ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: **«АРХІТЕКТУРА ІНТЕРНЕТ-МАГАЗИНУ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ КОМЕРЦІЙНИХ ПРОЦЕСІВ»**

на здобуття освітнього ступеня магістр
за спеціальності 123 Комп'ютерна інженерія
(код,найменування спеціальності)
освітньо-професійної програми Комп'ютерні системи та мережі
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Олександр РЯБОКОНЬ
(підпис) *(ім'я, ПРІЗВИЩЕ здобувача)*

Виконав: здобувач вищої освіти гр.КСДМ-62
Олександр РЯБОКОНЬ
(ім'я, ПРІЗВИЩЕ)

Керівник: Сергій КОРОТКОВ
к.т.н,доцент, *(ім'я, ПРІЗВИЩЕ)*

Рецензент: _____
(ім'я, ПРІЗВИЩЕ)

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерної інженерії
Ступінь вищої освіти «Магістр»
Спеціальність 123 Комп'ютерна інженерія
Освітньо-професійна прорама Комп'ютерні системи та мережі

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерної інженерії
Наталія ЛАЩЕВСЬКА
(ім'я, ПРІЗВИЩЕ)
"____" _____2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Рябоконю Олександру Олександровичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Архітектура інтернет-магазину для підвищення ефективності комерційних процесів
керівник роботи Сергій КОРОТКОВ phD (доктор філософії), доцент
(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від "15" жовтня 2024 р. №320
2. Строк подання кваліфікаційної роботи 29.12.2024р.
3. Вихідні дані кваліфікаційної роботи:
3.1. Аналіз предметної області.
3.2. Вибір технологій для проекту.
3.3. Проведення аналітики фреймворків, баз даних та додаткових бібліотек.
3.4. Науково-технічна література по темі магістрської роботи
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
3.5. Проектування та розробка інтернет-магазину.
3.6. Створення архітектури проекту, проектування бази даних.
3.7. Створення головної сторінки інтернет-магазину та впровадження модулів для підвищення ефективності комерційних процесів.
5. Перелік ілюстраційного матеріалу: *презентація*

4. Дата видачі завдання “01” вересня 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Підбір технічної літератури	10.10-12.10.2024	
2.	Дослідження інтернет-магазинів	12.10-15.10.2024	
3.	Реалізація проектування та розробки інтернет-магазину	15.10-28.11.2024	
4.	Висновки по роботі	28.11-30.11.2024	
5.	Оформлення роботи	30.11-02.12.2024	
6.	Розробка демонстраційних матеріалів, доповіді.	02.12-08.12.2024	
7.	Передзахист	09.12-11.12.2024	
8.	Здача в деканат	20.12-29.12.2024	

Здобувач вищої освіти_____ Олександр РЯБОКОНЬ

(підпис) *(ім’я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи_____ Сергій КОРОТКОВ

_____

(підпис) *(ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття ступня магістр: 72 стор., 49 рис., 2 табл., 22 джерел.

Мета роботи - дослідження методів розробки інтернет-магазинів, засоби їх створення, адміністрування та покращення функціональності з безпекою.

Об'єкт дослідження - процеси роботи інтернет-магазинів та аналіз їхньої діяльності, доступний у відкритому середовищі мережі Інтернет.

Предмет дослідження – інструменти для проектування інтернет-магазинів і комплект програмних засобів для розробки додатків.

Короткий зміст роботи: У першому розділі було розглянуто приклади інших популярних інтернет-магазинів для аналізу їх структури в подальшому впровадженні їх у проект.

Представлена таблиця порівняння найбільш популярних мов програмування, їх особливості в роботі, переваги та недоліки. Використання мов програмування в залежності від задачі та потреби.

Проведено аналіз фреймворків, баз даних та додаткових бібліотек для визначення кращого методу розробки інтернет-магазину.

У практичній частині виконується впровадження проекту в роботу, формування коду та формування інтернет-магазину.

КЛЮЧОВІ СЛОВА: БАЗА ДАНИХ, ФРЕЙМВОРК, ПРОГРАМУВАННЯ, АРХІТЕКТУРА ПРОЕКТУ, АДМІН ПАНЕЛЬ, ДОДАТКОВА БІБЛІОТЕКА, РЕЄСТРАЦІЯ, АВТОРИЗАЦІЯ, ХОСТИНГ.

ABSTRACT

Text part of the qualification work for the degree of Master: 72 pages, 49 figures, 2 tables, 22 sources.

The purpose of the work is to study the methods of developing online stores, the means of creating, administering and improving their security functionality.

The object of research is the processes of online stores and analysis of their activities available in the open environment of the Internet.

The subject of research is tools for designing online stores and a set of software tools for application development.

Summary of the work: In the first section, examples of other popular online stores were considered to analyze their structure for further implementation in the project.

A table comparing the most popular programming languages, their features, advantages and disadvantages is presented. The use of programming languages depending on the task and need.

An analysis of frameworks, databases, and additional libraries is carried out to determine the best method of developing an online store.

In the practical part, the project is implemented, the code is formed and the online store is created.

KEYWORDS: DATABASE, FRAMEWORK, PROGRAMMING, PROJECT ARCHITECTURE, ADMIN PANEL, ADDITIONAL LIBRARY, REGISTRATION, AUTHORIZATION, HOSTING.

ЗМІСТ

ВСТУП……. 9

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ……...…...11

РОЗДІЛ 2 ВИБІР ТЕХНОЛОГІЙ ДЛЯ ПРОЕКТУ……... 21

2.1 Мова програмування……. 21

2.2 Фреймворк……. 24

2.3 База даних……..……..25

2.4 Додаткові бібліотеки……..26

2.5 Хостинг…….31

РОЗДІЛ 3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ…...34

3.1 Створення архітектури проекту……34

3.2 Проектування бази даних…….. 41

3.3 Реєстрація та авторизація користувачів……44

3.4 Адмін-панель…….. 49

3.5 Додавання каталогу товарів та фільтр……..50

3.6 Оформлення замовлення через кошик……. 52

РОЗДІЛ 4 ВЕБ-ДИЗАЙН ТА ІНТЕРФЕЙС ДЛЯ КОРИСТУВАЧА……54

4.1 Створення головної сторінки з товарами…….…54

4.2 Створення вікна реєстрації та авторизації……56

4.3 Оформлення замовлення…….60

4.4 Оформлення скролу товарів……...62

4.5 Створення розділу обране……..63

4.6 Розділ відгуків…….64

4.7 Створення endpoint…….66

ВИСНОВКИ…… 69

ПЕРЕЛІК ПОСИЛАНЬ……70

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація).....72

ВСТУП

Актуальність теми. У сучасному світі інтернет-магазини відіграють значну роль у житті суспільства. Завдяки зручності, швидкості та багатьом іншим перевагам, вони стали одним із найпоширеніших способів здійснення покупок. Користувачі можуть у будь-який зручний час переглянути асортимент, обрати потрібний товар і замовити його з доставкою прямо до дверей. Ці фактори роблять інтернет-магазини невід'ємною частиною повсякденного життя більшості людей.

Головною умовою для користування інтернет-магазином є наявність пристрою та доступу до Інтернету. Сьогодні понад 90% населення має все необхідне для цього. У нашій країні практично немає проблем із підключенням до мережі — Інтернет доступний майже в кожному будинку, на вулицях, у парках і навіть у віддалених місцях, таких як ліси чи озера. Завдяки технологічному прогресу люди змогли значно спростити своє повсякденне життя.

Користувачі пристроїв із доступом до Інтернету мають майже необмежені можливості. Людина може використовувати ці технології для задоволення своїх потреб у найрізноманітніших формах, що робить їх надзвичайно зручними. Усе, що потрібно, — це дістати телефон із кишені або скористатися іншим доступним пристроєм, щоб виконати необхідні дії.

Мета роботи – вивчення методів і засобів створення, підтримки та управління інтернет-магазинами, включаючи аспекти адміністрування, забезпечення безпеки й функціональності, з подальшою розробкою веб-сторінки інтернет-магазину на основі отриманих знань.

Для виконання поставленої задачі, у магістерській роботі розроблено та виконано наступні завдання:

- аналіз предметної області;
- вибір технологій для проекту;
- проектування та розробка інтернет-магазину.

Об'єкт дослідження – процеси роботи інтернет-магазинів та аналіз їхньої діяльності, доступний у відкритому середовищі мережі Інтернет.

Предмет дослідження – інструменти для проектування інтернет-магазинів і комплект програмних засобів для розробки додатків.

Методи дослідження. У магістерській роботі використовувалися методи інтерактивної інформації, методи штучного інтелекту, математичні підходи до системного аналізу та методи технології розподіленого керування..

Джерела дослідження.

- https://ela.kpi.ua/handle/123456789/33885;

- https://www.education.ua/blog/48050/;

- https://hashdork.com/uk/

- https://er.dduvs.in.ua/bitstream/123456789/2459/1/6.pdf;

- http://4ua.co.ua/sociology/qb3ad69a4d43b88521306d37_0.html.

Наукова новизна одержаних результатів. У магістерській роботі визначені відомості по архітектурі проекту інтернет-магазину, виконано розробку інтернет-магазину та його рекомендації практичної побудови.

Враховуючи особливості створення інтернет-магазину виконано аналіз та підбір найбільш ефективної мови програмування, вибір фреймворків, баз даних та додаткових бібліотек.

Практична значущість одержаних результатів. Отримані результати можна використовувати в подальшій роботі для реалізації інших проектів програмування.

Апробація результатів магістерської роботи. Основні положення та результати магістерської роботи доповідались та обговорювались на двох науково-практичних конференціях.

Публікації. За матеріалами роботи опубліковано одну статтю у науковому журналі.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Спочатку потрібно знайти схожі інтернет-ресурси, не зважаючи на те, на якому саме пристрої вони розглядатимуться (телефон, планшет, комп'ютер, ноутбук тощо). Це дозволить порівняти різні підходи до створення та дизайну сайтів, обравши найбільш підходящий варіант. Як перший приклад було обрано сайт інтернет-магазину Книгарня "Є" [1] (рис. 1.1).

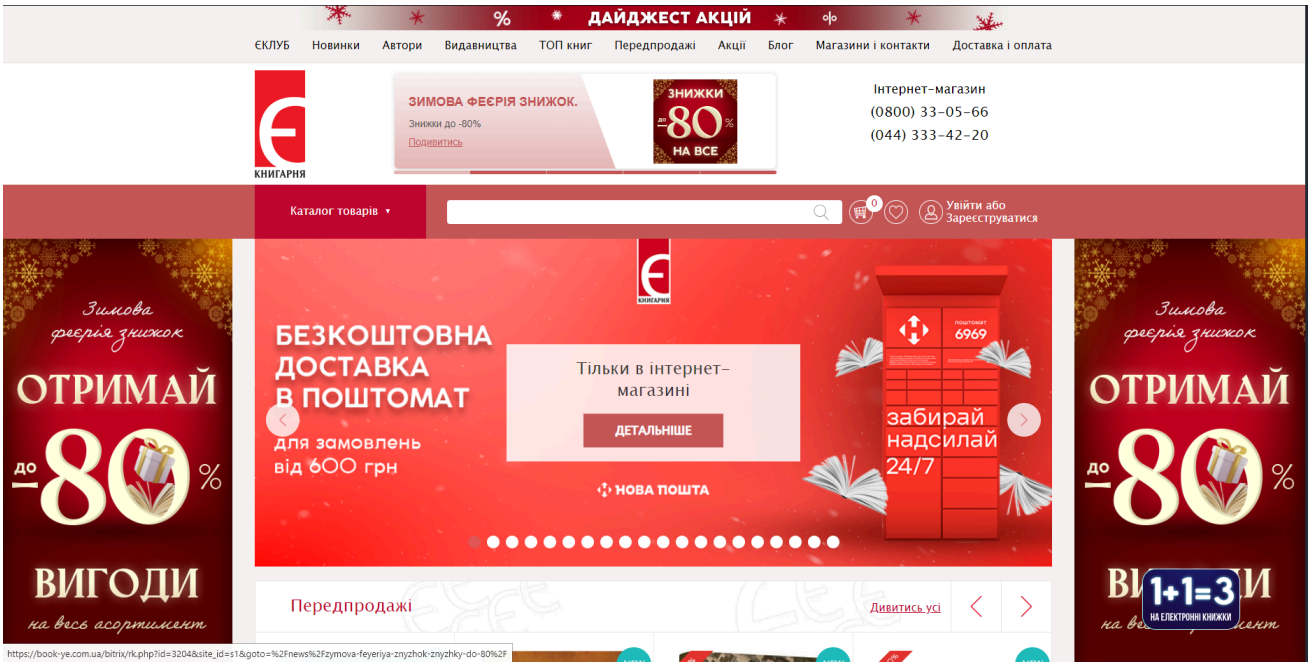


Рисунок 1.1 - Головна сторінка веб сайту «Книгарня "Є"»

Цей сайт пропонує такий набір функцій:

1. Широкий вибір товарів з детальним описом, якісними фотографіями, вказанням ціни, ваги та унікального артикулу для кожного продукту. Можливість ознайомитися з відгуками інших покупців (рисунок 1.2);
2. Реєстрація та авторизація для створення особистого кабінету (рисунок 1.3);
3. Додавання товарів до кошика та просте оформлення замовлення (рисунок 1.4 та рисунок 1.5);

4. Інтуїтивно зрозуміле пошукове поле для швидкого знаходження потрібного товару за різними параметрами (рисунок 1.6);

5. Наявність розділу з додатковою інформацією про магазин, умови доставки та оплати (рисунок 1.7).

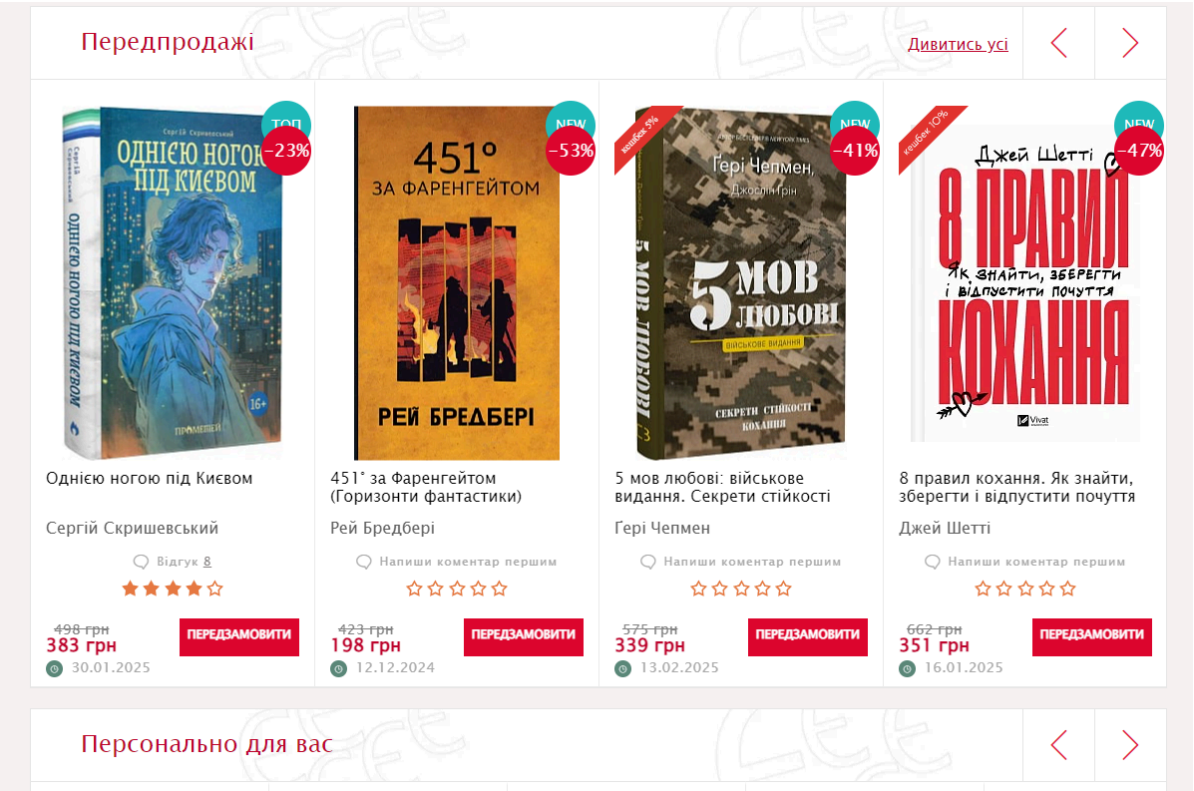


Рисунок 1.2 - Каталог товарів

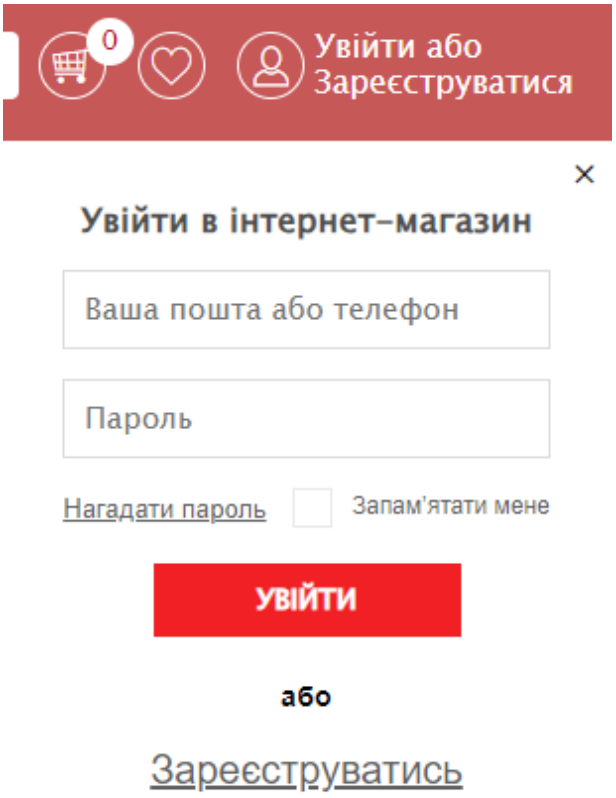


Рисунок 1.3 - Вікно авторизації та реєстрації

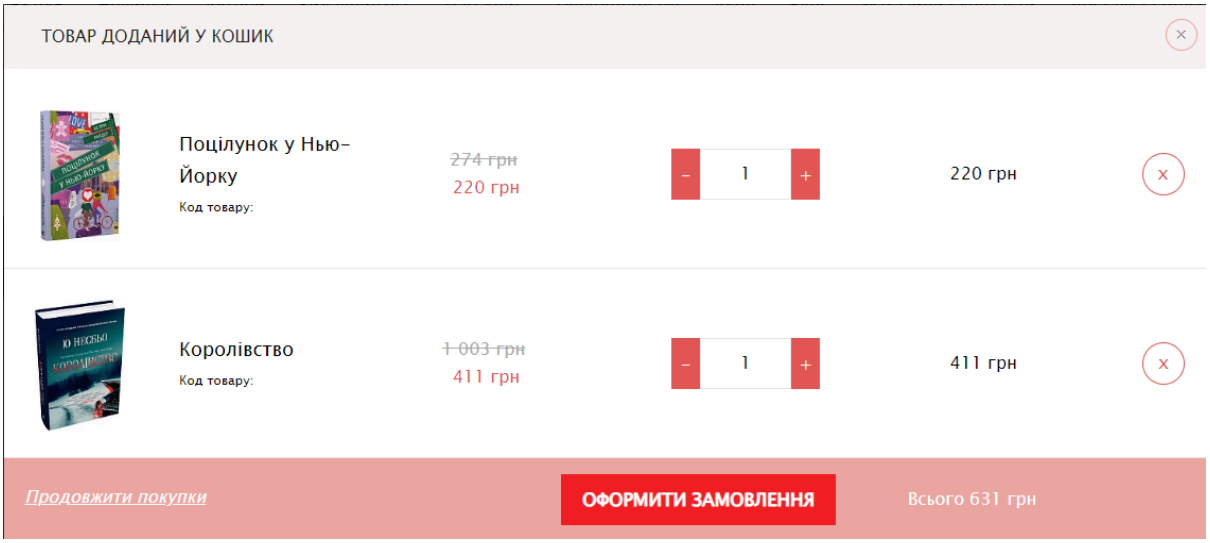


Рисунок 1.4 Кошик наших товарів

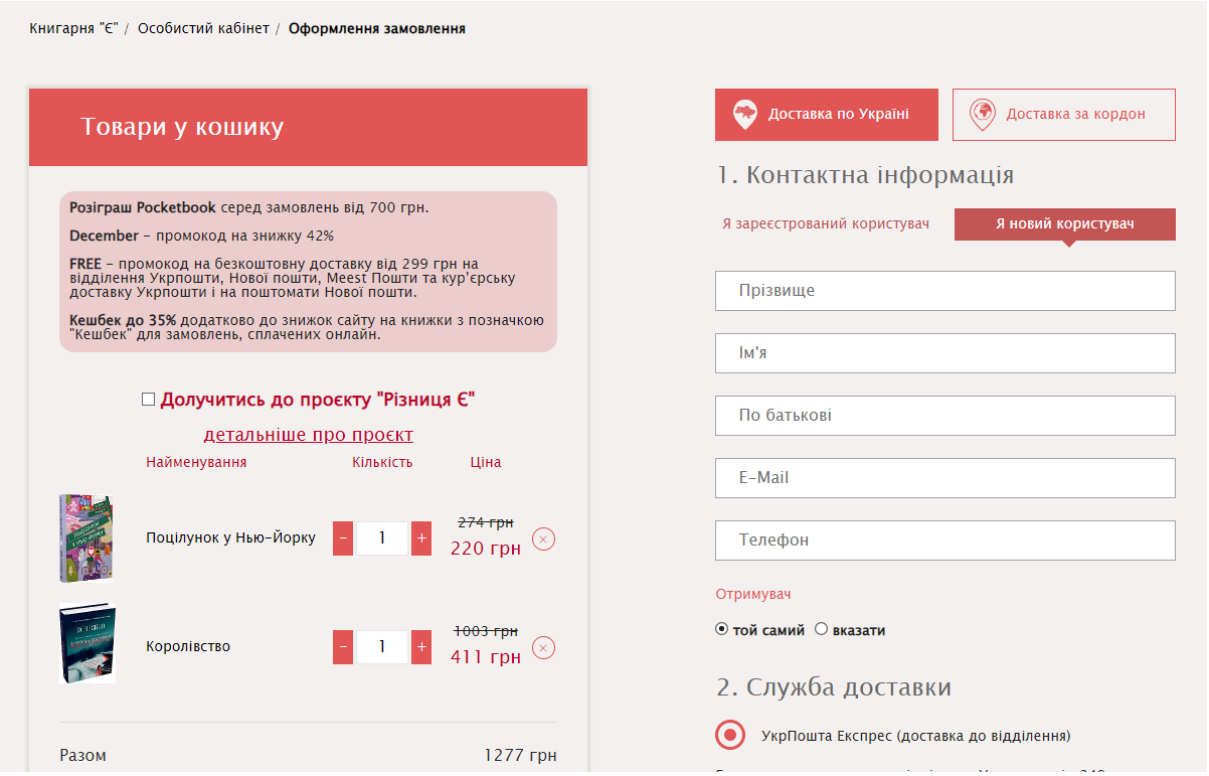


Рисунок 1.5 - Оформлення замовлення

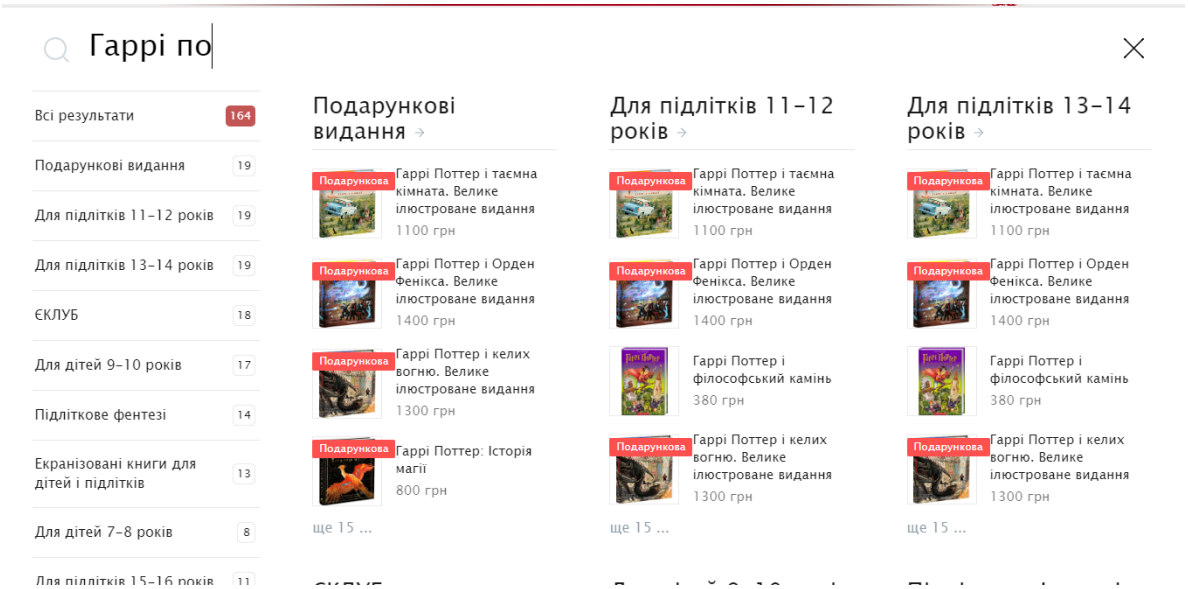


Рисунок 1.6 Пошук необхідних товарів

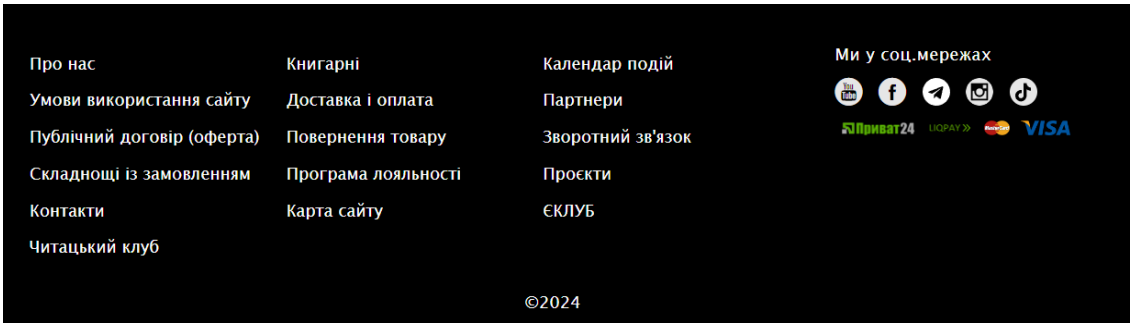


Рисунок 1.7 - Додаткова інформація

Для поглибленого вивчення різноманітних підходів до створення інтернет-магазинів та обрати оптимальне рішення проаналізуємо ще один веб-ресурс.

Другий інтернет магазин, який буде розглянуто та порівняно з попереднім – Ostriv [2] (рис. 1.8).

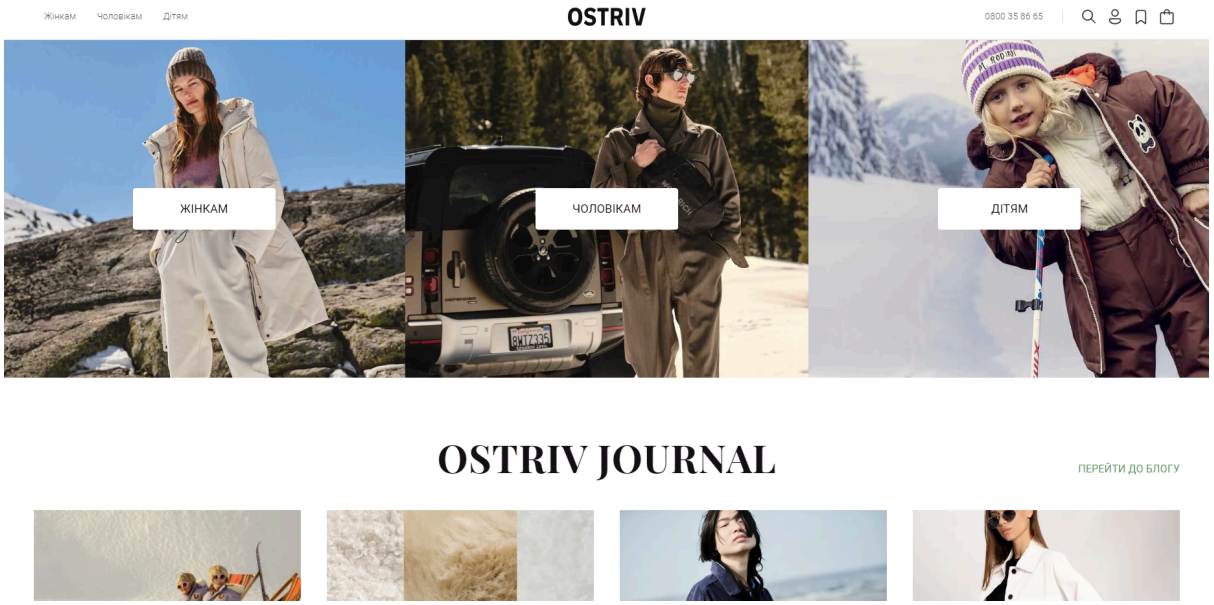


Рисунок 1.8 - Головна сторінка інтернет магазину

На відміну від попереднього інтернет-магазину, новий сайт має більш сучасний та інтуїтивно зрозумілий інтерфейс. Змістовні опції пошуку та яскрава проста гама роблять його більш привабливим для користувачів.

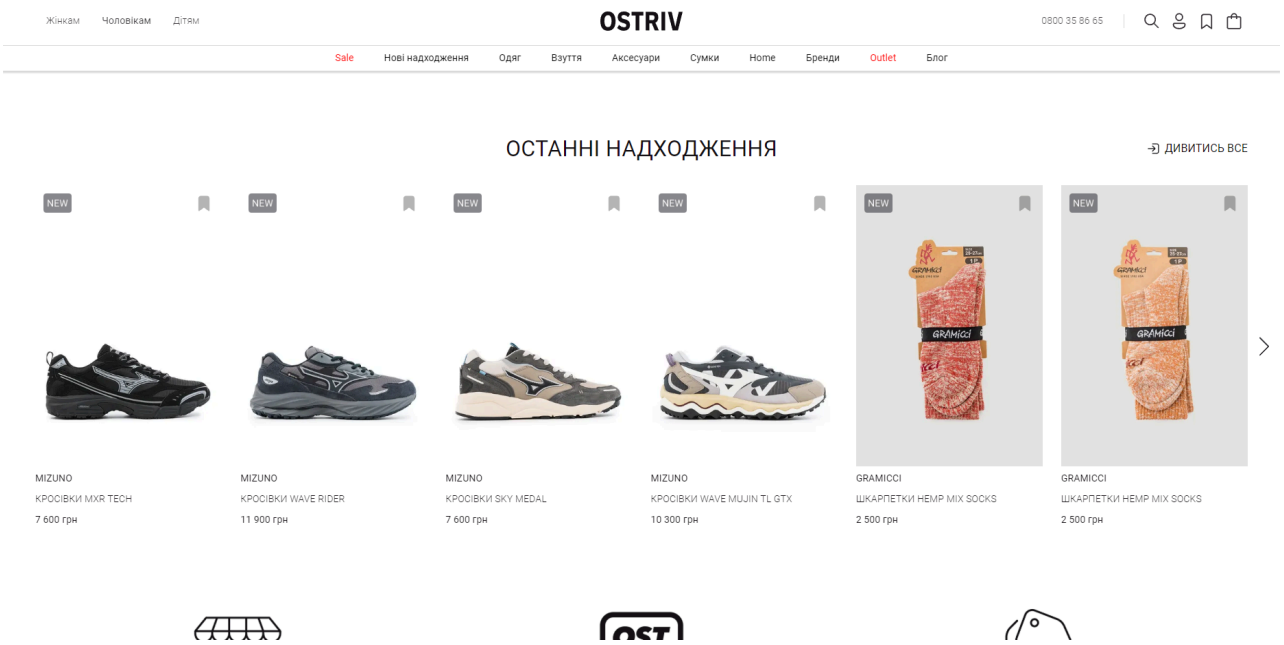


Рисунок 1.9 - Каталог товарів Ostriv

Можна помітити різницю між оформленням каталогів між цими двома магазинами. В книгарня “Є” йде більш детальний опис товару, його рейтинг та можна переглянути відгуки.

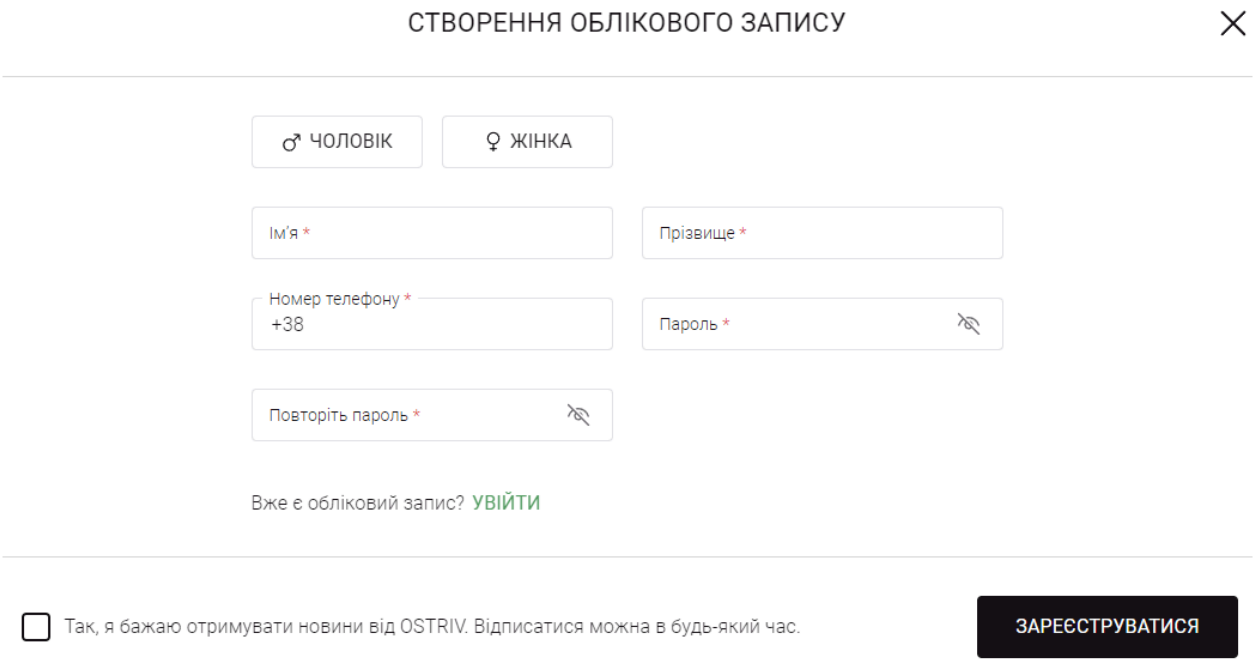


Рисунок 1.10 - Вікно авторизації та реєстрації Ostriv

Різниця між вікном реєстрації та авторизації – велика. Є можливість вказати більш детальну інформацію про себе але при цьому дана форма інтерфейсу не має нічого лишнього. Форма авторизації та реєстрації містить мінімальний набір полів, що дозволяє швидко і легко пройти процедуру.

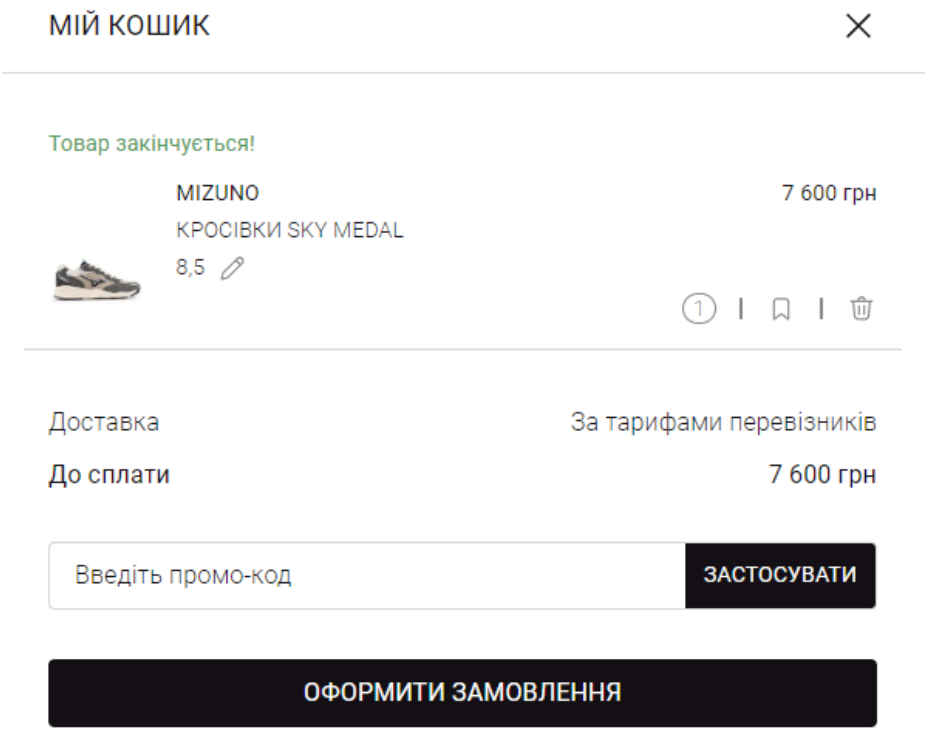


Рисунок 1.11 - Кошик наших товарів Ostriv

Порівняння дизайну оформлення замовлення показало незначні відмінності між платформами. Винятком є Ostriv, де реалізована функція введення промокоду на товар у випадку його наявності.

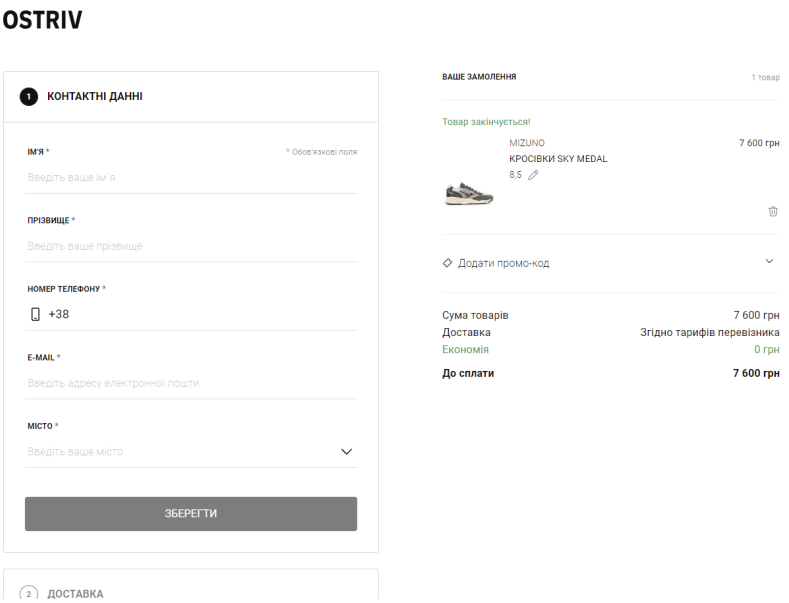


Рисунок 1.12 - Вікно оформлення замовлення Ostriv

Процес оформлення замовлення на всіх сайтах досить стандартний і передбачає введення схожої інформації: ім'я, прізвище, адресу тощо. Тому візуальне оформлення цих форм зазвичай мало відрізняється.

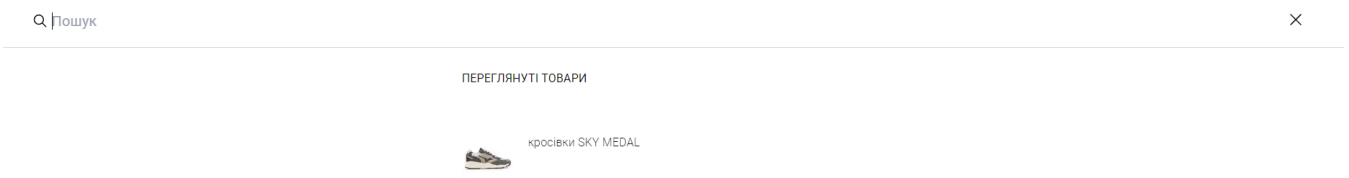


Рисунок 1.13 - Пошук необхідних товарів Ostriv

Можна помітити що на першому прикладі магазин вже надає нам список рекомендованих товарів який базується на попередніх вподобаннях. Водночас магазин Ostriv немає таких функцій, він лише зберігає в історії товари що були переглянуті користувачем раніше.

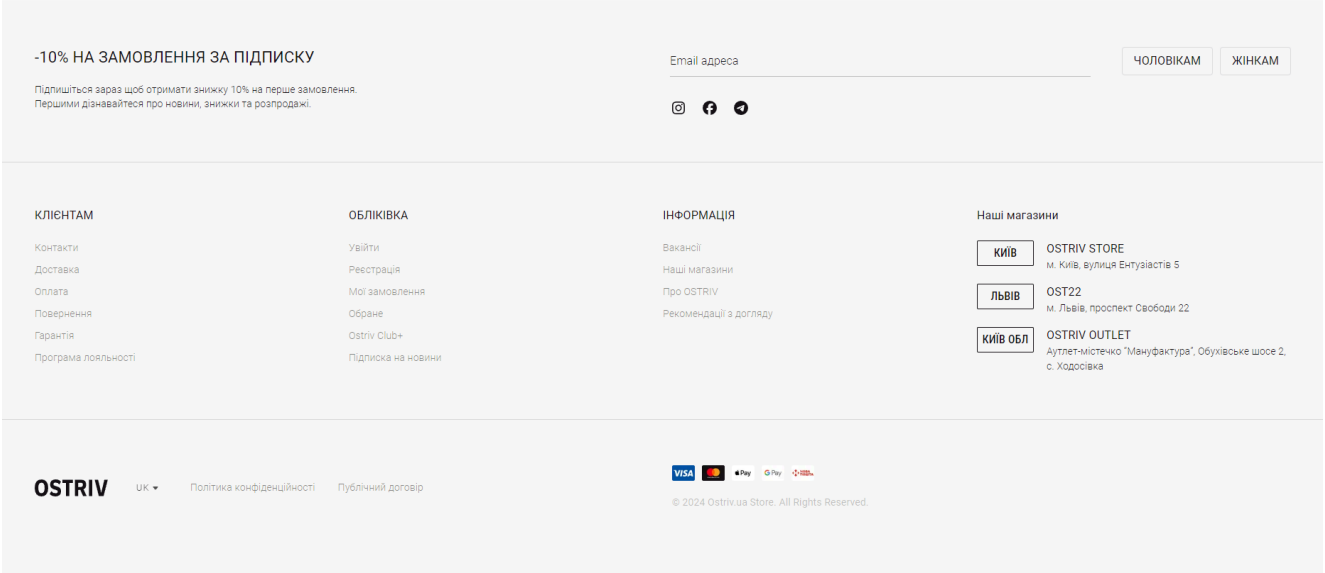


Рисунок 1.14 - Додаткова інформація інтернет магазину Garne

Візуальне оформлення інтернет-магазинів може бути різним, але інформаційне наповнення сторінок оформлення замовлення, як правило, схоже. Кожен магазин доповнює цю інформацію додатковими даними на свій розсуд.

Для полегшення вибору мови програмування та фреймворку для виконання свого інтернет магазину виконано таблицю з порівняннями одних з найпопулярніший сайтів світу, на якій мові вони були написані та які фреймворки були задіяні (табл. 1.1).

Веб-сайт	Мови програмування	Фреймворки	Сфери застосування
Amazon	Python, C++, Perl	Apache Struts	Роздрібна торгівля
eBay	Python	Apache Wicket	Електронна комерція
Alibaba	Python	Spring MVC	Міжнародна торгівля
Etsy	Python, PHP	Django, Laravel	Ручна робота, мистецтво
Shopify	Ruby	Ruby on Rails	Електронна комерція

Flipkart	Python	Play Framework	Роздрібна торгівля
Walmart	Python	Spring MVC	Роздрібна торгівля
Target	Python	Apache Struts	Роздрібна торгівля
Best Buy	Java, PHP	Laravel, Symfony	Електроніка
Newegg	Python	Codelgniter	Комп’ютерна техніка
Google Store	Python	Django, Flask	Електроніка
Zappos	Python	Flask	Взуття, аксесуари
BigBasket	Python	Django, Flask	Інтернет-гастрономія

Таблиця 1.1 – Аналітика вибору мови програмування та фреймворку у популярних світових торговельних гігантів

Висновок. Дослідження інтернет-магазинів виявило, що незважаючи на різноманітність дизайну та кольорової гами, основна структура та функціональність сайтів досить схожі. Це пов'язано з прагненням створити максимально простий та зручний інтерфейс для користувачів.

РОЗДІЛ 2 ВИБІР ТЕХНОЛОГІЇ ПРОЕКТУ

2.1 Мова програмування

Існує величезна кількість мов програмування, кожна з яких має свою унікальну синтаксичну конструкцію та призначення. Також кожна мова має свої недоліки та переваги.

Мова програмування - це штучна мова, створена для передачі команд машинам, зокрема комп'ютерам [3]. Мови програмування слугують інструментом для створення програм, які задають правила роботи комп'ютерів.

Найпопулярнішими в світі вважаються такі мови програмування як JavaScript, Java, Python [4], C# [5], TypeScript та PHP [6]. Кожна з цих мов має свої переваги та недоліки (табл. 2.1):

Особливості	Python	C#	PHP
Тип мови	Інтерпретована, високорівнева	Компільована, високорівнева	Інтерпретована, високорівнева
Призначення	Загальне програмування, веб-розробка наукові обчислення,	Розробка Windows-додатків, ігрова розробка, веб-розробка	Веб-розробка, розробка динамічних сторінок
Синтаксис	Простий, з лаконічною граматикою	Синтаксис, подібний до C/C++	Синтаксис, подібний до C/C++, але зі своїми особливостями

ОПП підтримка	Повна підтримка ООП	Повна підтримка ООП	Часткова підтримка ООП
Управління пам’яттю	Автоматична збірка сміття	Управління пам'яттю з допомогою збирача сміття.	Автоматична збірка сміття
Використання веб-фреймворків	Широкий вибір веб-фреймворків, таких як Django, Flask	Використання фреймворків .NET, таких як ASP.NET	Широкий вибір веб-фреймворків, таких як Laravel, Symfony
Розширюваність	Велика кількість модулів та бібліотек	Велика кількість бібліотек, особливо для Windows-розробки	Широкий спектр розширень та бібліотек
Використання в інших областях	Наукові дослідження, аналіз даних, штучний інтелект	Розробка десктопних додатків, ігрова розробка, розробка мобільних додатків з використанням Xamarin	Веб-розробка, розробка динамічних сторінок

Таблиця 2.1 - Порівняння популярних мов програмування

Таблиця 2.1 є своєрідним довідником для розробника, який допомагає оцінити переваги та недоліки різних мов програмування. Завдяки цьому можна підібрати оптимальний інструмент для реалізації будь-якої ідеї. Широкий спектр мов програмування дозволяє створювати як прості скрипти, так і складні програмні системи. Кожен розробник має можливість обрати мову, яка найбільш відповідає його стилю роботи та вимогам проекту. Саме ця різноманітність і є рушійною силою розвитку програмної індустрії.

Кожен веб-сайт має унікальний програмний код, який реалізований за допомогою однієї або кількох мов програмування. На прикладі Amazon можна побачити, що великі інтернет-магазини часто використовують комбінацію різних мов для забезпечення оптимальної продуктивності та функціональності. Найбільш поширені мови програмування, якими користується Amazon, включають:

- Java: Java є однією з основних мов програмування, яку використовує Amazon для розробки своїх серверних компонентів, веб-додатків та внутрішніх систем.

- C++: Amazon також активно використовує мову програмування C++ для низькорівневих розробок, оптимізації продуктів та компонентів, а також для побудови швидких та ефективних систем.

- Python: Python широко використовується в Amazon для автоматизації, розробки скриптів, аналізу даних та створення прототипів нових продуктів.

- JavaScript: Amazon використовує JavaScript для розробки клієнтської частини своїх веб-додатків та веб-інтерфейсів, включаючи фронтенд Amazon.com та інші веб-сайти.

- Ruby: Деякі сервіси Amazon, зокрема Amazon Elastic Beanstalk, використовують Ruby для розробки та керування веб-додатками.

Це лиш кілька прикладів мов програмування, які використовуються в Amazon. Компанія використовує різні мови програмування, щоб адаптуватися до мінливих вимог проектів.

Аналізуючи технологічний стек популярних українських інтернет-магазинів, варто звернути увагу на Rozetka. Основна мова програмування цього масштабного проекту – це Java. Саме на цій мові програмування написана більша частина серверної логіки, яка відповідає за обробку замовлень, управління товарами та

взаємодію з базою даних. Однак, крім Java, у розробці Rozetka беруть участь і інші мови, кожна з яких виконує свою специфічну функцію. Такий підхід дозволяє створювати гнучкі, масштабовані та надійні веб-додатки, здатні задовольнити потреби мільйонів користувачів.

Крім того, для фронтенду та клієнтської частини, використовуються такі мови:

- JavaScript: JavaScript використовується для розробки динамічного та інтерактивного фронтенду, включаючи валідацію форм, анімацію та взаємодію з користувачем.
- HTML/CSS: HTML використовується для створення структури веб-сторінок, а CSS - для оформлення та стилізації їх зовнішнього вигляду.
- SQL: Для зберігання та управління даними використовується мова SQL (Structured Query Language). Вона використовується для взаємодії з базою даних та виконання запитів.
- Інші мови: Крім цього, можуть використовуватися інші мови програмування або технології для конкретних компонентів або сервісів на сайті Rozetka.

Варто відзначити, що це загальна інформація про мови програмування, які зазвичай використовуються на сайті Rozetka. Актуальний технічний стек або мови можуть змінюватись з часом, оскільки веб-розробка є динамічним процесом.

2.2 Фреймворк

Фреймворк — інфраструктура програмних рішень, що полегшує розробку складних систем [7]. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою, але при цьому вона має ряд обмежень, що задають правила створення структури проекту та написання коду.

Фреймворки поділяються на:

- Бекенд-фреймворки;
- Фронтенд-фреймворки;
- Фуллстек-фреймворки.

Бекенд фреймворки – це бібліотеки серверних мов програмування [7]. Вони допомагають у створенні внутрішньої конфігурації веб-застосунків.

Внутрішні веб-платформи пропонують готовий набір інструментів, які спрощують розробку веб-додатків. Вони беруть на себе рутинні завдання, такі як автентифікація користувачів, захист даних та управління базами даних, дозволяючи розробникам зосередитися на унікальній функціональності своїх проектів.

Фронтенд-фреймворки забезпечують однакове відображення веб-сторінки у всіх браузерах [7]. Спрощують роботу зі стилями, анімацією та дозволяють створювати привабливі інтерактивні інтерфейси. Серед найпоширеніших фронтенд-фреймворків – Vue. js, Bootstrap, Foundation, Angular та інші.

Фуллстек-фреймворки - це універсальні фреймворки, які здатні розв'язувати задачі як на стороні клієнта, так і на стороні сервера [7]. Серед них:

- Meteor. Працює з JavaScript як на фронтенді, так і на бекенді. Завдяки режиму реального часу розробник відразу бачить зміни у проекті, що суттєво скорочує час на тестування та розробку.
- Next.js. Це окреме рішення, створене для JavaScript, яке допомагає вирішувати проблему React.js. Відображення на стороні сервера на React.js вирішується погано, тому і був розроблений цей фреймворк.
- Nuxt. Додаток до Vue.js, що допомагає робити універсальні програми. Працює з JavaScript.

Як правило, фулстек фреймворки використовуються як додаткові інструменти для вирішення конкретних завдань. Але все залежить від проекту. Перед вибором стеку технологій, наші аналітики вивчають особливості бізнес-процесів вашої компанії, щоб підібрати правильні рішення.

2.3 База даних

SQLite - це вбудована реляційна база даних, яка надає легку та ефективну систему керування базами даних [8]. Особливістю SQLite є те, що вона працює без окремого сервера баз даних та зберігає всі дані в одному файлі на диску. Ось деякі ключові риси та властивості SQLite:

- Вбудована база даних: SQLite побудована як бібліотека, яку можна вбудовувати безпосередньо в програми. Вона не потребує окремого сервера баз даних або налаштувань, що робить її легкою у використанні та розгортанні.

- Нульовий настрій: SQLite підтримує нульовий настрій, що означає, що база даних може бути доступна для читання та запису одночасно без блокування всієї бази даних. Це робить її ефективною для багатопотокових та багатокористувацьких сценаріїв.

- Підтримка SQL: SQLite використовує SQL (Structured Query Language) для взаємодії з базою даних. Вона підтримує багато розширень SQL, включаючи складні запити, об'єднання, сортування, групування та багато інших функцій.

- Малий розмір та ефективність: SQLite має невеликий розмір файлу та мінімальні вимоги до ресурсів. Вона ефективно використовує пам'ять та процесор, що робить її швидкою та легкою у використанні навіть на пристроях з обмеженими ресурсами.

- Транзакційна безпека: SQLite підтримує ACID-властивості (Atomicity, Consistency, Isolation, Durability) транзакцій, що забезпечує цілісність та надійність даних.

- Кросплатформеність: SQLite підтримується на багатьох операційних системах, включаючи Windows, macOS, Linux, Android та інші. Це робить його універсальним рішенням для розробки кросплатформенних програм.

SQLite широко використовується в різних сферах, включаючи мобільну розробку, вбудовані системи, десктопні додатки та багато інших сценаріїв, де потрібна легка та ефективна база даних.

2.4 Додаткові бібліотеки

Python має кілька вбудованих бібліотек, які постачаються разом з інсталяцією мови. Ось декілька основних вбудованих бібліотек Python [9]:

- math: Ця бібліотека надає математичні функції та константи для виконання різних операцій над числами.

- random: Вона використовується для генерації випадкових чисел, вибірки зі списків, перемішування елементів тощо.

- datetime: Ця бібліотека надає функціональність для роботи з датами і часом, включаючи створення, форматування та виконання операцій над датами.

- os: Вона дозволяє взаємодіяти з операційною системою, надаючи функції для створення / видалення файлів та папок, отримання шляху до поточної директорії, взаємодії зі змінними середовища тощо.

- sys: Ця бібліотека надає доступ до функцій і змінних, пов'язаних з інтерпретатором Python. Вона дозволяє керувати аргументами командного рядка, взаємодіяти з виключеннями та іншими системними параметрами.

- json: Вона надає функції для роботи з форматом даних JSON, зокрема для серіалізації (перетворення об'єктів Python у рядок JSON) та десеріалізації (перетворення рядка JSON у об'єкти Python).

- re: Це регулярні вирази, які надають потужні засоби для роботи з текстом, пошуку та заміни певних патернів.

Це частина вбудованих бібліотек Python. Існує ще великий обсяг інших, які надають функціональність для різних задач.

У Python існує безліч популярних бібліотек, які надають різноманітні функціональні можливості. Ось кілька з них:

- NumPy: Бібліотека для наукових обчислень, яка надає потужні масиви та функції для роботи з ними.

- Pandas: Бібліотека для аналізу та обробки даних. Вона надає інструменти для маніпулювання, фільтрації, групування та агрегування даних.

- Matplotlib: Бібліотека для візуалізації даних. За допомогою Matplotlib можна створювати графіки, діаграми, гістограми та інші типи візуалізаційних елементів.

- TensorFlow: Відкрита бібліотека для машинного навчання та глибинного навчання. Вона надає інструменти для побудови та навчання нейронних мереж.

- Scikit-learn: Бібліотека для машинного навчання, яка містить реалізації різних алгоритмів класифікації, регресії, кластеризації та інших.

- Keras: Високорівневий інтерфейс для побудови нейронних мереж, що працює поверх TensorFlow. Він спрощує розробку та навчання моделей глибинного навчання.

Django: Веб-фреймворк, що допомагає створювати веб-додатки та веб-сервери з використанням Python. Він надає зручні інструменти для роботи зі сторінками, маршрутизації, базами даних тощо.

- Flask: Легкий веб-фреймворк для створення веб-додатків та API. Він забезпечує просту структуру та розширені можливості налаштування.

- PyTorch: Відкрита бібліотека для машинного навчання та глибинного навчання, яка активно використовується у дослідженнях та промислових застосуваннях.

- Requests: Бібліотека для взаємодії з мережевими ресурсами, зокрема HTTP-запитами. Вона дозволяє здійснювати запити до веб-серверів та обробляти отримані відповіді.

Наведений список бібліотек Python є далеко не вичерпним. Існує велика кількість сторонніх бібліотек, які розширюють функціональність мови Python. Вибір бібліотеки залежить від специфічних вимог проекту та особистих переваг розробника.

В цьому проекті використано наступні бібліотеки: pillow, beautifulsoup4, numpy, pandas, six. Про pandas та numpy я коротку інформацію вказав, давайте розглянемо інші.

Pillow - це популярна бібліотека для обробки зображень у мові програмування Python. Вона надає широкий набір функцій для відкриття, маніпулювання та збереження зображень у різних форматах.

Основні можливості бібліотеки Pillow включають:

- Завантаження та збереження зображень: Pillow дозволяє відкривати зображення з файлів у різних форматах, таких як JPEG, PNG, BMP, TIFF та інші. Ви також можете зберегти зображення в потрібному форматі.

- Маніпуляція зображеннями: Pillow надає широкий набір функцій для маніпулювання зображеннями. Ви можете змінювати розмір зображення, обрізати його, обертати, перевертати, виконувати зміни кольору, налаштовувати яскравість, контрастність та інші атрибути.

- Робота з пікселями: Ви можете отримувати доступ до окремих пікселів зображення та змінювати їх значення. Це дозволяє вам виконувати детальну обробку зображень, таку як фільтрація, ретушування або застосування ефектів.

- Обробка зображень у масштабі: Pillow підтримує операції масштабування зображень, включаючи збільшення та зменшення розміру зображення зі збереженням деталей та якості.

- Робота з кольором: Бібліотека Pillow надає можливості для роботи з кольором зображень. Ви можете змінювати кольорові простори, перетворювати зображення в чорно-біле або змінювати інтенсивність кольору.

- Налаштування зображень: Ви можете виконувати різні операції з налаштуванням зображень, такі як зміна розміру, обрізка, обертання, перевертання та інші.

Pillow є простою та потужною бібліотекою для обробки зображень у Python. Вона широко використовується у різних сферах, включаючи наукові дослідження, комп'ютерне зору, веб-розробку та обробку зображень у загальному.

BeautifulSoup4 (або просто BeautifulSoup) є популярною бібліотекою для парсингу HTML і XML даних у Python. Вона надає зручний і простий спосіб витягувати потрібну інформацію з веб-сторінок або XML-файлів, а також здійснювати маніпуляції з цими даними.

Основні риси BeautifulSoup4:

- Парсинг HTML і XML: BeautifulSoup4 дозволяє зручно і легко зчитувати HTML- і XML-документи. Вона автоматично розпізнає кодування документа і здатна працювати з різними видами розмітки.

- Навігація по документу: BeautifulSoup4 надає потужні засоби для навігації по структурі документа. Ви можете переходити між елементами, отримувати їх атрибути, знаходити батьківські або дочірні елементи, а також шукати елементи за різними критеріями.

- Вибірка даних: За допомогою BeautifulSoup4 можна витягувати потрібні дані з HTML-сторінок або XML-файлів. Ви можете шукати елементи за тегами, класами, ідентифікаторами або за вмістом елементів. BeautifulSoup4 надає зручні методи для отримання тексту, атрибутів, URL-посилань, форм, таблиць та інших елементів.

- Маніпуляція з даними: BeautifulSoup4 дозволяє змінювати структуру HTML-документа або XML-файлу. Ви можете додавати, видаляти або модифікувати елементи, атрибути, текст або інші властивості документа.

- Інтеграція з парсерами: BeautifulSoup4 підтримує різні парсери, такі як "html.parser", "lxml" і "html5lib". Ви можете вибрати оптимальний парсер в залежності від ваших потреб.

- Простий у використанні: BeautifulSoup4 має простий та інтуїтивно зрозумілий інтерфейс, що дозволяє швидко розпочати роботу з парсингом даних.

Узагальнюючи, BeautifulSoup4 є потужним інструментом для парсингу HTML і XML, який допомагає отримувати, аналізувати та маніпулювати даними з веб-сторінок або XML-файлів у зручному форматі.

Бібліотека six є важливим інструментом в екосистемі Python, який надає допоміжні функції для підтримки сумісності між версіями Python 2 і Python 3. Основна мета six - зробити розробку коду, який може працювати як у середовищі Python 2, так і у середовищі Python 3, більш простою і зручною.

Основні риси six:

- Сумісність версій Python: six надає набір функцій, класів і декораторів, які дозволяють розробникам писати код, який буде працювати як у Python 2, так і у Python 3. Це дозволяє зменшити зусилля, необхідні для підтримки обох версій мови.

- Універсальні типи даних: six надає крос-платформні типи даних, які працюють і в Python 2, і в Python 3. Наприклад, вона має six.text_type, який представляє тип unicode в Python 2 і тип str в Python 3. Це допомагає забезпечити єдиний код, який може оперувати з різними типами даних, залежно від версії Python.

- Крос-платформні функції: six містить набір корисних функцій, які працюють однаково в Python 2 і Python 3. Наприклад, вона має функцію six.print_, яка може бути використана для друку тексту безпосередньо в коді, незалежно від версії Python.

- Декоратори і контекстні менеджери: six надає декоратори і контекстні менеджери, які спрощують роботу з різними версіями Python. Наприклад,

декоратор six.add_metaclass дозволяє встановлювати метаклас для класу безпосередньо в коді, що допомагає підтримувати сумісність з Python 2 і Python 3.

- Інші корисні функції: six також має інші корисні функції, такі як six.moves, яка надає альтернативи для модулів і функцій, що змінили назву між версіями Python, і six.raise_from, яка дозволяє викидати виключення зберігаючи контекст виникнення.

- Користуватися six дуже просто: достатньо імпортувати необхідні функції, класи чи декоратори з модуля six і використовувати їх у своєму коді.

Узагальненням, six є потужним інструментом для розробників Python, який допомагає підтримувати сумісність між версіями Python 2 і Python 3, спрощуючи написання єдиного коду, який може працювати на обох платформах.

Висновок. Другий розділ присвячений аналізу технологічного стеку проекту. Після ретельного порівняння різних варіантів, ми зупинилися на Python завдяки його високій продуктивності та широким можливостям. Для прискорення розробки веб-магазину ми обрали фреймворк Django, який пропонує готові компоненти для створення динамічних веб-сайтів. А для зберігання даних ми обрали SQLite, яка є простим у використанні та добре інтегрується з Django

2.5 Хостинг

Хостинг — послуга надання дискового простору, підключення до мережі та інших ресурсів для розміщення фізичної інформації на сервері, що постійно перебуває в мережі (наприклад інтернеті) [10].

Поняття хостингу включає в себе широкий спектр послуг із використанням різного апаратного та програмного забезпечення. Зазвичай під поняттям послуги хостингу мають на увазі, як мінімум, послугу розміщення файлів сайту на сервері, на якому запущене ПЗ, необхідне для обробки запитів до цих файлів (веб сервер). Як правило, до послуг хостингу вже входить надання місця для поштової кореспонденції, баз даних, DNS файлового сховища тощо, а також підтримка функціонування відповідних сервісів, однак вони можуть надаватися і окремо. А саме, послуга може бути обмежена розміщенням поштової кореспонденції та відповідного ПЗ (поштовий хостинг), клієнтських файлів (файловий хостинг),

виключно відео файлів (відео хостинг) або інших файлів певного типу та за певними умовами [10].

Хостинг є ключовим елементом для існування сайту, оскільки саме завдяки йому сайт стає доступним для користувачів в інтернеті. Основні його ролі:

- Зберігання файлів сайту: хостинг забезпечує місце на сервері для зберігання всіх файлів сайту: HTML, CSS, JavaScript, зображень, баз даних тощо.
- Доступність сайту цілодобово: сервер хостингу працює постійно, забезпечуючи доступ до сайту в будь-який час з будь-якої точки світу.
- Обробка запитів користувачів: коли користувач вводить адресу сайту в браузері, сервер хостингу обробляє цей запит, знаходить потрібні файли та надсилає їх на пристрій користувача.
- Підтримка доменного імені: хостинг часто інтегрується з доменним ім'ям, дозволяючи пов'язати URL-адресу сайту з його фізичним розташуванням на сервері.
- Захист та безпека: багато хостинг-провайдерів пропонують додаткові функції, як-от захист від атак DDoS, регулярні резервні копії даних, SSL-сертифікати для захищеного з'єднання тощо.
- Продуктивність та швидкість: якість хостингу впливає на швидкість завантаження сайту. Сервери з потужним обладнанням і оптимізованим програмним забезпеченням забезпечують швидке обслуговування запитів.
- Масштабованість: хостинг дозволяє масштабувати ресурси сайту відповідно до його потреб. Наприклад, якщо сайт починає отримувати більше трафіку, можна збільшити обсяг серверних ресурсів.

Типи хостингу:

Віртуальний (shared hosting): Один сервер використовується кількома сайтами.

Віртуальний виділений сервер (VPS): Окрема віртуальна частина сервера для одного користувача.

Виділений сервер: Сервер цілком належить одному користувачеві.

Хмарний хостинг: Дані розміщуються на кількох серверах одночасно, забезпечуючи гнучкість та надійність.

Без хостингу сайт просто не зможе існувати в інтернеті, оскільки йому потрібна інфраструктура для функціонування.

РОЗДІЛ 3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ

3.1 Створення архітектури проекту

Архітектура проекту програмування - це структура та організація програмного проекту, яка визначає, як різні компоненти системи взаємодіють між собою, як вони організовані та як вони вирішують поставлені завдання [11]. Архітектура визначає основні принципи, патерни, структури та механізми, які використовуються для побудови програмного проекту.

Основні поняття та складові архітектури проекту програмування:

- Компоненти: Це основні частини системи, які мають визначену функціональність. Компоненти можуть бути класами, модулями, пакетами або сервісами. Кожен компонент повинен мати чітко визначений обсяг відповідальності і виконувати конкретну роль у системі.

- Модулі: Модулі представляють логічні блоки функціональності, які групуються разом у відповідності до певних критеріїв, таких як схожість функцій, залежності або обсяг відповідальності. Вони допомагають організувати код і забезпечують модульність.

- Структури даних: Це способи організації та зберігання даних в системі. Це можуть бути списки, масиви, хеш-таблиці, дерева або інші структури, які відповідають вимогам проекту.

- Інтерфейси: Інтерфейси визначають, як компоненти або модулі взаємодіють між собою. Це можуть бути API, протоколи обміну даними, контракти або специфікації поведінки компонентів.

- Патерни проектування: Патерни проектування - це визначені рекомендації та шаблони розробки, які допомагають вирішувати типові проблеми при побудові програмних систем. Вони надають стандартизований підхід до організації коду та забезпечують гнучкість та перевикористання.

- Механізми комунікації: Це способи передачі даних та повідомлень між компонентами або модулями системи. Це можуть бути синхронні або асинхронні виклики, події, черги повідомлень або інші механізми.

- Управління залежностями: Це механізми та практики для керування залежностями між компонентами або модулями. Це можуть бути інструменти збирання залежностей, ін'єкція залежностей або конфігураційні файли.

- Безпека: Архітектура повинна враховувати питання безпеки, такі як захист даних, автентифікація, авторизація, контроль доступу і забезпечення конфіденційності.

Архітектура проекту програмування покликана забезпечити чітку організацію та розподіл відповідальностей, забезпечити перевикористання коду, зручну розширюваність та підтримку. Коректно спроектована архітектура сприяє розробці, тестуванню, супроводженню та масштабуванню програмного проекту.

Віртуальне середовище (Virtual Environment) - це ізольоване середовище у веб-розробці, яке дозволяє вам встановлювати та управляти окремими версіями пакетів Python для кожного проекту [11]. Віртуальне середовище дозволяє уникнути конфліктів між різними версіями пакетів та забезпечує чистоту і порядок у вашому проекті.

Основні переваги використання віртуального середовища:

- Ізоляція: Кожне віртуальне середовище має свій власний набір пакетів і залежностей, що дозволяє уникнути конфліктів між різними проектами. Ви можете мати різні версії пакетів для кожного проекту без впливу на інші проекти.

- Чистота: Віртуальне середовище допомагає зберігати ваш проект чистим і організованим. Ви можете встановлювати лише необхідні пакети для вашого проекту, не завантажуючи зайві компоненти.

- Легкість керування залежностями: Ви можете легко керувати версіями пакетів у вашому віртуальному середовищі. Ви можете встановлювати, оновлювати та видаляти пакети лише для конкретного проекту без впливу на інші проекти.

- Переносимість: Віртуальне середовище дозволяє вам легко перенести ваш проект на іншу систему або передати його іншим розробникам. Ви можете

створити віртуальне середовище, включивши список встановлених пакетів, і передати цей список для встановлення тим, хто працюватиме з проектом.

- Тестування: Віртуальне середовище дозволяє вам створювати окремі середовища для тестування різних версій пакетів і залежностей. Це сприяє відладці та впровадженню проекту.

У Python існує кілька інструментів для створення та керування віртуальними середовищами, таких як venv, virtualenv, conda, pipenv тощо. Ви можете вибрати той, який найкраще відповідає вашим потребам.

Для створення проекту в PyCharm, нам необхідно натиснути на розділ File і у відкритому вікні обрати New Project. Після цих дій відкривається вікно створення нашого проекту, де необхідно вказати назву проекту та обрати версію Python. Версію необхідно встановити якомога більш останню.

Після виконання цих дій у нас створюється автоматично віртуальне середовище проекту яке ми можемо використовувати та виконувати в ньому необхідні для нас дії.

Далі необхідно встановити Django використовуючи pip.

PIP (Package Installer for Python) - це стандартний пакетний менеджер для мови програмування Python [12]. Він дозволяє легко встановлювати, оновлювати та керувати пакетами та залежностями Python у вашій системі.

Основні функції та можливості pip включають:

- Установка пакетів: За допомогою pip ви можете легко встановлювати пакети Python з репозиторіїв PyPI (Python Package Index) та інших джерел. Просто виконайте команду pip install package_name для встановлення пакета.

- Оновлення пакетів: Ви можете оновлювати встановлені пакети до останніх доступних версій за допомогою команди pip install --upgrade package_name або pip install --upgrade для оновлення всіх пакетів.

- Керування залежностями: pip дозволяє керувати залежностями вашого проекту, забезпечуючи, що всі необхідні пакети будуть встановлені та оновлені. Ви можете визначити залежності у файлі requirements.txt та встановити їх командою pip install -r requirements.txt.

- Видалення пакетів: За допомогою pip ви можете видалити встановлені пакети з вашої системи за допомогою команди pip uninstall package_name.

- Пошук пакетів: pip надає можливість шукати пакети в репозиторії PyPI за допомогою команди pip search search_query.

Робота з віртуальними середовищами: pip добре поєднується з віртуальними середовищами, такими як venv або virtualenv. Ви можете встановлювати пакети та керувати залежностями в межах конкретного віртуального середовища.

Це лише кілька основних функцій pip. Існує багато додаткових можливостей та параметрів командного рядка для використання pip. Для отримання більш детальної інформації ви можете скористатися документацією, виконавши команду pip help або відвідати офіційний сайт pip.

Для встановлення Django треба використати ось таку команду в терміналі “pip install Django”

Ця команда завантажить в проект фреймворк Django. Далі потрібно саме створити проект Django використовуючи команду “django-admin startproject fin_server”

Після всіх дій проект буде мати ось такий скелет проекту:

```
ecommerce/
  manage.py
  ecommerce/
        settings.py
        wsgi.py
        urls.py
```

Одним з найголовнішим файлом проекту є - manage.py який є тісно пов’язаний з Django, він надає доступ до різних утиліт командного рядка, які допомагають керувати Django-проектом. Він знаходиться у кореневому каталозі вашого Django-проекту.

Завдяки файлу manage.py виконуються такі дії:

- Запуск сервера розробки: Ми можемо запустити вбудований сервер розробки Django за допомогою команди python manage.py runserver. Він запустить сервер, який буде слухати локальний хост і порт за замовчуванням (зазвичай http://localhost:8000/), на якому ми можемо переглядати наш Django-проект.

- Міграції бази даних: Django надає механізм міграцій для збереження та застосування змін до бази даних. Ми можемо створювати міграції, виконуючи

команди типу python manage.py makemigrations та застосовувати їх за допомогою python manage.py migrate.

- Створення адміністративного облікового запису: Ми можемо створити обліковий запис адміністратора для адміністративного інтерфейсу Django, використовуючи команду python manage.py createsuperuser.

- Виконання тестів: Django дозволяє нам створювати тести для вашого проекту. Ми можемо запустити тести, виконавши команду python manage.py test.

- Виконання інших утиліт: У файлі manage.py також доступні інші утиліти командного рядка, такі як shell, check, collectstatic, flush тощо.

Файл manage.py забезпечує зручний спосіб керування різними аспектами нашого Django-проекту безпосередньо з командного рядка. Manage.py є скриптом, який використовується в багатьох проектах на основі фреймворку Django. Цей скрипт забезпечує зручний спосіб управління різними аспектами проекту, включаючи запуск сервера розробки, управління міграціями бази даних, створення адміністративних команд та інші операції.

Скрипт manage.py зазвичай знаходиться в кореневому директорії проекту Django і використовується для виконання різних команд через командний рядок або термінал. Ось деякі з команд, які часто використовуються, які можна виконати за допомогою manage.py:

- runserver: Запускає вбудований сервер розробки Django, що дозволяє переглядати проект у локальному середовищі розробки.

- makemigrations: Створює файли міграції на основі змін у моделях проекту. Міграції використовуються для автоматичного оновлення баз даних при зміні схеми моделей.

- migrate: Застосовує міграції та оновлює базу даних відповідно до поточного стану міграцій.

- createsuperuser: Створює суперкористувача Django, який має повний доступ до адміністративної панелі.

- shell: Запускає інтерактивну оболонку Python із попередньо завантаженим контекстом проекту Django.

- test: Запускає тести для проекту Django та виводить результати.

- startapp: Створює нову програму Django, включаючи необхідну структуру файлів та директорій.

Це лише кілька прикладів команд, доступних за допомогою manage.py. Крім того є можливість створювати власні команди, розширюючи функціональність Django, і запускати їх через manage.py.

settings.py - це модульний конфігураційний файл у проекті Django, який містить настройки для всього проекту. Цей файл визначає параметри, такі як база даних, статичні файли, маршрутизація URL, автентифікація, міжсайтовий захист від підробки запитів (CSRF), локалізація та інші параметри, що впливають на поведінку проекту.

Ключові налаштування, які можна знайти у файлі settings.py:

- Debug: Булеве значення, яке вказує, чи активовано режим налагодження. Коли DEBUG встановлений у True, Django буде виводити докладні повідомлення налагодження при виникненні помилок.

- Allowed_Hosts: Список доменних імен або IP-адрес, які можуть опрацьовувати запити в режимі розгортання. Це необхідно для забезпечення безпеки та запобігання атакам типу "відмова в обслуговуванні".

- Installed_Apps: Список інстальованих програм Django. Тут перераховуються всі програми, які мають бути включені до проекту. Кожна програма представлена рядком у форматі 'app_name'.

- Databases: Словник, що визначає налаштування бази даних. Тут ви вказуєте тип бази даних, ім'я користувача, пароль, хост та інші подробиці, необхідні підключення до бази даних.

- Static_Url та Static_Root: Static_Url визначає Url-шлях для статичних файлів, таких як CSS, JavaScript та зображення. Static_Root вказує на директорію, в якій зберігаються статичні файли, зібрані при виконанні команди collectstatic.

- Media_Url та Media_Root: Media_Url визначає Url-шлях для завантажених користувачем медіафайлів, таких як зображення або відео. Media_Root вказує на директорію, де зберігаються ці медіафайли.

Це лише деякі з основних параметрів, які можна знайти у файлі settings.py. Файл також може містити інші налаштування, які можуть бути специфічними для кожного проекту або сторонніх пакетів.

Файл urls.py є одним із ключових файлів у фреймворку Django, який використовується для визначення маршрутів (URL-шаблонів) веб-програми. У цьому файлі визначаються зв'язки між URL-адресами та уявленнями (views), які обробляють запити від користувачів.

У urls.py визначено такі основні елементи:

- urlpatterns: Це змінна, яка містить список всіх URL-шаблонів вашої програми. Кожен URL-шаблон описується за допомогою функції path() або re_path() і складається з двох основних аргументів: шаблону URL-адреси (рядок) та подання (view) або функції, яка буде викликана при збігу URL.

- path() або re_path(): Це функції, які використовуються для визначення URL-шаблонів. path() призначена для простих URL-шаблонів, які не використовують регулярні вирази, а re_path() використовується, коли вам потрібно використовувати регулярні вирази для більш гнучкого зіставлення URL-адрес.

- views: Модуль або файл, який містить уявлення (views) для обробки запитів. Подання може бути функцією або класом, який приймає запит та повертає відповідь.

- name: Атрибут, який присвоюється кожному URL-шаблону та використовується для ідентифікації URL-адрес у коді. Він корисний, коли потрібно створити посилання на певні сторінки в програмі, і замість жорсткого кодування URL-адрес ви можете використовувати їх імена.

Основні елементи, які зазвичай трапляються у файлі urls.py. Він відіграє важливу роль у визначенні структури та маршрутизації веб-додатку Django.

Файл wsgi.py є частиною веб-застосунків, заснованих на Python, і використовується для запуску програми за допомогою WSGI (Web Server Gateway Interface) серверів. WSGI є стандартом інтерфейсу між веб-сервером та веб-додатком на мові Python.

Зазвичай файл wsgi.py містить кілька рядків коду, які налаштовують та запускають WSGI-сумісну програму.

Файл wsgi.py в основному служить для ініціалізації та запуску вашого WSGI-програми, і його вміст може змінюватись в залежності від вимог вашого проекту.

3.2 Проектування бази даних

Тепер необхідно розробити базу даних, та зробити міграції в неї. Це створить стандартні таблиці які використовує Django.

Автоматично будуть сформовані такі таблиці:

- auth_user - це таблиця містить інформацію про зареєстрованих користувачів
- django_session - ця таблиця зберігає сеансові дані користувачів
- django_migrations - в цій таблиці зберігаються дані про всі виконані міграції
- django_admin_log - тут зберігаются всі зміні які буде зроблені адміністратором в адмін-панелі
- django_content_type - ця таблиця містить інформацію про типи моделей в проекті

Виконано проектування бази даних, яка буде містити всі таблиці та поля які нам допоможуть під час в розробки функціоналу.

В проекті є три таблиці category, product та shippingaddress для збереження всієї необхідної інформації щодо інтернет магазину.

Проведено аналіз змісту кожної окремої таблиці та відмінностей між ними. Перша таблиця - shippingaddress має зберігати інформацію про замовлення покупця:

- first_name: ім’я покупця;
- last_name: прізвище покупця;
- email: електронна адреса покупця;
- address: адреса покупця;
- city: місто покупця;
- products: обраний товар покупцем;
- zipcode: індекс користувача;
- user: обліковий запис користувача.

Таблиця category буде мати інформацію щодо категорій які розміщені на нашому магазині:

- name: назва категорії;
- slug: індекс товару.

Таблиця product містить інформацію для заповнення щодо товару:

- category: категорія;
- title: назва товару;
- brand: назва бренду товару;
- description: опис товару;
- slug: індекс товару;
- price: ціна товару;
- image: рисунок товару.

Це три основні таблиці необхідні для збереження даних. Ці таблиці в коді мають інший вигляд (рис. 3.1 та 3.2):

```
8   class ShippingAddress(models.Model):
9
10      first_name = models.CharField(max_length=64)
11
12      last_name = models.CharField(max_length=64)
13
14      email = models.EmailField(max_length=256)
15
16      address = models.CharField(max_length=256)
17
18      city = models.CharField(max_length=255)
19
20      products = models.ManyToManyField(Product)
21
22      zipcode = models.CharField(max_length=255, null=True, blank=True)
23
24      # Authenticated
25
26      user = models.ForeignKey(User, on_delete=models.CASCADE, null=True, blank=True)
27
28      class Meta:
29
30          verbose_name_plural = 'Адреса для відправлення'
31
32      def __str__(self):
33
34          return 'Торгова адреса - ' + str(self.id)
35
36
```

Рисунок 3.1 - Модель в додатку payment

```python
class Category(models.Model):
    name = models.CharField(max_length=250, db_index=True)
    slug = models.SlugField(max_length=250, unique=True)

    class Meta:
        verbose_name_plural = 'Категорії'

    def __str__(self):
        return self.name

    def get_absolute_url(self):

        return reverse('list-category', args=[self.slug])


class Product(models.Model):

    category = models.ForeignKey(Category, related_name='product', on_delete=models.CASCADE, null=True)

    title = models.CharField(max_length=250)

    brand = models.CharField(max_length=250, default='un-branded')

    description = models.TextField(blank=True)

    slug = models.SlugField(max_length=255)

    price = models.DecimalField(max_digits=4, decimal_places=2)

    image = models.ImageField(upload_to='images/')

    class Meta:
        verbose_name_plural = 'Продукти'

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse('product-info', args=[self.slug])
```

Рисунок 3.2 - Модель в додатку store

Після створення моделей в проекті, потрібно виконати створення міграції командою “python manage.py makemigrations” далі застосовуємо міграції “python manage.py migrate”. Після цих команд в базу даних зробились міграції, це означає, що можна вже розглядати таблиці які було створено.

Кінцевий вигляд бази даних буде мати такий вигляд (рис. 3.3):

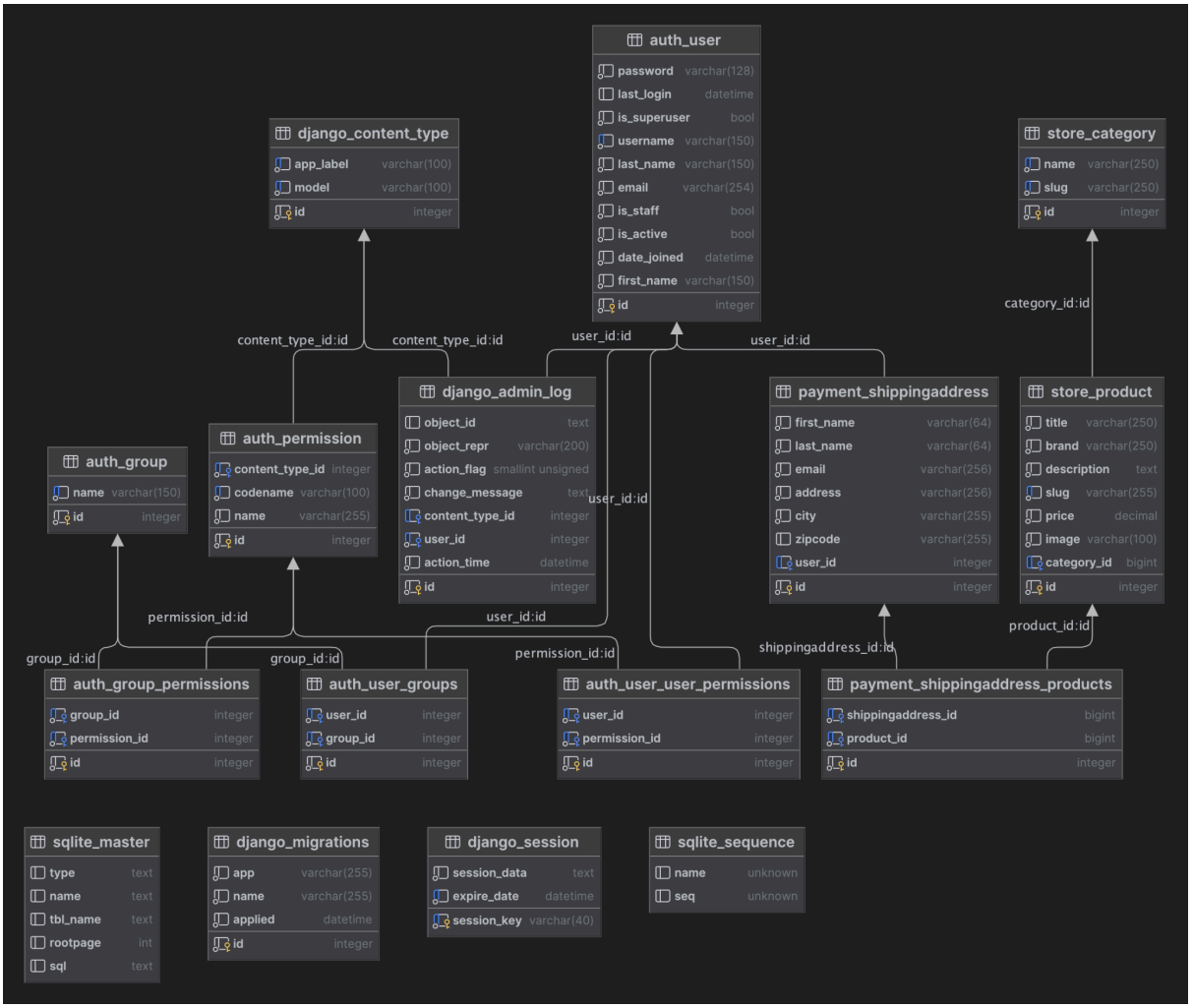


Рисунок 3.3 - Діаграма бази даних

3.3 Реєстрація та авторизація користувачів

Реєстрація – це процес, під час якого користувачі створюють облікові записи або профілі на веб-сайтах або програмах, щоб отримати доступ до додаткових функцій, ресурсів або персоналізованих послуг [14]. Він дозволяє користувачам ідентифікувати себе та встановлювати унікальні облікові дані для подальшої автентифікації.

Веб-сайти та програми можуть надавати різні методи реєстрації, але зазвичай процес включає такі кроки:

- Форма реєстрації: Користувач заповнює форму реєстрації, в якій може знадобитися вказати таку інформацію, як ім'я користувача, адресу електронної

пошти та пароль. Додаткові поля можуть бути пов'язані з вимогами конкретного веб-сайту або програми.

- Перевірка даних: Веб-сайт або програма перевіряє введені дані, щоб переконатися в їх правильності та відповідності певним правилам. Наприклад, може бути перевірка унікальності імені користувача чи правильності формату адреси електронної пошти.

- Створення облікового запису: Якщо дані пройшли перевірку, створюється обліковий запис користувача. Зазвичай, інформація про користувача зберігається в базі даних, і користувачеві присвоюється унікальний ідентифікатор.

- Підтвердження електронною поштою (опціонально): У деяких випадках користувачеві може бути надіслано листа з підтвердженням на вказану адресу електронної пошти. Користувач повинен перейти за посиланням для підтвердження своєї реєстрації.

- Опціональні кроки: У процесі реєстрації можуть бути додані додаткові кроки, такі як налаштування профілю користувача, завантаження фотографії, вибір конфіденційності та інші настройки, специфічні для конкретного веб-сайту або програми.

Після завершення реєстрації користувач може увійти до системи, використовуючи свої облікові дані, які будуть автентифікуватися веб-сайтом або програмою для надання доступу до персоналізованих функцій та ресурсів.

Авторизація - це процес перевірки та підтвердження ідентифікації користувача або суб'єкта перед наданням доступу до обмежених ресурсів, систем або інформації [14]. У багатьох веб-сервісах, програмах та інших системах авторизація використовується для забезпечення безпеки і контролю доступу до конфіденційної інформації.

Процес авторизації зазвичай включає в себе наступні кроки:

- Ідентифікація: користувач надає свої ідентифікаційні дані, такі як ім'я користувача або електронна пошта, для ідентифікації себе перед системою.

- Перевірка: система перевіряє надані ідентифікаційні дані, зазвичай з допомогою пароля або іншого типу аутентифікаційної інформації, яку користувач повинен ввести.

- Авторизація: якщо ідентифікація успішна і користувач підтверджує свою власність, система надає йому певні привілеї або доступ до обмежених ресурсів. Це може включати доступ до веб-сторінок, функцій програм або конфіденційних даних.

Авторизація є важливою складовою безпеки і захисту даних. Вона допомагає убезпечити систему від несанкціонованого доступу і забезпечити, що лише правомірні користувачі отримують доступ до конфіденційної інформації.

Додавши обробку запиту на реєстрацію маємо такий вигляд (рис. 3.4):

```
def register(request):

    form = CreateUserForm()

    if request.method == 'POST':
        form = CreateUserForm(request.POST)
        if form.is_valid():
            user = form.save()
            user.is_active = False
            user.save()
            # Email verification setup (template)
            current_site = get_current_site(request)
            subject = 'Електронна пошта для підтвердження облікового запису'
            message = render_to_string('account/registration/email-verification.html', {
                'user': user,
                'domain': current_site.domain,
                'uid': urlsafe_base64_encode(force_bytes(user.pk)),
                'token': user_tokenizer_generate.make_token(user),
            })
            user.email_user(subject=subject, message=message)
            return redirect('email-verification-sent')
    context = {'form': form}
    return render(request, 'account/registration/register.html', context=context)
```

Рисунок 3.4 - Функція реєстрації в додатку account

Дана функція працює при авторизації таким чином:

- Користувач вказує свої параметри та надсилає їх на перевірку системі;

- Якщо користувач вказав всі поля коректно то запит автоматично виконується;

- Якщо користувач вкаже невірно - система видасть повідомлення про помилку та вкаже де саме була допущена помилка.

При реєстрації процес схожий але містить більше своїх особливостей:

- Користувач вводить всі свої параметри які аналогічно авторизації перевіряються системою;

- У випадку якщо людина вказала всі параметри вірно, то їх на пошту приходить смс повідомлення з підтвердженням реєстрації на сайті.

- Після підтвердження людина автоматично отримує можливість входити до інтернет магазину під своїм приватним обліковим записом та виконувати всі покупки вже не як гість, а авторизований користувач.

```
def email_verification(request, uidb64, token):
    unique_id = force_str(urlsafe_base64_decode(uidb64))
    user = User.objects.get(pk=unique_id)
    # Success
    if user and user_tokenizer_generate.check_token(user, token):
        user.is_active = True
        user.save()
        return redirect('email-verification-success')
    # Failed
    else:
        return redirect('email-verification-failed')


def email_verification_sent(request):
    return render(request, 'account/registration/email-verification-sent.html')


def email_verification_success(request):
    return render(request, 'account/registration/email-verification-success.html')


def email_verification_failed(request):
    return render(request, 'account/registration/email-verification-failed.html')
```

Рисунок 3.5 - Детальний вигляд функцій email підтвердження

Додатково для задовільної роботи магазину необхідно мати просту та зрозумілу сторінку входу та виходу на сайті (рис 3.6):

```python
def my_login(request):
    form = LoginForm()
    if request.method == 'POST':
        form = LoginForm(request, data=request.POST)
        if form.is_valid():
            username = request.POST.get('username')
            password = request.POST.get('password')
            user = authenticate(request, username=username, password=password)
            if user is not None:
                auth.login(request, user)
                return redirect("dashboard")
    context = {'form':form}
    return render(request, 'account/my-login.html', context=context)


def user_logout(request):
    auth.logout(request)
    return redirect("store")
```

Рисунок 3.6 - Детальний вигляд функцій login та logout

За потреби необхідно мати видалення облікового запису, адже людина може випадково зареєструватися не під своїми даними або допустити орфографічну помилку. Для цього створюємо функцію delete_account (рис. 3.7):

```python
@login_required(login_url='my-login')
def delete_account(request):

    user = User.objects.get(id=request.user.id)

    if request.method == 'POST':

        user.delete()

        return redirect('store')



    return render(request, 'account/delete-account.html')
```

Рисунок 3.7 - Детальний вигляд функції delete_account

Кожен користувач має право налаштовувати свій обліковий запис так як йому це подобається. Саме для цього додано до інтернет магазину можливість редагування основного свого профілю за допомогою функції profile_management (рис. 3.8):

```
@login_required(login_url='my-login')
def profile_management(request):
    if request.method == 'POST':
        user_form = UpdateUserForm(request.POST, instance=request.user)
        if user_form.is_valid():
            user_form.save()
            return redirect('dashboard')
    user_form = UpdateUserForm(instance=request.user)
    context = {'user_form':user_form}
    return render(request, 'account/profile-management.html', context=context)
```

Рисунок 3.8 - Детальний вигляд функції profile_management

3.4 Адмін-панель

Адмін-панель у Django - це вбудований інструмент, який надає інтерфейс для адміністрування та керування даними веб-додатком, побудованим на фреймворку Django [15]. Це потужний інструмент, який дозволяє адміністраторам здійснювати різні операції з базою даних, моделями, користувачами та іншими аспектами додатка.

Основні функції адмін-панелі Django включають:

- Автогенерація інтерфейсу: Адмін-панель автоматично генерує інтерфейс для кожної моделі в додатку. Це означає, що ви можете переглядати, створювати, редагувати та видаляти записи бази даних безпосередньо з адмін-панелі, без необхідності власноручно писати код для створення форм і видів.
- Конфігурація моделей: Ви можете визначати, які поля моделі відображатимуться в адмін-панелі, як їхні значення валідуються та які дії доступні для виконання над записами.

- Пошук і фільтрація: Адмін-панель надає можливість шукати та фільтрувати записи за певними критеріями, що спрощує навігацію та редагування даних.
- Аутентифікація та авторизація: Адмін-панель використовує вбудовану систему аутентифікації та авторизації Django, що дозволяє обмежити доступ до адмін-панелі тільки для адміністраторів та інших користувачів з відповідними правами.
- Розширення та налаштування: Ви можете розширювати функціональність адмін-панелі шляхом написання власних додаткових модулів, переозначення стандартних шаблонів або налаштування параметрів адмін-панелі.

Загалом, адмін-панель Django робить процес адміністрування та керування даними веб-додатком швидким і зручним, забезпечуючи потужні інструменти для роботи з базою даних та моделями.

В адмін панелі є можливість виконувати додавання товарів на головну сторінку магазину. Розміщення їх ціни, опису, рисунку. Також в адмін панелі є можливість коригуванню облікових записів існуючих, налаштування категорій та перегляд активних замовлень.

3.5 Додавання каталогу товарів та фільтр

Зайшовши на сторінку інтернет магазину покупець повинен бачити доступний товар, щоб обрати необхідний та придбати.

```
def store(request):
    all_products = Product.objects.all()

    context = {'my_products':all_products}

    return render(request, 'store/store.html', context)
```

Рисунок 3.9 - Детальний вигляд функції store

Таким чином зайшовши на сайт, користувач побачить всі доступні товари інтернет магазину.

Також для більш зручного сортування необхідних товарів можна додати функцію categories (рис. 3.10), яка розділяє товари між собою враховуючи їх категорію. Тобто обравши необхідну категорія користувач буде бачити лише товари обраної ним категорії.

```
def categories(request):
    all_categories = Category.objects.all()

    return {'all_categories': all_categories}
```

Рисунок 3.10 - Детальний вигляд функції categories

Щоб функція categories (рис. 3.10) працювала необхідно пов’язати її додатково з функцією list_categories (рис. 3.11). Ця функція дозволяє користувачеві вивести на екран список всіх доступних категорій та обрати для нього свою необхідну.

```
def list_category(request, category_slug=None):

    category = get_object_or_404(Category, slug=category_slug)

    products = Product.objects.filter(category=category)

    return render(request, 'store/list-category.html', {'category': category, 'products': products})
```

Рисунок 3.11 - Детальний вигляд функції list_categories

Щоб користувач мав можливість проглянути інформацію яка вказана біля товару (ціна, опис, рисунок та індекс) необхідно додати функцію product_info (рис. 3.12). Цей перегляд дає можливість користувачу легко порівняти товари за цікавими для нього критеріями.

```
def product_info(request, product_slug):
    product = get_object_or_404(Product, slug=product_slug)

    context = {'product': product}

    return render(request, 'store/product-info.html', context)
```

Рисунок 3.12 - Детальний вигляд функції product_info

Всі ці функції також дають можливість адміністратору магазину додавати нову інформацію в кожне необхідне поле. Зайшовши на адмін панель можна відредагувати вже існуючі товари, додати нові категорії відборів і товарів.

3.6 Оформлення замовлення через кошик

Для оформлення замовлення необхідно додати свої функції оформлення замовлення. Після того коли людина обрала необхідний товар їй необхідно виконати замовлення товару. Для цього додаємо функцію manage_shipping (рис. 3.13). Ця функція направляє користувача на форму оформлення замовлення.

```
def manage_shipping(request):

    form = ShippingForm()

    if request.method == 'POST':

        form = ShippingForm(request.POST)

        if form.is_valid():

            shipping_user = form.save()

            shipping_user.user = request.user

            shipping_user.save()

            return redirect('dashboard')
```

Рисунок 3.13 - детальний вигляд функції manage_shipping

Наступна функція complete_order (рис. 3.14) буде формою оформлення замовлення де користувач буде вказувати всю детальну інформацію замовлення та особисті дані.

```
def complete_order(request):

    if request.POST.get('action') == 'post':

        name = request.POST.get('name')
        name1 = request.POST.get('name1')
        email = request.POST.get('email')
        address = request.POST.get('address')
        city = request.POST.get('city')
        zipcode = request.POST.get('zipcode')
        products = request.POST.get('products')
        if request.user.is_authenticated:

            order = ShippingAddress(first_name=name, last_name=name1, email=email, address=address, city=city, zipcode=zipcode,
                                    user=request.user)
            order.save()

            return JsonResponse({'success': True})

    return render(request, 'complete-order.html')
```

Рисунок 3.14 - Детальний вигляд функції complete_order

Після оформлення замовлення воно потрапляє в базу даних. Адміністратор сайту має можливість відкрити на адмін панелі оформлені замовлення та виконувати їх. Також в цій же адмін панелі адміністратор має можливість змінювати замовлення у випадку коректив від замовника, або взагалі видаляти.

РОЗДІЛ 4 ВЕБ-ДИЗАЙН ТА ІНТЕРФЕЙС ДЛЯ КОРИСТУВАЧА

4.1 Створення головної сторінки з товарами

Для створення головної сторінки необхідно створити файл формату html в якому буде міститися код програми який відображатиме нашу головну сторінку та гіпер посилання на інші сторінки.

```
{% load static %}
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8"/>
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <!-- CSS -->
        <link rel="stylesheet" href="https://bootswatch.com/5/flatly/bootstrap.min.css">
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
        <link rel="stylesheet" href="{% static 'css/styles.css' %}">
        <!-- JavaScript -->
        <script src="https://code.jquery.com/jquery-3.5.1.min.js" integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0=" crossorigin="anonymous"></script>
        <!-- JavaScript Bundle with Popper -->
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-0ERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJua0e923+m
     <!-- Bootstrap Max CDN -->
      <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    </head>
    <title> Django </title>
    <nav class="navbar navbar-expand-lg bg-black justify-content-center">
        <a class="navbar-brand main-heading text-white" href="{% url 'store' %}">
            Django
        </a>
        <button
            class="navbar-toggler"
            type="button"
            data-toggle="collapse"
            data-target="#navbarNavDropdown"
            aria-controls="navbarNavDropdown"
            aria-expanded="false"
            aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarCollapse">
            <ul class="navbar-nav me-auto mb-2 mb-md-0">
                <li class="nav-item dropdown">
                    <a class="button  dropdown-toggle text-white" href="#"  id=" navbarDropdown" role="button"
                    data-bs-toggle="dropdown" aria-expanded="false">
                        Категорії
                    </a>
                    <ul class="dropdown-menu " aria-labelledby="navbarDropdown">
```

Рисунок 4.1 - Код до головної сторінки №1

```
base.html
                    <li>
                        <a class="dropdown-item" href="{% url 'store' %}"> Всі </a>
                    </li>
                    {% for category in all_categories %}
                    <li>
                        <a class="dropdown-item " href="{{ category.get_absolute_url }}"> {{ category.name | capfirst }} </a>
                    </li>
                    {% endfor %}
                </ul>
            </li>
        </ul>
    </div>
    <div class="collapse navbar-collapse text-center" id="navbarNavDropdown">
        <ul class="navbar-nav ms-auto">
            {% if user.is_authenticated %}
                <li class="nav-item">
                    <a class="button btn-alert navbar-btn text-white" type="button"  href="{% url 'dashboard' %}"> Мій кабінет </a>
                </li>
                {% else %}
                <li class="nav-item">
                    <a class="button btn-alert navbar-btn text-white" type="button"  href="{% url 'register' %}"> Зареєструватися </a>
                </li>
            {% endif %}
            {% if user.is_authenticated %}
                 <li class="nav-item">
                    <a class="button btn-alert navbar-btn text-white" type="button"  href="{% url 'user-logout' %}"> Вихід </a>
                </li>
                {% else %}
                <li class="nav-item">
                    <a class="button btn-alert navbar-btn text-white" type="button"  href="{% url 'my-login' %}"> Вхід </a>
                </li>
            {% endif %}
                 
            <li class="nav-item">
                <a type="button" role="button" href="{% url 'cart-summary'  %}" class="button btn-outline-secondary">
                    <i class="fa fa-shopping-cart" aria-hidden="true"> </i>

                    <div id="cart-gty" class="d-inline-flex">
                        {% with qty_amount=cart|length %}
                            {% if qty_amount > 0 %}
```

Рисунок 4.2 - Код до головної сторінки №2

```
                    <div id="cart-gty" class="d-inline-flex">
                        {% with qty_amount=cart|length %}
                            {% if qty_amount > 0 %}
                                {{ qty_amount }}
                                {% else %}
                                    0
                            {% endif %}
                        {% endwith %}
                    </div>
                </a>
            </li>

        </ul>
    </div>
</nav>
<body >
    {% block content %}
    {% endblock %}
<br><br><br>
</body>
</html>
```

Рисунок 4.3 - Завершення головної сторінки

Після виконання коду необхідно запустити розроблений сайт та перевірити коректність введених даних. Для запуску проекту необхідно прописати в терміналі:

- cd..: повернення з директорії в початкове положення;
- venv1\scripts\activate: активація віртуального середовища;
- cd ecommerce: перехід в папку ecommerce;
- python manage.py runserver: запуск серверу.

Після прописання необхідних команд, у вказаному вище порядку, у терміналі відкривається посилання на сам інтернет магазин (рис. 4.4).

```
PS C:\Users\User\Desktop\Проект норм\devv\ecommerce> cd..
PS C:\Users\User\Desktop\Проект норм\devv> venv1\Scripts\activate
(venv1) PS C:\Users\User\Desktop\Проект норм\devv> cd ecommerce
(venv1) PS C:\Users\User\Desktop\Проект норм\devv\ecommerce> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
June 11, 2023 - 21:42:05
Django version 4.2.1, using settings 'ecommerce.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Рисунок 4.4 - Посилання на інтернет магазин

Переходимо по посиланню після чого в браузері який встановлений на нашому комп'ютері за замовчуванням відкривається головне вікно магазину де знаходиться каталог товарі з фільтром по категоріям, можливість авторизації або реєстрації на сайті та наш кошик з обраним товаром.

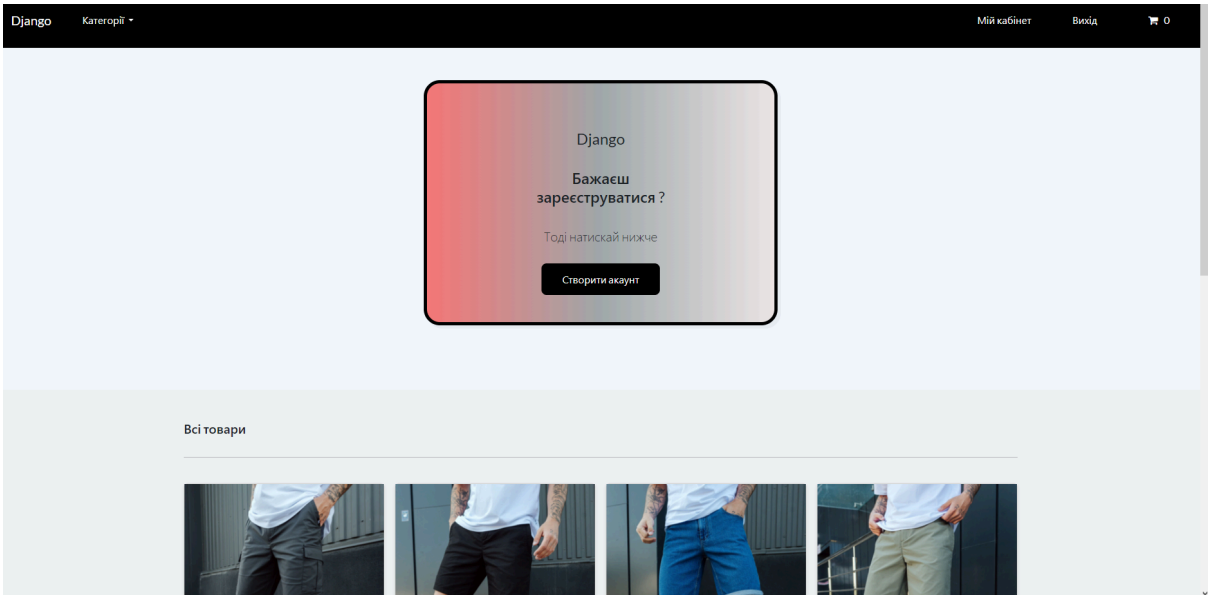


Рисунок 4.5 - Головна сторінка інтернет магазину

4.2 Створення вікна реєстрації та авторизації

Для створення вікна реєстрації необхідно створити новий файл формату html якому ми надамо назву register. Після створення файлу необхідно прописати параметри які ми вважаємо за потрібне для візуального відображення сторінки.

```
register.html
{% include "store/base.html" %}
{% load static %}
{% load i18n %}
{% load crispy_forms_tags %}
{% block content %}
    <style>
        body{
            background-color: gray;
        }
        #button{
            background-color: black;
            border-radius: 8px;
            border: none;
            color: white;
            padding: 15px 32px;
            text-align: center;
            text-decoration: none;
            display: inline-block;
            font-size: 16px;
            -webkit-transition-duration: 0.4s; /* Safari */
            transition-duration: 0.4s;
        }
        #button:hover{
            background-color: red; /* Green */
            color: white;
        }
    </style>
    <body>
        <br> <br>
        <div class="container bg-white shadow-md p-5 form-layout">
            <h4 class="text-center"> Створення облікового запису </h4>
            <br>
            <br>
            <form method="POST" autocomplete="off">
                {% csrf_token %}
                {{form.username|as_crispy_field}}
                <br> <br>
                {{form.email|as_crispy_field}}
                <br> <br>
style
```

Рисунок 4.6 - Перша частина коду від сторінки реєстрації

```
    </style>
    <body>
        <br> <br>
        <div class="container bg-white shadow-md p-5 form-layout">
            <h4 class="text-center"> Створення облікового запису </h4>
            <br>
            <br>
            <form method="POST" autocomplete="off">
                {% csrf_token %}
                {{form.username|as_crispy_field}}
                <br> <br>
                {{form.email|as_crispy_field}}
                <br> <br>
                {{form.password1|as_crispy_field}}
                <br> <br>
                {{form.password2|as_crispy_field}}
                <br> <br>
                <button id="button"  class="btn-lg w-100 btn-block p-2" >  Створити обліковий запис </button>
            </form>
        </div>
    </body>
    <br> <br>
{% endblock %}
```

Рисунок 4.7 - Друга частина коду від сторінки реєстрації

Далі на головній сторінці розробленого сайту натискаємо в правому верхньому куті на “зареєструватись”, після чого переходимо на поле реєстрації (рис. 4.8).

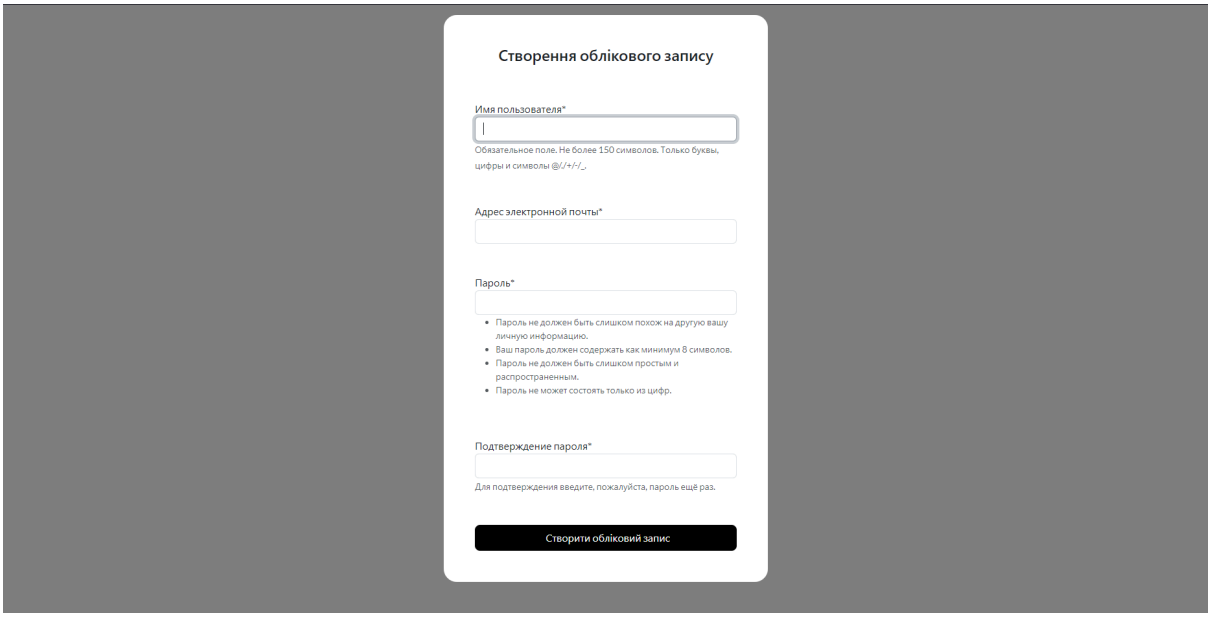


Рисунок 4.8 - Зовнішній вигляд сторінки реєстрації

На цій сторінці користувач вказує свої параметри щоб потім мати змогу заходити на сайт під своїм обліковим записом. При введені паролю є обмеження, яких необхідно дотримуватися. Після того коли користувач вказав всі свої необхідні дані на його пошту яку він вказав у вікні реєстрації надійде лист підтвердження (рис 4.9) в якому потрібно підтвердити свою реєстрацію перейшовши за посиланням.

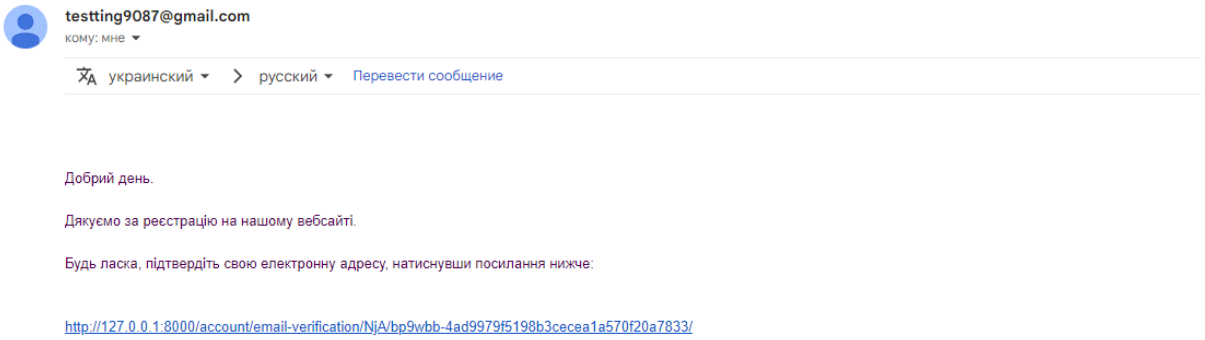


Рисунок 4.9 - Лист підтвердження реєстрації

Після підтвердження адміністратор системи побачить на адмін панелі користувачів новий обліковий запис якому може надати новий статус чи змінити його права.

Для авторизації користувача також необхідно створити новий файл формату html (рис. 4.10).

```
{% include "store/base.html" %}

{% load static %}

{% load crispy_forms_tags %}


{% block content %}

    <style>

        body{

            background-color: gray;
        }

        #button{
            background-color: black;
            border-radius: 8px;
            border: none;
            color: white;
            padding: 15px 32px;
            text-align: center;
            text-decoration: none;
            display: inline-block;
            font-size: 16px;
            -webkit-transition-duration: 0.4s; /* Safari */
            transition-duration: 0.4s;
        }

        #button:hover{
            background-color: red; /* Green */
            color: white;
        }


    </style>

```

Рисунок 4.10 - Код сторінки авторизації облікового запису

Якщо користувач вже має обліковий запис, йому необхідно на головній сторінці інтернет магазину в правому верхньому куті натиснути на “вхід”, після чого відкриється вікно авторизації (рис. 4.11) де користувач вказує свій пароль та ім’я користувача після чого він автоматично авторизується до системи.

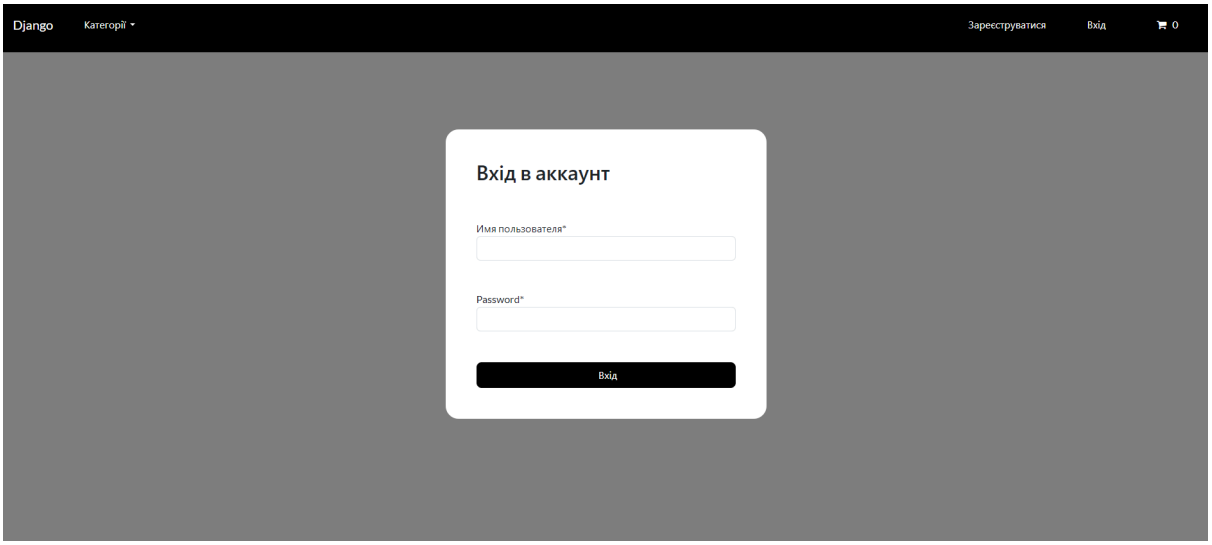


Рисунок 4.11 - Зовнішній вигляд сторінки авторизації

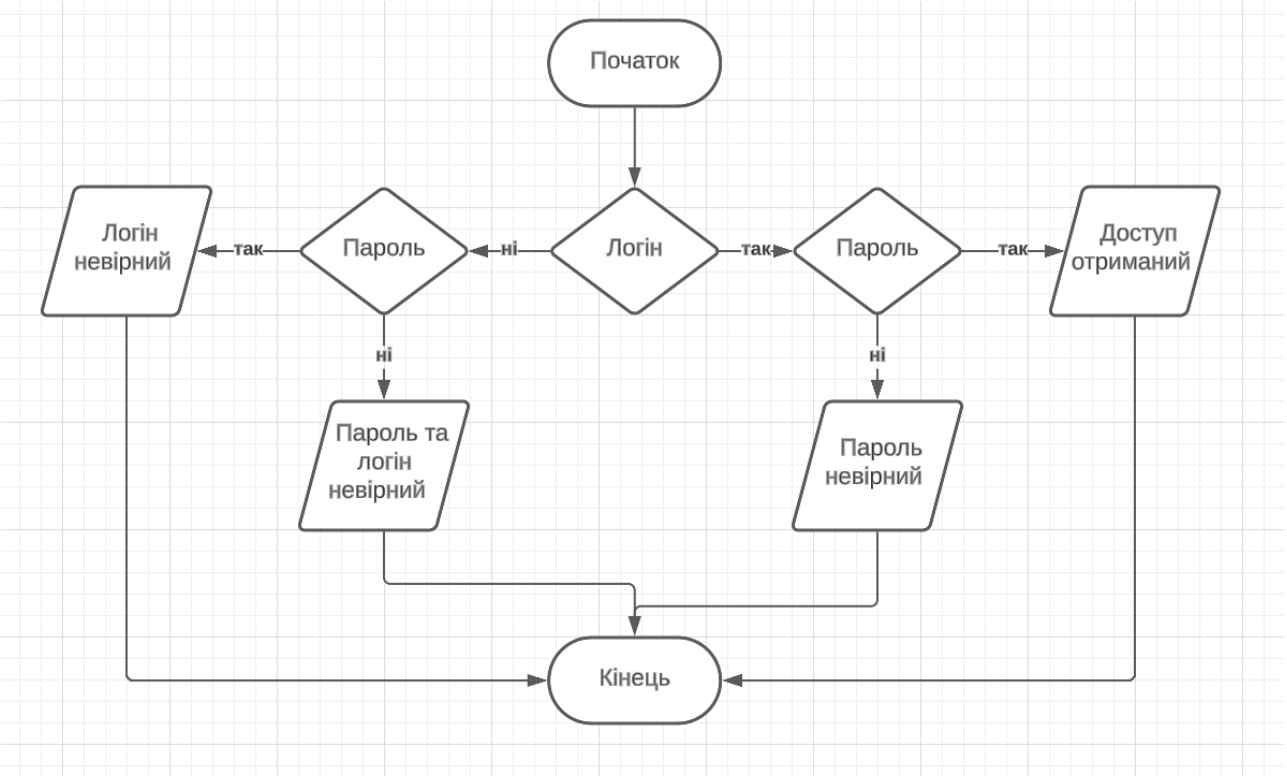


Рисунок 4.12 – Алгоритм роботи реєстрації та авторизації

4.3 Оформлення замовлення

Щоб покупець мав змогу оформити замовлення необхідно створити окремий файл формату html, в якому ми додамо необхідні поля заповнення для формування замовлення покупцем (рис. 4.13 та 4.14).

```
register.html   email-verification.html   checkout.html   manage-shipping.html

{% include "store/base.html" %}
{% load static %}
{% block content %}
<style>
    body
    {
        background-color: gray;
    }
</style>
    <body>
        <br>
        <div class="container bg-white shadow-md p-5" style="...">
            <form id="form">
                <div>
                    <h3> <i class="fa fa-chevron-circle-right" aria-hidden="true"></i>  Останній крок. </h3>
                    <p> Будь ласка, введіть відповідну інформацію нижче: </p>
                    <hr>
                    <br>
                    <div class="form-field">
                        <input class="form-control validate" id="name" type="text" placeholder="Ім'я*" autocomplete="off" value="{{ shipping.first_name }}" required>
                    </div>
                    <br>
                     <div class="form-field">
                        <input class="form-control validate" id="name1" type="text" placeholder="Прізвище*" autocomplete="off" value="{{ shipping.last_name }}" required>
                    </div>
                    <br>
                    <div class="form-field">
                        <input class="form-control validate" id="email" type="email" placeholder="Email адреса*" autocomplete="off" value="{{ shipping.email }}" required
                    </div>
                    <br>
                    <div class="form-field">
                        <input class="form-control validate" id="address" type="text" placeholder="Адреса*" autocomplete="off" value="{{ shipping.address }}" required>
                    </div>
                    <br>
                    <div class="form-field">
                        <input class="form-control validate" id="city" type="text" placeholder="Місто*" autocomplete="off" value="{{ shipping.city }}" required>
                    </div>
                    <br>
                    <div class="form-field">
```

Рисунок 4.13 - Перша частина коду сторінки оформлення замовлення

```
                    <div class="form-field">
                        <input class="form-control" id="zipcode" type="text" placeholder="Zip код (необов'язково)" autocomplete="off" value="{{ shipping.zipcode }}">
                    </div>
                </div>
                <br>
                <br>
               <button id="complete-order" type="submit" class="btn btn-primary center  text-white">  Залишити замовлення </button>
            </form>
        </div>
        <br>
    </body>
    <script>
        $(document).on('submit', function(e){
            e.preventDefault();
            $.ajax({
                type: 'POST',
                url: '{% url "complete-order" %}',
                data: {
                    name: $('#name').val(),
                    name1: $('#name1').val(),
                    email: $('#email').val(),
                    address: $('#address').val(),
                    city: $('#city').val(),
                    zipcode: $('#zipcode').val(),
                    csrfmiddlewaretoken: "{{ csrf_token }}",
                    action: 'post'
                },
                success: function(json){
                    // console.log(json)
                    window.location.replace("{% url 'payment-success' %}");
                },
                error: function(xhr, errmsg, err){
                    window.location.replace("{% url 'payment-failed' %}");
                }
            });
        })
    </script>
{% endblock %}
```

Рисунок 4.14 - Друга частина коду сторінки оформлення замовлення

Таким чином після виконання цього файлу з’являється можливість оформлення замовлення. Користувач, обравши необхідний для нього товар може оформити замовлення перейшовши спочатку в свою корзину (правий врехній кут), а потім натиснувши нище “перейти до замовлення”. В корзині користувач має змогу перевірити своє замовлення перед тим як його оформлювати.

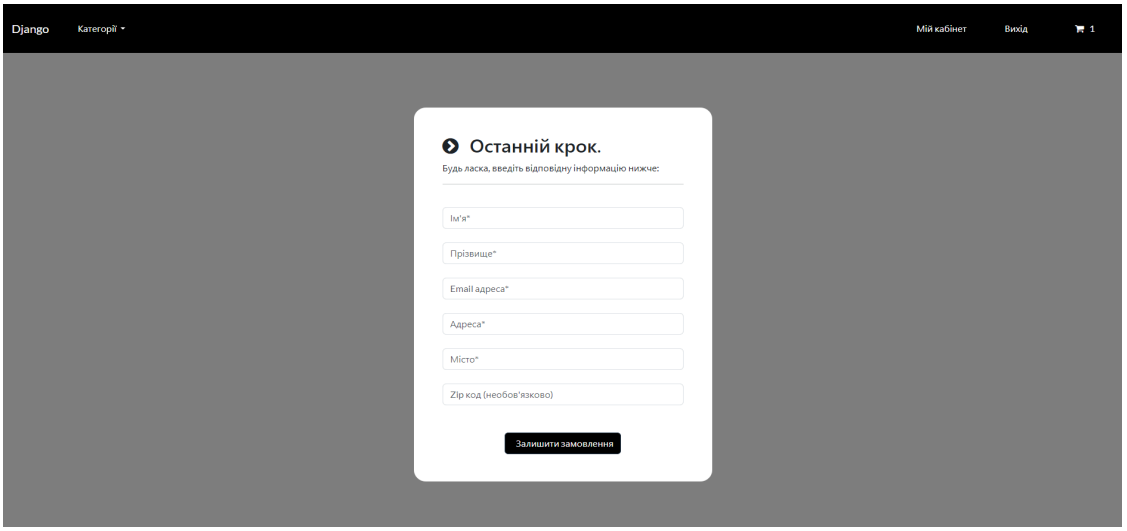


Рисунок 4.15 - Сторінка офомлення замовлення

Коли користувач оформив замовлення адміністратор може проглянути це замовлення у своїй адмін панелі. Також при необхідності адміністратор може відредагувати замовлення, видалити його або виконати.

4.4 Оформлення скролу товарів

Для виконання скролу товарів необхідно створити окремий файл skripts.js в якому встановлені необхідні налаштування скролу, а саме: напрямок скролу товарів, варіант скролу та властивість скролу.

Для додавання інтерактивності та покращення користувацького досвіду, була використана бібліотека Swiper.js. Swiper.js — це потужна та гнучка бібліотека для створення слайдерів, яка забезпечує плавну та інтерактивну взаємодію користувачів з контентом. Вона відзначається високою продуктивністю, підтримує адаптивний дизайн і жестове управління для мобільних пристроїв, що робить її ідеальною для сучасних веб-додатків. Крім того, Swiper.js легко налаштовується завдяки багатій системі API, а також сумісна з популярними фреймворками, такими як React, Vue та Angular. Її використання спрощує інтеграцію анімацій, автопрокрутки та інших візуальних ефектів, що значно покращує користувацький досвід.

```
document.addEventListener("DOMContentLoaded", (event) => {
  const swiper = new Swiper('.swiper', {
      // Optional parameters
      direction: 'horizontal',
      slidesPerView: 4,
      spaceBetween: 16,

      // If we need pagination
      pagination: {
        el: '.swiper-pagination',
        clickable: true,
      },

      // Navigation arrows
      navigation: {
        nextEl: '.swiper-button-next',
        prevEl: '.swiper-button-prev',
      },

      // And if we need scrollbar
      scrollbar: {
        el: '.swiper-scrollbar',
      },
  });
});
```

Рисунок 4.16 - Детальний опис виконання скролу

4.5 Створення розділу обране

Для створення розділу обране необхідно створити додати функції додавання та видалення визначеного товару до вкладення обране. Ця функція допомагає користувачу мати швидкий доступ до необхідного для нього товару.

```
<script>
    const addFavorite = (target, productId) => {
        $.ajax({
            type: 'POST',
            url: 'http://localhost:8000/favorite/',
            data: {
                product: productId,
                csrfmiddlewaretoken: "{{ csrf_token }}",
            },
            success: function(json) {
            },
            error: function(xhr, errmsg, err) {},
              complete: function() {
                target.attr('is-active', true);
                const filledIcon = target.find('.favorite-icon_filled');
                const solidIcon = target.find('.favorite-icon_solid');

                $(filledIcon).show();
                $(solidIcon).hide();
            }
        });
```

Рисунок 4.17 - Скрипт для додавання товарів у обране

Додавання та вилучення товарів в розділі обране відмічається натисканням на зображення серця в правому верхньому куті товару. Для більш зрозумілого візуального ефекту при додаванні товару в обране сердце стає червоного кольору.

Коректна робота перенесення товарів у розділ обране формується завдяки функціям FavoriteDeleteView, ProductsView та FavoriteListView (рис 3.18).

```
class FavoriteDeleteView(APIView):  1 usage
    def delete(self, request, pk):  2 usages (2 dynamic)
        try:
            favorite = Favorite.objects.get(product_id=pk)
            favorite.delete()
            return Response(status=status.HTTP_204_NO_CONTENT)
        except:
            return Response({'detail': 'Not Found'})

class ProductsView(generics.ListAPIView):  1 usage
    queryset = Product.objects.all()
    serializer_class = ProductsSerializers

class FavoriteListView(APIView):  1 usage
    def post(self, request, *args, **kwargs):
        products_ids = request.data.getlist('ids[]', [])
        products_ids = [int(num) for num in products_ids]
        data = Product.objects.filter(pk__in=products_ids).values('id', 'category', 'title', 'brand', 'description', 'slug', 'price', 'image')
        return Response(data)
```

Рисунок 4.18 - Детальний опис функцій FavoriteDeleteView, ProductsView та FavoriteListView

Розділ "Обране" на сайті з продажу одягу створювався з метою забезпечення зручного збереження та швидкого доступу до товарів, які сподобалися користувачам. Для цього було реалізовано функціонал інтерактивного додавання товарів у список обраного. Користувач може обрати товар із каталогу, натиснувши на іконку у вигляді серця. Ця дія автоматично виконує запит через бібліотеку Axios, що дозволяє додати товар до списку без оновлення сторінки. Після успішного виконання запиту іконка серця змінює колір на червоний, сигналізуючи про те, що товар успішно збережений. Завдяки використанню асинхронної технології цей процес є швидким та плавним, забезпечуючи приємний досвід для користувача.

Функція обраних товарів працює виключно у користувачів які авторизовані на сайті інтернет магазину.

4.6 Розділ відгуків

Розділ відгуків на сторінці товару було створено з використанням HTML, CSS та JavaScript у рамках фреймворку Django. У формі, розташованій на сторінці, було передбачено поле для введення тексту відгуку, випадаючий список для вибору оцінки від 1 до 5 та кнопку для відправки даних. JavaScript з використанням бібліотеки Axios забезпечував асинхронну відправку даних на сервер через POST-запит. Серверна частина, побудована на Django, приймала дані форми, створювала об'єкт моделі відгуку, зберігали його у базі даних та повертала у форматі JSON. Отриманий від сервера об'єкт відгуку використовувався для оновлення HTML без перезавантаження сторінки: новий відгук додавався до списку існуючих відгуків за допомогою методу insertAdjacentHTML. CSS відповідав за стилізацію форми та списку відгуків, надаючи їм сучасний і привабливий вигляд. Такий підхід забезпечив інтерактивність і зручність у використанні, дозволяючи користувачам миттєво бачити свої відгуки після їх додавання.

```
330     };
331
332     document.getElementById('add-review-button').addEventListener('click', () => {
333         const text = document.getElementById('new-review-input').value;
334         const stars = $('#review-stars option:selected').text();
335
336         $.ajax({
337             type: 'POST',
338             url: 'http://localhost:8000/feedback/',
339             data: {
340                 product,
341                 text,
342                 stars,
343                 user,
344                 csrfmiddlewaretoken: "{{ csrf_token }}",
345             },
346             success: function(json) {
347                 const html = renderReview(json);
348
349                 document.getElementById('product-reviews').insertAdjacentHTML('afterbegin', html);
350
351                 document.getElementById('new-review-input').value = '';
352             },
353             error: function(xhr, errmsg, err) {},
354         });
355     });
356
357 </script>
```

Рисунок 4.19 - Скрипт для додавання відгуку користувачам

```
const renderReview = json => {
    return `
        <li class="review-item">
            <div class="review-header">
                <div class="review-user">
                    <svg width="30px" height="30px" viewBox="0 0 16 16" fill="none" xmlns="http://www.w3.org/2000/svg"><path d="M8 7C9.65685 7 1
                </div>
                <div class="review-info">
                     <div class="review-top">
                        <p class="review-name">${json.username}</p>
                        <span class="review-date">${json.created}</span>
                    </div>
                    <ul class="review-score">
                        <li class=${json.stars >= 1 ? "active" : ""}>
                            <svg width="20px" height="20px" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg"><path d="M9.15316
                        </li>
                        <li class=${json.stars >= 2 ? "active" : ""}>
                            <svg width="20px" height="20px" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg"><path d="M9.15316
                        </li>
                          <li class=${json.stars >= 3 ? "active" : ""}>
                            <svg width="20px" height="20px" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg"><path d="M9.15316
                        </li>
                        <li class=${json.stars >= 4 ? "active" : ""}>
                            <svg width="20px" height="20px" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg"><path d="M9.15316
                        </li>
                        <li class=${json.stars >= 5 ? "active" : ""}>
                            <svg width="20px" height="20px" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg"><path d="M9.15316
                        </li>
                    </ul>
                </div>
            </div>
            <p class="review-text">${json.text}</p>
        </li>
    `;
};
```

Рисунок 4.20 - Скрипт відображення відгуку формату html

4.7 Створення endpoint

Endpoint (кінцева точка) – це конкретна адреса або місце в системі, де клієнт може взаємодіяти з сервером. Це може бути URL-адреса для веб-сервісів, метод у реляційній базі даних або навіть фізичний пристрій у мережі. По суті, endpoint – це "ворота", через які дані входять і виходять із системи.

Типи endpointів

Веб-сервіси:

REST API: Найпоширеніший тип, який використовує HTTP методи (GET, POST, PUT, DELETE) для виконання операцій над ресурсами.

GraphQL API: Альтернативний підхід, який дозволяє клієнту запитувати саме ті дані, які йому потрібні, зменшуючи обсяг переданих даних.

Бази даних:

SQL: Структуровані запити до реляційних баз даних.

NoSQL: Документоорієнтовані, графові або ключово-значення бази даних, які мають свої специфічні методи взаємодії.

Мікросервіси:

Кожен мікросервіс має свої власні endpoint, які можуть бути доступні через HTTP, gRPC або інші протоколи.

IoT пристрої:

Фізичні пристрої (сенсори, актуатори) мають свої endpoint для зчитування даних або керування ними.

Функціональність endpointів

Приймання даних: Endpoint отримує дані від клієнта (наприклад, дані форми, параметри запиту).

Обробка даних: Дані обробляються відповідно до логіки сервера (наприклад, валідація, зберігання в базі даних, виконання розрахунків).

Повернення результату: Endpoint повертає відповідь клієнту (наприклад, JSON, XML, HTML).

```
from django.urls import path
from rest_framework.routers import DefaultRouter

from . import views
from .views import FeedbackViewSet

router = DefaultRouter()
router.register( prefix: r'feedback', FeedbackViewSet, basename='feedback')

urlpatterns = [
    path('', views.store, name='store'),

    path('product/<slug:product_slug>/', views.product_info, name='product-info'),

    path('search/<slug:category_slug>/', views.list_category, name='list-category'),

    path('products/', views.ProductsView.as_view(), name='products-list'),
    path('favorite/', views.FavoriteListCreationView.as_view(), name='favorite-list-create'),
    path('favorite/<int:pk>/', views.FavoriteDeleteView.as_view(), name='favorite-delete'),
    path('favorite_list/', views.FavoriteListView.as_view(), name='favorite-list'),
    path('get_current_user/', views.GetCurrentUserListView.as_view(), name='getuser-list')
]
urlpatterns += router.urls
```

Рисунок 4.21 - Список endpoint

Обробка endpoint у Django базується на роботі з URL-адресами та функціями або класами, що реалізують логіку обробки запитів. Django використовує паттерн «MVC» (Model-View-Controller), який у рамках цього фреймворку інтерпретується як «MVT» (Model-View-Template). Endpoint обробляє HTTP-запити й повертає відповідь, зазвичай у вигляді HTML, JSON або іншого формату.

Основні компоненти обробки endpoint

1. Файл URLconf:

URLconf визначає маршрутизацію запитів, тобто пов’язує URL-адреси з відповідними функціями або класами. Це робиться за допомогою списків або шляхів, що описують шаблони URL.

2. View (Подання):

View реалізує логіку обробки запиту. Це може бути:

- Функція, яка приймає HTTP-запит і повертає відповідь.

- Клас-нащадок `View`, який підтримує різні методи для роботи з HTTP-методами (`GET`, `POST`, `PUT`, тощо).

3. Middleware (Проміжне програмне забезпечення):

Middleware обробляє запити й відповіді перед і після досягнення view. Вони можуть додавати заголовки, перевіряти автентифікацію або змінювати вхідні/вихідні дані.

4. HTTP-запити та методи:

Django підтримує різні HTTP-методи (наприклад, `GET`, `POST`, `PUT`, `DELETE`). У межах view можна обробляти дані запитів, виконувати бізнес-логіку та формувати відповідь.

5. Формування відповіді:

View повертає об’єкт відповіді, який може бути різних типів:

- Простий текст або HTML (`HttpResponse`).
- Дані у форматі JSON чи XML.
- Перенаправлення на інший URL.
- Помилки, наприклад, 404 або 403.

Валідація та обробка даних

Для обробки вхідних даних Django надає такі інструменти:

- Форми забезпечують валідацію та роботу з даними.
- Серіалізатори (у Django REST Framework): використовуються для перетворення даних між моделями й форматами, наприклад, JSON.
- Доступ до параметрів: Django дозволяє отримувати дані з URL, форми, JSON-запитів або заголовків HTTP.

Безпека та обмеження доступу

Django пропонує інструменти для підвищення безпеки endpoint:

- Механізми перевірки аутентифікації та авторизації.
- Захист від CSRF-атак за допомогою токенів.
- Ручна або автоматична обробка дозволів (permissions).

Логування та обробка помилок

Обробка endpoint може включати логування подій для відстеження роботи. Django підтримує систему обробки помилок із можливістю повернення спеціальних сторінок або повідомлень користувачам.

Загалом, обробка endpoint у Django гнучка та налаштовується під потреби конкретного проекту, що робить її ефективним інструментом для створення веб-додатків.

ВИСНОВКИ

У рамках дипломного проекту було проведено дослідження та розробка інтернет магазину з використанням мови програмування Python, фреймворку Django і бази даних SQLite. В ході аналізу існуючих інтернет магазинів були виявлені основні вимоги та функціональні можливості, які повинен мати сучасний інтернет магазин.

Проведено аналіз методів та засобів розробки інтернет-магазину. Під час вибору технологій для проекту були враховані такі фактори, як ефективність, швидкодія, простота розробки та підтримки, а також наявність великого співтовариства розробників.

У результаті проведеного аналізу методів та засобів розробки інтернет-магазину для реалізації проекту вибрано мову програмування Python, фреймворк Django та систему управління базами даних SQLite, завдяки чому забезпечено ефективну та швидку розробку, високу продуктивність, простоту підтримки та доступ до багатофункціональної екосистеми Python. Використання цих технологій дозволило створити стабільний та функціональний інтернет-магазин, який може задовольнити потреби користувачів.

Проведено проектування та розробку інтернет магазину. Створено архітектуру проекту, включаючи моделі даних, представлення та контролери. Реалізовано основні функціональні можливості, такі як додавання товарів до кошика, оформлення замовлення, реєстрація та аутентифікація користувачів. Забезпечено інтеграцію з платіжними шлюзами для оплати замовлень.

У результаті виконання дипломного проекту розроблено функціональний інтернет магазин з використанням мови програмування Python, фреймворку Django і бази даних SQLite. Розроблений інтернет магазин відповідає основним вимогам та функціональним можливостям, які були визначені на початку проекту. Застосування вищезазначених технологій та інструментів дозволило ефективно розробити та реалізувати інтернет магазин, що задовольняє потреби сучасних користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Приклад першого інтернет магазину. URL: https://book-ye.com.ua (дата звернення: 12.11.2024); [1]
2. Приклад другого інтернет магазину. URL: https://ostriv.ua (дата звернення: 13.11.2024); [2]
3. Сайт з інформацією про мову програмування. URL: https://naurok.com.ua (дата звернення : 10.11.2024); [3]
4. Офіційний сайт мови програмування Python. URL: https://www.python.org (дата звернення: 10.11.2024); [4]
5. Сайт з інформацією про мову програмування C#. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 10.11.2024); [5]
6. Сайт з інформацією про мову програмування PHP. URL: https://www.php.net/manual/en/intro-whatis.php (дата звернення: 11.11.2024); [6]
7. Сайт з інформацією про характеристики фреймворку. URL: https://wezom.com.ua/ua/blog/scho-take-freymvork-poyasnyujemo-prostimi-slovami (дата звернення: 11.11.2024); [7]
8. Сайт з інформацією про характеристику SQLite. URL: https://blog.ithillel.ua/ru/articles/introduction-to-sqlite-python (дата звернення: 11.11.2024); [8]
9. Сайт з інформацією про додаткові бібліотеки. URL: https://blog.ithillel.ua/articles/a-library-or-your-own-code (дата звернення: 11.11.2024); [9]
10. Сайт з інформацією про хостинг. URL: https://uk.wikipedia.org/wiki/Хостинг (дата звернення 14.11.2024); [10]
11. Сайт з інформацією про архітектуру проекту програмування. URL: https://www.kursak.com/arkhitektura-prohramnoho-zabezpechennia (дата звернення: 12.11.2024); [11]
12. Сайт з інформацією про віртуальне середовище. URL: https://docs.python.org/3/library/venv.html (дата звернення 14.11.2024); [12]
13. Сайт з інформацією пакетного менеджменту та його характеристик. URL: https://pip.pypa.io / (дата звернення: 14.11.2024); [13]
14. Сайт з інформацією про реєстрацією та авторизацією. URL: https://itvdn.com/ua/webinars/description/authorization-authentication-webapps (дата звернення 14.11.2024); [14]
15. Сайт з інформацією про адмін панелі та їх програмування. URL: https://hostiq.ua/blog/ukr/admin-panel/ (дата звернення: 15.11.2024); [15]
16. Сайт з інформацією щодо економіки. URL: https://studies.in.ua/bzakon (дата звернення 14.11.2024); [16]

17. Сайт з інформацією про капітальні вкладення. URL: https://pidru4niki.com/80399/ekonomika/kapitalni_vkladennya_yihnya_struktura (дата звернення 15.11.2024); [17]

18. Малишкін О. І. Навчальний посібник. Економіка виробництва. Видавництво «ТРОПЕА» - К.: 2022. – 128 с. [18]

19. Сайт з інформацією цін на електроенергію для промислових користувачів. URL: https://index.minfin.com.ua/ua/tariff/electric/prom/ (дата звернення 17.11.2024). [19]

20. Охорона праці. URL: https://osvita-docs.com/node/41 (дата звернення: 18.11.2024); [20]

21. Сайт з інформацією про ергономіку та її визначення. URL: https://termin.in.ua/erhonomika/ (дата звернення 20.11.2024); [21]

22. Сайт з інформацією про безпеку даних. URL: https://uk.theastrologypage.com/data-security (дата звернення 22.11.2024). [22]

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кваліфікаційна робота магістра
на тему «АРХІТЕКТУРА ІНТЕРНЕТ-МАГАЗИНУ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ КОМЕРЦІЙНИХ ПРОЦЕСІВ»

Виконав: студент групи КСДМ-62
Рябоконь О.О.
Керівник роботи:
phD (доктор філософії), доцент
Сергій КОРОТКОВ

Київ 2024

Мета, об’єкт та предмет дослідження

- **Об’єкт дослідження**. Процеси функціонування інтернет магазинів, аналітика роботи у відкритому доступі мережі інтернет.
- **Предмет дослідження**. Засоби розробки інтернет магазинів, набір інструментів розробки програм.
- **Метою роботи** є дослідження методів та засобів розробки та підтримки інтернет-магазинів, створення їх адміністрування, безпеки та функціональності, спираючись на яке розробити веб-сторінку інтернет магазину.

Аналіз предметної області

Веб-сайт	Мови програмування	Фреймворки	Сфери застосування
Amazon	Python, C++, Perl	Apache Struts	Роздрібна торгівля
eBay	Python	Apache Wicket	Електронна комерція
Alibaba	Python	Spring MVC	Міжнародна торгівля
Etsy	Python, PHP	Django, Laravel	Ручна робота, мистецтво
Shopify	Ruby	Ruby on Rails	Електронна комерція
Flipkart	Python	Play Framework	Роздрібна торгівля
Walmart	Python	Spring MVC	Роздрібна торгівля
Target	Python	Apache Struts	Роздрібна торгівля
Best Buy	Java, PHP	Laravel, Symfony	Електроніка
Newegg	Python	Codelgniter	Комп'ютерна техніка
Google Store	Python	Django, Flask	Електроніка
Zappos	Python	Flask	Взуття, аксесуари
BigBasket	Python	Django, Flask	Інтернет-гастрономія

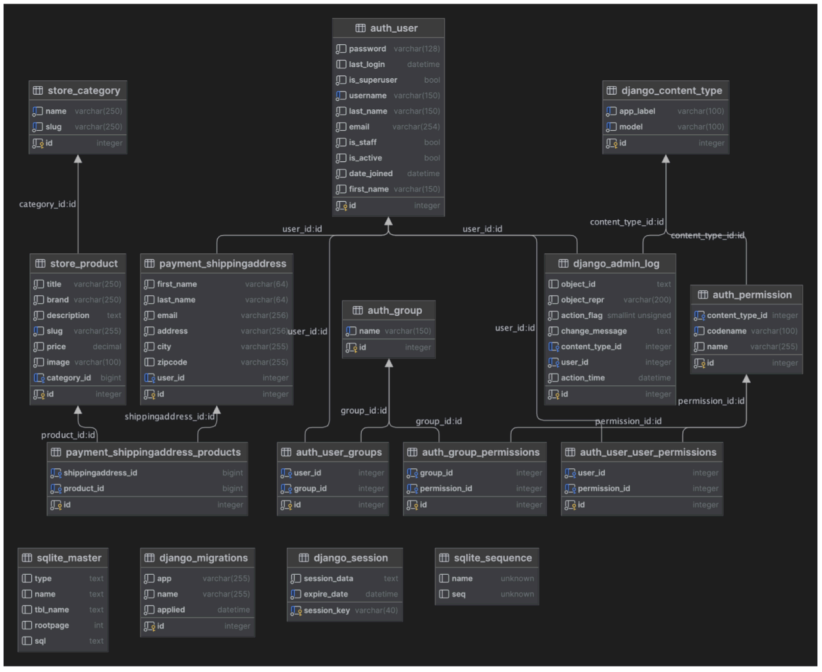
- Щоб легше було визначитись з мовою програмування та фреймворком для виконання інтернет магазину було виконано таблицю з порівнянням одних з найпопулярніший сайтів світу, на якій мові вони були написані та які фреймворки були задіяні

Вибір мови програмування

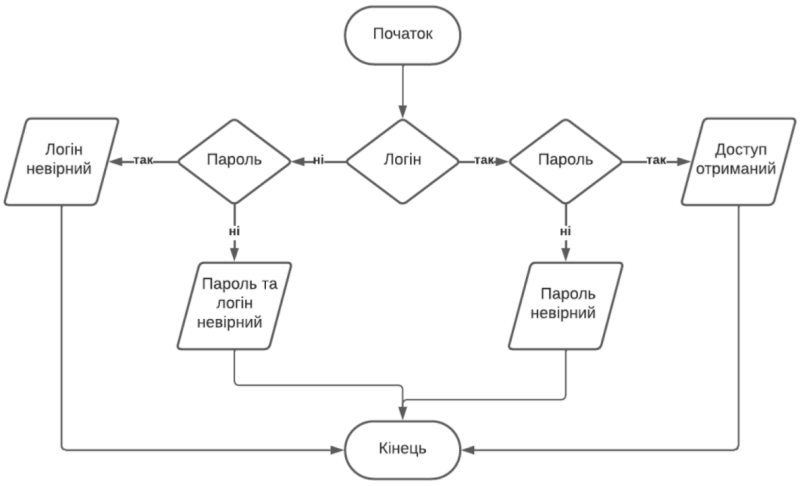
	Python	C#	PHP
Тип мови	Інтерпретована, високорівнева	Компільована, високорівнева	Інтерпретована, високорівнева
Призначення	Загальне програмування, веб-розробка наукові обчислення,	Розробка Windows-додатків, ігрова розробка, веб-розробка	Веб-розробка, розробка динамічних сторінок
Синтаксис	Простий, з лаконічною граматикою	Синтаксис, подібний до C/C++	Синтаксис, подібний до C/C++, але зі своїми особливостями
ОПП підтримка	Повна підтримка ООП	Повна підтримка ООП	Часткова підтримка ООП
Управління пам'яттю	Автоматична збірка сміття	Управління пам'яттю з допомогою збирача сміття, але також можлива ручна робота з пам'яттю	Автоматична збірка сміття
Використання веб-фреймворків	Широкий вибір веб-фреймворків, таких як Django, Flask	Використання фреймворків .NET, таких як ASP.NET	Широкий вибір веб-фреймворків, таких як Laravel, Symfony
Розширюваність	Велика кількість модулів та бібліотек	Велика кількість бібліотек, особливо для Windows-розробки	Широкий спектр розширень та бібліотек
Використання в інших областях	Наукові дослідження, аналіз даних, штучний інтелект	Розробка десктопних додатків, ігрова розробка, розробка мобільних додатків з використанням Xamarin	Веб-розробка, розробка динамічних сторінок

- **Python** - це високорівнева, інтерпретована, загально призначена мова програмування. Вона була розроблена у 1991 році Гвідо ван Россумом і має простий синтаксис, що полегшує читання і розуміння коду. Python популярний серед програмістів завдяки своїй простоті, ефективності та широким можливостям.

Діаграма бази-даних



Блок-схема логіки авторизації користувачів



Фреймворк

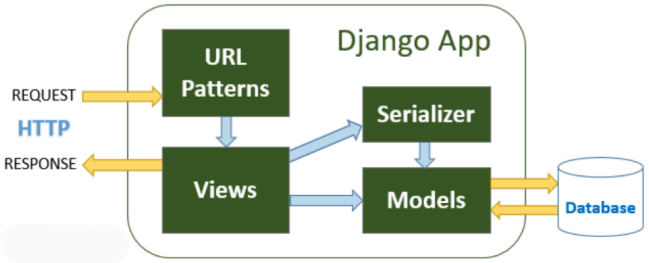
Фреймворк — інфраструктура програмних рішень, що полегшує розробку складних систем [7]. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою, але при цьому вона має ряд обмежень, що задають правила створення структури проекту та написання коду.

Фреймворки поділяються на:

- Бекенд-фреймворки;
- Фронтенд-фреймворки;
- Фуллстек-фреймворки.

Бекенд фреймворки – це бібліотеки серверних мов програмування [7]. Вони допомагають у створенні внутрішньої конфігурації веб-застосунків.

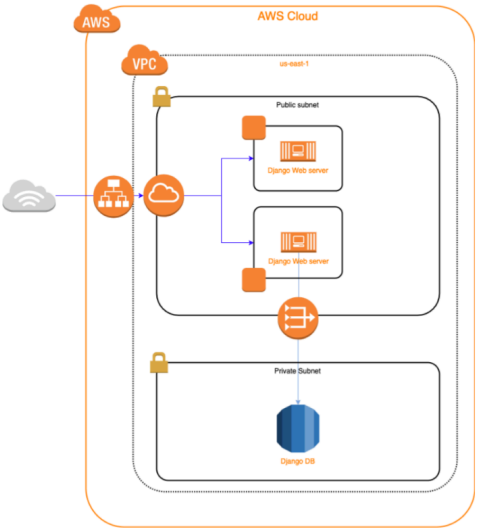
Внутрішні веб-платформи пропонують готовий набір інструментів, які спрощують розробку веб-додатків. Вони беруть на себе рутинні завдання, такі як автентифікація користувачів, захист даних та управління базами даних, дозволяючи розробникам зосередитися на унікальній функціональності своїх проектів.



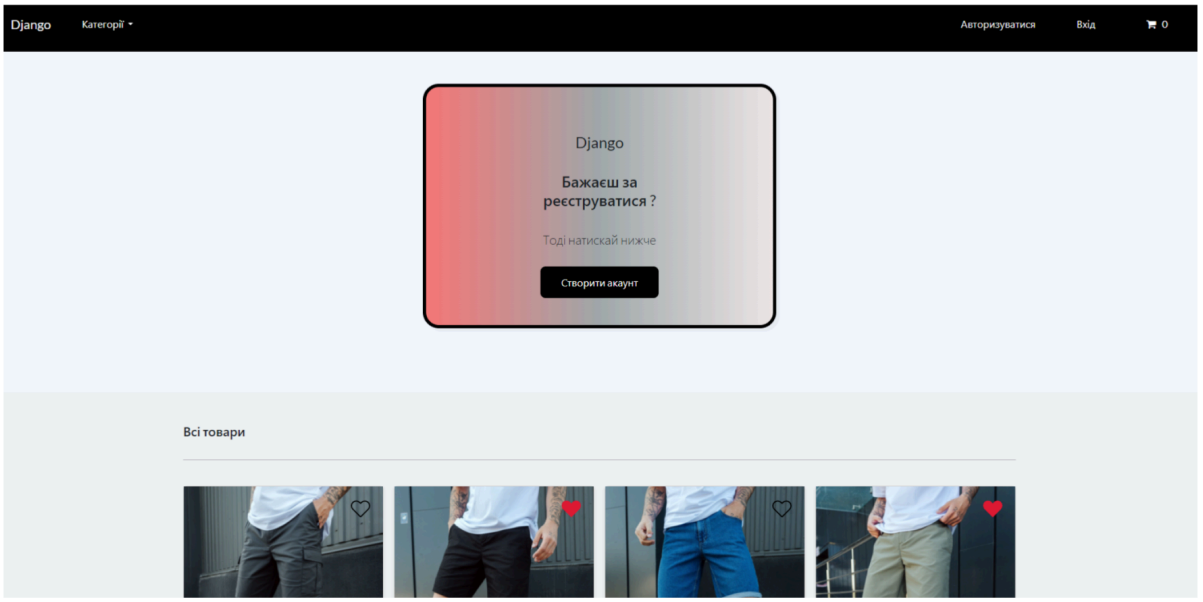
Хостинг

Хостинг — послуга надання дискового простору, підключення до мережі та інших ресурсів для розміщення фізичної інформації на сервері, що постійно перебуває в мережі (наприклад інтернеті).

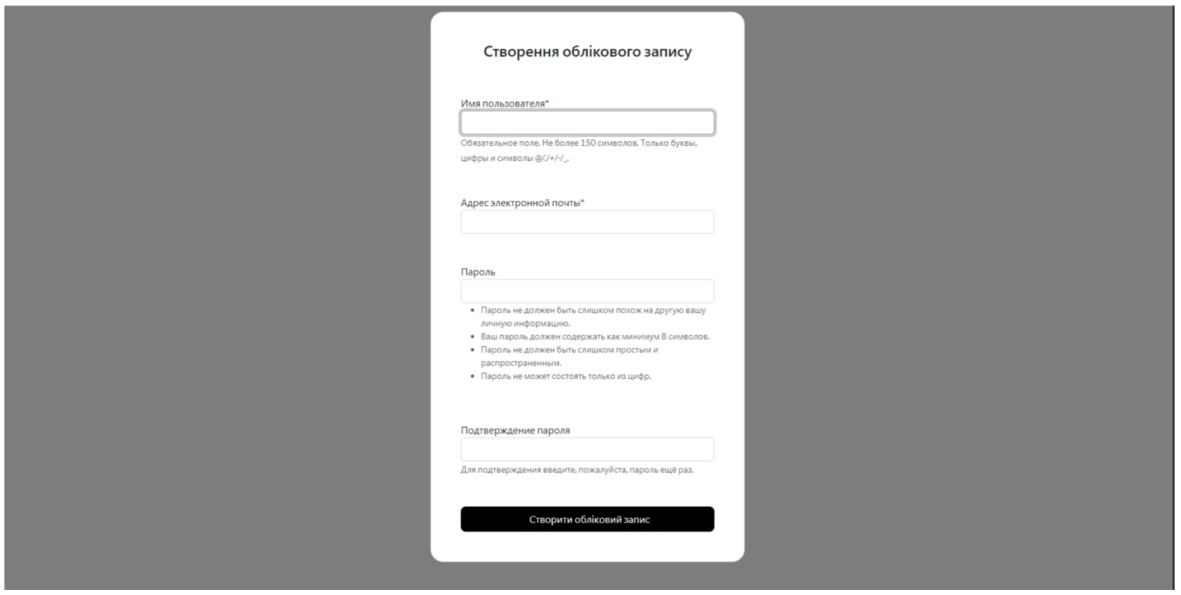
Поняття хостингу включає в себе широкий спектр послуг із використанням різного апаратного та програмного забезпечення. Зазвичай під поняттям послуги хостингу мають на увазі, як мінімум, послугу розміщення файлів сайту на сервері, на якому запущене ПЗ, необхідне для обробки запитів до цих файлів (веб сервер). Як правило, до послуг хостингу вже входить надання місця для поштової кореспонденції, баз даних, DNS файлового сховища тощо, а також підтримка функціонування відповідних сервісів, однак вони можуть надаватися і окремо. А саме, послуга може бути обмежена розміщенням поштової кореспонденції та відповідного ПЗ (поштовий хостинг), клієнтських файлів (файловий хостинг), виключно відео файлів (відео хостинг) або інших файлів певного типу та за певними умовами



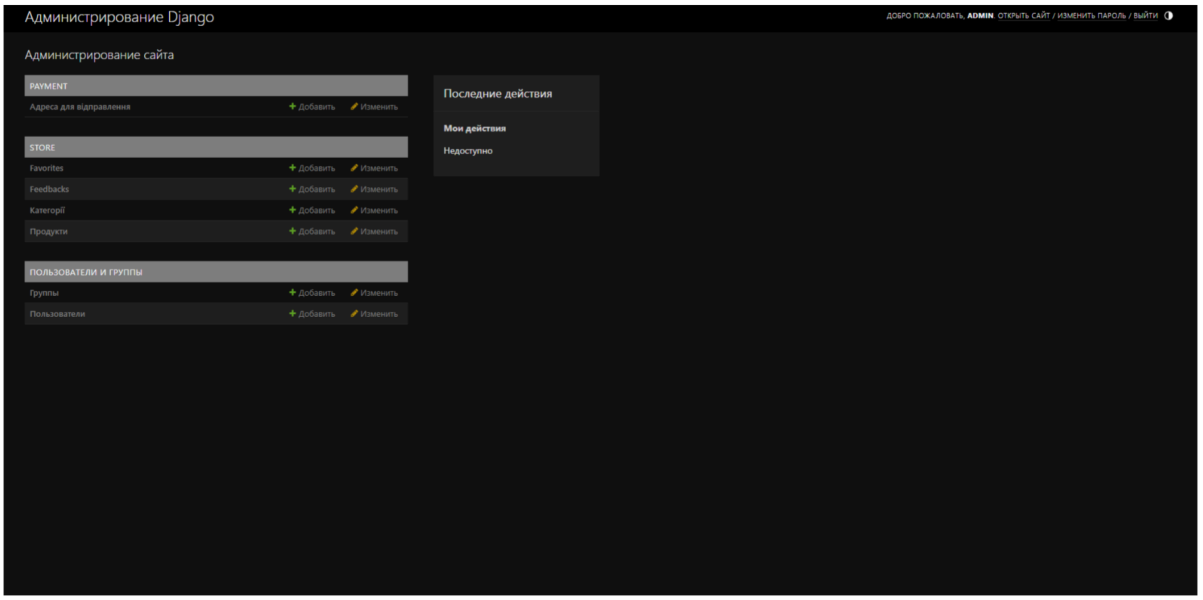
Головна сторінка інтернет-магазину



Реєстрація користувача



Адмін панель інтернет магазину



Переваги

До переваг можна віднести:

Безпека даних користувачів: Ви можете акцентувати увагу на безпеці покупців, наприклад, через використання сучасних протоколів шифрування для захисту фінансових і персональних даних.

Адаптивність до змін у попиті: Використання алгоритмів прогнозування для адаптації до сезонних змін або актуальних трендів дає можливість бути на крок попереду конкурентів, збільшуючи запаси популярних товарів та знижуючи ризики надлишку товарів, що не користуються попитом.

Оптимізація для швидкого завантаження сторінок: Швидкість завантаження є важливим фактором для зручності користувачі. Ви можете наголосити на оптимізації контенту та серверної частини для забезпечення миттєвого завантаження сторінок.

Зручність: Для оформлення замовлення покупець витрачає мінімум часу так як інтерфейс має дуже спрощений та зрозумілий вигляд