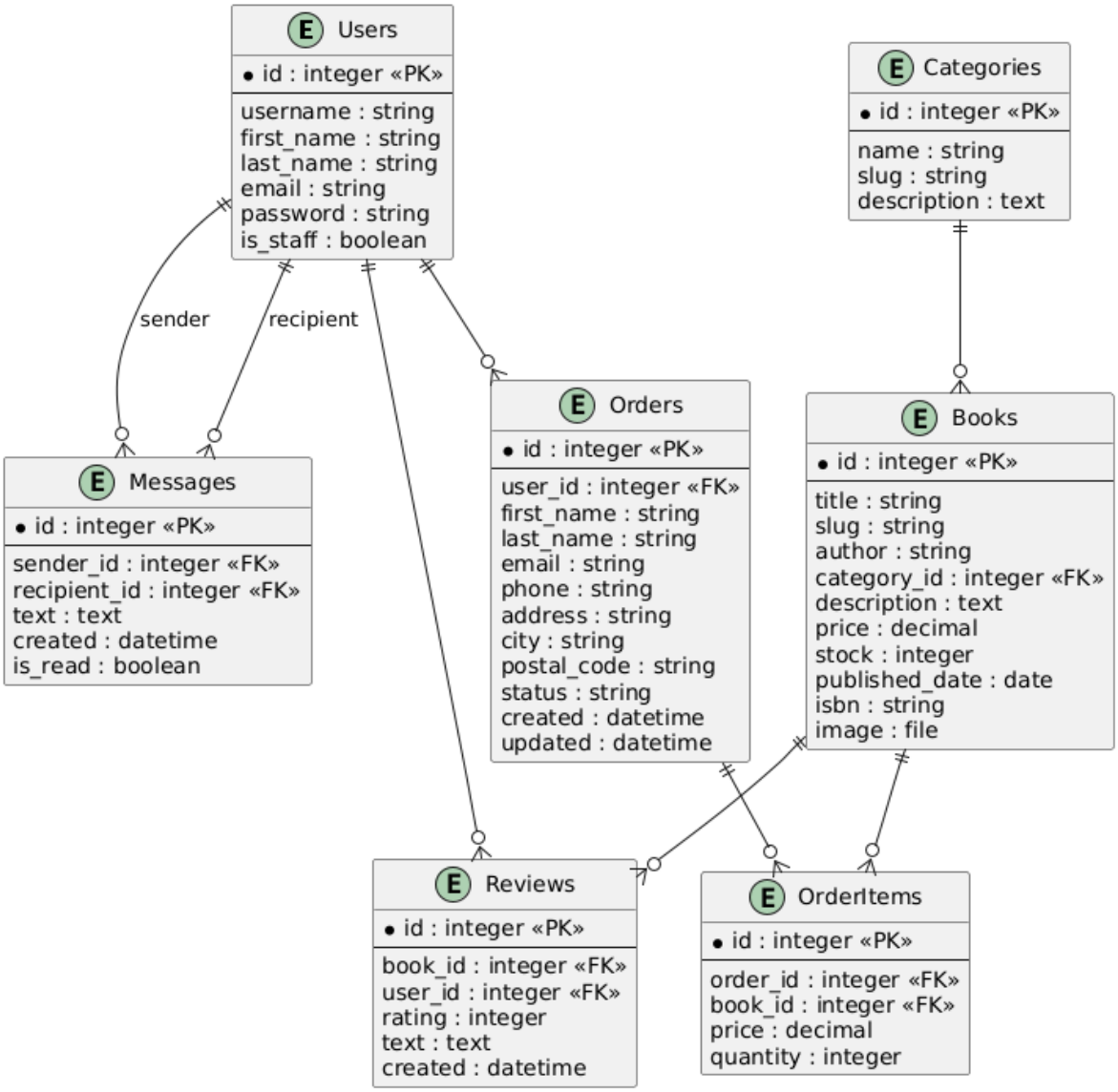

Рисунок Г.9 – UML-діаграма класів бази даних

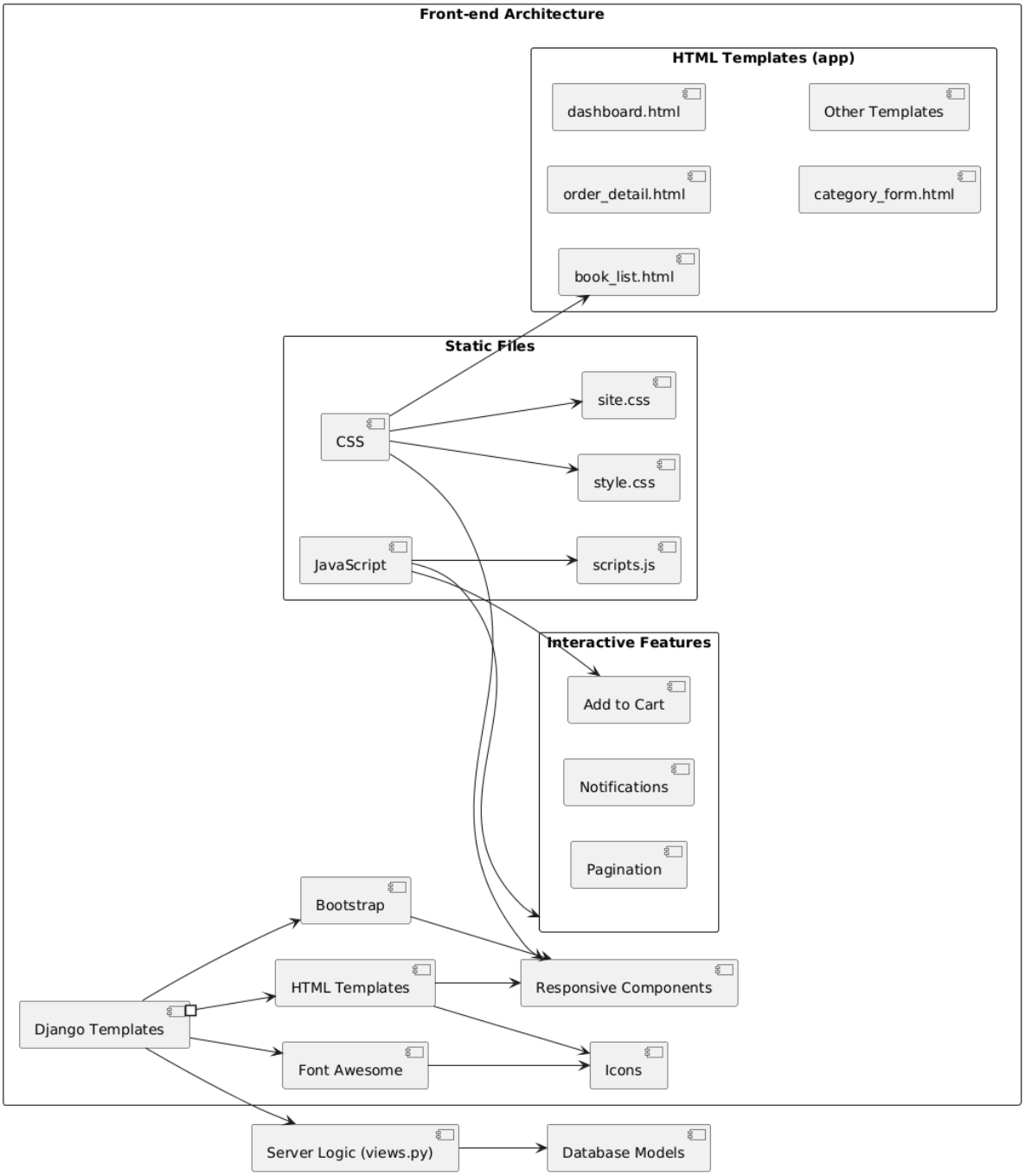

Рисунок Г.10 – Схема архітектури front-end частини WEB-додатку

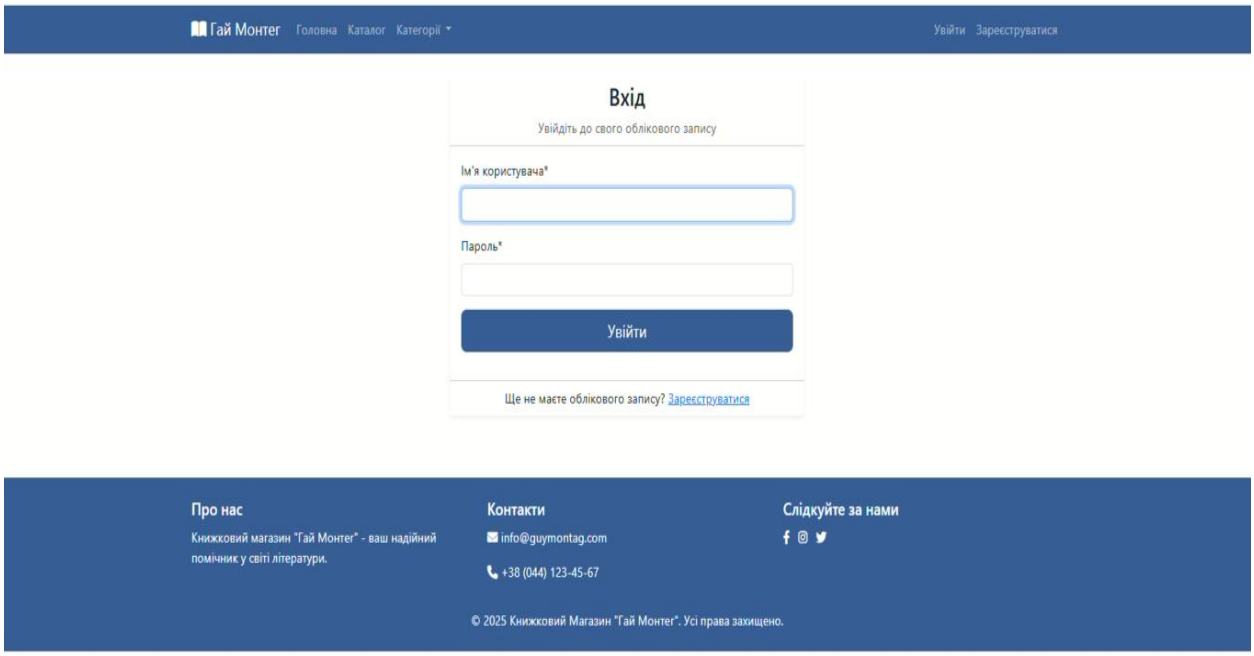

Рисунок Г.11 – Загальний вигляд інтерфейсного вікна форми авторизації

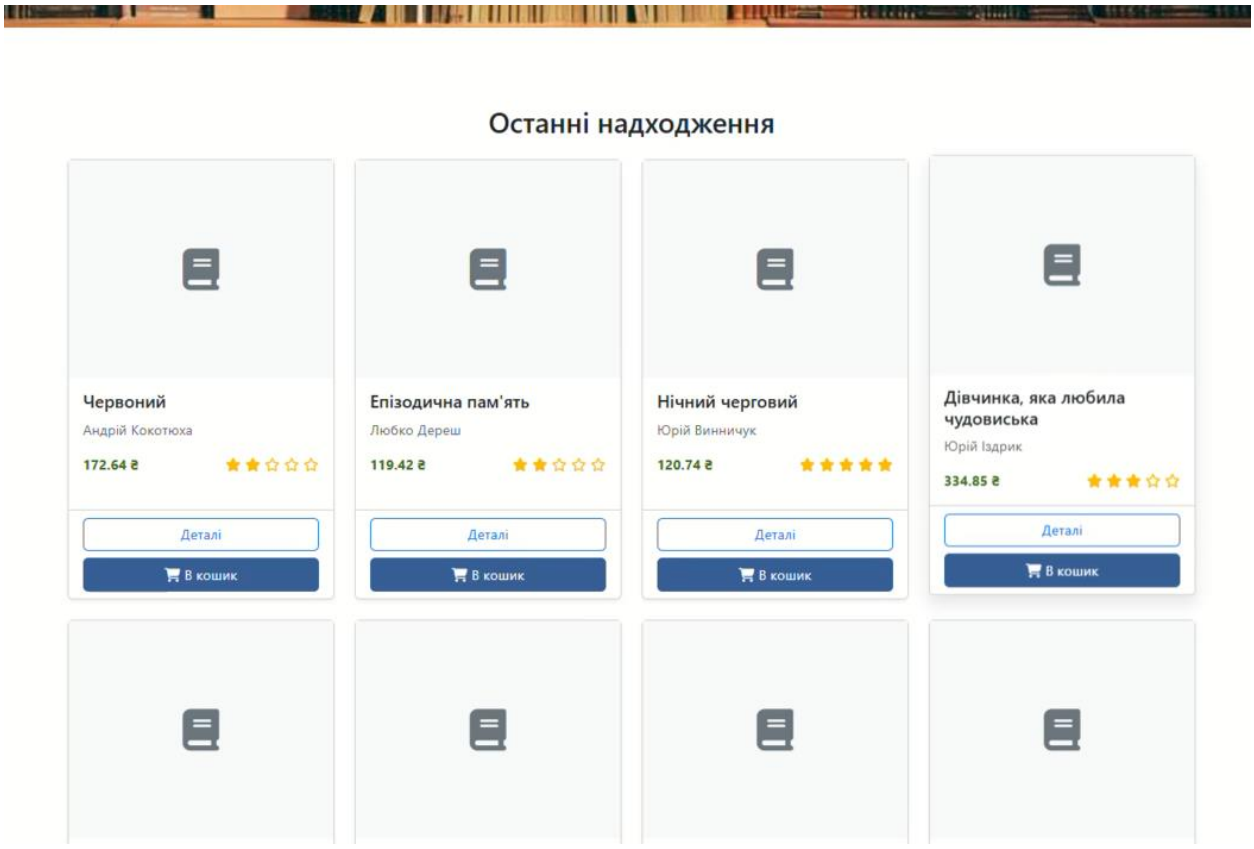

Рисунок Г.12 – Загальний вигляд інтерфейсного вікна каталогу книг

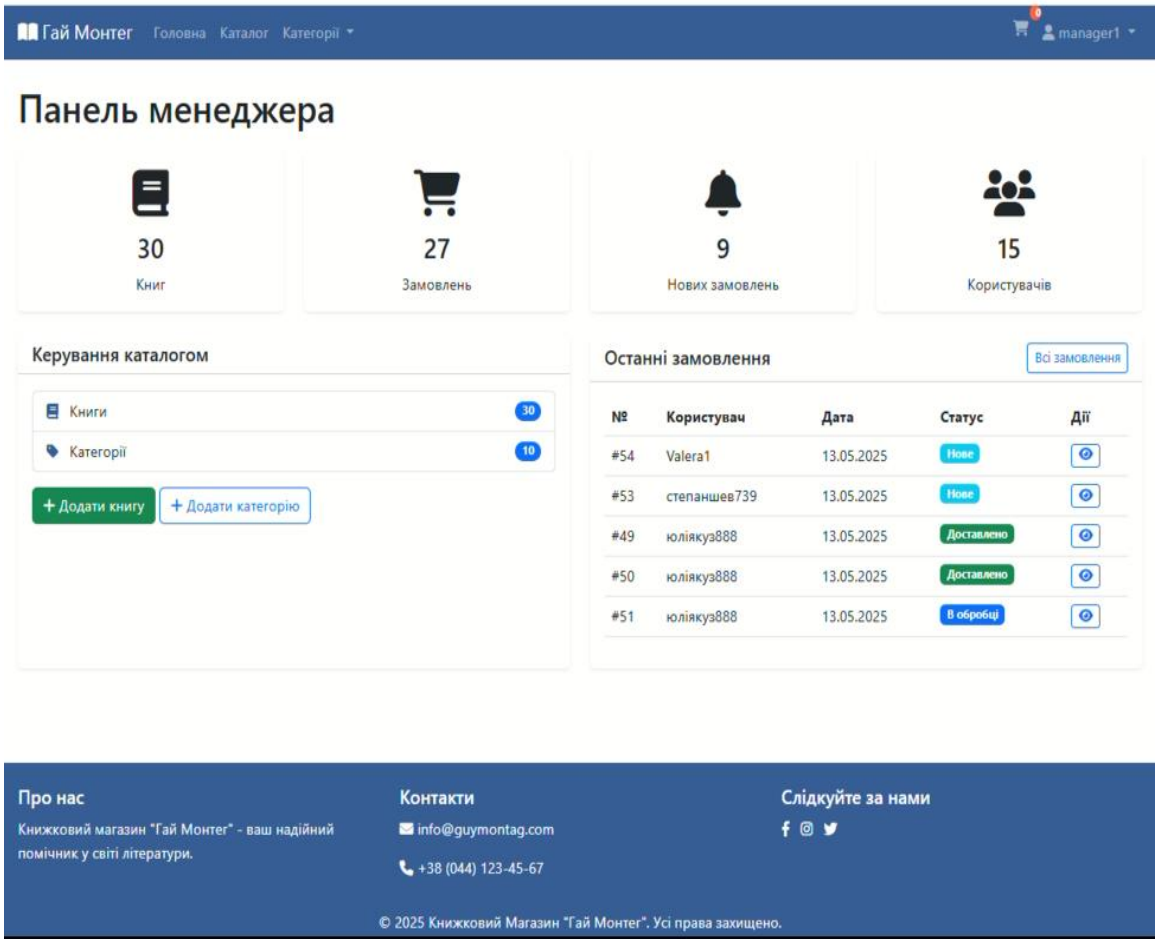

Рисунок Г.13 – Загальний вигляд інтерфейсного вікна панелі менеджера

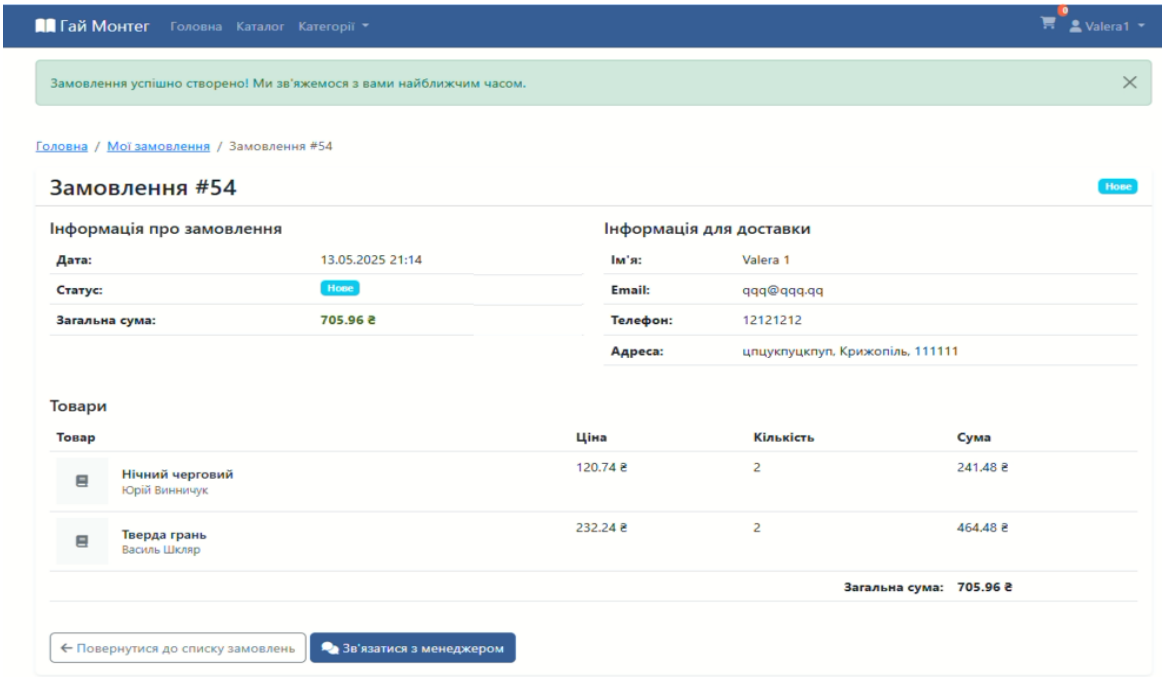

Рисунок Г.14 – Загальний вигляд інтерфейсного вікна деталей замовлення