

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

«Створення веб-сайту книжкового магазину»

Виконав студент групи ЗПЗк-22-1
БЕРЕЗЮК Олексій Віталійович

Керівник роботи:
МОРОЗОВ Андрій Васильович

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

Житомир – 2024

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **БЕРЕЗІЮК Олексій Віталійович**

Керівник роботи: **МОРОЗОВ Андрій Васильович**

Тема роботи: «Створення веб-сайту книжкового магазину»,
затверджена Наказом закладу вищої освіти від «23» лютого 2024 р., №75/с

Вихідні дані для роботи: методи та засоби розробки зручного сервісу для купівлі книг в мережі інтернет, аналіз проблематики подібних сервісів, використання фреймворку Django для розробки веб-сайту.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Морозов А.В.	02.03.24	02.03.24
2	Морозов А.В.	05.04.24	05.04.24
3	Морозов А.В.	01.05.24	01.05.24

РЕФЕРАТ

Робота складається з 70 сторінок, 36 ілюстрацій і 12 використаних джерел.

Мета роботи полягає в тому, щоб розробити повністю функціональний сайт книжкового магазину за допомогою Python Django та низкою допоміжних бібліотек.

У роботі наведено повний процес розробки починаючи від аналізу конкурентів та розробки базової структури, закінчуючи написанням кінцевого коду, що вирішує низькопріоритетні задачі користувача.

Загалом проєкт представляє собою динамічний веб-додаток, побудований на клієнт-серверній архітектурі. Проєкт ще є куди розвивати, але оскільки кінцева мета не несе в собі отримання комерційної вигоди — було реалізовано лише базовий функціонал.

КЛЮЧОВІ СЛОВА: ІНТЕРНЕТ МАГАЗИН, КНИГА, СЕРВЕР

					ІПЗ.КР.Б – 121 – 24 – ПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата				
Розроб.	Березюк О.В.				Створення веб-сайту книжкового магазину	Літера	Аркуш	Аркушів
Керівник	Морозов А.В.						3	70
Н.контр.	Вакалюк Т.А.					Житомирська політехніка, Група ЗІПЗк-22-1		
Затв.	Вакалюк Т.А.							

ABSTRACT

The paper consists of 70 pages, 36 illustrations, and 12 references.

The aim of the work is to develop a fully functional bookstore website using Python Django and a number of auxiliary libraries.

The paper presents the full development process from competitor analysis and basic structure development to writing the final code that solves low-priority user tasks.

Overall, the project is a dynamic web application built on a client-server architecture. The project still has room for development, but since the ultimate goal is not to generate commercial benefits, only the basic functionality has been implemented.

KEYWORDS: BOOK, ONLINE STORE, SERVER

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ ТА ОСОБЛИВОСТЕЙ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ВЕБ-САЙТУ.....	10
1.1 Постановка задачі.....	10
1.2 Аналіз аналогів програмного продукту.....	11
1.3 Вибір архітектури інформаційної системи.....	20
1.4 Інструментальні засоби реалізації програмного продукту та вимоги до апаратних засобів.....	22
Висновки до першого розділу.....	22
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБ-САЙТУ КНИЖКОВОГО МАГАЗИНУ	24
2.1 Визначення варіантів використання та структури системи.....	24
2.2 Розробка бази даних системи.....	30
2.3 Проєктування та реалізація алгоритмів роботи.....	40
2.4 Реалізація функціоналу веб-сайту.....	42
Висновки до другого розділу.....	49
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-САЙТОМ.....	50
3.1 Порядок встановлення та налаштування системи.....	50
3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з веб-сайтом.....	51
3.3 Тестування роботи веб-сайту.....	59
Висновки до третього розділу.....	60
ВИСНОВКИ.....	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	62
ДОДАТКИ.....	64

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ORM — технологія, що спрощує взаємодію з базою даних.

SASS — препроцесор CSS, що додає більш гнучкий функціонал при роботі зі стилями.

SSR (Server-Side Rendering) — тип роботи веб-сайту, коли всі сторінки відтворюються на сервері та відправляються клієнту в готовому вигляді.

MVC (Model-view-controller) — архітектурний шаблон Django, за яким виконується вибірка даних, відображення та керування.

DRY(Don't repeat yourself) — принцип розробки програмного забезпечення, що каже «Не повторюй себе».

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Аркуш
						6
Зм.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. В сучасному світі існує досить велике різноманіття різного роду веб-сайтів. В мережі існує безліч різного роду онлайн крамниць, метою яких є отримання комерційної вигоди чи розширення масштабів бізнесу, а це в свою чергу ніколи не втратить актуальність в капіталістичному світі.

Метою проекту є проектування та розробка веб-сайту книжкового магазину. Проект покликаний продемонструвати основні проблеми пов'язані з розробкою подібного роду веб-застосунків, та способи їх вирішення. Мета проекту також потребує виконання таких завдань як:

1. Аналіз конкурентів. Вибірка переваг кожного, а також їх недоліків;
2. На основі зібраної інформації та теми проекту сформувати базовий функціонал, опираючись на кінцевого користувача;
3. Визначення відповідних технологій;
4. Побудова інформаційної моделі та бази даних;
5. Розробка архітектури та дизайну;
6. Розробка серверної частини;
7. Впровадження дизайну, розробка клієнтської частини;
8. Тестування.

Об'єктом дослідження є проектування, розробка та реалізація веб-сайту книжкового магазину.

Методом дослідження є використання сучасних веб-технологій, об'єктно орієнтованого програмування та дизайну, для забезпечення якісного вихідного продукту.

Випускна робота складається з вступу, 3-х розділів, висновків до кожного розділу, списку використаних джерел із 12 найменувань та 36 ілюстрацій по тексту. Повний обсяг роботи становить 70 сторінок, з них 63 сторінки основного тексту.

РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ ТА ОСОБЛИВОСТЕЙ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ВЕБ-САЙТУ

1.1 Постановка задачі

Основною метою будь-якого інтернет-магазину є продаж.

В наш час відвідування веб-сайтів є чимось на зразок рутини, оскільки інтернет дає змогу робити наше життя простішим. В тому числі простішим для ведення бізнесу чи будь-якої іншої підприємницької діяльності.

В світі інтернет речей сприйняття користувача дещо відрізняється від реального світу. Купить користувач книгу чи ні багато в чому залежить від досвіду користувача — від його сприйняття інтерфейсу та зручності користування.

Досвід користувача не може існувати без самого користувача, тому завжди при розробці будь-якого додатку UX-дизайнери проводять співбесіду з потенційними користувачами. Окрім того цей досвід можна частково перейняти аналізуючи вже готові продукти на ринку, інтерв'ю та скарги користувачів.

Тому для успішного виконання поставленої задачі потрібно:

- Розробити Технічне Завдання продукту;
- Розробити інформаційну систему згідно якої буде розроблятися дизайн, frontend та backend частини веб-сайту;
- Розробити чергу UML-діаграм для полегшення розробки та уникнення непорозумінь в послідовності виконання завдань;
- Розробити дизайн згідно інформаційної системи та потреб користувачів;
- Перенести дизайн в реалізацію на Frontend;
- Розробити Backend частину згідно ТЗ та інформаційної архітектури;
- Зв'язати Frontend та Backend частини в одне ціле;
- Розробити схему розгортання серверу та власне розгорнути веб-сайт.

1.2 Аналіз аналогів програмного продукту

На сьогоднішній день лише серед українського сегменту в інтернеті існує безліч веб-сайтів по продажу книг не беручи до уваги маркетплейс гігантів по типу Rozetka, що містять в собі безліч товарів різного роду.

Зрештою наша розробка виключно спеціалізується на книгах, тому щоб сильно не розтягувати проаналізуємо лише декілька з них.

Першим на огляд обрали магазин «Є-Книгарня». На момент написання цього абзацу цей книжковий магазин видавався першим за запитом в Google тому логічно буде припустити, що він є один з найпопулярніших магазинів.

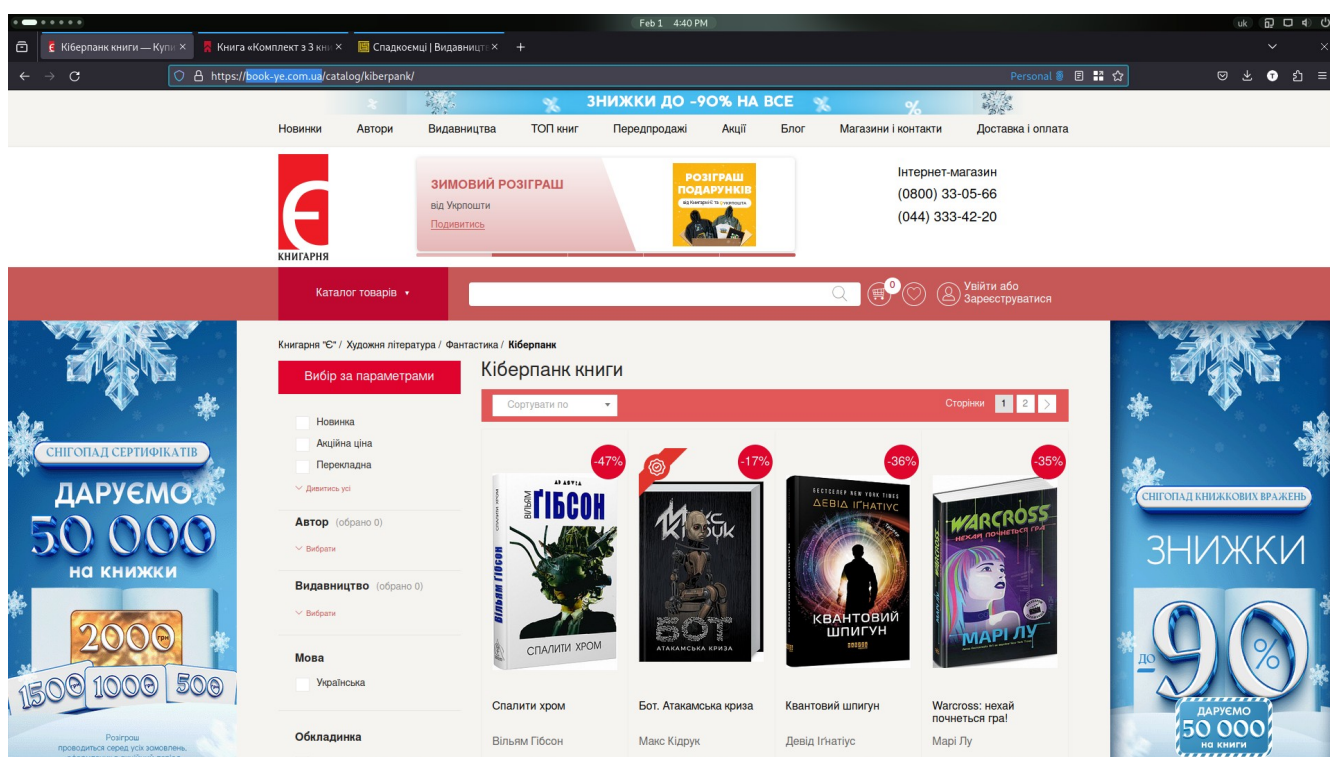


Рис 1.1 (book-ye.com.ua)

На перший погляд інтерфейс здається надзвичайно складним - багато яскравих елементів, що забирають на себе когнітивне навантаження та розсіює увагу. У всьому іншому нам зустрічає меню товарів у вигляді звичної всім плитки та доволі мінімалістичним меню сортування (зазвичай при виборі книги більшого і не потрібно).

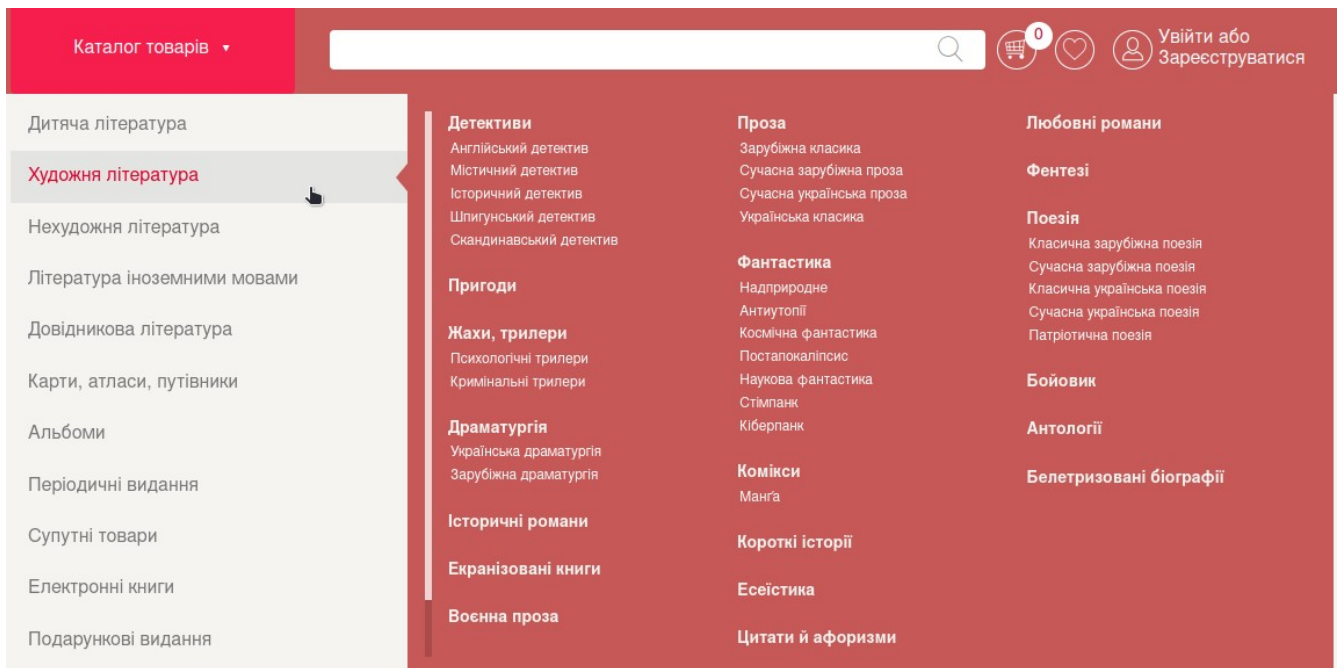


Рис 1.2 (book-ye: каталог товарів)

Каталог товарів доволі простий та зрозумілий, завжди в полі зору користувача. Вся література відсортована відповідно по категоріям, окрім того деякі категорії мають підкатегорії для більш специфічних запитів у користувача.

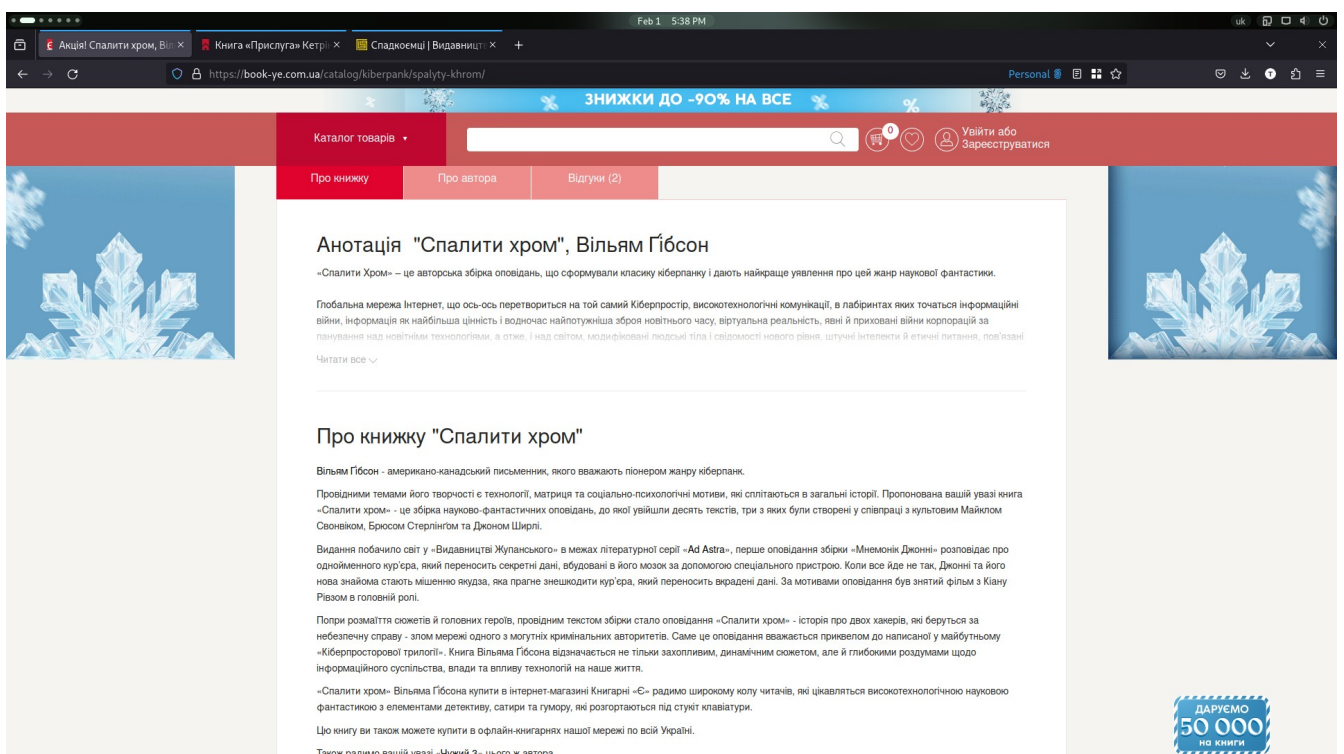


Рис 1.3 (book-ye: опис)

					Аркуш
					10
Зм.	Арк.	№ докум.	Підпис	Дата	

На сторінці товару можна знайти всю необхідну інформацію про товар. Єдине, що не зазначається - це тип обкладинки та фізичні характеристики книги, попри те можна оглянути фрагмент книги перед покупкою. Завдяки користувальницькому акаунту можна в майбутньому залишити відгук чи поставити питання.

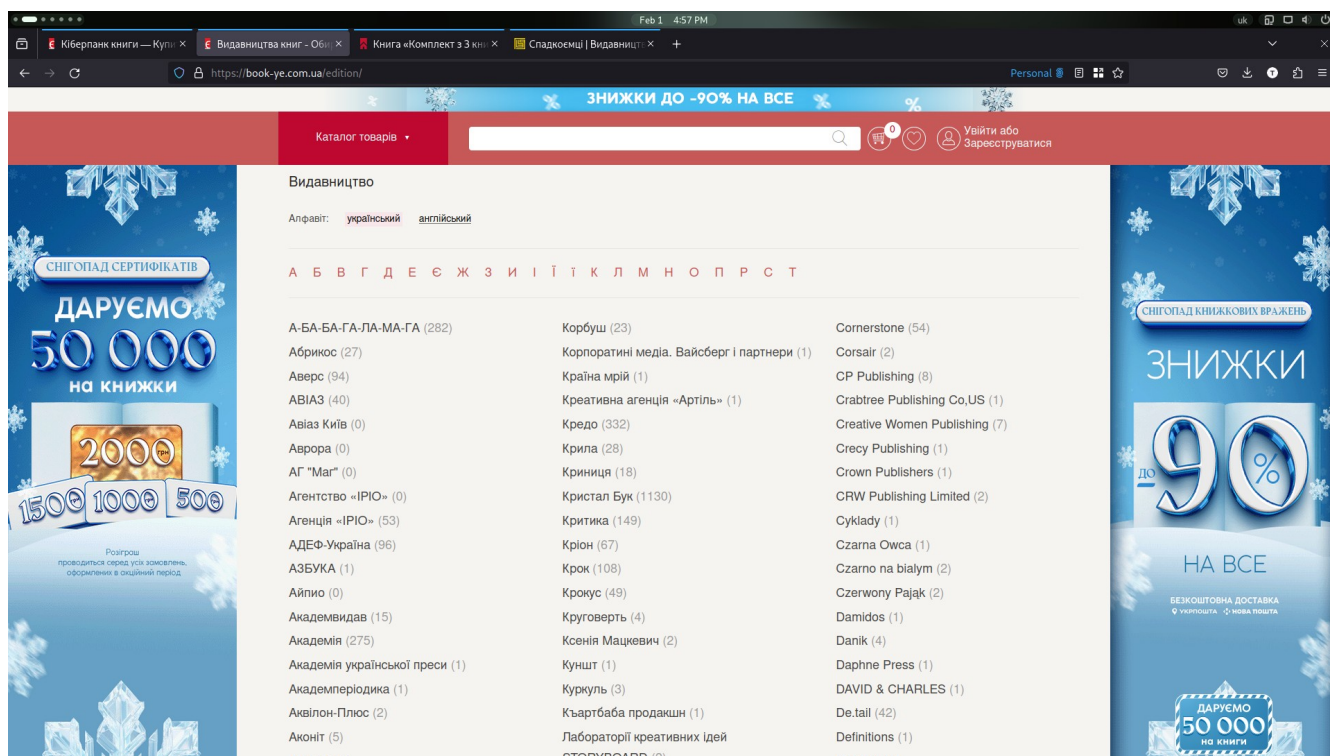


Рис 1.4 (book-ye: видавництва)

Також варто зазначити про деякі особливості крамниці..

Першою такою особливістю є окремий пункт різних видавництв. Їх може налічуватись декілька сотень, тому величезним плюсом є алфавітний покажчик.

Друга і остання його особливість це наявність сторінки «Передпродажі книг». На цій сторінці користувач може створити передзамовлення на книгу, що ще не з'явилась на полицях крамниць і бути в числі перших хто її отримає.

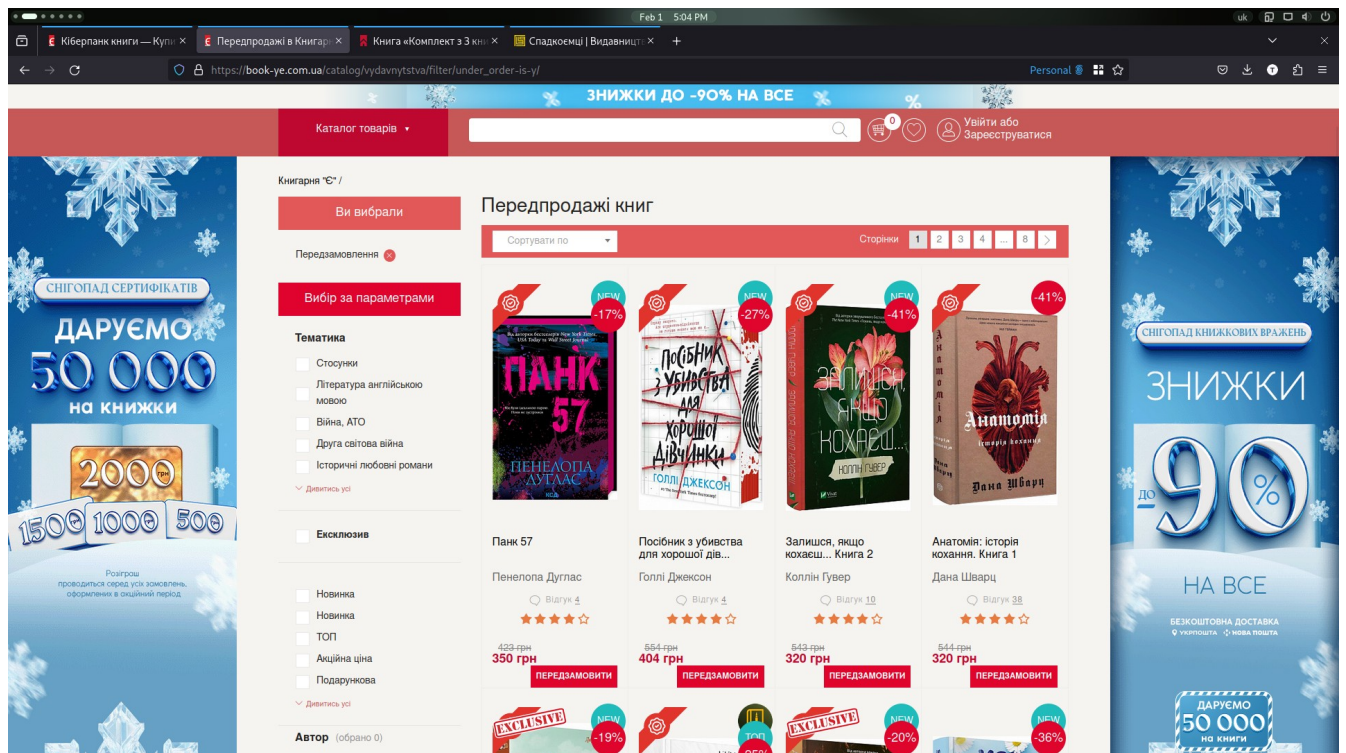


Рис 1.5 (book-ye: передзамовлення)

Підсумок:

- Перевантажений інтерфейс;
- Зручний каталог;
- Неповний опис книги;
- Є можливість передзамовлення;
- Є можливість переглянути фрагмент книги;
- Доволі примітивне сортування та фільтрування товарів;
- Є окрема сторінка з видавництвами на запит користувача.

Наступна на огляд крамниця— «Клуб Сімейного Дозвілля». Вона існує на ринку України ще з 2000-их років і має чи не найбільше досвіду з-поміж інших крамниць.

При вході на сайт нас зустрічає доволі простий але приємний лендінг, що коротко дає зрозуміти про основні переваги сайту. Окрім того наведені приклади самих популярних книг (чи навіть збірок).

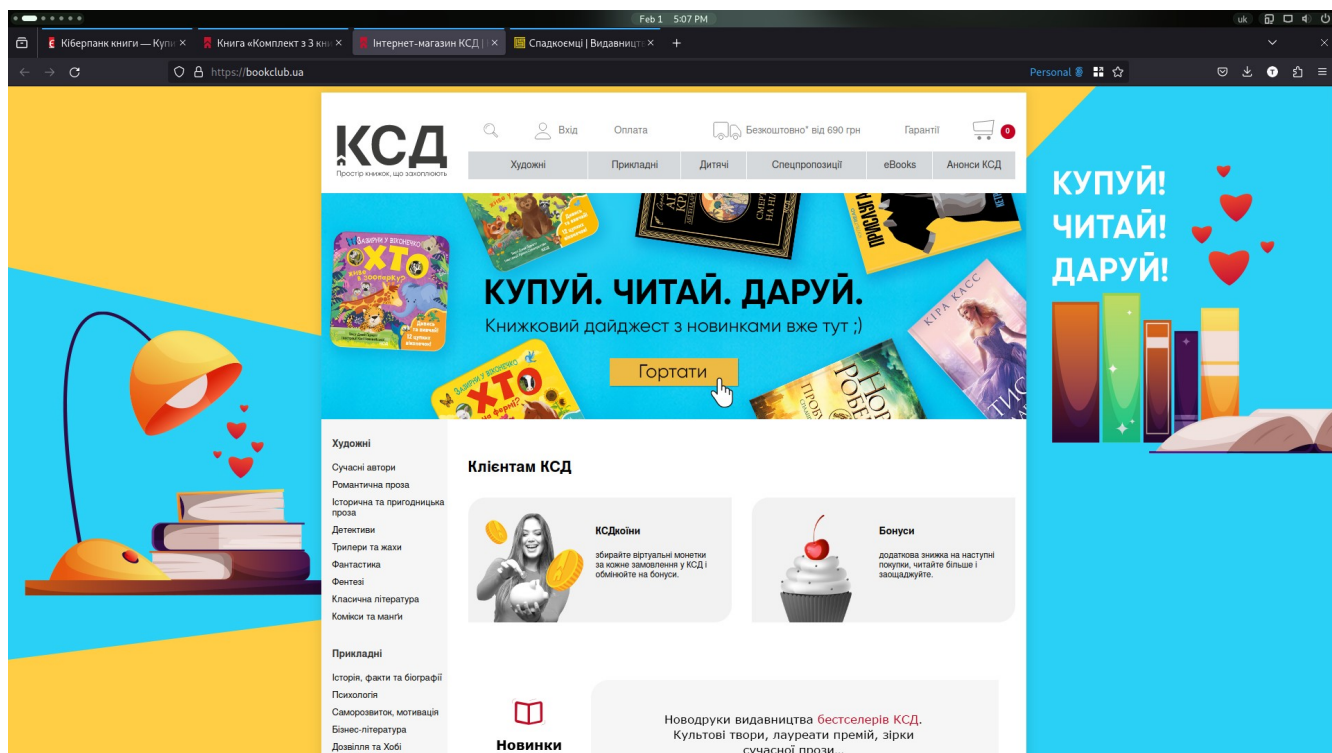


Рис 1.6 (bookclub.ua)

Одна з головних особливостей - на даному сайті клікабельний динамічний фон. Він змінюється залежно від того на якій сторінці ви перебуваєте. Це скоріше варто віднести до мінусів, адже це не дає ніяких приємних вражень користувачу, а скоріше навпроти - може зіпсувати досвід користувача великим когнітивним навантаженням. При кліку на фон спрацьовує посилання, що відкриває сторінку з журналом від КСД. Цей момент теж варто занести в мінуси оскільки подібні журнали вже непотрібні сучасному користувачу. Набагато краще було замінити його розділом «Новини» (який до речі кажучи теж є).

Зм.	Арк.	№ докум.	Підпис	Дата

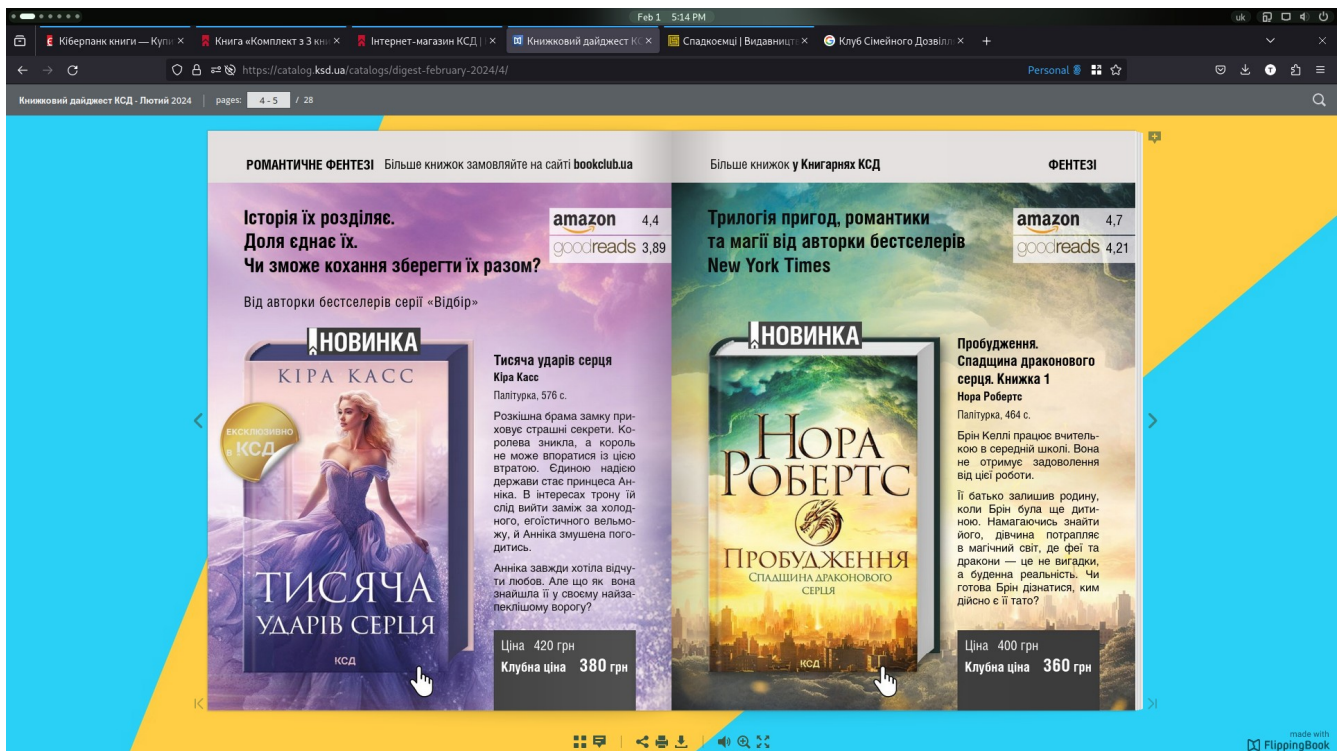


Рис 1.7 (bookclub: журнал)

На сайті немає такого ж каталогу, як в попередньому прикладі.

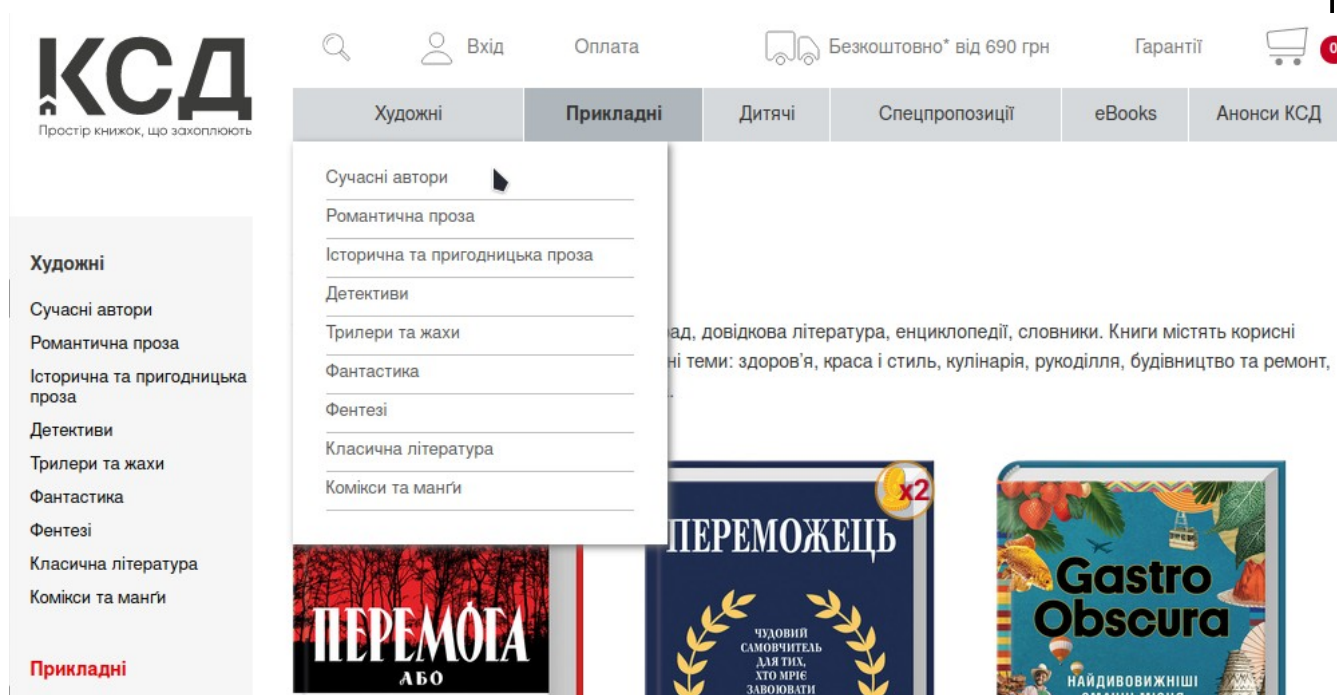


Рис 1.8 (bookclub: каталог)

Вся література поділена на прикладу, художню та дитячу з власними підкатегоріями. Цей каталог також завжди на очах, проте інформацію

дублюється з колонкою ліворуч, що є поганим прикладом інформаційної архітектури. Також з точки зору користувацького досвіду, деякі дії на сайті не є передбачуваними, що теж негативно впливає на сприйняття.

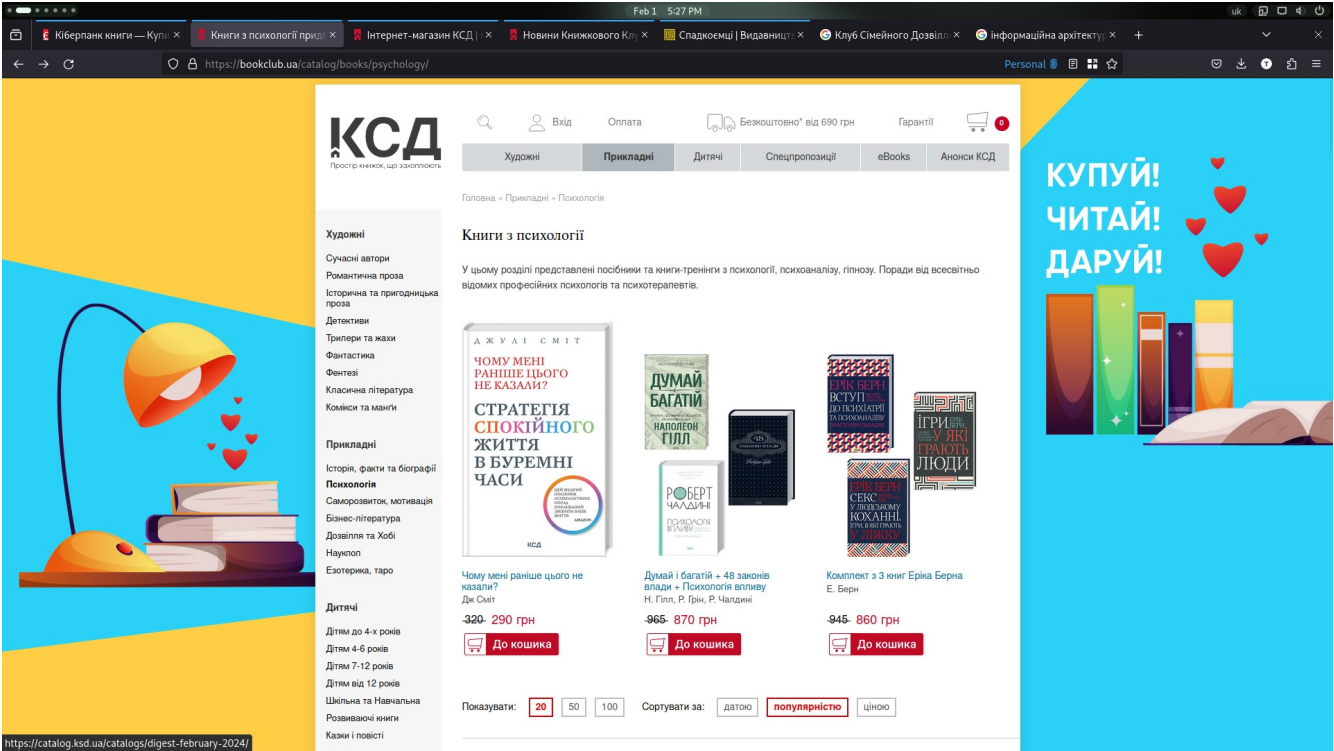


Рис 1.9 (bookclub: товари)

І також хочеться зазначити про простоту візуального оформлення плитки товарів. Окрім того, є зручніші режими сортування ніж на минулому сайті.

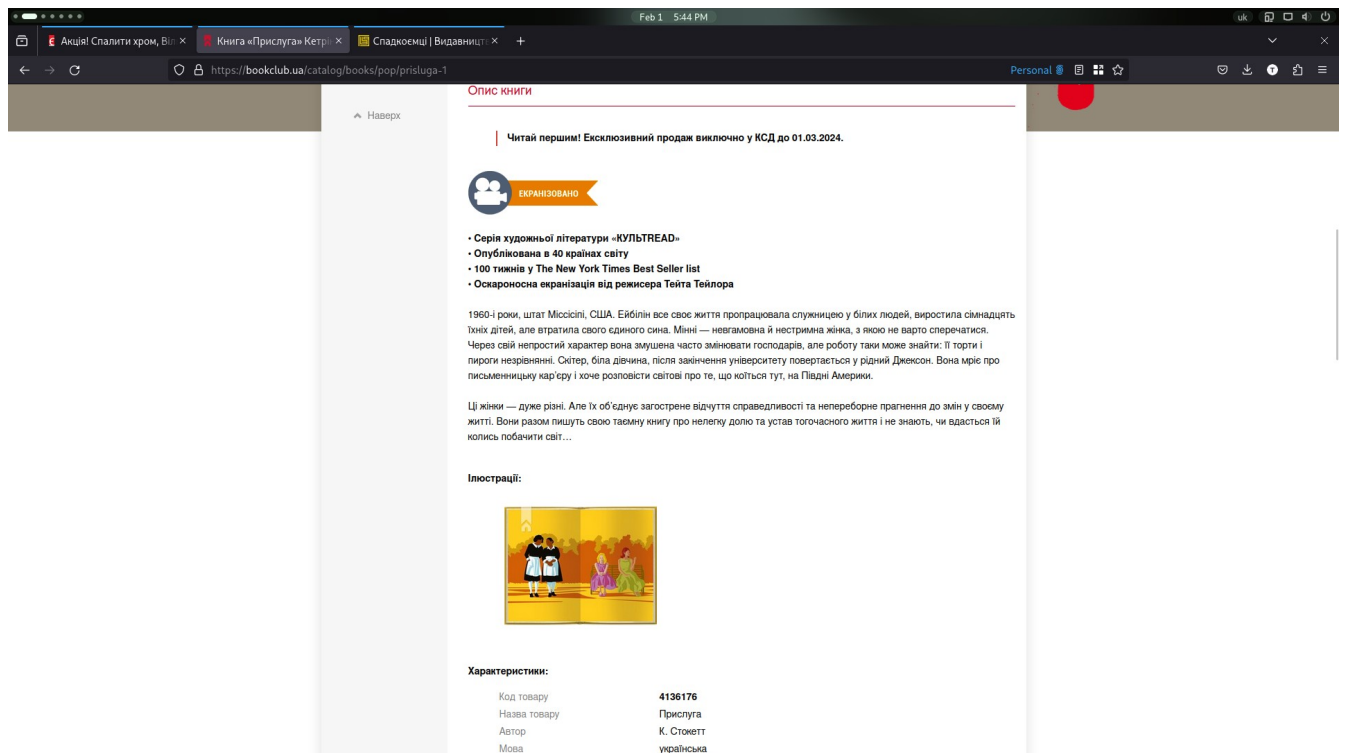


Рис 1.10 (bookclub: опис)

Опис книги, як і в попередньому прикладі містить в собі всю необхідну інформацію та фізичні характеристики.

Підсумок:

- Легкий і красивий інтерфейс;
- Посередній каталог;
- Повний опис книги;
- Сторінка з анонсами та спецпропозиціями на різні смаки;
- Є можливість переглянути фрагмент книги;
- Доволі примітивне але зручне сортування та фільтрування товарів.

Останнім на аналіз йде магазин «Вавилонська Бібліотека». З ходу сайт зустрічає дуже мінімалістичним і приємним оформленням.

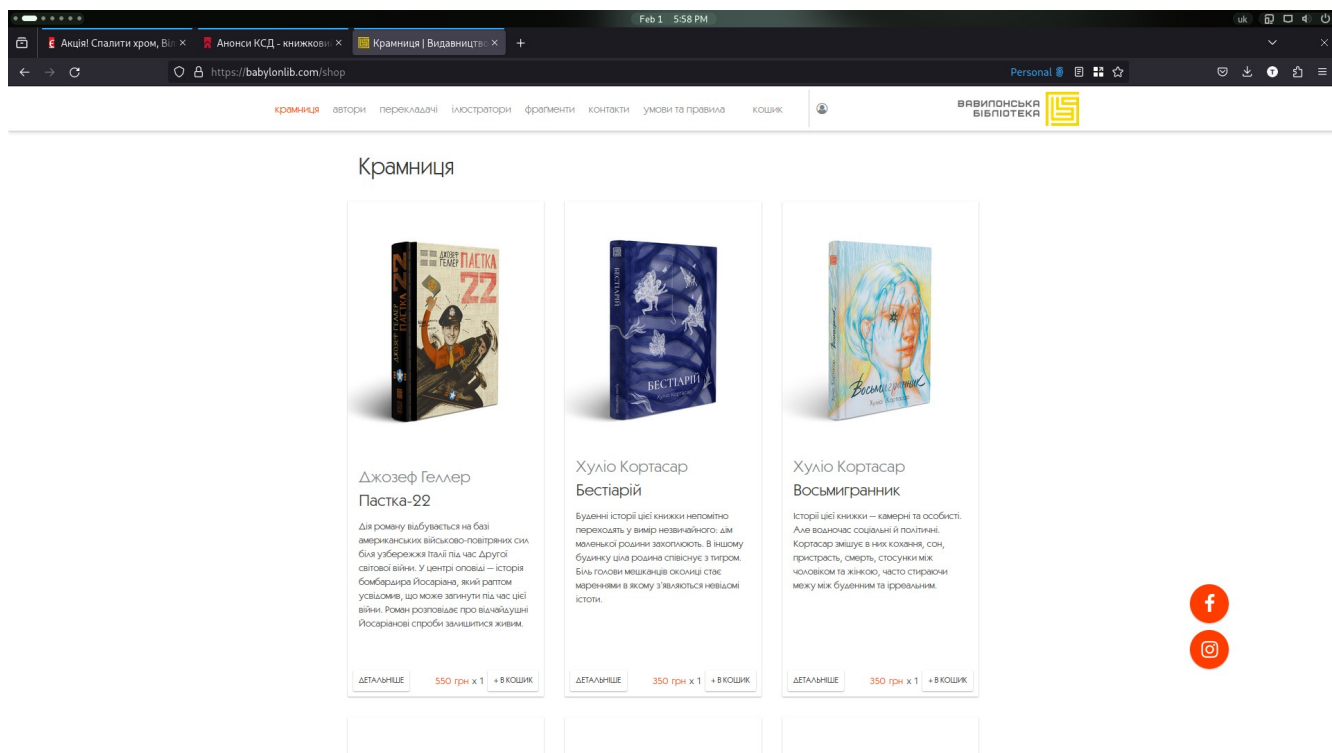


Рис 1.11 (babylonlib: головна)

Оформлення хоч і приємне, проте розмір шрифтів краще зробити більшим.

Також на сайті немає каталогу. Це пов'язано з тим, що він продає доволі мало книг, проте на сайті є розділи з перекладачами та ілюстраторами, що є його приємною особливістю (інтернет має знати героїв в лице).

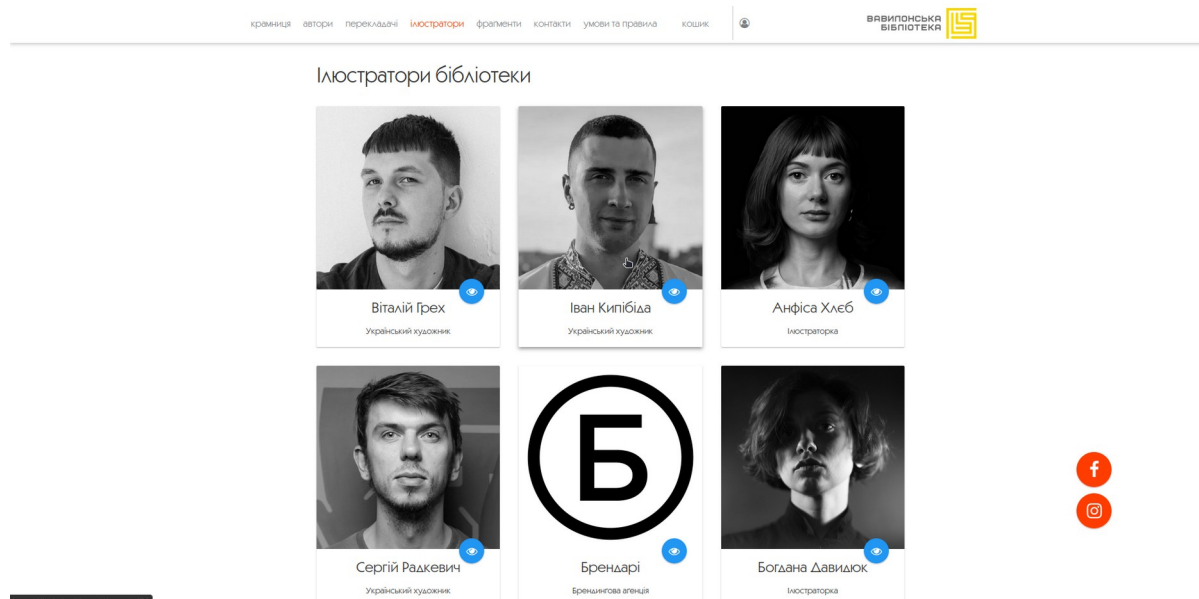


Рис 1.12 (babylonlib: ілюстратори)

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		17

Опис книг дуже мінімалістичний. Все необхідне можна побачити на одному екрані без необхідності скролу.

Підсумок:

- Легкий і красивий інтерфейс;
- Відсутній каталог та сортування/фільтрування;
- Дуже малий опис книги;
- Сторінки перекладачів та ілюстраторів;
- Є можливість переглянути фрагмент книги.

1.3 Вибір архітектури інформаційної системи

Наступним важливим етапом розробки веб-сайту є розробка його архітектури, адже якою б красивою не була машина без двигуна вона нікуди не поїде.

Будь-який повноцінний веб-додаток має Backend та Frontend частини. Окрім того додаток буде мати клієнтську та серверну частину. Архітектура клієнт-сервер є домінуючою концепцією у створенні розподілених мережних застосунків.

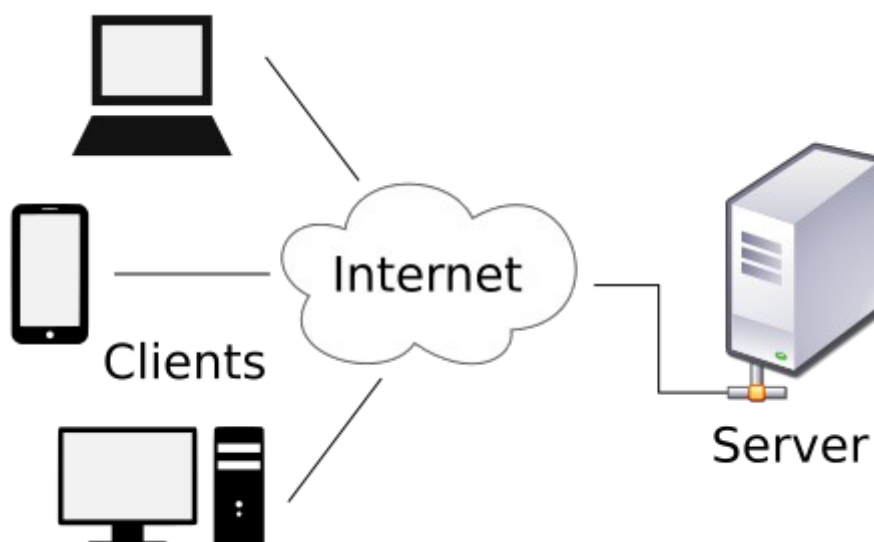


Рис 1.13 (Джерело: Вікіпедія — клієнт-серверна архітектура)

Зм.	Арк.	№ докум.	Підпис	Дата

Окрім того, веб-сайти, що оперують великими обсягами інформації часто використовують власну базу даних. Тому сервер потрібен бути розрахованим для зберігання великої кількості інформації, в тому числі медіа (зразки книг, фотографії обкладинок тощо).

А тепер по порядку...

Для backend (або ж серверної) частини найкращим чином буде використати Model-View-Template архітектуру. Архітектура Model-View-Template, також відома як MVT, є шаблоном проектування програмного забезпечення, який використовує Django. Весь концепт шаблону зводиться до трьох понять:

- models — Моделі;
- templates — Шаблони;
- views — Представлення.

Дана архітектура чудово підходить проектам, в яких потрібно багато взаємодіяти з базами даних. Вибірка необхідної інформації та рендер сторінки відбувається зі сторони сервера. Таким чином можна зменшити навантаження зі сторони frontend, в тому числі і для розробника.

Для клієнтської частини (frontend) звичайно що буде використовуватись архітектура Multi Page Application або ж Багатосторінковий Додаток. Багатосторінковий додаток (МРА) складається з декількох сторінок із статичною інформацією (текстом, зображеннями тощо) та посиланнями на інші сторінки з тим самим вмістом. Вибір цієї архітектури є очевидним, оскільки від доволі зручно лягає під архітектуру серверної частини та мінімізує час на розробку.

1.4 Інструментальні засоби реалізації програмного продукту та вимоги до апаратних засобів.

Починати реалізацію такого продукту звісно, що з концепт дизайнів. Це допоможе нам краще зрозуміти архітектуру застосунку, та сформувати його фінальне бачення ще на ранніх стадіях розробки. Окрім того робота повинна мати презентабельний та привабливий вигляд. Для виконання цього завдання чудово підійде інструмент під назвою Figma.

Наступним етапом буде розробка Backend частини та бази даних, яку вона буде використовувати для зберігання власне самих даних. Найкращим варіантом буде фреймворк Django в сумісності з мовою програмування Python. Django має в собі реалізовану ORM, що в свою чергу неабияк спростить взаємодію з базою даних, оскільки візьме реалізацію SQL-запитів на себе. Окрім того, даний фреймворк використовує MVC архітектуру, що ідеально підходить для подібних проєктів.

Реалізація Front-End частини буде відбуватись за посередництвом мови розмітки HTML5 та препроцесора SASS, який неабияк пришвидшить процес написання стилів для кожної сторінки. Окрім того, SASS сам генерує файли CSS, що в свою чергу слідує принципу DRY, та захищає нашу таблицю стилів від повторень.

Для особливих випадків, також може згодитись скриптова мова програмування JavaScript, що може взяти на себе виконання сценаріїв, в яких ми не можемо скористатись сервер рендерингом.

Щодо вимог до апаратного засобу - підійде будь-який персональний комп'ютер з можливістю виходу в мережу інтернет за допомогою сучасних веб-браузерів. Оскільки це веб-додаток більшого і не потрібно.

Висновки до першого розділу

У першому розділі даної роботи було сформовано основні етапи виконання завдання, такі як аналіз конкурентів та було визначено за допомогою яких технологій найкраще буде реалізувати свою ідею. По результатам аналізу ми дійшли висновку, що найкращим чином буде створити багатосторінковий додаток з рендером на стороні серверу, з Model-View-Template архітектурою.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Аркуш
						21
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБ-САЙТУ КНИЖКОВОГО МАГАЗИНУ

2.1 Визначення варіантів використання та структури системи

В першу чергу потрібно зрозуміти які вимоги має задовільняти даний продукт для потенційного покупця. Зробити це неважко, якщо грамотно систематизувати зібрану інформацію з Розділу №1. Ця інформація допоможе нам забезпечити краще розуміння того, що має вийти в кінцевому результаті.

Бізнес-вимоги:

1. Основні цілі: Проєкт створюється з метою збору, обробки й зберігання даних про книжковий магазин, його товари та розпродажі. На головних сторінках реалізована функції пошуку та фільтрації даних, на основі того, що зберігається в базі даних. Окрім того, кожен користувач має можливість додавати товари в обране, замовляти та залишати відгуки.

2. Представлення проєкту: Проєкт буде реалізовано у вигляді веб-сайту, що містить структуровану інформацію по даній тематичній області.

Вимоги користувачів:

Зовнішні користувачі:

1. Повна навігація по сайту: можливість переглядати каталог та його вміст, пошук та фільтрація та відповідних сторінках продукту;
2. Можливість авторизації чи реєстрації на сайті;
3. Можливість перегляду всієї інформації на сайті включно з відгуками користувачів;
4. Можливість замовлення товару без необхідності реєстрації.

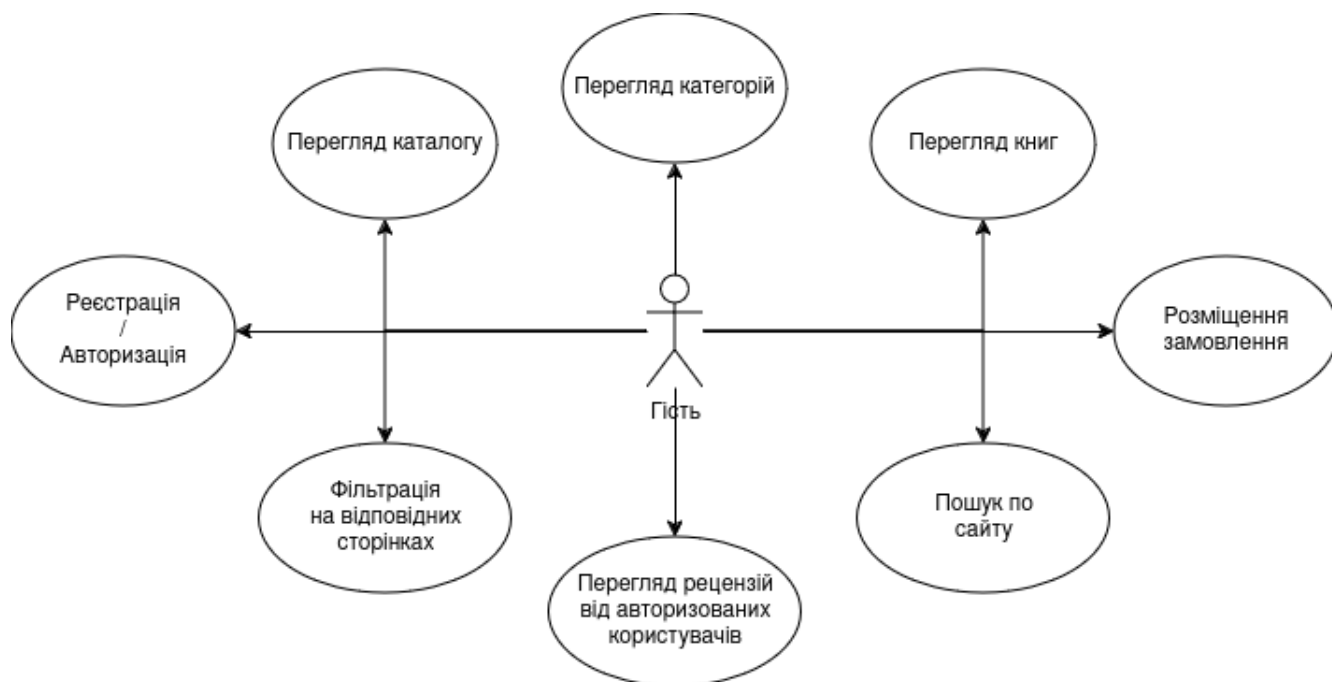


Рис. 2.1 Діаграма варіантів використання (зовнішній користувач)

Внутрішні користувачі:

1. Наслідування всіх можливостей зовнішніх користувачів;
2. Можливість додавати товар в обране;
3. Можливість додавати товар в кошик;
4. Можливість замовляти товар з кошика;
5. Можливість зміни логіну / паролю облікового запису користувача;
6. Можливість видалення власного облікового запису користувача;
7. Можливість розміщення відгуку на сторінці продукту;
8. Можливість видалення власного відгуку на сторінці продукту.

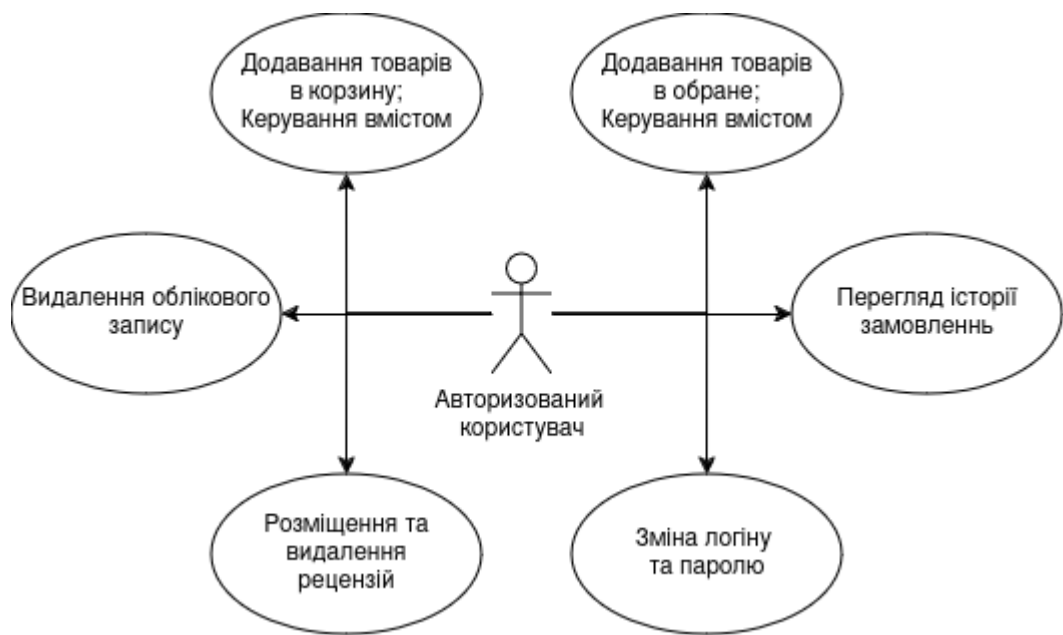


Рис. 2.2 Діаграма варіантів використання (внутрішній користувач)

Персонал

1. Можливість зручного внесення інформації про товари;
2. Керування обліковими записами користувачів;
3. Додавання знижок на обрані товари;
4. Можливість зміни інформації та її видалення;
5. Можливість приховувати товари.

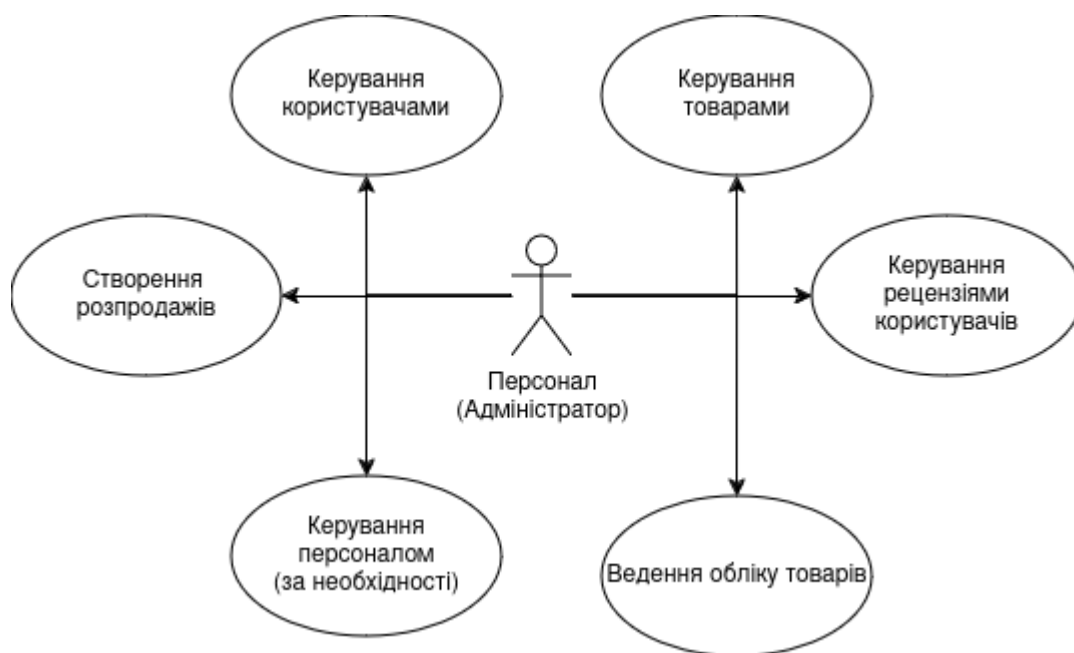


Рис. 2.3 Діаграма варіантів використання (персонал)

Функціональні вимоги:

1. Авторизація / реєстрація. Користувачі повинні мати змогу створювати та керувати власними обліковими записами.
2. Конфіденційність. На сайті не повинна зберігатись будь-яка конфіденційна інформація користувача.
3. Створення, редагування та видалення продуктів, рецензій, облікових записів.
4. Система повинна коректно та автоматично відображати необхідну інформацію.

Нефункціональні вимоги:

1. Сприйняття:

- Час, необхідний для початку вільного користування сайтом має не перевищувати 3-ьох хвилин;
- Час відгуку системи для будь-яких запитів не повинен перевищувати 0.5 мс;

Зм.	Арк.	№ докум.	Підпис	Дата

- Інтерфейс повинен бути дружнім до користувача та інтуїтивно зрозумілий.

2. Надійність:

- Середній час безвідмовної роботи не менше 30 днів;
- Норма помилок на кожні десять тисяч рядків коду не повинна перевищувати 2;

3. Швидкість:

- Сайт повинен швидко виконувати запити користувачів.

4. Можливості використання:

- Масштабування — додавання нових функцій не повинно призводити до збою роботи веб-сайту.

Використовуючи зібрану інформацію, а саме функціональні вимоги та діаграми використання ми можемо виявити ключові елементи, необхідні для розробки програмного каркасу системи. Але перед тим, як формувати діаграму класів, базу даних та діаграму активностей, потрібно трішки знати про те, як влаштований фреймворк Django.

В першу чергу Django складається з застосунків. Кожен застосунок містить свої власні моделі бази даних, представлення, фільтри і так далі. І кожен застосунок покликаний вирішувати задачі в межах своїх моделей, а саме вносити зміни у власні моделі та формувати їх представлення для потенційного користувача.

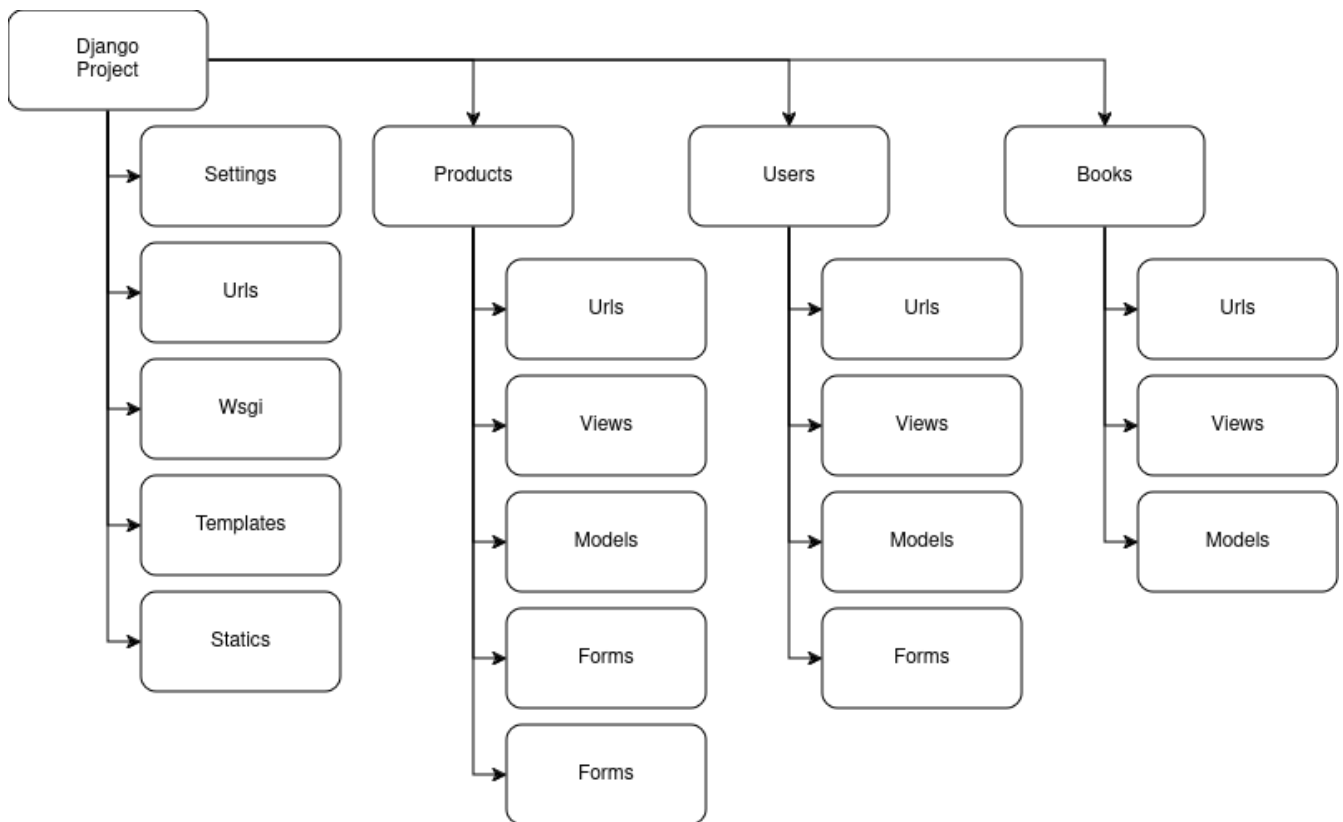


Рис 2.4 Структура системи

Одже для забезпечення всіх необхідних потреб, нам достатньо всього трьох основних застосунків, а саме Products, Users та Books.

Застосунок Books відповідає за керування книгами, їх жанрами, авторами та класифікації.

Застосунок Product покликаний вирішити питання з керуванням вмісту крамниці та його відображенням.

Застосунок Users бере на себе зобов'язання по взаємодії з авторизованими користувачами, корзиною та списком бажань.

Кожен з цих застосунків матиме свої таблиці в адмін панелі, що дає можливість адміністратору доволі зручно керувати вмістом веб-сайту.

2.2 Розробка бази даних системи

Попри це, з яких би застосунків не складався веб-сайт, ми повинні чітко розуміти, який вигляд матиме база даних. Для цього використаємо зібрані дані на основі Розділу №1 та функціональних вимог Розділу №2.

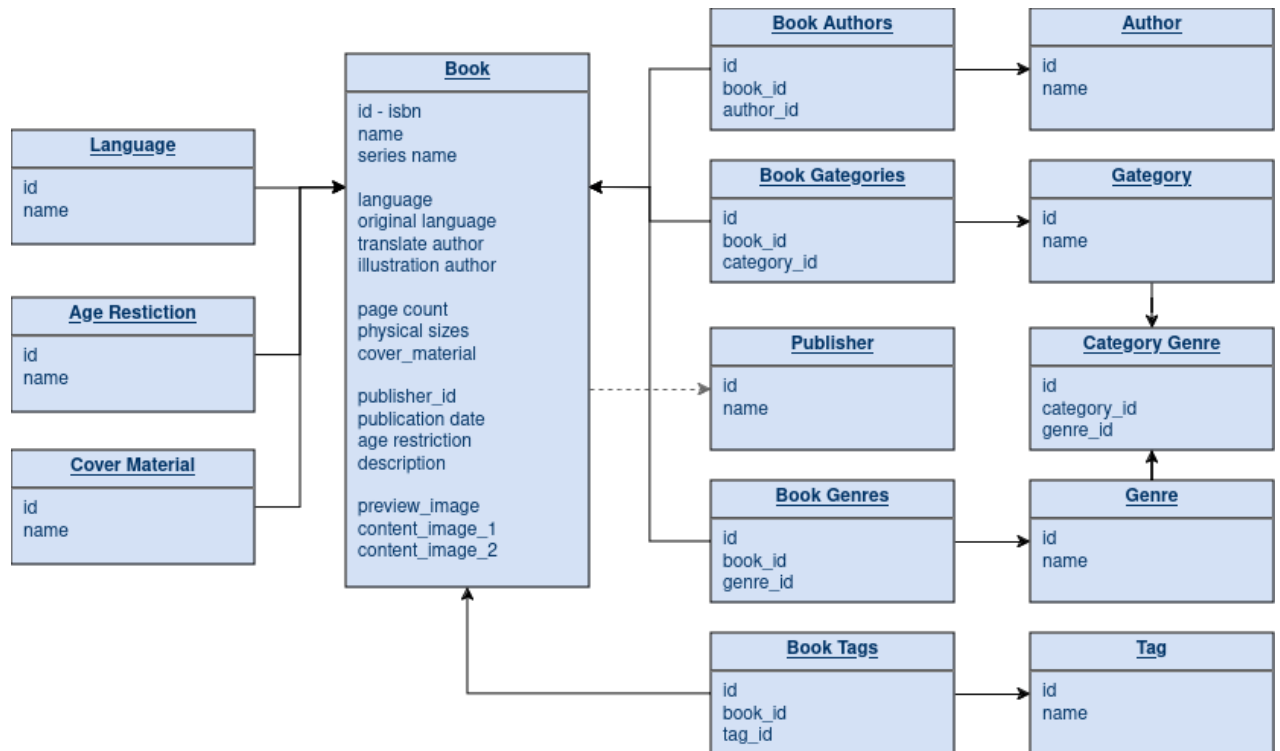


Рис 2.5 Схематичне зображення бази даних (Книги)

Таблиця з книгами є ключовою таблицею в базі даних. Кожна книга може мати довільну кількість авторів, жанрів та відноситись до декількох категорій одночасно. Окрім того, кожна книга буде містити в собі теги, за якими можна спросити пошук по сайту.

Наступними важливими таблицями бази даних є таблиці продуктів, які візьмуть на себе завдання бізнес вимог Розділу №2.

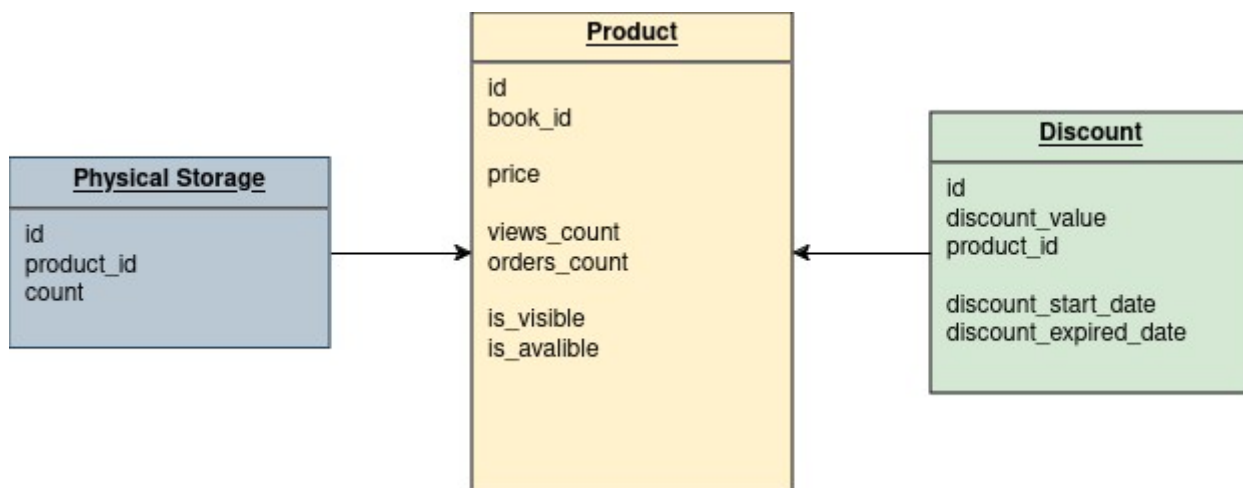


Рис 2.6 Схематичне зображення бази даних (Продукти)

Кожен продукт додатково містить в собі лічильник переглядів та замовлень, та має прапорці видимості та доступності. Таблиця зі знижками слугує для створення розпродаж. Кожен розпродаж має свою дату початку та кінця. Таблиця складу слугує для ведення обліку.

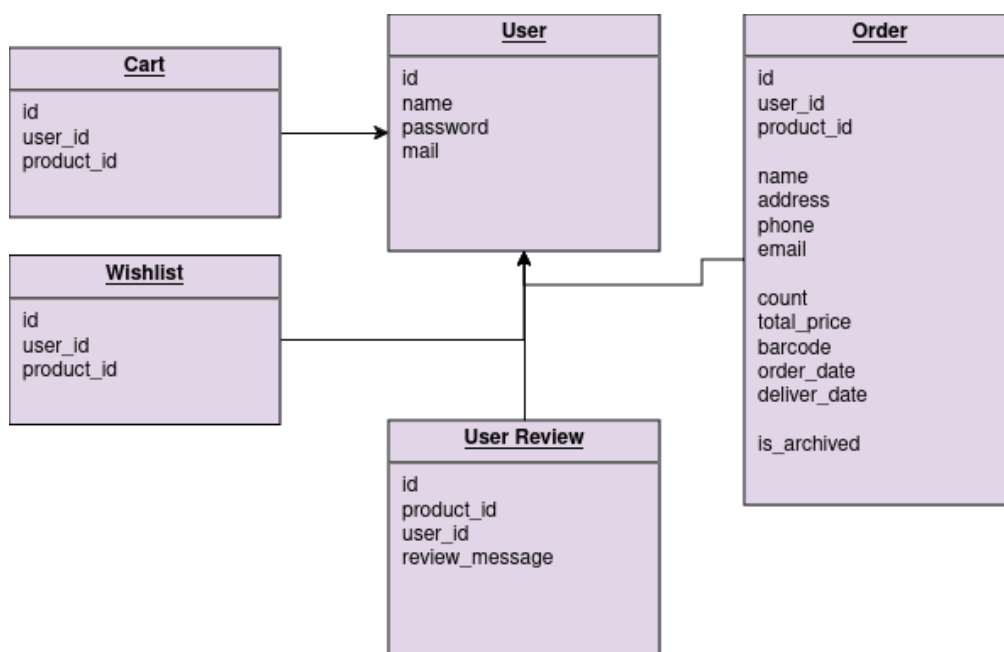


Рис 2.7 Схематичне зображення бази даних (Користувачі)

Таблиці пов'язані з користувачами задовільняють вимоги лише авторизованих користувачів, за виключенням таблиці замовлень. Адже бізнес має бути зацікавленим в тому, щоб продавати власні товари й для тих

користувачів, які не хочуть реєструватись. В таких випадках ідентифікатор користувача в таких таблицях буде позначатися як Null. Дані про замовлення будуть видимі лише адміністраторам чи відповідному персоналу.

Що стосується ключової таблиці користувача — вона має досить мінімалістичний вигляд. Не містить в собі ні полів з правами доступу, ні полів статусу і тому подібних, що на перший погляд може здатися досить дивним. Річ в тому, що дана таблиця зображена досить образно. Django надає власну систему управління користувачами та групами і автоматично створює власні моделі користувачів, надає їм базові права, стежить за сесіями. За необхідності цю модель можна перевизначити, додавши власні поля чи прапорці, проте поки що такої необхідності немає.

Реалізація бази даних буде здійснюватися на основі реляційної бази даних SQLite3 під управлінням Django ORM. Почнемо з налаштування файлу конфігурації. Для забезпечення з'єднання з базою даних потрібно вказати наступні параметри:

DATABASES — словник, що містить в собі перелік баз даних, які використовуються в проєкті;

default — база даних в словнику, що використовується по змовчанню;

ENGINE - бекенд баз даних;

NAME - ім'я бази даних.

Окрім того, є додаткові параметри - USER, PASSWORD, HOST та PORT, які необхідно вказати при використанні баз даних типу PostgreSQL, MySQL, Oracle і тд.

У висновку маємо наступне:

```
DATABASES = {  
    'default': {
```

```

        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

```

Можемо переходити до реалізації моделей, опираючись на схематичні зображення баз даних. (Рис 2.5 — 2.7). Діаграма класів не буде сильно відрізнятись від діаграми бази даних. Зрідка окремі класи будуть містити окремі методи, але розберемо це трішки нижче.

Для керування книгами, створимо модель Book в додатку Books.

```

class Book(models.Model):
    class Meta:
        verbose_name = _("Book")
        verbose_name_plural = _("Books")

    def __str__(self) -> str:
        return f"{self.title}"

```

Клас Meta потрібен для коректного відображення даних в адмін панелі, вказання методу сортування по змовчуванню тощо. Окрім того, кожен клас моделей має містити в собі метод __str__(), який конвертує об'єкт класу у звичайний рядок для відображення текстом.

```

        isbn = models.DecimalField(max_digits=13,
        decimal_places=0, primary_key=True)
        title = models.CharField(max_length=100, blank=False)
        series = models.CharField(max_length=100, blank=True,
        null=True, default=None)
        description = models.CharField(max_length=3000,
        blank=False)
        age_restriction = models.ForeignKey(to=AgeRestriction,
        null=True, default=None, on_delete=models.DO_NOTHING)
        page_numbers = models.IntegerField(default=0)

```

```

        physical_size = models.CharField(max_length=30)
        cover_material = models.ForeignKey(to=CoverMaterial,
null=True, on_delete=models.DO_NOTHING)
        language = models.ForeignKey(
            to=Language,
            on_delete=models.SET_NULL,
            null=True,
            related_name='book_language'
        )
        publisher = models.ForeignKey(to=Publisher,
on_delete=models.SET_NULL, null=True, default=None)
        release_date = models.DateField(blank=True, null=True,
default=None)
        book_authors = models.ManyToManyField(to=Author)

```

Поле автора фактично є кінцем головних полів, що має містити в собі будь-яка книга. Починаючи з мови оригіналу буде йти додаткова інформація, яка є необхідною для потенційного покупця:

```

        original_language = models.ForeignKey(
            to=Language,
            on_delete=models.SET_NULL,
            null=True,
            related_name='book_original_language'
        )
        translate_author = models.CharField(max_length=100,
blank=True)
        illustration_author = models.CharField(max_length=100,
blank=True)

```

Поля категорій, жанрів та тегів необхідні для здійснення пошуку та сортування по сайту:

```

categories = models.ManyToManyField(to=Category)
genres = models.ManyToManyField(to=Genre)
tags = models.ManyToManyField(to=Tag)
preview_image = models.ImageField(

```

```

        max_length=1024,
        upload_to=get_preview_image_path,
        verbose_name=_("Preview Image"),
        validators=(FileExtensionValidator(['png', 'jpg']),),
        blank=False
    )

```

І власне картинки, щоб покупець знав, що купує:

```

content_image_1 = models.ImageField(
    max_length=1024,
    upload_to=get_content_image_1_path,
    verbose_name=_("Content Image 1"),
    validators=(FileExtensionValidator(['png', 'jpg']),),
    blank=False
)

content_image_2 = models.ImageField(
    max_length=1024,
    upload_to=get_content_image_2_path,
    verbose_name=_("Content Image 2"),
    validators=(FileExtensionValidator(['png', 'jpg']),),
    blank=False
)

```

Ще раз стисло про поля класу Book та його методи:

Поля:

- ISBN (DecimalField) — слугує унікальним ідентифікатором кожної книги. Це міжнародний стандарт, який містить в собі послідовність чисел, що мітять в собі відповідні коди видавництв, країн, порядкового номеру і так далі;
- Title (CharField) — Повна назва книги;
- Series (CharFields) — належність книги до певної серії чи збірки (поле можна залишити пустим);

- Description(CharFiels) — Опис книги;
- Age Resctiction (Foreing Key) — вікові обмеження;
- Page Numbers (IntegerField) — кількість сторінок книги;
- Physical Size (CharField) — фізичні розміри в міліметрах;
- Cover Material(Foreing Key) — тип обкладинки;
- Language (Foreing Key) — мова якою надрукована книга;
- Original Language (Foreing Key) — мова оригіналу;
- Translate Author (CharField) — автор перекладу (поле можна залишити пустим);
- Illustration Author (CharField) — автор ілюстрацій (поле можна залишити пустим);
- Book Author (ManyToManyField) — автор книги;
- Publisher (Foreing Key) — видавець книги;
- Release Date (DateField) — дата публікації;
- Categories (ManyToManyField) — категорії до яких можна віднести книгу;
- Genres (ManyToManyField) — жанри до яких можна віднести книгу;
- Tags (ManyToManyField) — теги до яких можна віднести книгу;
- preview_image (ImageField) — фото обкладинки;
- content_image_1 (ImageField) — фото вмісту №1;
- content_image_2 (ImageField) — фото вмісту №2.

З цікавих моделей додатку Books , також є таблиця Category:

```
class Category(models.Model):
    name = models.CharField(max_length=50, unique=True)
    genres = models.ManyToManyField(to=Genre)
```

```
class Meta:
    verbose_name = _("Category")
    verbose_name_plural = _("Categories")

    def __str__(self) -> str:
        return f"{self.name}"
```

Поля:

- Name (CharField) — назва категорії;
- Genres(ManyToManyField) — список жанрів які дана категорія може в собі містити; Комбінації не є унікальними, що надає можливість відносити книгу одного жанру до декількох категорій одночасно, розширюючи можливості каталогу.

Більшість інших моделей додатку Books переважно матимуть в собі лише поле імені, що відповідно вказуватиме на унікальну назву категорії, жанру, тегу, автора, тощо. Наприклад:

```
class Author(models.Model):
    name = models.CharField(max_length=100, unique=True)

    class Meta:
        verbose_name = _("Author")
        verbose_name_plural = _("Authors")

    def __str__(self) -> str:
        return f"{self.name}"
```

Моделі застосунку Products:

Модель Product

Поля:

- id — унікальний ідентифікатор

- book (Foreing Key) — унікальне поле, що містить в собі книгу;
- price (DecimalField) — поточна ціна книги;
- view_count (IntegerField) — число переглядів на сайті (за змовчуванням нуль);
- orders_count (IntegerField) — число замовлень на сайті (за змовчуванням нуль);
- is_available(BooleanField) — чи доступний продукт для замовлення (за змовчуванням False);
- is_visible(BooleanField) — чи доступний продукт для відображення на сайті (за змовчуванням False);

Методи:

- view_count_increment() — оновлює статистику переглядів на товар;
- order_count_increment() — оновлює статистику замовлень на товар;

Модель Discount:

Поля:

- product (Foreing Key) — унікальне поле, що містить в собі продукт;
- discount_value (IntegerField) — відсоткове значення знижки на товар;
- start_date та expired_date (DateTimeField) — час протягом якого буде діяти знижка;

Модель Storage:

Поля:

- product (Foreing Key) — унікальне поле, що містить в собі продукт;
- count (IntegerField) — кількість товару, що є на складі.

Та моделі застосунку Users:

Модель UserReview:

- user (Foreing Key) — поле, що містить в собі користувача;
- product (Foreing Key) — поле, що містить в собі продукт;
- date (DateField) — дата публікації відгуку (задається автоматично);
- review_msg (CharField) — власне сам відгук.

Модель Cart та Wishlist:

- user (Foreing Key) — поле, що містить в собі користувача;
- product (Foreing Key) — поле, що містить в собі продукт;

Модель Oders — зберігає інформацію про замовлення:

- user (Foreing Key) — поле, що містить в собі користувача (може бути Null);
- product (Foreing Key) — поле, що містить в собі продукт (може бути Null);
- fullname (CharField) — повне ім'я покупця;
- address (CharField) — адреса покупця;
- phone (DecimalDield) — телефон покупця;
- email (CharField) — електронна адреса покупця;
- product_count (IntegerFiled) — кількість замовленого товару (за змовчуванням 1);
- still_price (FloatField) — ціна на момент замовлення з врахування знижки;
- order_date(DateTimeField) — дата замовлення (задається автоматично);
- is_archiver (BooleanField) — якщо True, замовлення вважатиметься виконаним;

- `is_cancelled` (BooleanField) - Якщо True, замовлення вважатиметься скасованим незалежно від значення поля `is_archiver`.

2.3 Проектування та реалізація алгоритмів роботи

Веб-сайт є досить простим у користуванні, та не містить в собі ніяких прихованих функцій, що дозволить потенційним відвідувачам чи постійним користувачам легко орієнтуватися в його навігації. Алгоритм роботи веб-сайту зображено на рис 2.8.

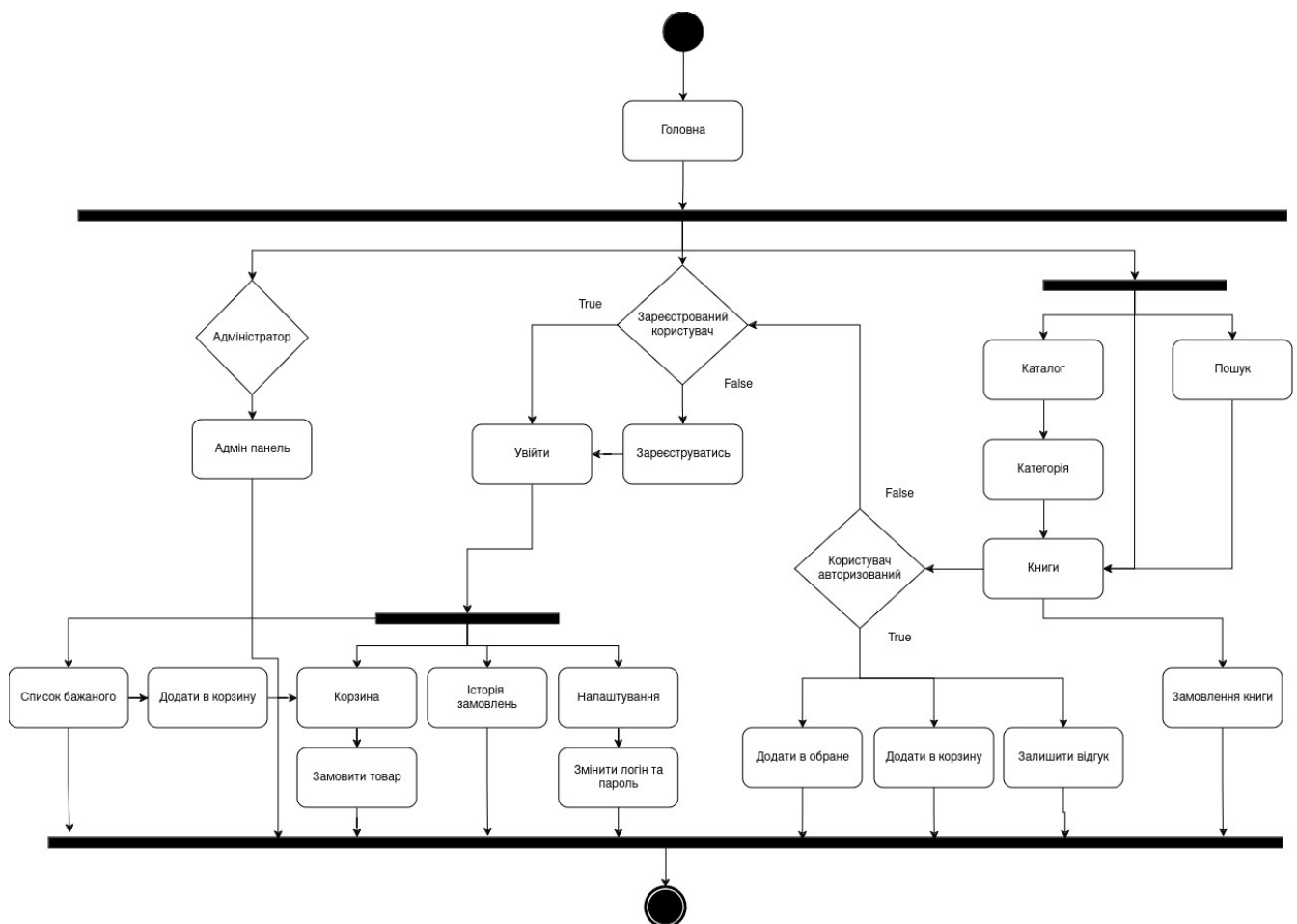


Рис 2.8 Діаграма активності

Детальніше про його пункти...

Якщо користувач не авторизований, він має такі можливості:

- Зареєструватися чи авторизуватися;

- Переглянути головну сторінку;
- Переглянути каталог;
- Переглянути сторінку книги;
- Переглянути сторінку категорії з каталогу;
- Скористатися пошуком по сайту;
- Скористатися фільтрами де це можливо;
- Розмістити замовлення на книгу;

Якщо регулярний користувач авторизований, він додатково має наступні можливості:

- Перейти на сторінку профілю;
- Проглянути вміст корзини;
- Проглянути вміст списку бажаного;
- Переглянути історію замовлень;
- Зміна логіну та пароллю;
- Додавати товар в корзину;
- Додавати товар в бажане;
- Замовляти товар через корзину;
- Переміщувати товар з бажаного в корзину;
- Вийти з облікового запису;
- Видалити обліковий запис;

Якщо адміністратор авторизований, він додатково має наступні можливості:

- Додавати чи змінювати книги, авторів, опис, категорії, жанри і тд;

- Додавати чи змінювати товари, знижки. Вносити бухгалтерську інформацію;
- Керувати діяльністю користувачів;
- Створювати довільних акторів з довільними правами.

2.4 Реалізація функціоналу веб-сайту

Як було неодноразово сказано, основний функціонал веб-сайту буде реалізовано з допомогою фреймворку Django. Повернімось до структури системи (Рис 2.4). На даному етапі ми маємо реалізовану базу даних models.py та можемо приступати до створення запитів до них через користувальницький інтерфейс, а саме представлення або ж Views.

Оскільки дана робота не є гайдом з програмування, описувати розробку та функціонал кожного представлення немає сенсу, адже підхід до створення кожного з них досить схожий.

Почнімо з представлення каталогу застосунку Books.

```
from django.views.generic import ListView
from books.models import Category

class CategoryView(ListView):
    model = Category
    context_object_name = "categories_list"
    template_name = "books/categories.html"
```

Однією з ключових переваг Django є відсутність необхідності «Розробляти велосипед». Фреймворк надає весь необхідний функціонал. Наприклад в базових відображеннях існують класи DetialView та ListView, які автоматично виконують вибірку з бази даних та готують об'єкт моделі до рендеру на сторінці.

В даному прикладі ми створюємо свій власний клас наслідуючись від класу `ListView`. `ListView` — це клас призначений для відображення списку об'єктів. Все що потрібно вказати, це модель з якої ми беремо вибірку, шаблон, в який потрібно помістити список, та яку назву присвоїти цьому списку для взаємодії в шаблоні.

Використовуючи мову шаблонів Django маємо наступний шаблон:

```
<div class="list">
    {% for category in categories_list %}
    <div class="element">
        <span> {{ category.name }} </span>
        <ul>
            {% for genre in category.genres.all %}
                <li>{{ genre.name }}</li>
            {% endfor %}
        </ul>
    </div>
    {% endfor %}
</div>
```

Навігація по сторінкам в Django здійснюється за допомогою `urls.py`. Кожен такий файл містить в собі шлях, представлення та коротка назва для використання її в шаблонах.

Для застосунку Books файл матиме наступний вигляд:

```
from django.urls import path
from books import views

app_name = 'books'
urlpatterns = [
    path("", views.CategoryView.as_view(), name="catalog"),
]
```

Також варто не забувати, включити його в загальний `urls.py`, який і слугує ключовим елементом навігації по сайту. Як правило він створюється разом з проєктом, та лежить поруч з файлами налаштувань.

```
from django.conf.urls.static import static
from django.contrib import admin
from django.urls import path, include
from bookstore import settings

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

А тепер перейдімо до більш складних та цікавих прикладів, а саме сторінки авторизації. Для цього використаємо застосунок Users. Хорошою практикою в Django вважається створення класів форми, та передачі їх не рендер в форму, за допомогою представлень. Це забезпечить нам повне розуміння того, як працювати з даними, що туди потрапили. Одже першим чином в файлі `forms.py` створимо форму.

```
from django import forms
from django.contrib.auth.models import User
from django.utils.translation import gettext_lazy as _

class UserLoginForm(forms.Form):
    username = forms.CharField(
        label="Username",
        min_length="3",
        max_length="25",
        required=True
    )
    password = forms.CharField(
        label="Password",
```

```

        min_length="5",
        max_length="100",
        required=True,
        widget=forms.PasswordInput
    )

    def clean(self) -> dict:
        cleaned_data = self.cleaned_data

        try:
            user =
User.objects.all().get(username=cleaned_data["username"])
        except User.DoesNotExist:
            raise forms.ValidationError("Username or password
is invalid.")

        if user.check_password(cleaned_data["password"]):
            cleaned_data['user'] = user
        else:
            raise forms.ValidationError("Username or password
is invalid.")

        return cleaned_data

```

Реалізація досить близька до класів моделей, проте використовує дещо інші класи та методи. Першим чином варто вказати поля, по яким ми здійснюватимемо вхід, а саме username та password. Кожне з цих полів повинно мати свої правила, на основі яких ми зможемо обмежити фантазії потенційного користувача. Наприклад в моделях ми не можемо встановити мінімальну довжину символів, проте ми можемо це зробити в формі. Якщо цього мало, також Django дає можливість створити власний метод clean(). Він стежить за тим, щоб кожне поле відповідало певним правилам та обмеження, та в разі їх порушення генерувати помилку та сповіщати про це користувача. Також в

цьому методі, можна встановити додаткові перевірки. Наприклад в формі логіну варто встановити перевірку на те, чи користувач існує в системі.

Готову форму слід помістити в атрибути класу представлення. Саме представлення будемо наслідувати від базових представлень, а саме від класу `FormView` який і використовується для сторінок з формами.

```
class LoginView(FormView):
    template_name = "users/login.html"
    form_class = UserLoginForm
    success_url = "/"

    def dispatch(self, request, *args, **kwargs):
        if self.request.user.is_authenticated:
            return redirect(self.success_url)

        return super().dispatch(self.request, *args,
**kwargs)

    def form_valid(self, form):
        login(self.request, form.cleaned_data["user"])
        return super().form_valid(form=form)
```

У випадку, якщо всі умови форми будуть виконані, а користувач натисне клавішу «Відправити», форма автоматично попаде в метод `form_valid()` де ми зможемо опрацювати отримані дані.

Метод `dispatch()` приймає запит та його аргументи від клієнта та повертає йому відповідь. Якщо його перевизначити - можна помістити додаткові умови перевірки. В даному прикладі була додана перевірка на те, чи користувач не авторизований. У випадку, якщо все ж авторизований, спрацює функція перенаправлення, а представлення авторизації завершить свою роботу. Таким чином авторизований користувач ніколи не зможе зайти на сторінку авторизації вдруге.

Далі слід додати наше представлення в `urls.py` та згенерувати шаблон, як і в попередньому представленні.

Тепер розглянемо представлення з пагінацією та обов'язковою авторизацією на прикладі перегляду вмісту корзини.

```
class ProfileCartView(LoginRequiredMixin, ListView):
    model = Cart
    template_name = "users/profile/cart.html"
    current_page = 1
    login_url = "/profile/login/"
    redirect_field_name = "/profile/cart"

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

        try:
            cart =
Cart.objects.filter(user=self.request.user).order_by("-pk")
        except ObjectDoesNotExist:
            context["page"] = None
            context["num_pages"] = 0

        return context

    cart_pagination = Paginator(cart, 20)
    page = cart_pagination.page(self.current_page)
    context["page"] = page
    context["num_pages"] = cart_pagination.num_pages

    return context

    def dispatch(self, request, *args, **kwargs):
        if "page" in kwargs:
```

```

        self.current_page = kwargs["page"]

    return super().dispatch(self.request, *args,
**kwargs)

```

Клас `LoginRequiredMixin` забезпечує перевірку на те, чи юзер авторизований. У випадку, якщо ні — спрацює перенаправлення по атрибуту `login_url`. Окрім того, класи в Python мають змогу наслідувати одночасно декілька класів. Тому також ми можемо зробити вибірку товарів корзини з бази даних та передати їх на рендер в шаблон.

Для пагінації в Django існує клас `Paginator`. Для вибірки об'єктів по сторінкам необхідно лише вказати список об'єктів, кількість об'єктів на одній сторінці та через атрибути посилання отримати номер сторінки. Це можна зробити в `urls.py`.

```

urlpatterns = [
    path("cart/page/<int:page>",
views.ProfileCartView.as_view(), name="profile-cart-page"),
]

```

В даному прикладі `<int:page>` приймає через себе номер сторінки. Далі в методі представлення `dispatch` ми дістаємо змінну з списку переданих аргументів та кладемо в атрибут класу.

Користуватись методами класів можна навіть з шаблону. Проте варто не забувати про синтаксис.

```

{% if num_pages > 1 %}
    <div class="pages">
        {% if page.has_previous %}
            <a href="{% url 'users:profile-cart-page'
page.previous_page_number %}"><span class="material-symbols-
outlined">arrow_back</span></a>
        {% endif %}
        <span class="page_counter">{{ page.number }} /
{{ num_pages }}</span>

```

```

        {% if page.has_next %}
        <a href="{% url 'users:profile-cart-page'
page.next_page_number %}"><span class="material-symbols-
outlined">arrow_forward</span></a>
        {% endif %}
    </div>
    {% endif %}

```

Таким чином, використовуючи методи `has_previous` та `has_next` ми можемо автоматично створювати навігацію для довільної кількості об'єктів в базі даних.

Висновки до другого розділу

Другий розділ даної роботи демонструє ключові моменти розробки веб-сайту. На основі попереднього розділу, було сформовано схематичне зображення системи, враховуючи специфіку фреймворку. Також були розроблені схеми бази даних та їх реалізація.

На основі сформованих вимог було реалізовано діаграму діяльності, яка чудово демонструє можливості майбутнього програмного продукту. Діаграма діяльності в свою чергу послугувала фундаментом для реалізації всього функціоналу веб-сайту.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-САЙТОМ

3.1 Порядок встановлення та налаштування системи

Оскільки веб-сайт не перевантажений різного роду скриптами, він не потребує сильної машини для управління. Насправді все напряду залежить від кількості одночасних користувачів та трафіку, який вони будуть генерувати. На перші час достатньо буде:

- Процесор на 4 ядра з тактовою частотою 1-2ГГц;
- Оперативна пам'ять від 2 до 4Гб;
- Місце на диску від 5Гб (залежить від вмісту медіа в базі даних);

Щоб запустити серверну частину додатку потрібно виконати наступний ряд дій:

- Скопіювати репозиторій з кодом на локальну машину за допомогою `git clone`;
- Встановити віртуальне середовище за допомогою: **`python3 -m venv venv`**;
- Увійти в віртуальне середовище: **`source venv/bin/activate`**;
- Встановити залежності до проєкту: **`pip3 install -r requirements.txt`**;
- Створити **`.env`** файл зі змінними середовища;
- Провести міграції: **`python3 manage.py makemigration`** та **`python3 manage.py migrate`**;
- Запустити додаток проєкт **`python3 manage.py runserver`**.

Важливим моментом є те, що необхідно перед запуском впевнитись, що сервер не запуститься в режимі відлагодження. Для цього в `settings.py` потрібно

встановити відповідний прапорець на False, або вказати його в змінній середовища.

Для взаємодії з веб-сайтом підійде будь-який сучасний веб-браузер.

3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з веб-сайтом.

При першому вході, користувач опиниться на головній сторінці (рис 3.1).

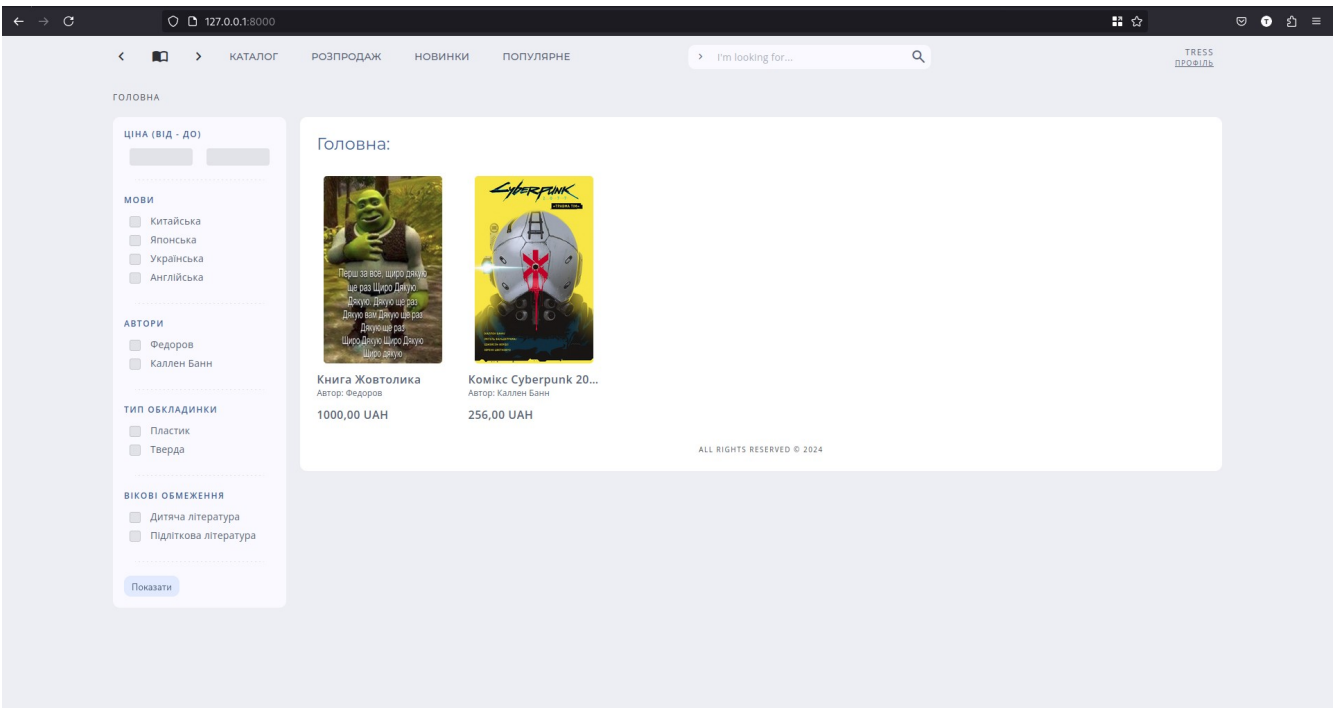


Рис 3.1 Головна сторінка

З даної сторінки користувач відразу матиме доступ до каталогу, сторінки розпродажів чи основний сортувальних фільтрів. Окрім того є окремі фільтри для головної сторінки. В даному прикладі, користувач авторизований, проте у протилежному випадку замість кнопки профіль буде кнопка авторизації, як це продемонстровано на рисунку 3.2.

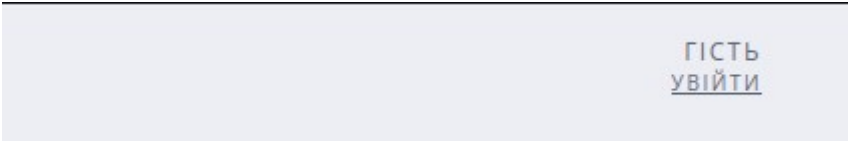


Рис 3.2 Неавторизований користувач

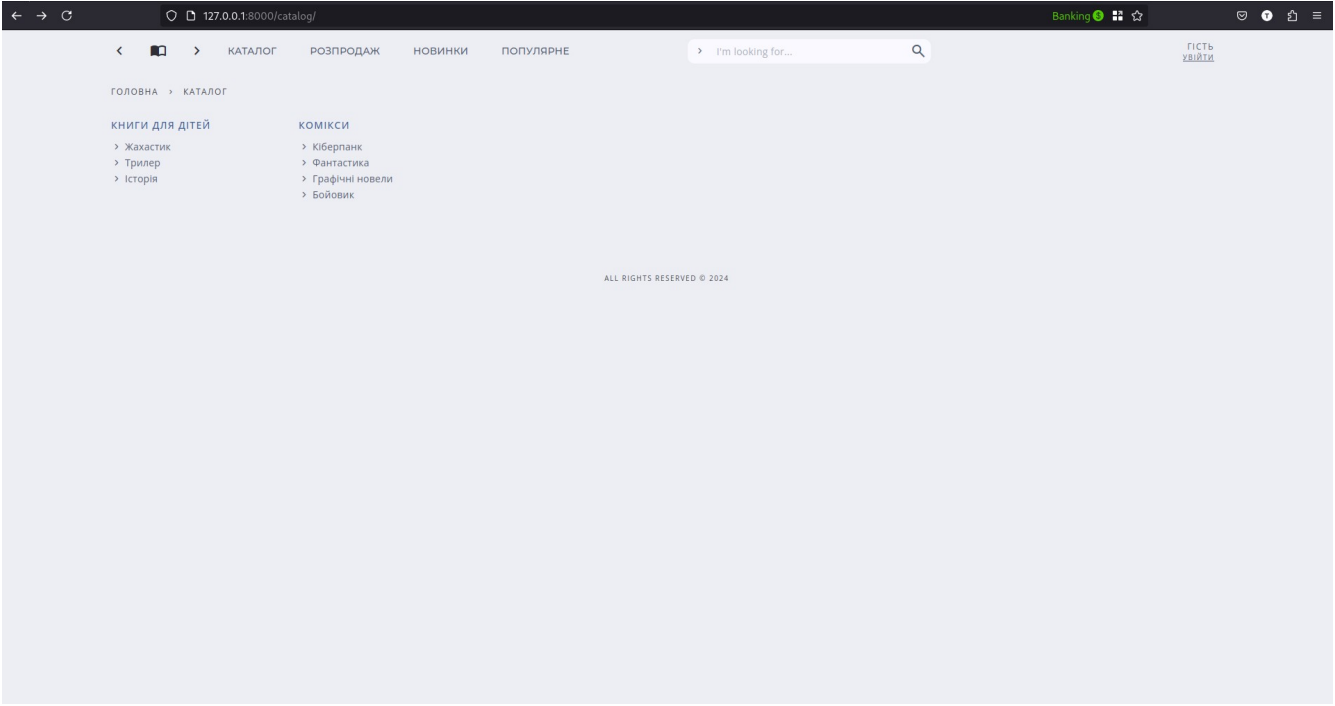


Рис 3.3 Каталог

Сторінка каталогу містить в собі всі категорії та жанри, пов’язані з ними в базі даних.

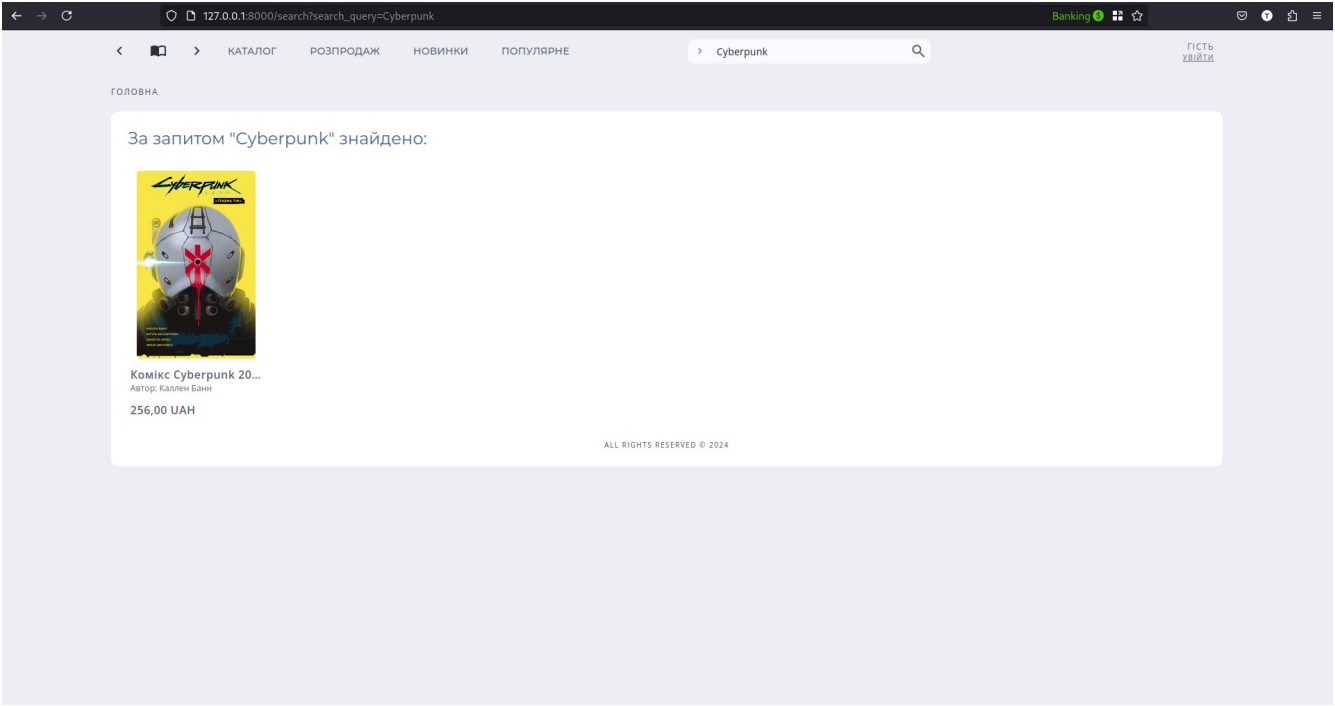


Рис 3.4 Пошук

Поле пошуку допомагає швидко знаходити необхідні книги, роблячи вибірку з бази даних по тегам книги.

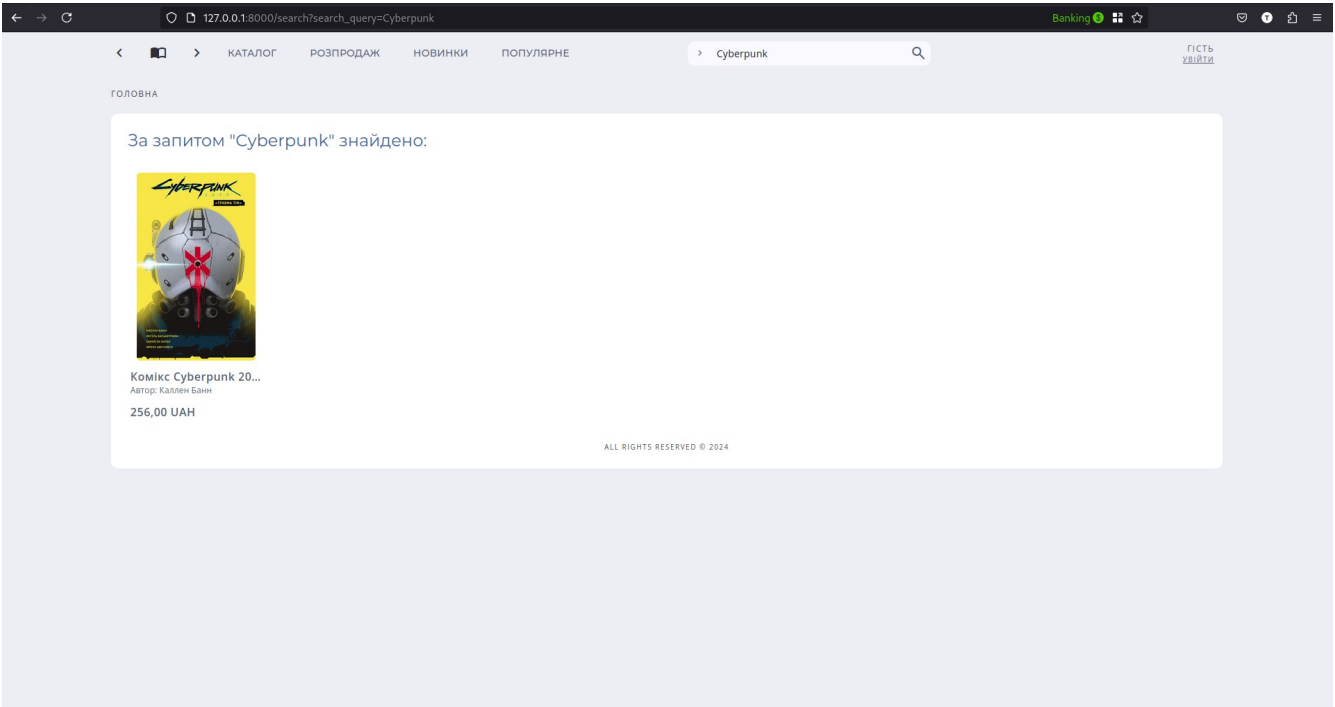


Рис 3.5 Сторінка книги

Сторінка книги містить в собі всю інформацію пов’язану з книгою. На даній сторінці, будучи авторизованим, можна додати книгу в кошик, бажане чи залити відгук до неї (рис 3.6).

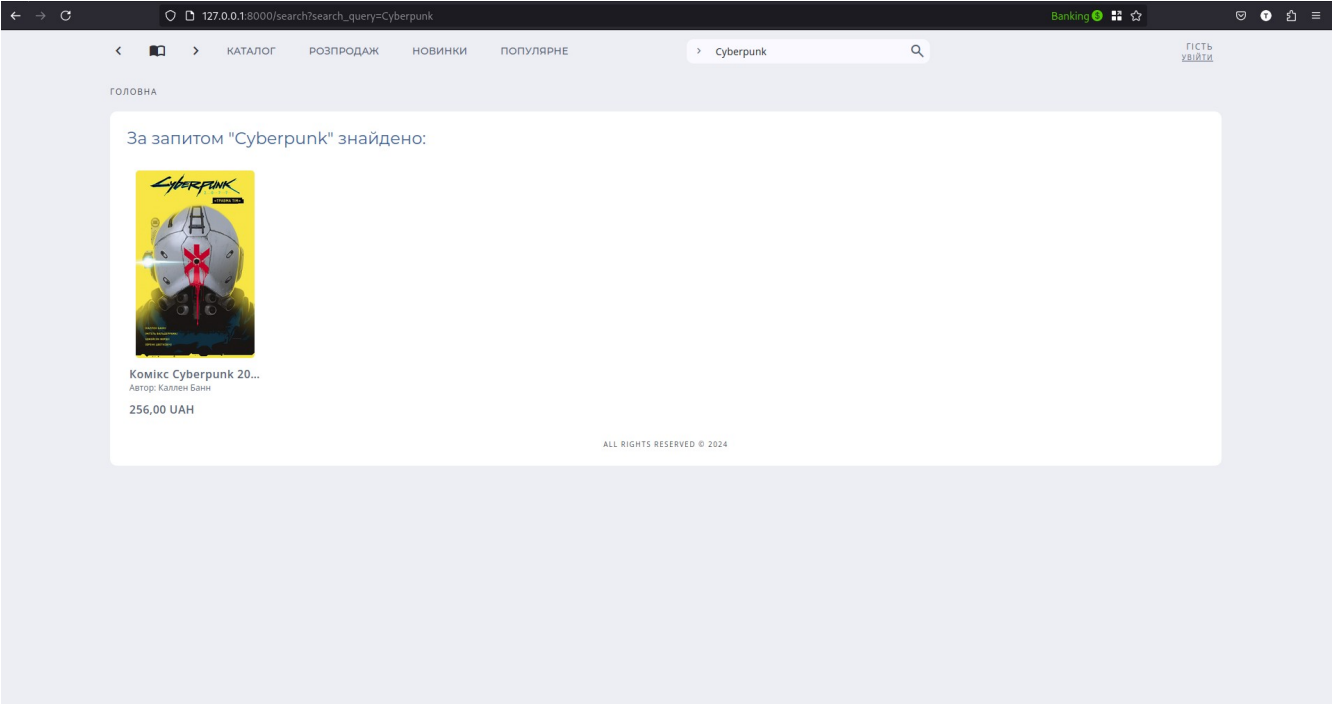


Рис 3.6 Відгуки користувачів

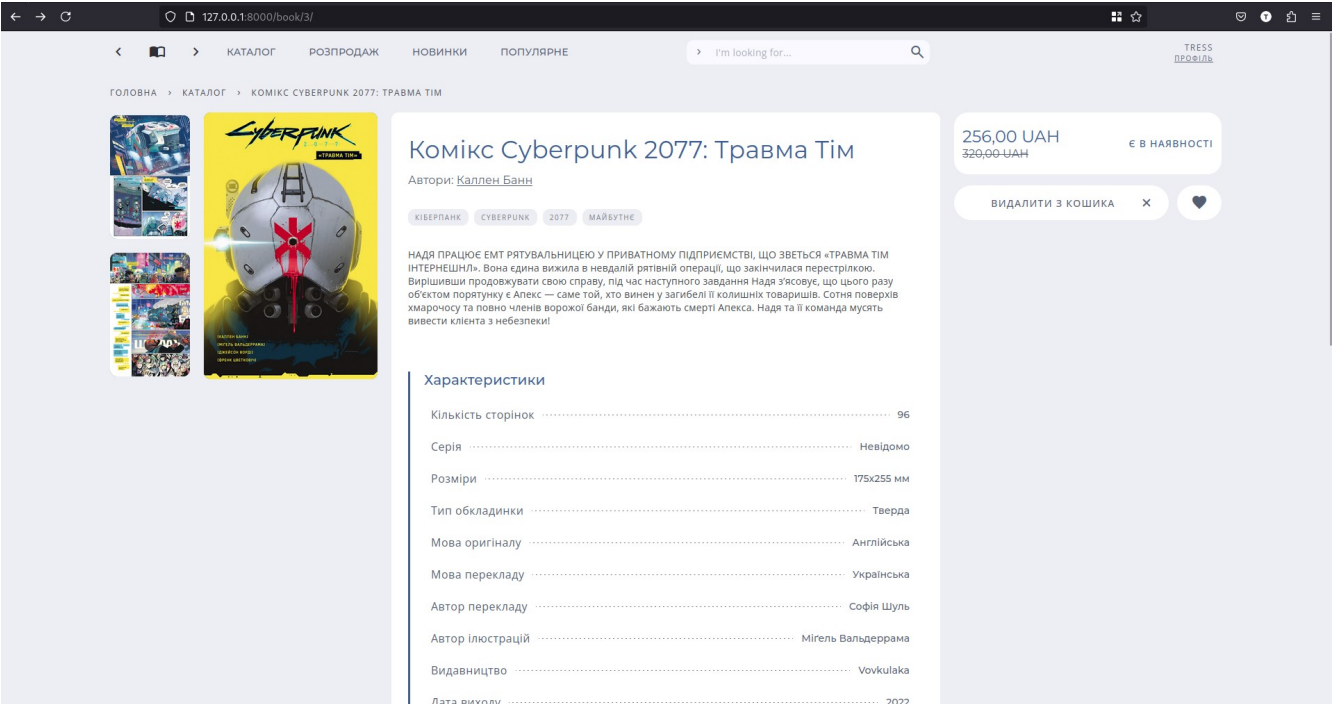


Рис 3.7 Вигляд сторінки авторизованого користувача

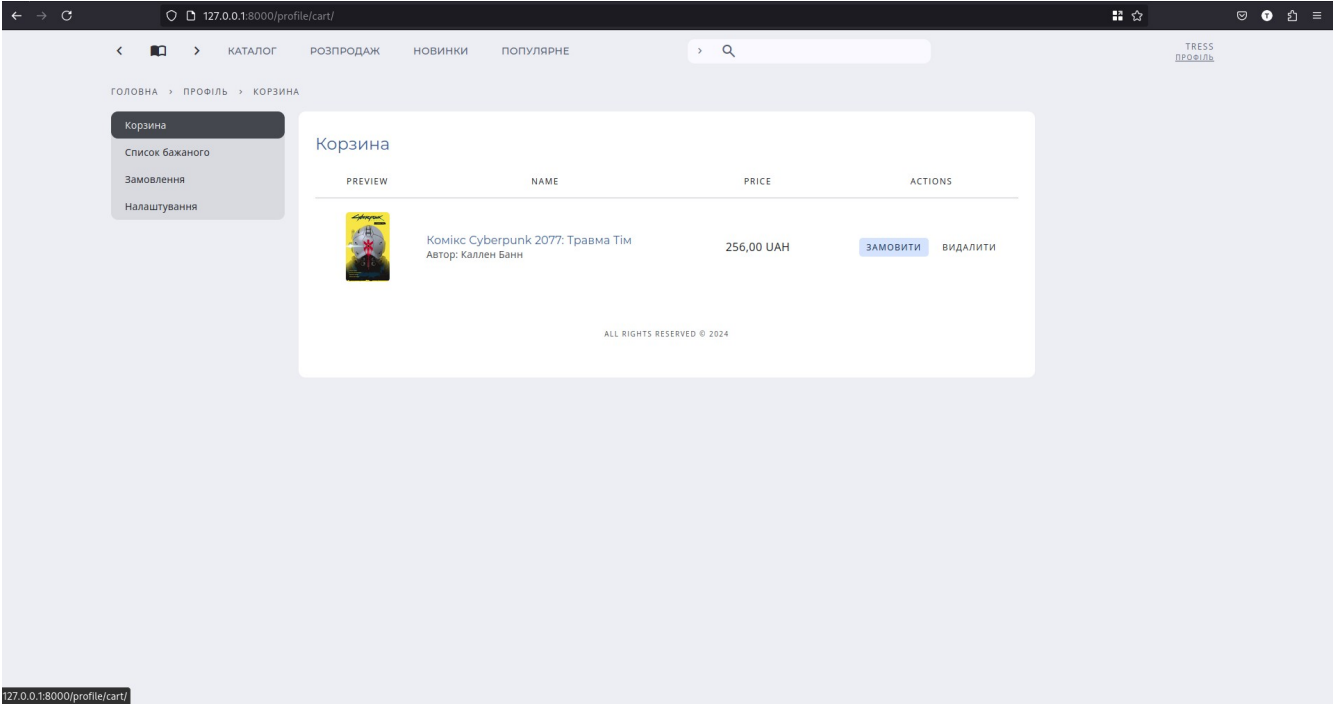


Рис 3.8 Профіль — корзина

Перейшовши в профіль ми автоматично попадемо в корзину, де відразу можна перейти до оформлення замовлення на товар.

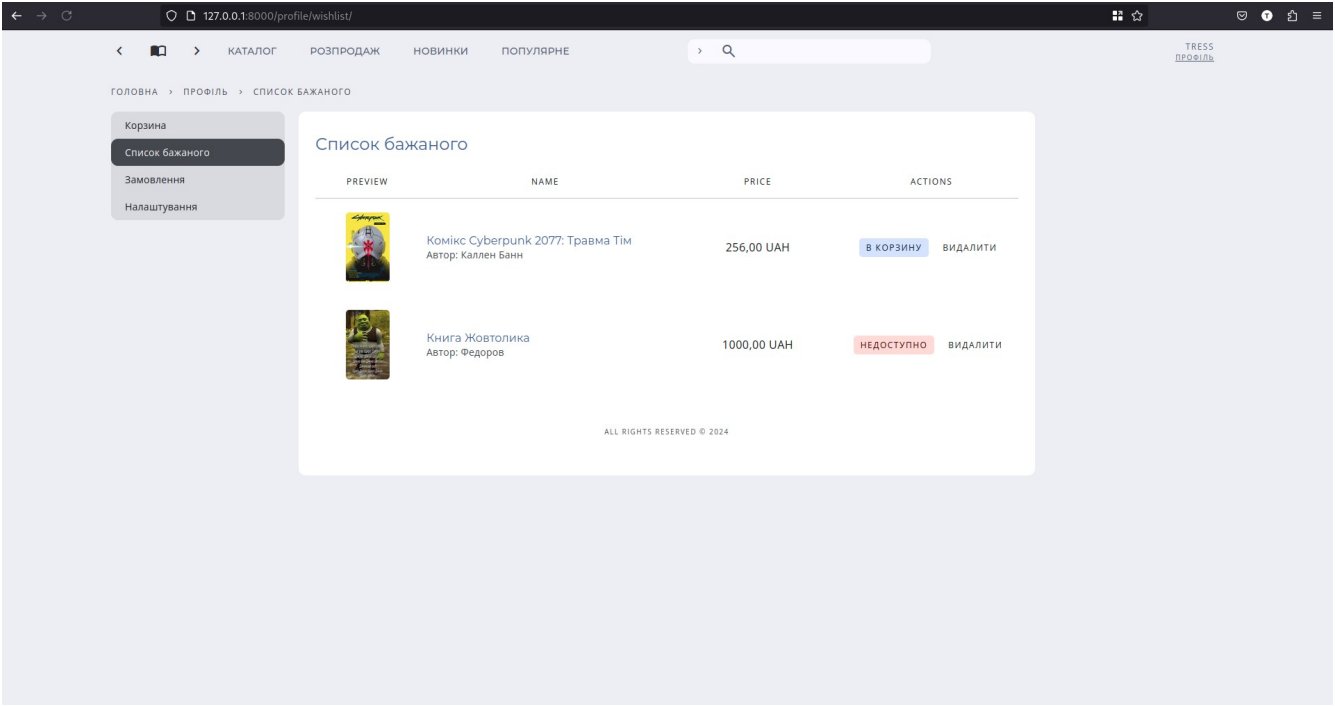


Рис 3.9 Профіль — бажане

Список бажаного подібний по функціоналу до кошику, проте ми не можемо перемістити в кошику товари, яких немає на складі. Натомість, можна розмістити їх в бажане, та дочекатись поновлення асортименту на складі.

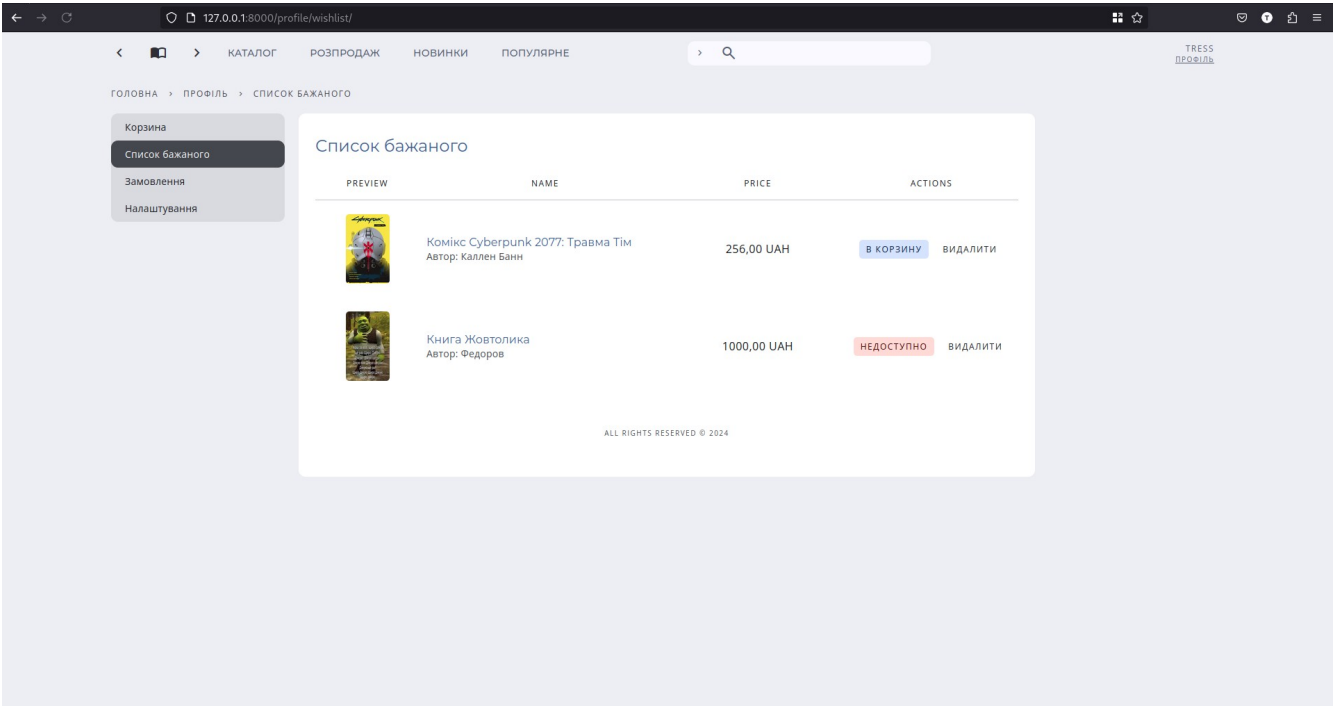


Рис 3.10 Історія замовлень

На сторінці історії замовлень можна проглянути всі свої замовлення та статус їх виконання. Окрім того, є доступний до перегляду чек.

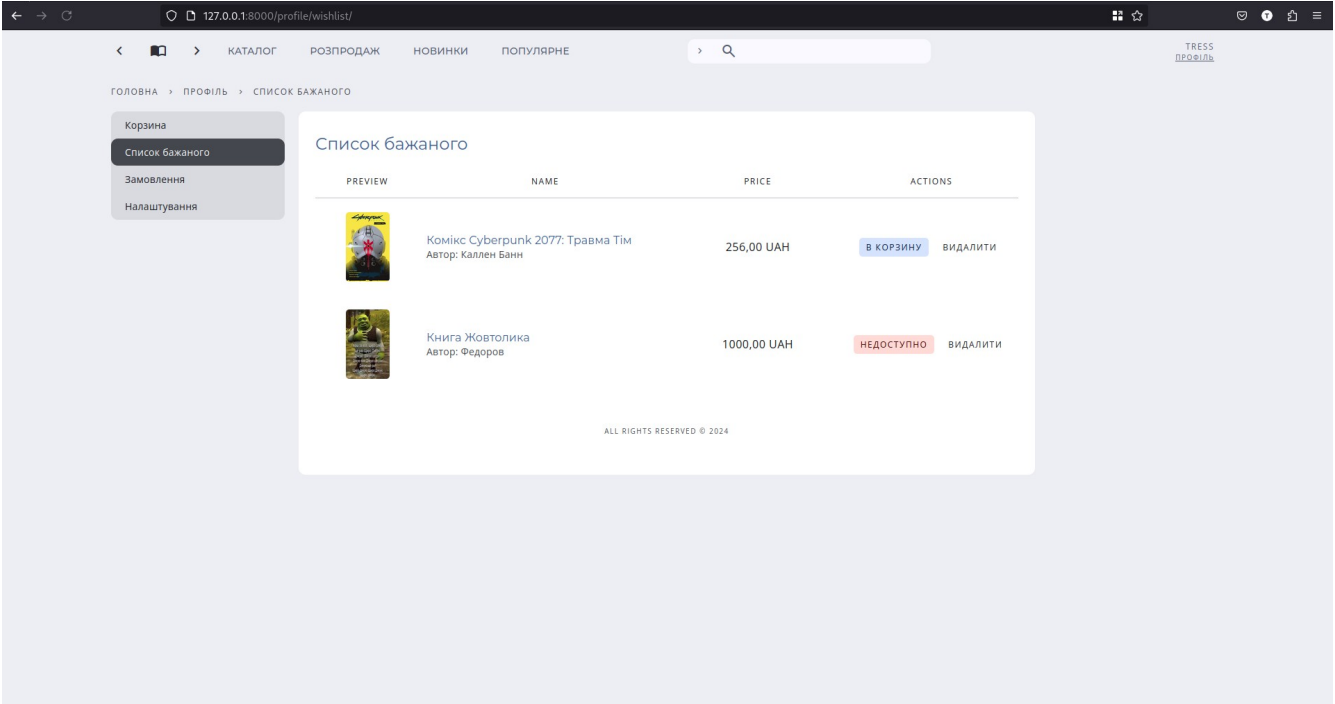


Рис 3.11 Налаштування

На сторінці налаштувань користувач має змогу змінити пароль, логін та видалити обліковий запис. Кожна з цих дій потребує введення поточного паролю.

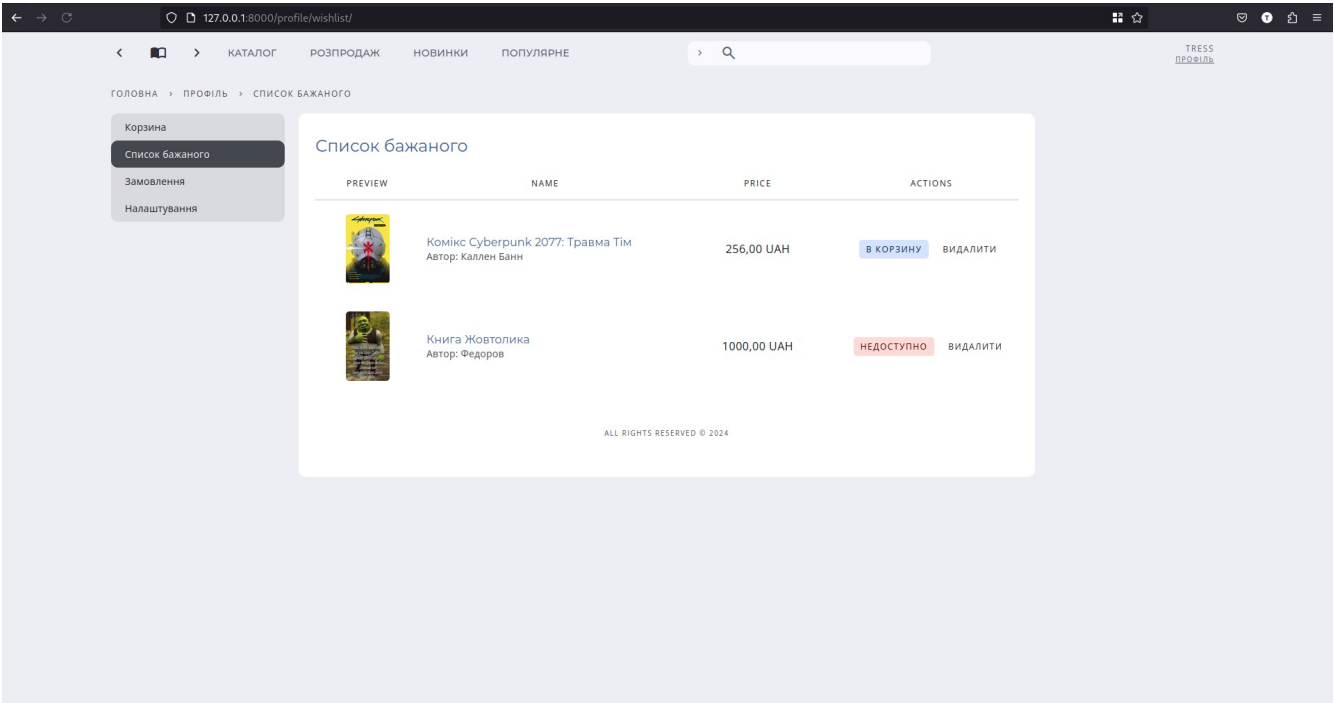


Рис 3.12 Помилка при спробі змінити налаштування без паролю.

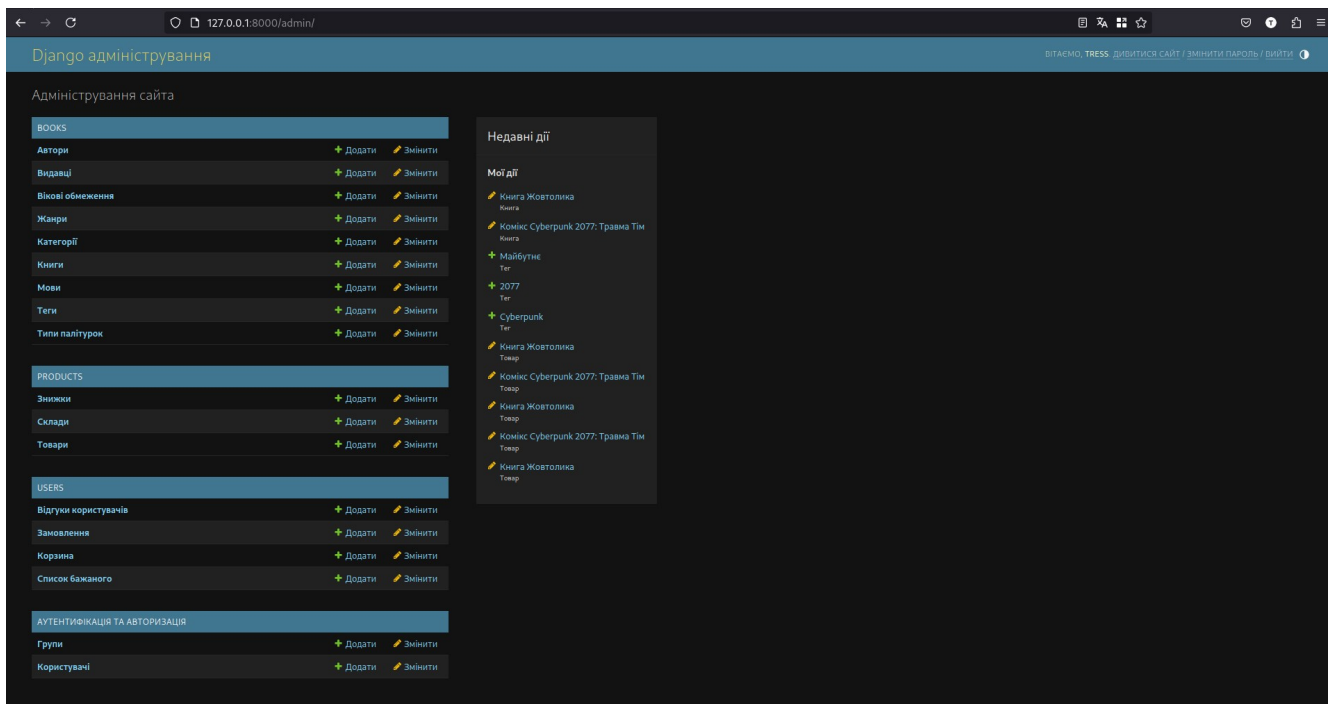


Рис 3.13 Адміністративна панель

Що стосується адміністративної панелі, вона генерується автоматично, на основі шаблонів самого Django. В ній відокремлені всі поточні моделі, та поділені на групи відповідно до застосунків.

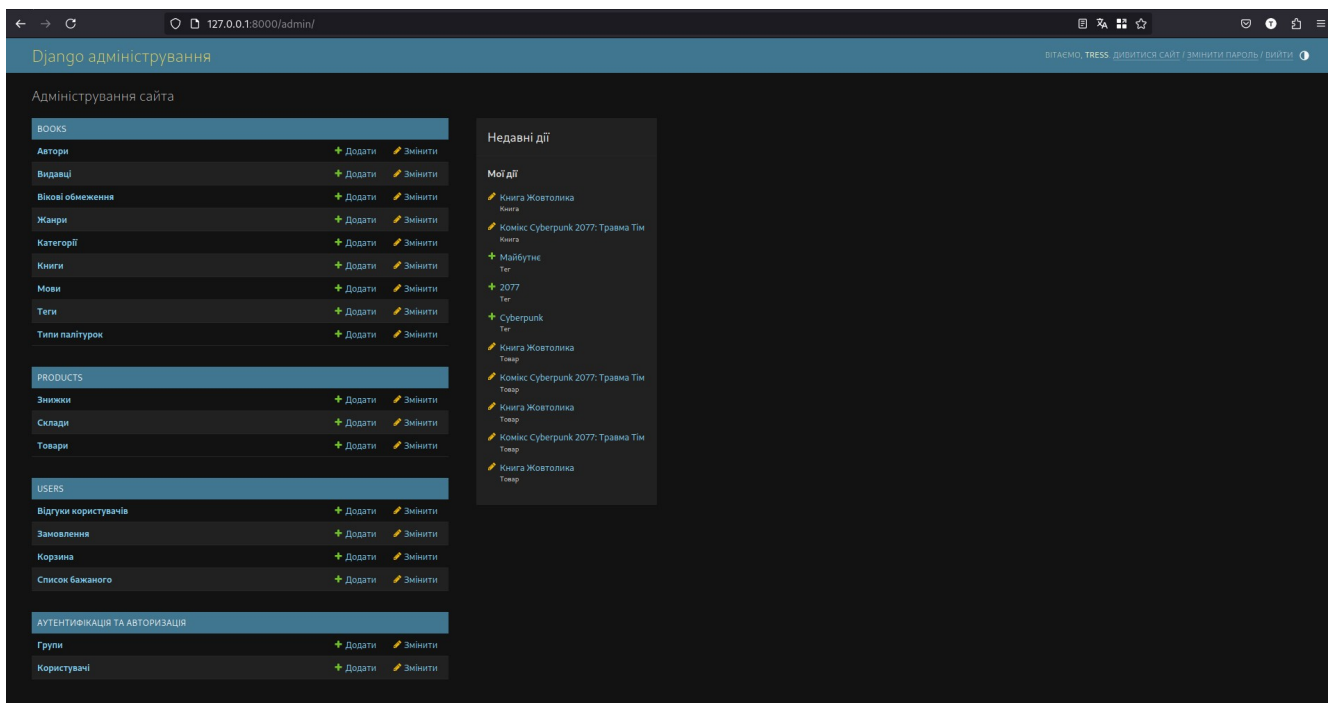


Рис 3.14 Вміст таблиці книг

Ми зручно можемо переглядати всі доступні екземпляри таблиці, редагувати їх чи додавати нові (рис 3.15).

Рис 3.15 Додавання книги

3.3 Тестування роботи веб-сайту.

По завершенню розробки веб-сайту, настає час переходити до тестування. Цей крок потрібен для того, аби переконатись, що сайт відповідає всім поставленим вимогам, та готовий до передачі масовому користувачу.

Для перевірки працездатності виконаємо деякі дії, які буде виконувати користувач, персонал чи незареєстрований юзер.

Додавання книги на веб-сайт

- **Передумова:** Після додавання нового товару (книги) в адмін панелі, він має відобразитись на головній сторінці, якщо він доступний для видимості;
- **Очікуваний результат:** товар відобразиться на головній сторінці;
- **Фактичний результат:** ...відповідає очікуваному результату.

Замовлення товару не маючи облікового запису

- **Передумова:** не авторизуючись, на сторінці книги спробувати придбати книгу;
- **Очікуваний результат:** переадресація на сторінку оформлення замовлення;
- **Фактичний результат:** переадресація на сторінку авторизації.

Спроба видалення рецензії іншого користувача

- **Передумова:** видалення коментарів працює так, що юзер переходить по посиланню яке в своєму шляху містить ідентифікатор рецензії в базі даних. Підставивши цей ідентифікатор у відповідне посилання можна спробувати надіслати запит на видалення;
- **Очікуваний результат:** коментар не видалиться, якщо він не належить користувачу, який надіслав запит;
- **Фактичний результат:** ... відповідає очікуваному результату.

Висновки до третього розділу

У даному розділі було наведено порядок встановлення веб-додатку та визначено конфігурацію базового сервера необхідного для роботи веб-сайту.

Також було продемонстровано основні сторінки веб-сайту, включаючи в себе адміністративну панель управління.

В результат тестування було знайдено одну грубу помилку.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра був розроблений веб-сайт книжкового магазину.

Першим етапом було чітке формування поставленої задачі та базових вимог, яким повинна відповідати розробка. Було проаналізовано аналоги, з врахуванням усіх переваг та недоліків. В результаті аналізу цієї інформації було сформоване нечітке бачення майбутнього додатку, та були обрані технології, які найліпше підходять для розробки подібного роду веб-сайтів.

Другим етапом виконання, було формування чітких функціональних та нефункціональних вимог. Була побудована діаграма активностей, яка послугувала фундаментом для розробки більшості класів, бази даних та представлення.

Третій етап дав нам змогу подивитись на розробку очима базового користувача, та вказав всі проблемні питання пов'язані з користувацьким інтерфейсом веб-сайту.

Веб-сайт не є повністю завершеним, та потребує допрацювання. Проте він може послугувати чудовим фундаментом для будь-якого книжкового інтернет магазину.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційний сайт фреймворку Django [Електронний ресурс] – Режим доступу до ресурсу: <https://www.djangoproject.com/>.
2. Офіційний сайт мови програмування Python [Електронний ресурс] – Режим доступу до ресурсу: <https://www.python.org/>.
3. Документація по django-filters [Електронний ресурс] – Режим доступу до ресурсу: <https://django-filter.readthedocs.io/en/stable/>.
4. Документація по git [Електронний ресурс] – Режим доступу до ресурсу: <https://git-scm.com/>.
5. Стаття по роботі з фільтрами [Електронний ресурс] – Режим доступу до ресурсу: <https://letscodemore.medium.com/how-to-add-modelmultiplechoicefilter-of-django-filters-in-django-7cedb682ede2>.
6. Запити до бази даних в Django [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.djangoproject.com/en/5.0/topics/db/queries/>.
7. Мова шаблонів Django [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.djangoproject.com/en/5.0/ref/templates/>.
8. Моделі Django Django [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.djangoproject.com/en/5.0/topics/db/models/>.
9. Користувачі в Django [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.djangoproject.com/en/5.0/ref/contrib/auth/>.
10. Форми в Django [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.djangoproject.com/en/5.0/topics/forms/>.
11. Налаштування проєкту Django [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.djangoproject.com/en/5.0/topics/settings/>.

12. Препроцесор SASS [Електронний ресурс] – Режим доступу до ресурсу: <https://sass-lang.com/>.
13. Стаття по тегам в Django [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.djangoproject.com/en/5.0/ref/templates/builtins/>.
14. Django Tamplate API [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.djangoproject.com/en/5.0/ref/templates/api>.
15. Ключові команди Django [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.djangoproject.com/en/5.0/ref/django-admin/>

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Аркуш
						62
Зм.	Арк.	№ докум.	Підпис	Дата		

Лістинг до прикладу реалізації каталогу

views.py

```
from django.views.generic import ListView
from books.models import Category
from products.forms import SearchForm
```

```
class CategoryView(ListView):
    model = Category
    queryset = Category.objects.all()
    context_object_name = "categories_list"
    template_name = "books/categories.html"

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super().get_context_data(**kwargs)
        context["search_form"] = SearchForm

        return context
```

models.py

```
from django.core.validators import FileExtensionValidator
from django.db import models
from django.utils.translation import gettext_lazy as _
```

```
def get_preview_image_path(instance, filename):
    return f"posts/{instance.isbn}/preview.{filename[-3:]}"
```

```
def get_content_image_1_path(instance, filename):
```

```

return f"posts/{instance.isbn}/content_1.{filename[-3:]}"

def get_content_image_2_path(instance, filename):
    return f"posts/{instance.isbn}/content_2.{filename[-3:]}"

class Publisher(models.Model):
    name = models.CharField(max_length=50, unique=True)

    class Meta:
        verbose_name = _("Publisher")
        verbose_name_plural = _("Publishers")

    def __str__(self) -> str:
        return f"{self.name}"

class Author(models.Model):
    name = models.CharField(max_length=100, unique=True)

    class Meta:
        verbose_name = _("Author")
        verbose_name_plural = _("Authors")

    def __str__(self) -> str:
        return f"{self.name}"

class Genre(models.Model):
    name = models.CharField(max_length=50, unique=True)

    class Meta:

```

```

        verbose_name = _("Genre")
        verbose_name_plural = _("Genres")

    def __str__(self) -> str:
        return f"{self.name}"

class Category(models.Model):
    name = models.CharField(max_length=50, unique=True)
    genres = models.ManyToManyField(to=Genre)

    class Meta:
        verbose_name = _("Category")
        verbose_name_plural = _("Categories")

    def __str__(self) -> str:
        return f"{self.name}"

class Tag(models.Model):
    name = models.CharField(max_length=50, unique=True)

    class Meta:
        verbose_name = _("Tag")
        verbose_name_plural = _("Tags")

    def __str__(self) -> str:
        return f"{self.name}"

class Language(models.Model):
    name = models.CharField(max_length=50, unique=True)

```

```

class Meta:
    verbose_name = _("Language")
    verbose_name_plural = _("Languages")

def __str__(self) -> str:
    return f"{self.name}"

class AgeRestriction(models.Model):
    name = models.CharField(max_length=50, unique=True)

    class Meta:
        verbose_name = _("Age restriction")
        verbose_name_plural = _("Age restrictions")

def __str__(self) -> str:
    return f"{self.name}"

class CoverMaterial(models.Model):
    name = models.CharField(max_length=50, unique=True)

    class Meta:
        verbose_name = _("Cover material")
        verbose_name_plural = _("Cover materials")

def __str__(self) -> str:
    return f"{self.name}"

class Book(models.Model):
    class Meta:
        verbose_name = _("Book")

```

```

        verbose_name_plural = _("Books")

    def __str__(self) -> str:
        return f"{self.title}"

    isbn = models.DecimalField(max_digits=13, decimal_places=0,
primary_key=True)
    title = models.CharField(max_length=100, blank=False)
    series = models.CharField(max_length=100, blank=True,
null=True, default=None)
    description = models.CharField(max_length=3000, blank=False)

    age_restriction = models.ForeignKey(to=AgeRestriction,
null=True, default=None, on_delete=models.DO_NOTHING)
    page_numbers = models.IntegerField(default=0)
    physical_size = models.CharField(max_length=30)
    cover_material = models.ForeignKey(to=CoverMaterial,
null=True, on_delete=models.DO_NOTHING)

    language = models.ForeignKey(
        to=Language,
        on_delete=models.SET_NULL,
        null=True,
        related_name='book_language'
    )
    original_language = models.ForeignKey(
        to=Language,
        on_delete=models.SET_NULL,
        null=True,
        related_name='book_original_language'
    )

    publisher = models.ForeignKey(to=Publisher,
on_delete=models.SET_NULL, null=True, default=None)

```

```

    release_date = models.DateField(blank=True, null=True,
default=None)

    translate_author = models.CharField(max_length=100,
blank=True)
    illustration_author = models.CharField(max_length=100,
blank=True)
    book_authors = models.ManyToManyField(to=Author)

    categories = models.ManyToManyField(to=Category)
    genres = models.ManyToManyField(to=Genre)
    tags = models.ManyToManyField(to=Tag)

    preview_image = models.ImageField(
        max_length=1024,
        upload_to=get_preview_image_path,
        verbose_name=_("Preview Image"),
        validators=(FileExtensionValidator(('png', 'jpg'))),),
        blank=False
    )

    content_image_1 = models.ImageField(
        max_length=1024,
        upload_to=get_content_image_1_path,
        verbose_name=_("Content Image 1"),
        validators=(FileExtensionValidator(('png', 'jpg'))),),
        blank=False
    )

    content_image_2 = models.ImageField(
        max_length=1024,
        upload_to=get_content_image_2_path,
        verbose_name=_("Content Image 2"),
        validators=(FileExtensionValidator(('png', 'jpg'))),),

```

```

        blank=False
    )

```

Urls.py

```

from django.urls import path
from books import views

```

```

app_name = 'books'
urlpatterns = [
    path("", views.CategoryView.as_view(), name="catalog"),
]

```

categories.html

```

{% extends "../index.html" %}
{% block title %} Bookstore | Catalog {% endblock %}

{% load static %}
{% block style %}
<link rel="stylesheet" href="{% static 'css/catalog.css' %}">
{% endblock %}

{% block breadcrumbs %}
<span class="material-symbols-outlined">chevron_right</span>
<a href="{% url 'books:catalog' %}">Каталог</a>
{% endblock %}

{% block content %}
<main>
    <div class="categories">
        <div class="list">
            {% for category in categories_list %}

```

```

        <div class="element">
            <span> {{ category.name }} </span>
            <ul>
                {% for genre in category.genres.all %}
                    <li><span class="material-symbols-
outlined">chevron_right</span> <a href="#">{{ genre.name
}}</a></li>

                    {% endfor %}
                </ul>
            </div>
        {% endfor %}
    </div>

    <div class="footer">
        <span> All Rights Reserved © 2024 </span>
    </div>
</div>
</main>
{% endblock content %}

```