МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНТЕРНЕТ МАГАЗИНУ ВЗУТТЯ З ВИКОРИСТАННЯМ DJANGO»**

Виконала: студентка 4 курсу, групи 6.1219-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
О.Р. Смотрицька
(ініціали та прізвище)

Керівник завідувач кафедри програмної інженерії, доцент, к.ф.-м.н. Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення

(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.

(підпис)

« 07 » 02 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Смотрицькій Олександрі Романівні

(прізвище, ім'я та по-батькові)

1. Тема роботи (проєкту) Розробка інтернет магазину взуття з використанням Django

керівник роботи (проєкту) Лісняк Андрій Олександрович, к.ф.-м.н., доцент

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 26 » січня 2023 року № 102-с

2. Строк подання студентом роботи 07.06.2023

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.
3. Практична реалізація коду.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Реалізація та тестування вебсайту.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис, дата**	
		завдання видав	**завдання прийняв**

7. Дата видачі завдання 07.02.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.02.2023	
2.	Збір вихідних даних.	13.02.2023	
3.	Обробка методичних та теоретичних	20.02.2023	
	джерел.		
4.	Розробка першого розділу.	08.03.2023	
5.	Розробка другого розділу.	23.03.2023	
6.	Розробка третього розділу.	25.04.2023	
7.	Розробка четвертого розділу.	17.05.2023	
8.	Оформлення та нормоконтроль	01.06.2023	
	кваліфікаційної роботи.		
9.	Захист кваліфікаційної роботи.	21.06.2023	

Студент _____ О.Р. Смотрицька
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.О. Лісняк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інтернет магазину взуття з використанням Django»: 60 с., 30 рис., 7 табл., 20 джерел, 7 додатків.

CSS, DJANGO, HTML, PYTHON, SQLITE, БАЗА ДАНИХ, ВЕБСАЙТ, ІНТЕРНЕТ МАГАЗИН.

Об’єкт дослідження – процес розробки інтернет-магазину взуття.

Мета роботи – розробка інтернет-магазину взуття з використанням фреймворку Django.

Предмет дослідження – Pycharm, Django, draw.io.

Методи дослідження – методи об’єктно-орієнтованого програмування, методи програмної інженерії, системний аналіз.

У кваліфікаційній роботі проаналізовано фреймворки Python для створення інтернет магазинів. Для створення інтернет-магазину обрано Django, оскільки цей фреймворк дозволяє створювати високопродуктивні та функціональні вебдодатки. У Django є багато розроблених модулів, які спрощують процес розробки, також вбудована адміністративна панель для зручного управління сайтом. Фреймворк також має легку роботу з базами даних та дає можливість створення REST API інтернет-магазину. Всі ці функції стануть в пригоді при створенні потрібного інтернет-магазину.

В результаті роботи спроєктовано та створено інтернет магазин взуття з використанням Django.

SUMMARY

Bachelor’s Qualifying Paper «Development of The Shoe Online Store using Django»: 60 pages, 30 figures, 7 tables, 20 references, 7 supplements.

CSS, DATABASE, DJANGO, HTML, ONLINE STORE, PYTHON, SQLITE, WEBSITE.

The object of the study is the process of developing an online shoe store.

The aim of the study is to develop an online shoe store using the Django framework.

The subject of the study is the Django web framework (Python).

The methods of research are object-oriented programming methods, software engineering methods, system analysis.

The paper analyzes Python frameworks for creating online stores. Django was chosen for creating the online store, as this framework allows creating high-performance and functional web applications. Django has many developed modules that simplify the development process, as well as a built-in administrative panel for easy website management. The framework also has an easy-to-use database system and the ability to create REST API for the online store. All of these features will come in handy when creating the desired online store.

As a result of the work, an online shoe store was designed and created using Django.

ЗМІСТ

ВСТУП

У сучасному світі розробка інтернет-магазинів є великою потребою, оскільки все більше людей шукають зручний спосіб придбання товарів в Інтернеті. Фреймворк Django є одним з найпопулярніших фреймворків для розробки вебдодатків на мові програмування Python, і він має багато корисних функцій та можливостей для створення високоякісних інтернет-магазинів.

Отже, актуальною задачею є створення функціонального та зручного для користувача інтернет-магазину взуття, який має такі функції: оформлення замовлень, кошик та оплату.

Метою роботи є розробка інтернет-магазину взуття з використанням фреймворку Django.

Задачі, які необхідно розв’язати для досягнення поставленої мети:

– проаналізувати існуючі вебфреймворки Python;

– проаналізувати предметну область;

– спроєктувати та реалізувати базу даних;

– реалізувати інтерфейс;

– протестувати роботу сайту.

Об’єкт дослідження – процес розробки інтернет-магазину взуття.

Предмет дослідження – Pycharm, Django, draw.io.

Методи дослідження – методи об’єктно-орієнтованого програмування, методи програмної інженерії, системний аналіз.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Аналіз популярних фреймворків для розробки вебдодатків

У якості інструментів для розробки сучасних інтернет-магазинів досить часто використовують різноманітні фреймворки або системи управління контентом. Фреймворки налічують широкі можливості для створення функціонального та налаштованого під конкретного користувача вмісту сайту. За допомогою розширень та плагінів, фреймворки можуть бути розширені та адаптовані до різних потреб проєкту. Системи управління контентом надають простий інтерфейс для керування вмістом сайту, включаючи додавання, редагування та видалення сторінок, публікацію новин і інформації про товари. Ці системи мають загальну структуру сайту і підходять для тих, хто шукає зручне та легке управління сайтом без глибоких технічних знань. Отже, фреймворки в даному випадку мають перевагу для розробки сучасних інтернет-магазинів завдяки їх потужності, гнучкості та можливості створювати унікальний функціонал, що відповідає потребам конкретного проєкту.

Серед великої кількості фреймворків існують платформи, які можна розглядати для розробки інтернет-магазину такі, як PHP фреймворки, Node.js платформи та Python платформи.

PHP є мовою програмування загального призначення, яка досить часто використовується для розробки інтернет-магазинів, особливо в поєднанні з фреймворками та системами управління контентом. PHP пропонує розширені можливості для обробки даних, маніпулювання базами даних та роботи з вебсерверами. Одним з найпопулярніших фреймворків на основі PHP для розробки інтернет-магазинів є Laravel. Він надає зручну та елегантну синтаксичну структуру, що спрощує процес розробки. Laravel пропонує ряд функціональних можливостей, включаючи маршрутизацію, керування

сесіями, аутентифікацію користувачів та роботу з базами даних. Іншим популярним фреймворком на основі PHP є Symfony. Він відомий своєю стабільністю та розширюваністю. Symfony надає широкий спектр готових компонентів та бібліотек, що спрощують розробку інтернет-магазину [1–4].

Node.js є популярною платформою для розробки серверних додатків. Використовуючи JavaScript як основну мову програмування, Node.js дозволяє розробникам створювати ефективні та масштабовані додатки, які працюють як на стороні сервера, так і на клієнтському боці. Один з найпопулярніших фреймворків на основі Node.js для розробки інтернет-магазинів – це Express.js. Він є легким та гнучким фреймворком, який дозволяє швидко створювати серверні додатки. Express.js надає просту синтаксичну структуру та має велику кількість розширень, що допомагають в управлінні маршрутизацією, сесіями, обробкою запитів та іншими задачами. Ще одним популярним фреймворком на основі Node.js є Nest.js. Це прогресивний, модульний та розширюваний фреймворк, який базується на концепції «Angular» для створення ефективних та масштабованих додатків [5–7].

Ще одним популярним фреймворком на основі Node.js для розробки інтернет-магазинів є Nest.js. Nest.js є прогресивним, модульним та розширюваним фреймворком, який базується на концепції "Angular" для створення ефективних та масштабованих додатків.

Однією з досить перспективних платформ є саме Python. Для Python існують такі фреймворки Django, Pyramid, Web2py, Flask, Bottle, Tornado, CherryPy, TurboGears та інші, які можна використати для розробки інтернет-застосунку.

Мова Python нині стає все більш популярною у різних сферах розробки програмного забезпечення. Зокрема, ця мова застосовується в наукових обчисленнях, штучному інтелекті та машинному навчанні, для збору та аналізу великих обсягів даних, під час розробки вебзастосунків, для автоматизації дій користувача та в інших галузях. Перевагами мови Python є простота вивчення, швидкість написання коду, наявність багатьох бібліотек

та фреймворків з підтримкою широких функціональних можливостей. Проте суттєвим недоліком цієї мови є швидкість виконання коду [8].

Порівняємо декілька Python фреймворків для веброзробки, а саме: Django, Flask, Pyramid та Bottle.

Django – це високорівневий фреймворк, який дозволяє швидко створювати повноцінні вебдодатки. Він має вбудовану адміністративну панель, високу продуктивність та багато інших переваг. Django відносно простий у використанні, але потребує певного рівня знань Python [9]. Django пропонує широкий спектр можливостей для розробки вебдодатків, включаючи систему маршрутизації, шаблони, аутентифікацію та авторизацію користувачів, адміністративну панель, ORM та багато іншого [10].

Однією з головних переваг Django є його висока продуктивність. Фреймворк забезпечує швидкий доступ до баз даних за допомогою своєї ORM та використанням кешування. Крім того, він має велику та дружелюбну спільноту розробників, яка надає багато готових рішень та плагінів для розширення можливостей фреймворку [11].

Основні можливості Django включають:

– адміністративний інтерфейс: Django надає готовий адміністративний інтерфейс для керування базою даних та іншими аспектами додатку;

– ORM: Django має вбудовану ORM (об’єктно-реляційне відображення), що дозволяє працювати з базою даних у стилі ООП замість SQL;

– маршрутизація: Django має зручний механізм маршрутизації URL, що дозволяє легко визначати, який код виконуватиметься при запиті на певний URL;

– шаблонізація: Django надає вбудований движок шаблонів, що дозволяє легко відділяти логіку від представлення;

– безпека: Django має вбудовану систему аутентифікації та авторизації, що дозволяє захистити вебдодаток від несанкціонованого доступу;

– міжнародність: Django надає зручні інструменти для локалізації та

інтернаціоналізації вебдодатка.

Django є одним з найбільш популярних фреймворків для веброзробки на Python та використовується на багатьох відомих сайтах, таких як Instagram, Pinterest, Mozilla та інші.

Ще один популярний фреймворк для веброзробки на Python – це Flask Flask – це мінімалістичний фреймворк, який дозволяє розробникам створювати вебдодатки швидко та ефективно. Фреймворк має легковажний підхід до розробки та дозволяє використовувати різні бібліотеки Python. Flask добре підходить для розробки додатків середньої складності, але може виявитися складним для більш великих проєктів [12]. Одним з недоліків Flask є те, що він не має вбудованої підтримки для багатьох функцій, які має Django, таких як адміністративний інтерфейс.

Pyramid – це високопродуктивний вебфреймворк, який дозволяє розробляти вебдодатки будь-якої складності та розміру. Основними перевагами Pyramid є його простота, гнучкість та розширюваність (за допомогою пакетів, які можна встановлювати через pip). Недоліками Pyramid можуть бути відносно низька популярність порівняно з Django та Flask, що може зробити важчим знаходженням документації та рішень для конкретних проблем. Крім того, у Pyramid можуть бути деякі проблеми з продуктивністю, якщо вебдодатки дуже складні та великі [13].

Bottle є легковажним, простим і зрозумілим фреймворком веброзробки на Python. Він має мінімальну кількість залежностей і підтримує вбудований вебсервер. Більшість функцій фреймворку вбудовано в один файл, що робить його легко зрозумілим та налаштовуваним (Flask та Pyramid мають більшу складність). Основні можливості Bottle включають маршрутизацію, обробку запитів та шаблонів. Фреймворк також має підтримку куки та сеансів, декількох видів баз даних, аутентифікацію користувачів та інтеграцію з WSGI-серверами [14]. Одним з недоліків Bottle є те, що фреймворк має меншу спільноту, порівняно з Django, Flask та Pyramid, що може ускладнити пошук допомоги в разі проблем. Також, Bottle не має такої широкої

функціональності, як Django та Flask, що може бути недоліком для більш складних проєктів.

Отже, Django має перевагу серед інших фреймворків при створенні інтернет-магазину завдяки своїм вбудованим можливостям, таким як адміністративний інтерфейс, ORM, маршрутизація, шаблонізація, безпека та висока продуктивність. Завдяки цим функціям, Django дозволяє швидко розробляти повноцінні веб-додатки, керувати базою даних та забезпечувати безпеку користувачів, що особливо важливо для ефективної роботи інтернет-магазину.

1.2 Технічне завдання

1.2.1 Опис предметної області

Розробляється вебсайт інтернет-магазину взуття. На головній сторінці сайту розташуються зображення товарів та їхній опис. Користувач буде мати можливість додавати товари до кошику, змінювати їхню кількість, видаляти товари. Роблячи онлайн замовлення, покупець повинен заповнити форму для надання інформації про себе (прізвище, ім’я, номер телефону, електронна адреса) та про доставку (місто, адреса, індекс). Після оплати замовлення користувач отримає чек за оплату.

1.2.2 Функціональні вимоги

Функціональне призначення інтернет-магазину – реалізувати можливість купити товар, в даному випадку взуття, в інтернет магазині.

Загальні функціональні можливості додатку:

1) вебсайт магазину надає користувачу такі можливості:

– перегляд каталогу товарів;
– додавання товарів у кошику;
– видалення товарів у кошику;
– оформлення замовлення;
– оплата замовлення;

2) вебсайт магазину надає менеджеру такі можливості:
– заповнення сайту;
– оновлення інформації про товар (наявність моделі, розміру);
– додавання нових товарів;
– видалення товарів;
– обробка замовлень;
– керування замовленнями.

1.2.3 Нефункціональні вимоги

Нефункціональні вимоги до інтернет-магазину взуття включають:

– продуктивність: швидкість завантаження сторінок та відповіді системи на запити користувачів;

– надійність: стабільна та безперебійна робота системи, мінімізація відмов та збереження даних користувачів;

– безпека: захист особистої інформації користувачів, безпека транзакцій та захист від шахрайства;

– масштабованість: здатність системи розширюватися та обробляти зростаюче навантаження при збільшенні кількості користувачів та товарів;

– сумісність: підтримка різних пристроїв, браузерів та операційних систем для забезпечення доступності магазину для широкого кола користувачів;

– виконання стандартів: відповідність веб-стандартам та правилам електронної торгівлі, забезпечення зручності та навігації на сайті;

– дизайн та зручність використання: інтуїтивно зрозумілий та привабливий дизайн, зручна навігація та простота використання інтерфейсу для користувачів.

Висновки до розділу 1

У першому розділі було з'ясовано, що мова програмування Python є все більш популярною у різних сферах розробки програмного забезпечення та проведено аналіз популярних фреймворків Python для розробки вебдодатків, а саме Django, Flask, Pyramid та Bottle. Для розробки вебсайту інтернет-магазину взуття було обрано Django так, як він є високорівневим фреймворком, який дозволяє швидко створювати повноцінні вебдодатки, а також має вбудовану адміністративну панель, високу продуктивність та багато інших переваг. У розділі також було описано основні функції та можливості майбутнього вебсайту інтернет-магазину взуття.

2 ОГЛЯД МЕТОДІВ ПРОЄКТУВАННЯ

2.1 Побудова діаграми прецедентів

Діаграма прецедентів або діаграма випадків (варіантів) використання (use case diagram) – це діаграма, яка описує взаємозв’язки і залежності між групою випадків використання (прецедентами) і акторами, що беруть участь у процесі [15].

Прецедент визначає групу дій у системі, які призводять до конкретного видимого результату. Прецеденти відповідають зовнішньому інтерфейсу системи і визначають форму вимог до того, що має робити система. Актор — це зовнішній чинник, який взаємодіє з системою шляхом участі у випадку використання [15].

Діаграму прецедентів предметної області «Інтернет магазин взуття» зображено на рис. 2.1.

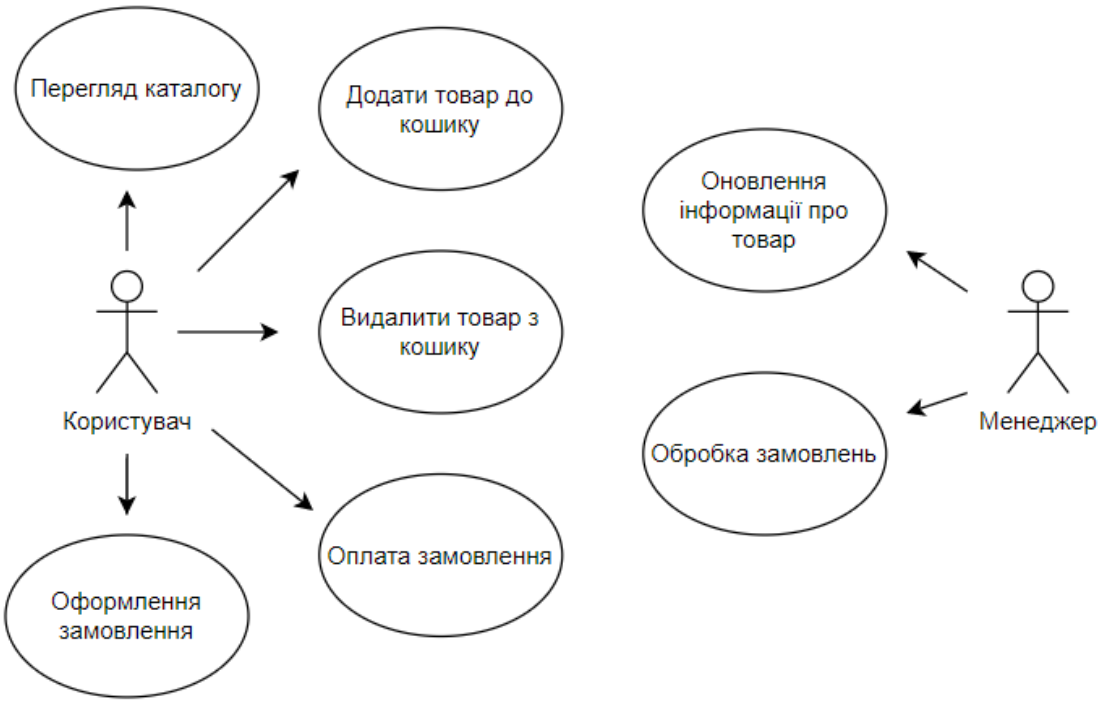

Рисунок 2.1 – Діаграма прецедентів

Business Use Case (бізнес-прецедент) – це тип прецеденту в діаграмі прецедентів, який описує функціональність системи з точки зору бізнес-процесів, ділових сценаріїв або виконання певної бізнес-задачі. Він фокусується на взаємодії між системою і зовнішніми акторами з орієнтацією на бізнес-потреби та цілі.

Опис Business Use Case представляє собою специфікацію, яка, подібно до звичайного варіанту використання, складається з наступних пунктів [16]:

– найменування;
– короткий опис;
– цілі і результати;
– опис сценаріїв;
– спеціальні вимоги;
– розширення;
– зв’язок з іншими Business Use Case;
– діаграми діяльності.

Розглянемо сценарії інтернет-магазину взуття.

Додати товар до кошику.

Найменування – додати товар до кошику.

Короткий опис – даний Business Use Case реалізує процес додавання товару до кошику користувачем на сайті.

Цілі – додати товар та побачити його у кошику.

Основний сценарій:

– користувач заходить на головну сторінку сайту;
– користувач обирає товар та додає його до кошику;
– користувач бачить доданий товар у кошику.

Видалити товар з кошику.

Найменування – видалити товар з кошику.

Короткий опис – даний Business Use Case реалізує процес видалення товару з кошику користувачем на сайті.

Цілі – видалити товар та побачити його відсутність у кошику.

Основний сценарій:

– користувач заходить на головну сторінку сайту;

– користувач переходить на сторінку кошика;

– користувач видаляє товар з кошику;

– користувач бачить, що видалений товар відсутній у кошику.

Оформлення замовлення.

Найменування – оформлення замовлення.

Короткий опис – даний Business Use Case реалізує процес оформлення замовлення користувачем на сайті.

Цілі – оформити замовлення та побачити чек.

Основний сценарій:

– користувач заходить на головну сторінку сайту;

– користувач додає товари до кошика;

– користувач натискає «замовити» та вводить свої особисті дані;

– користувач отримує чек замовлення.

Альтернативний сценарій: введені дані невірні – користувач бачить повідомлення про невірно введені дані та необхідність повторного вводу особистих даних.

Оплата замовлення.

Найменування – оплата замовлення.

Короткий опис – даний Business Use Case реалізує процес оплати замовлення користувачем на сайті.

Цілі – здійснити успішну оплату замовлення.

Основний сценарій:

– користувач заходить на головну сторінку сайту;

– користувач додає товари до кошика;

– користувач натискає «замовити» та вводить свої особисті дані;

– користувач отримує чек замовлення;

– користувач натискає «оплатити» та вводить дані для оплати;

– користувач отримує повідомлення про успішну оплату.

Альтернативний сценарій: введені дані для оплати невірні – користувач бачить повідомлення про невірно введені дані та необхідність повторного вводу даних.

Оновлення інформації про товар.

Найменування – оновлення інформації про товар.

Короткий опис – даний Business Use Case реалізує процес оновлення інформації про товар на сайті менеджером, а саме додавання нових товарів або їх видалення, редагування графи про наявність товару та інше.

Цілі – додати/видалити новий товар та побачити/не побачити його відображення на сайті.

Основний сценарій:

– менеджер заходить в адмін-панель;

– менеджер додає/видаляє товари;

– менеджер додає фото до товарів при додавання нового товару.

Обробка замовлення.

Найменування – обробка замовлення.

Короткий опис – даний Business Use Case реалізує процес обробки замовлення менеджером на сайті.

Цілі – обробити замовлення та направити інформацію про замовлення пакувальникам для подальшої доставки.

Основний сценарій:

– менеджер отримує нове замовлення;

– менеджер бачить, що замовник сплатив замовлення;

– менеджер перевіряє наявність товару;

– менеджер передає інформацію про замовлення пакувальникам;

– менеджер надсилає замовнику дані про доставку.

Альтернативний сценарій: замовленого товару немає в наявності – менеджер повідомляє замовники про відсутність товару та пропонує змінити замовлення.

2.2 Побудова діаграми послідовності

Діаграма послідовності – це одна з видів діаграм, яка використовується для візуалізації послідовності взаємодій між об'єктами або компонентами системи в певному сценарії. Вона дозволяє показати порядок виконання повідомлень та взаємодіючих об'єктів протягом певного часового періоду. Її основна мета полягає в тому, щоб виявити порядок виконання дій, залежності та обмін повідомленнями, викликами методів і відповідями. Це дозволяє краще розуміти, як об'єкти взаємодіють між собою в рамках певного сценарію або функціональності системи.

На діаграмах послідовностей показано обмін повідомленнями (тобто виклик методів) між декількома об’єктами у окремій обмеженій часом ситуації. Об’єкти є екземплярами класів. Основний наголос на діаграмах послідовностей робиться на порядок і моментах часу, у які повідомлення надсилаються об’єктам. На діаграмах об’єкти буде показано вертикальними штриховими лініями з назвою об’єкта над ними. Вісь часу також має вертикальний напрямок, її спрямовано вниз, повідомлення, які надсилаються від одного об’єкта до іншого, буде позначено стрілками з назвами операції і параметрів [15].

Повідомлення можуть бути або синхронними, звичайного типу повідомленнями, за виклику яких керування передається викликаному об’єкту до завершення виконання методу, або асинхронними, за виклику яких керування передається назад напряму об’єкту, який здійснював виклик. За використання синхронного повідомлення збоку від викликаного об’єкта буде показано вертикальний блок, який показуватиме перебіг виконання програми [15].

Діаграму послідовності предметної області «Інтернет магазин взуття» для сценарію оформлення замовлення зображено на рис. 2.2.

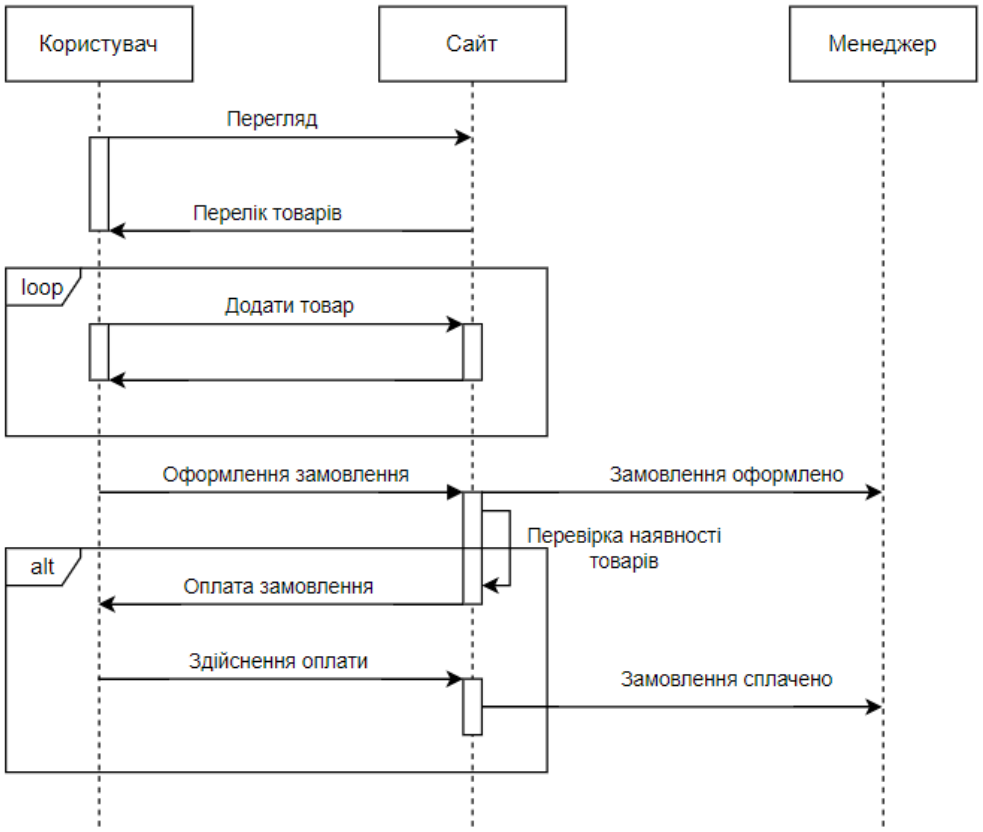

Рисунок 2.2 – Діаграма послідовності оформлення замовлення

Діаграму послідовності предметної області «Інтернет магазин взуття» для сценарію обробки замовлення зображено на рис. 2.3.

Рисунок 2.3 – Діаграма послідовності обробки замовлення

2.3 Проєктування моделей даних

Виходячи з аналізу вимог можна виділити наступні концептуальні сутності:

– сутність «Продукт (Product)» зберігає інформацію про продукт, а саме назву продукту, модель, ціну, наявність та фотографію;

– сутність «Категорія (Category)» зберігає інформацію про модель взуття та її артикул у каталозі;

– сутність «Замовник (Customer)» зберігає інформацію про замовника, а саме ім’я, прізвище, електрону адресу та номер телефону;

– сутність «Замовлення (Order)» зберігає інформацію про замовлення, а саме дату заказу, номер транзакції, номер замовника, та статус доставки;

– сутність «Елемент замовлення (Order item)» зберігає інформацію про номер замовлення, дату, продукт, модель та кількість продукту у кошику;

– сутність «Інформація доставки (Shipping information)» зберігає інформацію про замовника, номер замовлення, населений пункт та адресу куди потрібно доставити товар;

– сутність «Чек (Receipt)» зберігає інформацію про дату, замовника, номер замовлення, та загальну суму до сплати.

ER-діаграма (діаграма сутності-зв'язок) є зручним інструментом для моделювання реляційних баз даних. Вона складається з сутностей, атрибутів і зв'язків між ними [17].

Сутність – це окремий об'єкт або поняття, яке містить дані, що можуть бути збережені в базі даних. Атрибут – це характеристика, що відноситься до конкретної сутності, і описує властивості або аспекти цієї сутності. Зв'язок – це відношення між двома або більше сутностями, яке описує, як вони пов'язані між собою.

Кожна сутність представляється прямокутником, а кожен атрибут – еліпсом, який з'єднується з відповідною сутністю лінією. Зв'язки між сутностями показуються лініями, які з'єднують відповідні сутності [17].

ER-діаграми є важливим інструментом для проєктування баз даних і допомагають розуміти зв'язки між різними сутностями та їх атрибутами. Крім того, вони можуть бути використані для автоматичної генерації коду для створення таблиць і зв'язків в базі даних [18].

Основна ідея моделі реляційних даних полягає в тому, щоб використовувати таблиці для представлення інформації. Кожна таблиця має назву і складається з рядків і стовпців. Рядки таблиці представляють записи, а стовпці – атрибути цих записів [19].

У наступних таблицях (2.1 – 2.7) будуть розглянуті необхідні для побудови ER-діаграми сутності, атрибути, типи даних та ключі.

Таблиця 2.1 – Таблиця для сутності «Продукт (Product)»

Атрибут	Тип даних	Ключ
Id	Числовий	Первинний
Title	Текстовий	
Model	Текстовий	
Price	Текстовий	
Stock	Логічний	
Picture	Текстовий	

Таблиця 2.2 – Таблиця для сутності «Категорія (Category)»

Атрибут	Тип даних	Ключ
Id	Числовий	Первинний
Title	Текстовий	
Description	Текстовий	

Таблиця 2.3 – Таблиця для сутності «Замовник (Customer)»

Атрибут	Тип даних	Ключ
Id	Числовий	Первинний

Продовження табл. 2.3

Атрибут	Тип даних	Ключ
Name	Текстовий	
Surname	Текстовий	
Email	Текстовий	
Phone	Числовий	

Таблиця 2.4 – Таблиця для сутності «Замовлення (Order)»

Атрибут	Тип даних	Ключ
Id	Числовий	Первинний
Date_ordered	Дата	
Transaction_id	Числовий	
Customer_id	Числовий	Зовнішній
Complete	Логічний	

Таблиця 2.5 – Таблиця для сутності «Елемент замовлення (Order item)»

Атрибут	Тип даних	Ключ
Id	Числовий	Первинний
Quantity	Числовий	
Date_added	Дата	
Order_id	Числовий	Зовнішній
Product_id	Числовий	Зовнішній
Model_id	Числовий	Зовнішній

Таблиця 2.6 – Таблиця для сутності «Інформація доставки (Shipping information)»

Атрибут	Тип даних	Ключ
Id	Числовий	Первинний
City	Текстовий	

Продовження табл. 2.6

Атрибут	Тип даних	Ключ
State	Текстовий	
Address	Текстовий	
Zip_code	Числовий	
Date_added	Дата	
Customer_id	Числовий	Зовнішній
Order_id	Числовий	Зовнішній

Таблиця 2.7 – Таблиця для сутності «Чек (Receipt)»

Атрибут	Тип даних	Ключ
Id	Числовий	Первинний
Datetime	Дата	
Customer_id	Числовий	Зовнішній
Order_id	Числовий	Зовнішній
Total_price	Числовий	

При проєктуванні БД інформацію зазвичай розміщують у різних таблицях. Таблиці при цьому зв’язують.

Далі будуть розглянуті зв’язки між виявленими сутностями.

Між атрибутами Category та Order item буде зв’язок 1:1, тому що у замовленні може бути тільки одна модель товару.

Між атрибутами Product та Order item буде зв’язок 1:1, тому що у замовленні може бути тільки один тип продукту.

Між атрибутами Order item та Order буде зв’язок Б:1, тому що у замовленні може бути декілька товарів.

Між атрибутами Order та Customer буде зв’язок 1:Б, тому що у одного замовника може бути декілька замовлень.

Між атрибутами Order та Shipping information буде зв’язок Б:1, тому що

на одну адресу може надходити декілька замовлень.

Між атрибутами Customer та Shipping information буде зв’язок Б:Б, тому що на одну адресу може замовляти багато замовників.

Між атрибутами Customer та Receipt буде зв’язок 1:Б, тому що у одного замовника може бути багато чеків.

За допомогою програми Microsoft Visio було побудовано ER-діаграму предметної області «Інтернет-магазин взуття», яка зображена на рис. 2.4.

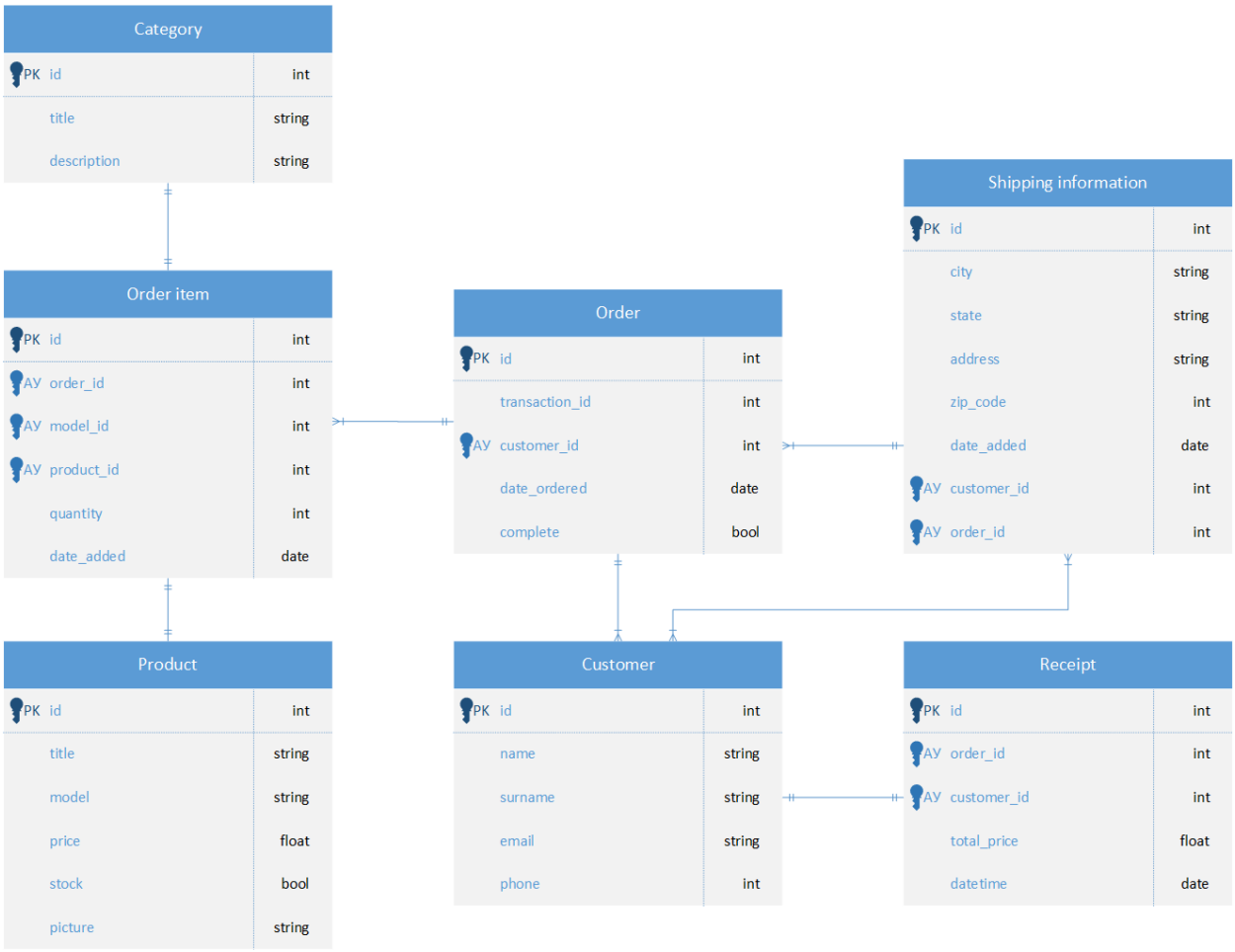

Рис. 2.4 – ER-діаграма предметної області «Інтернет-магазин взуття»

2.4 Побудова діаграми розгортання

Діаграма розгортання використовується для візуалізації фізичної архітектури системи. Вона надає засоби для представлення фізичних

компонентів системи, таких як апаратне забезпечення, програмне забезпечення та мережеві ресурси, а також зв'язків між ними.

На діаграмі розгортання можуть бути зображені такі елементи, про які описано нижче.

Вузли. Вони представляють фізичні або віртуальні пристрої, на яких працюють компоненти системи. Вузли можуть включати сервери, комп'ютери, мобільні пристрої або хмарні платформи. Кожен вузол може мати свої особливості, такі як апаратне забезпечення, операційну систему або доступні сервіси.

Компоненти. Вони представляють фізичні або логічні модулі програмного забезпечення, які виконують певні функції в системі. Компоненти можуть бути розміщені на вузлах та взаємодіяти через мережу. Наприклад, вебсервер може бути компонентом, розташованим на серверному вузлі, а клієнтське програмне забезпечення – на клієнтському вузлі.

Артефакти. Вони представляють фізичні файли або пакети, які використовуються в системі. Це можуть бути виконувані файли, конфігураційні файли, бази даних або інші ресурси, необхідні для роботи системи. Артефакти можуть бути пов'язані з компонентами і розташовуватися на відповідних вузлах.

Зв’язки. Вони показують комунікацію та залежності між компонентами і вузлами. Зв’язки вказують, як компоненти обмінюються повідомленнями або даними через мережу.

Діаграму розгортання предметної області «Інтернет магазин взуття» зображено на рисунку 2.5.

Ця діаграма складається із трьох вузлів пристроїв [20]:

– пристрій: будь-який пристрій, оснащений веббраузером, який слугує сполучним інтерфейсом між системою та користувачем;

– вебсервер: сервер, який власне містить бізнес-логіку системи, приймає та обробляє запити користувача та взаємодіє із сервером бази даних;

– сервер бази даних: сервер, який є інтерфейсом між вебсервером та

базою даних, тобто реагує на запити з боку вебсервера певними змінами у базі.

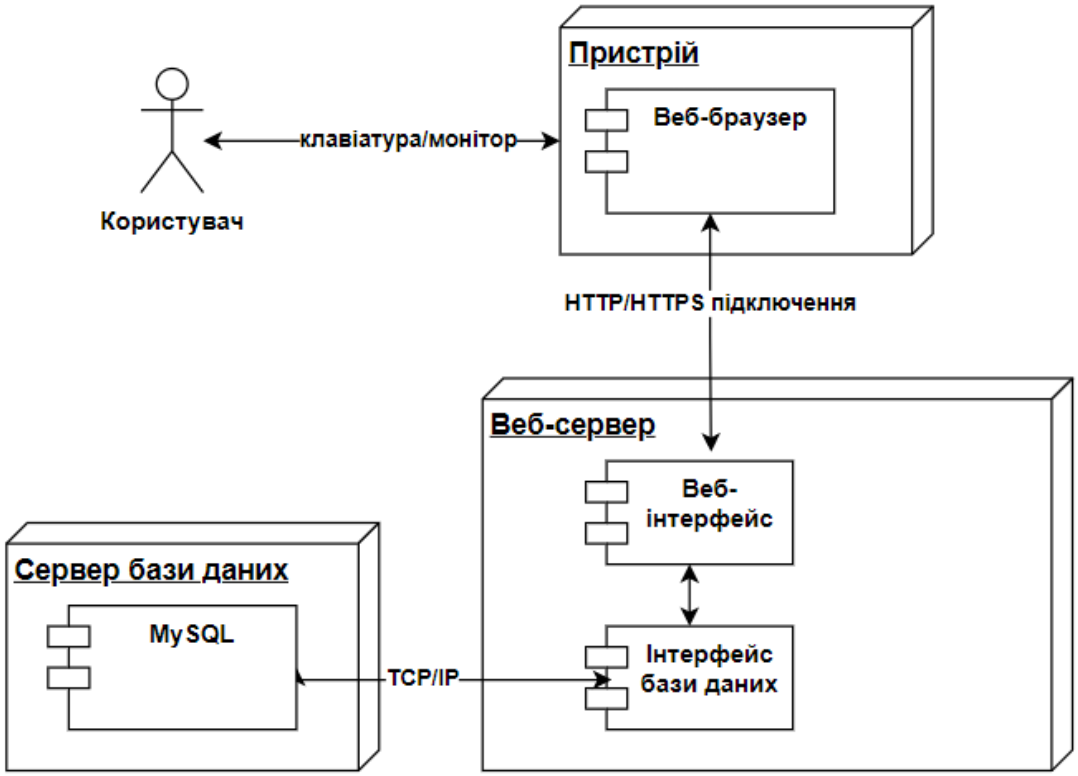

Рисунок 2.5 – Діаграма розгортання

Висновки до розділу 2

У другому розділі було розглянуто та побудовано ключові діаграми, необхідні для розробки інтернет-магазину взуття. Діаграма прецедентів відображає взаємодію користувачів з системою та визначає основні функціональні можливості, які має надавати магазин. ER-діаграма допомагає виявити та візуалізувати зв'язки між різними сутностями в системі, такими як товари, замовлення та клієнти. Було створено таблиці сутностей, у яких розглянуто їх атрибути, типи даних та ключі. За допомогою цих таблиць та програми Microsoft Visio була побудована ER-діаграма предметної області «Інтернет-магазин взуття». Діаграма послідовності дозволяє описати

послідовність взаємодій між компонентами системи, що стає основою для реалізації функціоналу магазину. Діаграма розгортання надає уявлення про фізичну архітектуру системи та розподіл компонентів на різних серверах або вузлах.

3 ПЕРВИННЕ НАЛАШТУВАННЯ ТА ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ БАЗИ ДАНИХ

3.1 Створення проєкту Django

Для початку роботи на ПК потрібно мати Python 3.9, відкривши консоль, за допомогою pip – системи керування пакунками, треба встановити django (рис. 3.1).

```
pip install django
```

Рисунок 3.1 – Установка фреймворку Django

Після вдалої установки треба створити новий проєкт за допомогою команди startproject (рис. 3.2).

```
django-admin startproject onlineshopping
```

Рисунок 3.2 – Створення нового проєкту

Після того, як команда виконана, буде створено новий проєкт під назвою todo_list (рис. 3.3).

```
onlineshopping/
    manage.py
    onlineshopping/
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
```

Рисунок 3.3 – Структура проєкту після виповнення команди startproject

Далі потрібно створити додаток за допомогою команди startapp, у якому буде проводитись подальша робота (див. рис. 3.4).

```
python manage.py startapp main
```

Рисунок 3.4 – Створення нового додатку main

Після виконання цієї команди, у проєкті буде створена нова папка main (рис. 3.5).

```
onlineshopping/
    manage.py
    main/
        __init__.py
        admin.py
        apps.py
        models.py
        tests.py
        views.py
        migrations/
            __init__.py

    onlineshopping/
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
```

Рисунок 3.5 – Структура проєкту після виповнення команди startapp

На цьому підготовка проєкту завершена, тепер за допомогою команди runserver, яка запускає локальний сервер (рис. 3.6) можна перевірити, чи все працює.

```
python manage.py runserver
```

Рисунок 3.6 – Запуск локального сервера

Після виконення команди runserver можна відкрити браузер та перейти за посиланням http://127.0.0.1:8000/, де відкриється вітальна сторінка.

Було проведено початкове налаштування та перший запуск проєкту. Можна переходити до подальшої реалізації.

3.2 Реалізація бази даних

У файлі models.py треба створити моделі до майбутніх таблиць бази даних, а саме – category, customer, order, orderitem, product, receipt та shippinginformation. Моделі Django описують структуру використовуваних даних. Використовувані дані зберігаються в базах даних, і за допомогою моделей здійснюється взаємодія з базою даних (рис. 3.7 – 3.13).

```
class Product(models.Model):
    title = models.CharField('Назва', max_length=50)
    model = models.CharField('Модель', max_length=50)
    price = models.IntegerField('Ціна')
    stock = models.BooleanField('У наявності')
    picture = models.ImageField('Фотографія')
    def __str__(self):
        return self.title
```

Рисунок 3.7 – Створення моделі Product

```
class Category(models.Model):
    title = models.CharField('Модель', max_length=50)
    description = models.TextField('Опис')
    def __str__(self):
        return self.title
```

Рисунок 3.8 – Створення моделі Category

```
class Customer(models.Model):
    user = models.OneToOneField(User, null=True, blank=True, on_delete=models.CASCADE)
    name = models.CharField('Ім'я', max_length=50, null=True)
    surname = models.CharField('Прізвище', max_length=50, null=True)
    email = models.EmailField('Електрона адреса', null=True)
    phone = models.CharField('Номер телефону', max_length=50, null=True)
    def __str__(self):
        return self.name
```

Рисунок 3.9 – Створення моделі Customer

```
class Order(models.Model):
customer = models.ForeignKey(Customer, on_delete=models.SET_NULL, null=True, blank=True)
date_ordered = models.DateTimeField('Дата', auto_now_add=True)
transaction_id = models.CharField('ID Транзакції', max_length=200)
complete = models.BooleanField('Сплачено', default=False)
def __str__(self):
return str(self.id)
```

Рисунок 3.10 – Створення моделі Order

```
class OrderItem(models.Model):
    product = models.ForeignKey(Product, on_delete=models.SET_NULL, null=True)
    model = models.ForeignKey(Category, on_delete=models.SET_NULL, null=True)
    order = models.ForeignKey(Order, on_delete=models.SET_NULL, null=True)
    quantity = models.IntegerField('Кількість', default=0, null=True, blank=True)
    date_added = models.DateTimeField('Дата', auto_now_add=True)

    @property
    def get_total(self):
        total = self.product.price * self.quantity
        return total
```

Рисунок 3.11 – Створення моделі OrderItem

```
class ShippingInformation(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.SET_NULL, null=True)
    order = models.ForeignKey(Order, on_delete=models.SET_NULL, null=True)
    city = models.CharField('Область', max_length=200, null=False)
    state = models.CharField('Населений пункт', max_length=200, null=False)
    address = models.CharField('Адреса', max_length=200, null=False)
    zip_code = models.CharField('Поштовий індекс', max_length=200, null=False)
    date_added = models.DateTimeField('Дата', auto_now_add=True)

    def __str__(self):
        return self.city
```

Рисунок 3.12 – Створення моделі ShippingInformation

```
class Receipt(models.Model):
    order = models.ForeignKey(Order, on_delete=models.SET_NULL, null=True)
    customer = models.ForeignKey(Customer, on_delete=models.SET_NULL, null=True)
    total_price = models.IntegerField('Сума', null=False)
    datetime = models.DateTimeField('Дата', auto_now_add=True)
```

Рисунок 3.13 – Створення моделі Receipt

Далі за допомогою команди makemigrations треба створити міграції (рис. 3.14). Міграції використовують для переносу змін в моделях (додавання полей, видалення моделі і т.д.) на структуру бази даних.

```
python manage.py makemigrations
```

Рисунок 3.14 – Створення міграцій

Після виконання цієї команди у папці migrations з’явиться файл 0001_initial.py, у якому буде створена міграція. Далі треба запустити міграції за допомогою команди migrate (рис. 3.15).

```
python manage.py migrate
```

Рисунок 3.15 – Запуск міграцій

Було проведено налаштування бази даних, створені та виповнені міграції. Можна переходити до подальшої реалізації сайту.

Фрагмент коду додавання товарів до кошику користувачем та внесення особистих даних та даних про доставку при оформленні замовлення зображено на рис. 3.16.

```
import json
from . models import *

def cookieCart(request):
    try:
        cart = json.loads(request.COOKIES['cart'])
    except:
        cart = {}

    items = []
    order = {'get_cart_total': 0, 'get_items_total': 0, 'get_cart_items': 0, 'shipping': True}
    cartItems = order['get_cart_items']
    for i in cart:
        try:
            cartItems += cart[i]['quantity']
            product = Product.objects.get(id=i)
            total = (product.price * cart[i]['quantity'])

            order['get_cart_total'] += total
            order['get_cart_items'] += cart[i]['quantity']
```

```
            item = {
                'product': {
                    'id': product.id,
                    'title': product.title,
                    'model': product.model,
                    'price': product.price,
                    'stock': product.stock,
                },
                'quantity': cart[i]['quantity'],
                'get_total': total
            }
            items.append(item)

            if product.stock is False:
                order['shipping'] = False

        except:
            pass

    return {'cartItems': cartItems, 'order': order, 'items': items}


def cartData(request):
    if request.user.is_authenticated:
        customer = request.user.customer
        order, created = Order.objects.get_or_create(customer=customer, complete=False)
        items = order.orderitem_set.all()
        cartItems = order.get_cart_items

    else:
        cookieData = cookieCart(request)
        cartItems = cookieData['cartItems']
        items = cookieData['items']
        order = cookieData['order']

    return {'cartItems': cartItems, 'order': order, 'items': items}

def guestOrder(request, data):
    name = data['form']['name']
    surname = data['form']['surname']
    email = data['form']['email']
    phone = data['form']['phone']

    cookieData = cookieCart(request)
    items = cookieData['items']
    Customer.objects.get_or_create(
        name=data['form']['name'],
        surname=data['form']['surname'],
        phone=data['form']['phone'],
        email=data['form']['email'],
    )
    customer, created = Customer.objects.get_or_create(
        name=name,
        email=email,
        surname=surname,
        phone=phone,
    )
    customer.save()

    order = Order.objects.create(
        customer=customer,
        complete=False,
```

```
            )
            for item in items:
                product = Product.objects.get(id=item['product']['id'])
                model = Category.objects.get(id=product.id)
                orderItem, created = OrderItem.objects.get_or_create(order=order, product=product,
quantity=item['quantity'])
                orderItem.model = model
                orderItem.save()

            return customer, order
```

Рисунок 3.16 – Створення замовлення

Фрагмент коду формування замовлення, яке додається до бази даних, та створення чеку зображено на рис. 3.17.

```
        def processOrder(request):
            transaction_id = datetime.datetime.now().timestamp()
            data = json.loads(request.body)

            if request.user.is_authenticated:
                customer = request.user.customer
                order, created = Order.objects.get_or_create(customer=customer, complete=False)

            else:
                customer, order = guestOrder(request, data)

            total = data['form']['total']
            order.transaction_id = transaction_id

            if int(total) == order.get_cart_total:
                order.complete = True
            order.save()

            if order.shipping == True:
                ShippingInformation.objects.create(
                    customer=customer,
                    order=order,
                    address=data['shipping']['address'],
                    city=data['shipping']['city'],
                    state=data['shipping']['state'],
                    zip_code=data['shipping']['zip_code'],
                )

                Receipt.objects.create(
                    customer=customer,
                    order=order,
                    total_price=int(data['form']['total'])
                )

                global cust
                cust = customer

            return JsonResponse('Payment complete!', safe=False)


        def receipt(request):
```

```
            orders = Order.objects.all()

            receipt = Receipt.objects.get(order_id=orders[::-1][0])
            items = OrderItem.objects.filter(order_id=orders[::-1][0])
            customer = Customer.objects.get(id=receipt.customer_id)

            return render(request, 'main/receipt.html', {'title': 'Чек', 'items': items, 'customer': customer, 'receipt':
receipt})
```

Рисунок 3.17 – Обробка замовлення

3.3 Реалізація сторінок

Прикладом опису реалізації сторінок буде розглянута головна сторінка, на якій розташовані усі товари, які можна додати до кошику, переглянути їх характеристики та перейти до самого кошику.

Для початку у папці main створено папку templates, потім у цій папці створено папку з назвою main, яка буде містити усі html файли.

Ще однією зручною річчю Django є розширення шаблонів. Це означає, що можна використовувати ті самі блоки HTML-коду для різних частин вебсайту. Так не доведеться повторюватися кожного разу, коли потрібно використовувати ту саму інформацію або структуру. Якщо з'явиться необхідність змінити щось, не потрібно вносити правки в кожну сторінку, достатньо скоригувати шаблон. Так у папці templates було створено базовий шаблон base.html, котрий містить навігаційне меню, та JavaScript функцію отримання cookie, csrftoken та елементи кошику для певного користувача (Додаток А).

Далі було створено файл index.html (Додаток Б), який і буде головною сторінкою (див. рис. 3.18).

Сторінка вже має налаштовані стилі за допомогою фреймворку Bootstrap, який використовується веброзробниками для швидкої верстки адаптивних дизайнів сайтів та вебдодатків.

Для доступу до об’єктів products, categories та cartItems прописано вигляд (view) у файлі views.py (див. рис. 3.19).

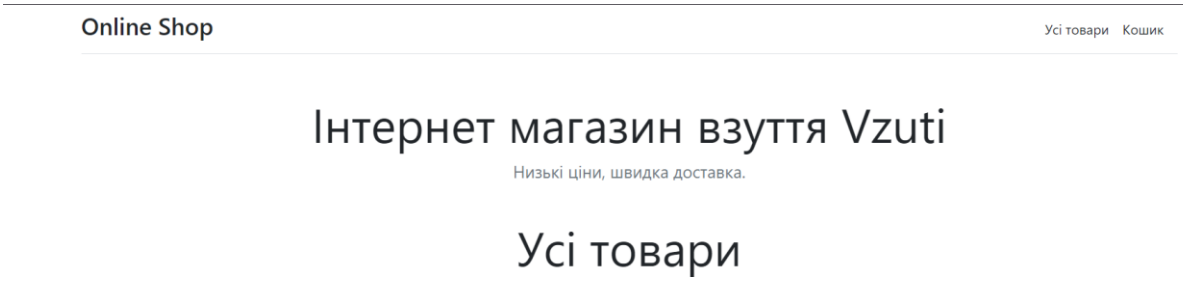

Рисунок 3.18 – Головна сторінка

```
        def index(request):
            products = Product.objects.all()
            categories = Category.objects.all()

            data = cartData(request)
            cartItems = data['cartItems']

            return render(request, 'main/index.html', {'title': 'Главная страница', 'products': products, 'categories': categories, 'cartItems': cartItems})
```

Рисунок 3.19 – View головної сторінки

Далі на сторінку додано товари та кнопки, за допомогою яких товари можна буде додати до кошику (рис. 3.20). Повний код сторінки можна побачити у Додатку Б.

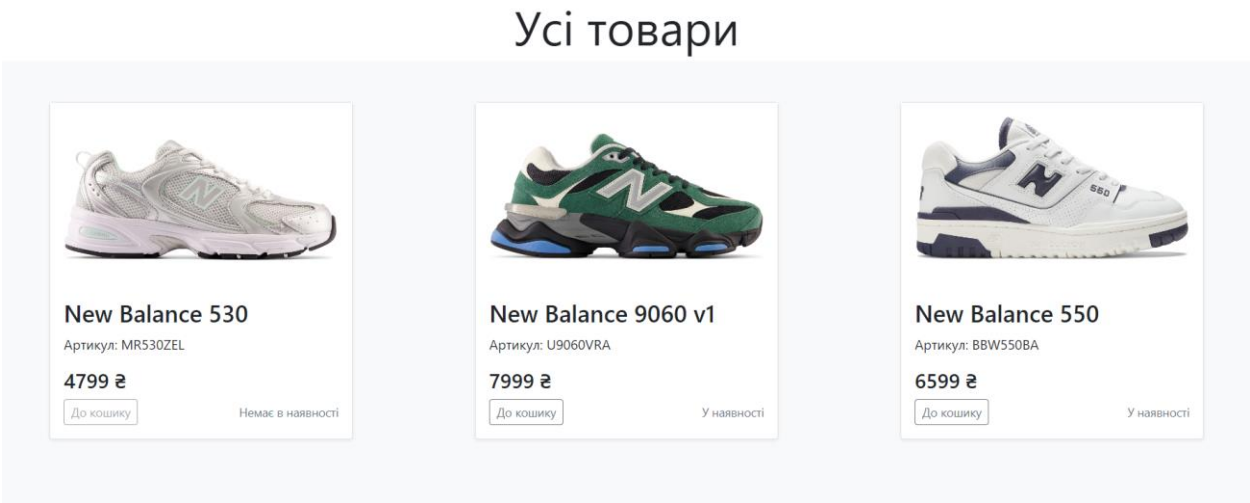

Рисунок 3.20 – Товари на головній сторінці

Висновки до розділу 3

Отже, було проведено первинне налаштування проєкту та перший запуск локального серверу. Були прописані моделі, створені та запущені міграції, завдяки яким була реалізована база даних. За допомогою технологій HTML та CSS фреймворку Bootstrap було розроблено сторінки вебсайту «Інтернет-магазин взуття», а прикладом опису реалізації сторінок було розглянуто головну сторінку. Програмну реалізацію уявлень (views), адреси сторінок (urls) та усіх сторінок сайту можна побачити у Додатках А – Ж.

4 ОПИС ІНТЕРФЕЙСУ СИСТЕМИ

4.1 Тестування сайту

Результатом роботи є вебсайт «Інтернет-магазин взуття» з використанням Django. Сайт складається з чотирьох сторінок – головна сторінка (рис. 4.1 та 4.3), кошик (рис. 4.4), сторінка замовлення (рис. 4.5, 4.6) та сторінка з чеком (рис.4.7).

Заходячи на сайт користувач бачить головну сторінку сайту (рис. 4.1). На ній він бачить перелік товарів, їхні фото та назву, кнопку «До кошику».

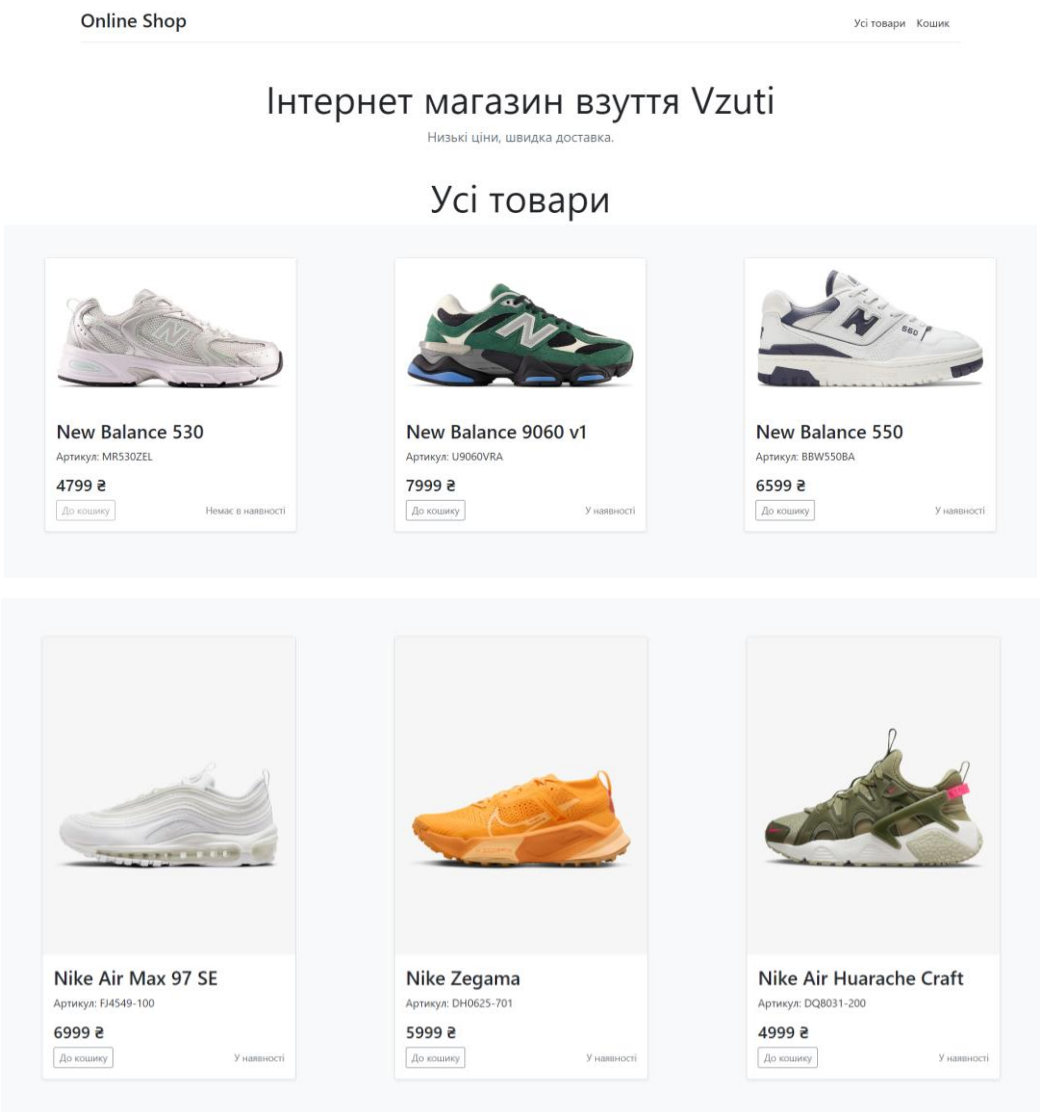

Рисунок 4.1 – Головна сторінка

Користувач має можливість додати товар до кошику, натиснувши кнопку «До кошику». Після цього на сторінці кошика з’являються додані товари, які можна прибрати або додати кількість товарів (рис. 4.2).

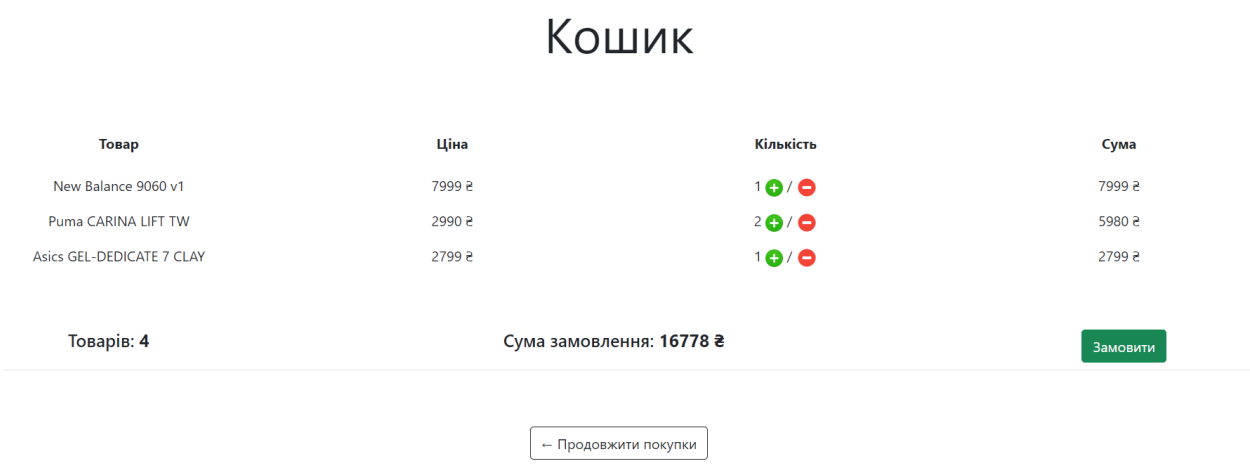

Рисунок 4.2 – Кошик

Далі користувач натискає кнопку «Замовити» й переходить до сторінки оформлення замовлення, де бачить форму (рис. 4.3), в яку треба внести особисті дані та дані про доставку (див. рис. 4.4).

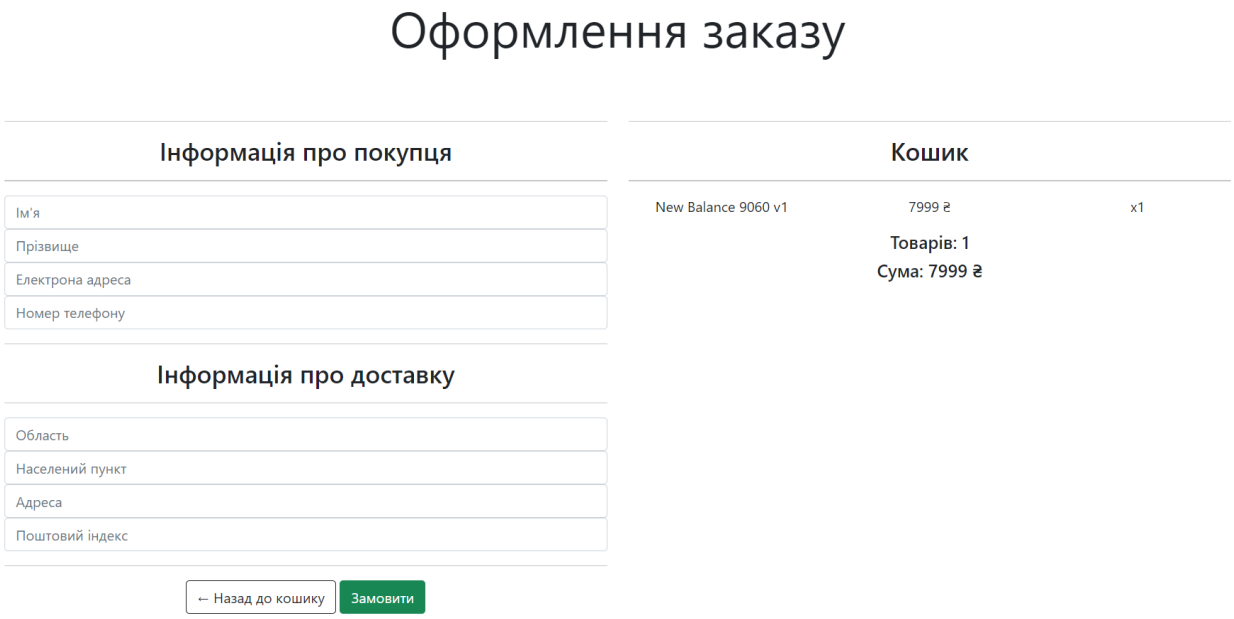

Рисунок 4.3 – Сторінка замовлення

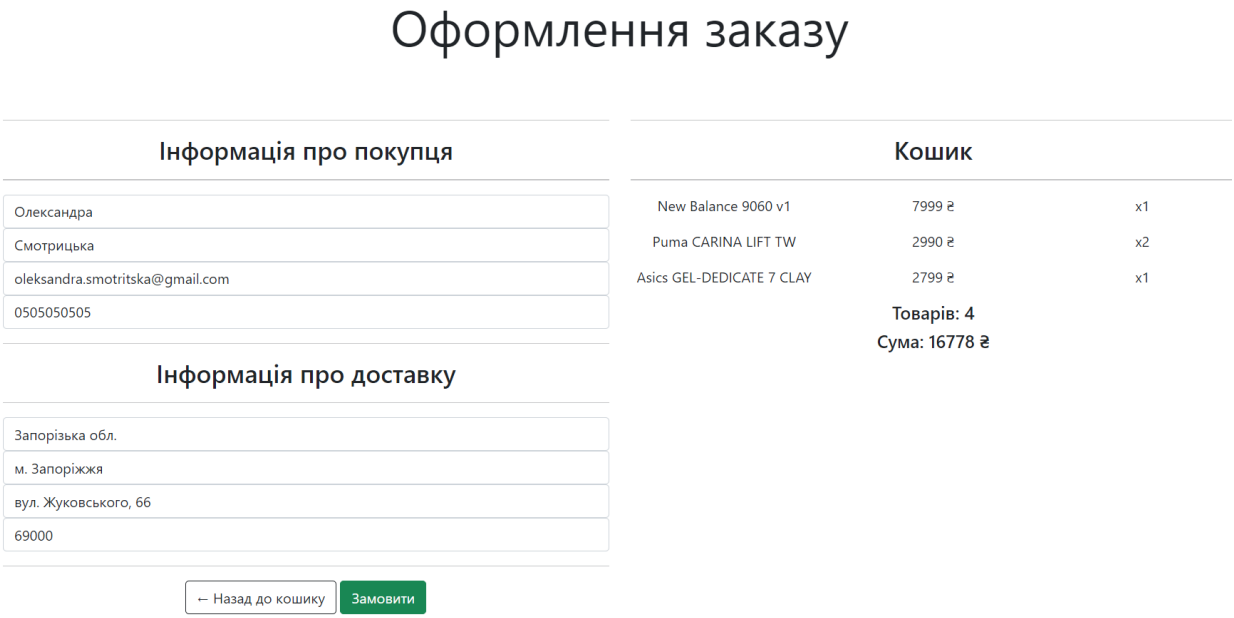

Рисунок 4.4 – Заповнена форма замовлення

Після того як користувач вніс необхідні дані та натиснув кнопку «Замовити» він бачить сторінку з чеком (рис. 4.5).

Рисунок 4.5 – Чек замовлення

Висновки до розділу 4

У цьому розділі було розглянуто усі сторінки вебсайту «Інтернет-магазин взуття», а саме головна сторінка, кошик, сторінка замовлення та сторінка з чеком.

ВИСНОВКИ

Отже, в результаті виконання кваліфікаційної роботи був розроблений вебсайт «Інтернет-магазин взуття» з використанням Django. Для зберігання інформації було обрано СКБД SQLite. За допомогою технологій HTML, фреймворку CSS – Bootstrap та JavaScript було розроблено інтерфейс прокту.

Згідно поставлених завдань було проаналізовано функціональні та технічні вимоги інтернет-магазину взуття. В результаті чого було побудовано Use-case діаграму та описано її прецеденти. Було побудовано діаграму розгортання та діаграми послідовності основних бізнес-процесів системи, а саме оформлення та обробка замовлення.

Було проаналізовано предметну область та спроєктовано ER-діаграму для зберігання інформації в системі. Розгорнуто новий проєкт Django та реалізовано всі необхідні функції бізнес логіки у вигляді контролера.

Було виконано ручне тестування та складено опис інтерфейсу системи.

Вебсайт «Інтернет магазин взуття» досить функціональний прототип, але для реального використання, може бути вдосконалений.

ПЕРЕЛІК ПОСИЛАНЬ

1. Engebreth G., Sahu S. PHP 8 Basics: For Programming and Web Development. New York : Apress, 2022. 352 p.

2. Sinha S. Beginning Laravel: Build Websites with Laravel 5.8 2nd Edition. New York : Apress, 2019. 439 p.

3. Stauffer M. Laravel: Up & Running: A Framework for Building Modern PHP Apps. Sebastopol : O'Reilly Media, 2019. 552 p.

4. Amarni B., Langlet E. Symfony 4 (LTS) – Développez des sites web PHP structurés et performants. Rome : ENI, 2021. 533 p.

5. Mardan A. Practical Node.js: Building Real-World Scalable Web Apps 2nd Edition. New York : Apress, 2018. 300 p.

6. Biswas N. MERN Projects for Beginners: Create Five Social Web Apps Using MongoDB, Express.js, React, and Node. New York : Apress, 2021. 300 p.

7. Bell J., Magolan G., Guijarro D., Peretti A., Housley P. Nest.js: A Progressive Node.js Framework. London : Bleeding Edge Press, 2018. 408 p.

8. Жереб К. А. Підвищення продуктивності коду мовою PYTHON з використанням техніки переписувальних правил. *Проблеми програмування*. 2020. № 2-3. С. 115-125.

9. The web framework for perfectionists with deadlines. URL: https://www.djangoproject.com/ (дата звернення: 03.03.2023).

10. Django documentation. URL: https://docs.djangoproject.com/en/3.2/ (дата звернення: 03.03.2023).

11. Greenfeld D., Roy A. Two Scoops of Django: Best Practices For Django 1.11. Los Angeles : Two Scoops Press, 2017. 553 p.

12. Welcome to Flask – Flask Documentation (2.1.x). URL: https://flask.palletsprojects.com/en/2.1.x/ (дата звернення: 06.03.2023).

13. The Pyramid Web Framework – The Pyramid Web Framework v2.0.1. URL: https://docs.pylonsproject.org/projects/pyramid/en/latest/ (дата звернення:

06.03.2023).

14. Bottle: Python Web Framework – Bottle 0.13-dev documentation. URL: https://bottlepy.org/docs/dev/ (дата звернення: 06.03.2023).

15. Підручник з Umbrello UML Modeller. URL: https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-basics.html (дата звернення: 15.03.2023).

16. RATIONAL UNIFIED PROCESS. URL: http://ni.biz.ua/8/8_8/8_88513_RATIONAL-UNIFIED-PROCESS.html (дата звернення: 21.03.2023).

17. Silberschatz A. Database system concepts (6th ed.). New York : McGraw Hill, 2011. 1376 p.

18. Ramakrishnan R., Gehrke J. Database Management Systems (3rd ed.). New York : McGraw Hill, 2003. 1104 p.

19. Elmasri R., Navathe S. B. Fundamentals of Database Systems (7th ed.). London : Pearson, 2010. 1280 p.

20. Литвин В. В., Наум О. М., Висоцька В. А., Дверій М. В. Архітектура системи онлайн-туризму для пошуку та планування подорожей із урахуванням потреб користувача. Львів : Нац. Ун-т “Львівська політехніка”, 2019. 22 с.

ДОДАТОК А

Програмна реалізація шаблонного файлу HTML

```
      <!doctype html>
      <html lang=ru>
      <head>
          <meta charset="UTF-8">
          <meta name="viewport" content="width=device-width, user-scalable=no,
initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
          <meta http-equiv="X-UA-Compatible" content="ie=edge">
          <title>{% block title %} {% endblock %}</title>
          <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css
">
      </head>
      <body>
          <div class="container py-2">
              <header>
                  <div class="d-flex flex-column flex-md-row align-items-
center pb-2 mb-4 border-bottom">
                      <a href="/" class="d-flex align-items-center text-dark
text-decoration-none">
                          <h3>Online Shop</h3>
                      </a>

                      <nav class="d-inline-flex mt-2 mt-md-0 ms-md-auto">
                          <a class="me-3 py-2 text-dark text-decoration-none"
href="{% url 'home' %}">Усі товари</a>
                          <a class="me-3 py-2 text-dark text-decoration-none"
href="{% url 'cart' %}">Кошик</a>
                      </nav>
                  </div>
              </header>
          </div>
          {% block body %} {% endblock %}

          <script type="text/javascript">
              var user = '{{request.user}}'

              function getToken(name) {
                  let cookieValue = null;
                  if (document.cookie && document.cookie !== '') {
                      const cookies = document.cookie.split(';');
                      for (let i = 0; i < cookies.length; i++) {
                          const cookie = cookies[i].trim();
                          if (cookie.substring(0, name.length + 1) === (name +
'=')) {
                              cookieValue =
decodeURIComponent(cookie.substring(name.length + 1));
                              break;
                          }
                      }
                  }
                  return cookieValue;
              }
              const csrftoken = getToken('csrftoken');

              function getCookie(name) {
```

```
                var cookieArr = document.cookie.split(';');

                for(var i=0; i<cookieArr.length; i++) {
                    var cookiePair = cookieArr[i].split('=');

                    if (name === cookiePair[0].trim()) {
                        return decodeURIComponent(cookiePair[1]);
                    }
                }

                return null
            }

            var cart = JSON.parse(getCookie('cart'))
            if(cart == undefined) {
                cart = {}
                console.log('cart was created')
                document.cookie = 'cart=' + JSON.stringify(cart) +
';domain=;path=/'
            }
            console.log('cart', cart)
        </script>
    </body>
    </html>
```

ДОДАТОК Б

Програмна реалізація головної сторінки

```
      {% extends 'main/base.html' %}
      {% load static %}
      {% block title %}
      {{ title }}
      {% endblock %}

      {% block body %}

          <div class="pricing-header p-3 pb-md-4 mx-auto text-center">
              <h1 class="display-4 fw-normal">Інтернет магазин взуття
Vzuti</h1>
              <p class="fs-5 text-muted">Низькі ціни, швидка доставка.</p>
          </div>
          <h1 class="display-4 fw-normal text-center"> Усі товари </h1>
          <div class="row row-cols-sm-3">
              {% for element in products %}
                  {% for elem in categories%}
                      {% if element.model == elem.title %}
                          <div class="album p-5 bg-light">
                              <div class="container">
                                  <div class="col">
                                      <div class="card shadow-sm">
                                          <img src="{{ element.picture.url
}}">
                                          <div class="card-body">
                                              <h3 class="card-text">{{
element.title }} {{ element.model}}</h3>
                                              <p class="card-text">{{
elem.description }}</p>
                                              <h4 class="card-text">{{
element.price }} ₴</h4>
                                              <div class="d-flex justify-
content-between align-items-center">
                                                  <div class="btn-group">
                                                      {% if element.stock ==
True %}
                                                          <button data-
product="{{ element.id }}" data-model="{{ element.model }}" data-action="add"
class="btn btn-sm btn-outline-secondary add-btn update-cart">До
кошику</button>
                                                      {% endif %}

                                                      {% if element.stock ==
False %}
                                                          <button data-
product="{{ element.id }}" data-action="add" class="btn btn-sm btn-outline-
secondary add-btn update-cart disabled">До кошику</button>
                                                      {% endif %}

                                                  </div>
                                                  <small class="text-muted">
                                                      {% if element.stock ==
True %}
                                                          У наявності
                                                      {% endif %}
```

```
                                                                {% if element.stock ==
False %}
                                                                    Немає в наявності
                                                                {% endif %}
                                                            </small>
                                                        </div>
                                                    </div>
                                                </div>
                                            </div>
                                        </div>
                                    </div>
                                {% endif %}
                            {% endfor %}
                        {% endfor %}

                        {% load static %}
                        <script src="{% static 'js/cart.js' %}"> </script>
                    </div>
                {% endblock %}
```

ДОДАТОК В

Програмна реалізація сторінки кошика

```
      {% extends 'main/base.html' %}

      {% block title %}
      {{ title }}
      {% endblock %}

      {% block body %}
          <h1 class="display-4 fw-normal text-center"> Кошик </h1>
          <br><br><br>
          <div class="row row-cols-sm-4 text-center">
              <div><strong>Товар</strong></div>
              <div><strong>Ціна</strong></div>
              <div><strong>Кількість</strong></div>
              <div><strong>Сума</strong></div>
          </div>
          <br>
          {% for item in items %}
              <div class="row row-cols-sm-4 text-center">
                  <div>
                      <p>{{ item.product.title }} {{ item.product.model }}</p>
                  </div>
                  <div>
                      <p>{{ item.product.price }} ₴</p>
                  </div>
                  <div>
                      {{ item.quantity }}
                      {% load static %}
                      <img data-product="{{ item.product.id }}" data-
action="add" class="update-cart" src="{% static 'images/plus.png' %}"
height="20" width="20">
                      /
                      <img data-product="{{ item.product.id }}" data-
action="remove" class="update-cart" src="{% static 'images/minus.png' %}"
height="24" width="24">
                  </div>
                  <div>
                      <p>{{ item.get_total }} ₴</p>
                  </div>
              </div>
          {% endfor %}

          {% load static %}
          <script src="{% static 'js/cart.js' %}"> </script>

          <div class="row row-cols-sm-3 text-center">
              <div class="col-lg-12">
                  <div class="box-element">
                      <br><br>
                      <table class="table">
                          <tr>
                              <th><h5>Товарів: <strong>{{ order.get_cart_items
}}</strong></h5></th>
                              <th><h5>Сума замовлення: <strong>{{
order.get_cart_total }} ₴</strong></h5></th>
                              <th>
```

```
                                    <a id= "order-button" class="btn btn-
success" href="{% url 'order' %}">Замовити</a>
                                </th>
                            </tr>
                        </table>
                    </div>
                    <br><br>
                    <a class="btn btn-outline-dark" href="{% url 'home'
%}">&#x2190; Продовжити покупки</a>
                </div>
            </div>

            <script type="text/javascript">
                var cart_items = '{{ order.get_cart_total }}'
                if (cart_items === '0'){
                    document.getElementById('order-
button').classList.add('disabled');
                }
            </script>
        {% endblock %}
```

ДОДАТОК Г

Програмна реалізація сторінки замовлення

```
      {% extends 'main/base.html' %}

      {% load static %}

      {% block title %}
      {{ title }}
      {% endblock %}

      {% block body %}
          <h1 class="display-4 fw-normal text-center"> Оформлення заказу </h1>

           <div class="row row-cols-sm-3 text-center p-5">
                  <div class="col-lg-6">
                        <div id="form-wrapper">
                              <form id="form">
                                    <div id="user-info">
                               <hr>
                                          <h3>Інформація про покупця</h3>
                                          <hr>
                                          <div class="form-field">
                                                <input required class="form-
control" type="text" name="name" placeholder="Ім'я">
                                          </div>
                                          <div class="form-field">
                                                <input required class="form-
control" type="text" name="surname" placeholder="Прізвище">
                                          </div>
                                <div class="form-field">
                                          <input required class="form-
control" type="email" name="email" placeholder="Електрона адреса">
                                          </div>
                                          <div class="form-field">
                                                <input required class="form-
control" type="text" name="phone" placeholder="Номер телефону">
                                          </div>
                                    </div>

                                    <div id="shipping-info">
                                          <hr>
                                          <h3>Інформація про доставку</h3>
                                          <hr>
                                          <div class="form-field">
                                                <input required class="form-
control" type="text" name="city" placeholder="Область">
                                          </div>
                                          <div class="form-field">
                                                <input required class="form-
control" type="text" name="state" placeholder="Населений пункт">
                                          </div>
                                          <div class="form-field">
                                                <input required class="form-
control" type="text" name="address" placeholder="Адреса">
                                          </div>
                                          <div class="form-field">
```

```
                                                <input required class="form-
control" type="text" name="zip_code" placeholder="Поштовий індекс">
                                          </div>
                                    </div>

                                    <hr>
                              <a class="btn btn-outline-dark" href="{% url 'cart'
%}">&#x2190; Назад до кошику</a>
                                    <input id="form-button" class="btn btn-
success btn-block" type="submit" value="Замовити">

                              </form>
                        </div>
                        <br>
                        <div class= "d-none" id="payment-info">
                              <small>Оплата</small>
                              <button id="make-payment">Сплатити</button>
                        </div>

                  </div>
                  <div class="col-lg-6">
                        <div class="box-element">
                              <hr>
                              <h3>Кошик</h3>
                              <hr>
                              {% for item in items %}
                                    <div class="row row-cols-sm-3 text-
center">
                                          <div><p>{{ item.product.title }} {{
item.product.model }}</p></div>
                                          <div><p>{{ item.product.price }}
₴</p></div>
                                          <div><p>x{{ item.quantity
}}</p></div>
                                    </div>
                              {% endfor %}
                              <h5>Товарів: {{ order.get_cart_items }}</h5>
                              <h5>Сума: {{ order.get_cart_total }} ₴</h5>
                        </div>
                  </div>
            </div>

            <script type="text/javascript">
                  var shipping = '{{ order.shipping }}'
                  var user = '{{ request.user }}'
                  var total = '{{ order.get_cart_total }}'

                  if (shipping === 'False') {
                        document.getElementById('shipping-info').innerHTML =
'Товару немає у наявності'
                        document.getElementById('user-info').innerHTML = ''
                        document.getElementById('payment-info').innerHTML = ''
                        document.getElementById('form-
button').classList.add('d-none');
                  }

                  if (user !== 'AnonymousUser') {
                        document.getElementById('user-info').innerHTML = ''
                  }

                  if (shipping === 'False' && user !== 'AnonymousUser') {
                        document.getElementById('form-
wrapper').classList.add('d-none');
```

```
                        document.getElementById('payment-
info').classList.remove('d-none');
                  }

                  var form = document.getElementById('form')
                  form.addEventListener('submit', function(e){
                        e.preventDefault()
                        console.log('Form Submitted...')
                        document.getElementById('form-
button').classList.add('d-none');
                        document.getElementById('payment-
info').classList.remove('d-none');
                  })

                  document.getElementById('make-
payment').addEventListener('click', function(e){
                        submitFormData()
                  })

                  function submitFormData() {
                        console.log('Payment data clicked')

                        var userFormData = {
                              'name': null,
                              'surname': null,
                              'email': null,
                              'phone': null,
                              'total': total,
                        }

                        var shippingInfo = {
                              'city': null,
                              'state': null,
                              'address': null,
                              'zip_code': null,
                        }

                        if (shipping !== 'False') {
                              shippingInfo.address = form.address.value
                              shippingInfo.city = form.city.value
                              shippingInfo.state = form.state.value
                              shippingInfo.zip_code = form.zip_code.value
                        }

                        if (user === 'AnonymousUser') {
                              userFormData.name = form.name.value
                              userFormData.surname = form.surname.value
                              userFormData.email = form.email.value
                              userFormData.phone = form.phone.value
                        }

                        var url = '/process_order/'
                        fetch(url, {
                              method: 'POST',
                              headers: {
                                    'Content-Type': 'application/json',
                                    'X-CSRFToken': csrftoken,
                              },
                              body: JSON.stringify({'form': userFormData,
'shipping': shippingInfo})
                        })
                                    .then((response) => response.json())
                                    .then((data) => {
```

```
                                        console.log('Success:', data);
                                        alert('Transaction completed');
                                        cart = {}
                                        document.cookie = 'cart=' +
JSON.stringify(cart) + ';domain=;path=/'

                                        window.location.href = '{% url
'receipt' %}'
                                    })
                            console.log('shipping info:', shippingInfo)
                            console.log('user info:', userFormData)
                    }
                </script>

        {% endblock %}
```

ДОДАТОК Д

Програмна реалізація сторінки з чеком

```
      {% extends 'main/base.html' %}

      {% load static %}

      {% block title %}
      {{ title }}
      {% endblock %}

      {% block body %}
      <h1 class="display-4 fw-normal text-center"> Дякуємо за замовлення </h1>
      <br>
      <h1 class="fw-normal text-center"> Чек </h1>

      <div class="fw-normal text-center">
          Дані покупця
          <br>
          Ім'я: {{customer.name}}
          <br>
          Прізвище: {{customer.surname}}
          <br>
          Електрона адреса: {{customer.email}}
          <br>
          Номер телефону: {{customer.phone}}
          <br><br>
          Товари
          <br>
          {% for i in items %}
              {{i.product}}
              {{i.product.model}}
              ..... x{{i.quantity}}
              <br>
          {% endfor %}

          <br>
          Дата замовлення
          <br>
          {{receipt.datetime}}
          <br><br>
          Сума замовлення
          <br>
          {{ receipt.total_price }} ₴
          <br><br><br>
          <a class="btn btn-outline-dark" href="{% url 'home' %}">На
головну</a>
      </div>

      {% endblock %}
```

ДОДАТОК Е

Уявлення проєкту (views)

```
      from django.shortcuts import render
      from django.http import JsonResponse
      from .models import *
      from . utils import cookieCart, cartData, guestOrder
      import json
      import datetime

      def index(request):
          products = Product.objects.all()
          categories = Category.objects.all()

          data = cartData(request)
          cartItems = data['cartItems']

          return render(request, 'main/index.html', {'title': 'Главная
страница', 'products': products, 'categories': categories, 'cartItems':
cartItems})

      def cart(request):
          data = cartData(request)

          items = data['items']
          order = data['order']
          cartItems = data['cartItems']

          return render(request, 'main/cart.html', {'title': 'Корзина',
'items': items, 'order': order, 'cartItems': cartItems})

      def products(request):
          return render(request, 'main/products.html')

      def order(request):
          data = cartData(request)

          items = data['items']
          order = data['order']
          cartItems = data['cartItems']

          return render(request, 'main/order.html', {'title': 'Оформление
заказа', 'items': items, 'order': order, 'cartItems': cartItems})

      def updateItem(request):
          data = json.loads(request.body)
          productId = data['productID']
          action = data['action']
          customer = request.user.customer

          product = Product.objects.get(id=productId)
          order, created = Order.objects.get_or_create(customer=customer,
complete=False)
          orderItem, created = OrderItem.objects.get_or_create(order=order,
product=product)
          model = Category.objects.get(id=productId)

          if action == 'add':
```

```
            orderItem.quantity = (orderItem.quantity + 1)
        elif action == 'remove':
            orderItem.quantity = (orderItem.quantity - 1)

        orderItem.model = model
        orderItem.save()

        if orderItem.quantity <= 0:
            orderItem.delete()

        return JsonResponse('Item was added', safe=False)

    def processOrder(request):
        transaction_id = datetime.datetime.now().timestamp()
        data = json.loads(request.body)

        if request.user.is_authenticated:
            customer = request.user.customer
            order, created = Order.objects.get_or_create(customer=customer,
complete=False)
        else:
            customer, order = guestOrder(request, data)

        total = data['form']['total']
        order.transaction_id = transaction_id

        if int(total) == order.get_cart_total:
            order.complete = True
        order.save()

        if order.shipping == True:
            ShippingInformation.objects.create(
                customer=customer,
                order=order,
                address=data['shipping']['address'],
                city=data['shipping']['city'],
                state=data['shipping']['state'],
                zip_code=data['shipping']['zip_code'],
            )

            Receipt.objects.create(
                customer=customer,
                order=order,
                total_price=int(data['form']['total'])
            )

            global cust
            cust = customer

        return JsonResponse('Payment complete!', safe=False)

    def receipt(request):
        orders = Order.objects.all()

        receipt = Receipt.objects.get(order_id=orders[::-1][0])
        items = OrderItem.objects.filter(order_id=orders[::-1][0])
        customer = Customer.objects.get(id=receipt.customer_id)

        return render(request, 'main/receipt.html', {'title': 'Чек',
'items': items, 'customer': customer, 'receipt': receipt})
```

ДОДАТОК Ж

Адреси сторінок (urls)

```
      from django.urls import path
      from django.contrib import admin
      from . import views

      from django.conf.urls.static import static
      from django.conf import settings

      urlpatterns = [
          path('admin/', admin.site.urls),
          path('', views.index, name='home'),
          path('cart', views.cart, name='cart'),
          path('order', views.order, name='order'),
          path('update_item/', views.updateItem, name='update_item'),
          path('process_order/', views.processOrder, name='process_order'),
          path('receipt/', views.receipt, name='receipt')
      ]

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