

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

«Веб-сайт з продажу мобільних пристроїв та аксесуарів»

Виконав студент групи ПЗ-19-3
НАКОНЕЧНИЙ Олексій Володимирович

Керівник роботи:
КРАВЧЕНКО Світлана Миколаївна

Рецензент:
ЧИЖМОТРЯ Олексій Володимирович

Житомир – 2023

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

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«02» серпня 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **НАКОНЕЧНИЙ Олексій Володимирович**

Керівник роботи: **КРАВЧЕНКО Світлана Миколаївна**

Тема роботи: **«Веб-сайт з продажу мобільних телефонів та аксесуарів»**
затверджена Наказом закладу вищої освіти від **«02» серпня 2023 р., №326с**

Вихідні дані для роботи: **методи та засоби розробки веб-додатків, аналіз проблематики веб-сайту з продажу мобільних телефонів та аксесуарів, даний сайт має відповідати сучасному рівню наявних технологій та бути зручним у використанні.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Кравченко С.М.	02.08.2023р.	02.08.2023р.
2	Кравченко С.М.	08.08.2023р.	08.08.2023р.
3	Кравченко С.М.	13.08.2023р.	13.08.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі	2 серпня 2023 – 4 серпня 2023	Виконано
2	Пошук, огляд та аналіз аналогічних розробок	4 серпня 2023 – 7 серпня 2023	Виконано
3	Формулювання технічного завдання та опрацювання літературних джерел	8 серпня 2023 – 12 серпня 2023	Виконано
4	Проектування структури системи	13 серпня 2023 – 15 серпня 2023	Виконано
5	Написання програмного коду	16 серпня 2023 – 22 серпня 2023	Виконано
6	Тестування роботи додатку	22 серпня 2023 – 24 серпня 2023	Виконано
7	Попередній захист	27 серпня 2023	Виконано

РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплексу сайту з продажу мобільних телефонів та аксесуарів і пояснювальної записки. Пояснювальна записка до випускної роботи містить 72 сторінки, 36 ілюстрацій та 1 таблицю.

Ця кваліфікаційна робота присвячена розробці веб-сайту для продажу мобільних телефонів та аксесуарів з використанням фреймворку Django. Вона розділена на три основних частини: теоретичний огляд, методологія розробки та тестування веб-додатку.

У першому розділі проведено аналіз аналогічних програмних продуктів, визначено архітектуру веб-додатку та вибрано інструменти та технології для його реалізації. Проведено вибір мови програмування, системи керування базами даних та фреймворку.

Другий розділ охоплює методологію та етапи розробки веб-сайту. Описано методи об'єктно-орієнтованого аналізу та проектування, зокрема діаграми варіантів використання, діяльності та послідовності. Розглянуто розробку бази даних, дизайн і верстку інтернет-магазину.

У третьому розділі детально описано розробку серверної частини веб-сайту на основі фреймворку Django та детально розглянуто процес тестування веб-сайту, включаючи ручне та автоматизоване тестування функціональності та сторінок. Звернута увага на тестування сторінок адміністрування, зокрема на додавання товару.

У висновках підсумовано досягнуті результати, важливі вибори під час розробки та основні аспекти тестування веб-додатку. Ця робота вносить важливий внесок у розробку інтернет-магазинів на основі Django, забезпечуючи структуровану архітектуру та надійне тестування.

КЛЮЧОВІ СЛОВА: ВЕБ-САЙТ, ІНТЕРНЕТ-МАГАЗИН, DJANG

					ІПЗ.КР.Б – 121 – 23 – ПЗ						
Зм.	Арк.	№ докум.	Підпис	Дата	Пояснювальна записка Розробка «Веб-сайт з продажу мобільних пристроїв та аксесуарів»			Літ.	Арк.	Аркуші	
Розроб.		О. Наконечний									
Керівник		С. Кравченко								4	73
Рецензент.		О. Чижмотря									
Зав. каф.		А. Морозов									
					Житомирська політехніка, група ІПЗ-19-3						

ABSTRACT

The bachelor's thesis consists of a software package for a website selling mobile phones and accessories and an explanatory note. The explanatory note to the final work contains 72 pages, 36 illustrations and 1 table.

This qualification work is devoted to the development of a website for the sale of mobile phones and accessories using the Django framework. It is divided into three main parts: theoretical overview, development methodology and testing of the web application.

The first section analyzes similar software products, defines the architecture of the web application, and selects tools and technologies for its implementation. The choice of a programming language, database management system, and framework is made.

The second section covers the methodology and stages of website development. Methods of object-oriented analysis and design are described, including use case, activity, and sequence diagrams. The development of a database, design and layout of an online store are also covered.

The third section describes in detail the development of the server side of the website based on the Django framework and discusses in detail the process of testing the website, including manual and automated testing of functionality and pages. Attention is paid to the testing of administration pages, in particular, to adding a product.

The conclusions summarize the results achieved, important choices during development, and the main aspects of testing a web application. This work makes an important contribution to the development of Django-based online stores by providing a structured architecture and reliable testing.

KEYWORDS: WEBSITE, ONLINE STORE, DJANGO, DEVELOPMENT.

					ІІЗ.КР.Б – 121 – 23 – ІІЗ						
Зм.	Арк.	№ докум.	Підпис	Дата	Пояснювальна записка Розробка «Веб-сайт з продажу мобільних пристроїв та аксесуарів»	Літ.		Арк.	Аркуші		
Розроб.		О. Наконечний						4	73		
Керівник		С. Кравченко									
Рецензент.		О. Чижмoтря									
Зав. каф.		А. Морозов									
						Житомирська політехніка, група ІІІЗ-19-3					

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП.....	8
РОЗДІЛ 1. ТЕОРЕТИЧНИЙ ОГЛЯД ТА АНАЛІЗ СУЧАСНИХ РІШЕНЬ У ГАЛУЗІ РОЗРОБКИ САЙТУ ДЛЯ ПРОДАЖУ МОБІЛЬНИХ ТЕЛЕФОНІВ ТА АКСЕСУАРІВ	10
1.1. Постановка задачі	10
1.2 Аналіз аналогічних програмних продуктів.....	11
1.3 Визначення архітектури веб-додатку	16
1.4 Аналіз інструментів та технологій для створення веб-сайту	18
1.4.1 Вибір мови програмування	18
1.4.2 Вибір системи керування базами даних	21
1.4.3 Вибір фреймворку	25
Висновки до розділу 1	28
РОЗДІЛ 2. МЕТОДОЛОГІЯ ТА ЕТАПИ РОЗРОБКИ САЙТУ З ПРОДАЖУ МОБІЛЬНИХ ТЕЛЕФОНІВ ТА АКСЕСУАРІВ	30
2.1 Методи та засоби об'єктно-орієнтованого аналізу і проектування	30
2.1.1 Діаграма варіантів використання.....	30
2.1.2 Діаграма діяльності (activitydiagram)	32
2.1.3 Діаграма послідовності (Sequencediagram)	34
2.1.4 Діаграма класів	35
2.2. Розробка бази даних	36
2.3. Розробка дизайну інтернет-магазину	39
2.4 Верстка сайту по створеному дизайну	44
Висновки до розділу 2.....	47
РОЗДІЛ 3. РОЗРОБКА ТА ТЕСТУВАННЯ ВЕБ-САЙТУ НА ОСНОВІ DJANGO	49

3.1 Структура веб-сайту.....	49
3.2 Програмування серверної частини веб-сайту	50
3.3. Тестування роботи веб-сайту.....	57
Висновки до розділу 3	62
ВИСНОВКИ	63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	64
ДОДАТКИ	66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AJAX - асинхронний JavaScript та XML

API - інтерфейс програмування додатків

CMS - система управління вмістом

CRUD - створення, читання, оновлення та видалення даних

CSS - каскадні таблиці стилів

IDE - інтегроване середовище розробки

JSON - обмін даними у форматі JavaScript

MVC - модель-вид-контролер (шаблон архітектури ПЗ)

ORM - об'єктно-реляційний мапер (для роботи з базами даних)

SVG - масштабована векторна графіка

UI - інтерфейс користувача

UX - користувацький досвід

БД – база даних

ПЗ - програмне забезпечення

ВСТУП

Актуальність теми. Сучасний темп життя заповнений технологічними новинками, що змінюють наше сприйняття та підходи в багатьох галузях. Однією з найвпливовіших сфер, яка безсумнівно відчуває це, є електронна комерція. Зростання популярності онлайн-торгівлі, зокрема продажу мобільних телефонів та аксесуарів, вимагає ефективних інструментів для створення та керування інтернет-магазинами. У цьому контексті виникає важливе завдання - розробити зручний веб-сайт для продажу мобільних пристроїв та додаткових аксесуарів.

Метою кваліфікаційної роботи є розробка та реалізація веб-сайту з продажу мобільних телефонів та аксесуарів з використанням мов програмування Python та фреймворку Django. Основним завданням є створення зручного та функціонального інтернет-магазину, який допоможе компанії привертати клієнтів та ефективно управляти продажами та запасами.

Об'єктом дослідження є процес створення та розвитку веб-сайту для продажу мобільних телефонів та аксесуарів. Дослідження включає в себе аспекти структури сайту, розробки функціональності для додавання товарів, оформлення замовлень, створення кошика покупок та інші аспекти, що стосуються ефективної роботи інтернет-магазину.

Предметом дослідження є взаємодія користувачів з веб-сайтом для продажу мобільних телефонів та аксесуарів, заснована на технологіях Python, Django, SQLite та Bootstrap. Розглядається представлення товарів, можливості вибору, оформлення замовлень та інші аспекти, які визначають зручність використання веб-сайту як для користувачів, так і для власників бізнесу.

Ця кваліфікаційна робота націлена на багатогранний підхід до створення веб-сайту для продажу мобільних телефонів та аксесуарів з метою створення зручного, привабливого та функціонального інтернет-магазину, який відповідає сучасним вимогам електронної комерції. Це буде робочий продукт, який може бути використаний реальними бізнесами для розширення їхньої онлайн-присутності та збільшення обсягів продажів.

РОЗДІЛ 1. ТЕОРЕТИЧНИЙ ОГЛЯД ТА АНАЛІЗ СУЧАСНИХ РІШЕНЬ У ГАЛУЗІ РОЗРОБКИ САЙТУ ДЛЯ ПРОДАЖУ МОБІЛЬНИХ ТЕЛЕФОНІВ ТА АКСЕСУАРІВ

1.1. Постановка задачі

Кваліфікаційна робота полягає у створенні функціонального інтернет-магазину з продажу мобільних телефонів та аксесуарів. Головною метою завдання є створення ефективного та зручного веб-середовища, яке допоможе клієнтам здійснювати покупки з мінімальними перешкодами, а власникам магазину - управляти та вдосконалювати бізнес-процеси.

Для реалізації поставленого завдання, основними етапами є:

1. Визначення основних вимог до функціональності і дизайну інтернет-магазину, зокрема, опцій вибору товарів, оформлення замовлення та інші.
2. Оцінка можливих технічних рішень для розробки веб-сайту, зокрема вибір мов програмування, системи керування базами даних та фреймворку.
3. Визначення загальної архітектури веб-додатку, у тому числі розподілення функцій між серверною (backend) та клієнтською (frontend) частинами, а також проектування бази даних.
4. Розробка веб-сайту згідно з визначеними вимогами та архітектурою, включаючи реалізацію функцій вибору товарів, оформлення замовлення та інші.
5. Проведення тестів на виявлення помилок. Внесення виправлень та покращень з метою забезпечення високої якості та зручності використання.
6. Розгортання веб-сайту на локальному сервері та запуск.

Ці етапи є ключовими для успішної реалізації проекту з розробки сайту з продажу мобільних телефонів та аксесуарів, що відповідає сучасним вимогам електронної комерції та надасть зручний і привабливий інтерфейс для користувачів.

1.2 Аналіз аналогічних програмних продуктів

Для вивчення ринкового оточення та здійснення порівняльного аналізу проведемо огляд трьох відомих інтернет-магазинів: **rozetka.com.ua**, **mojo.ua** та **zhuk.ua**. Це дозволить отримати глибше розуміння найкращих практик у галузі продажу мобільних телефонів та аксесуарів та визначити основні аспекти, які впливають на їхню ефективність та популярність.

Rozetka є одним з найбільших та найвпливовіших інтернет-магазинів в Україні [1], який відомий своїм широким асортиментом товарів та високою якістю обслуговування клієнтів (рис. 1.1).

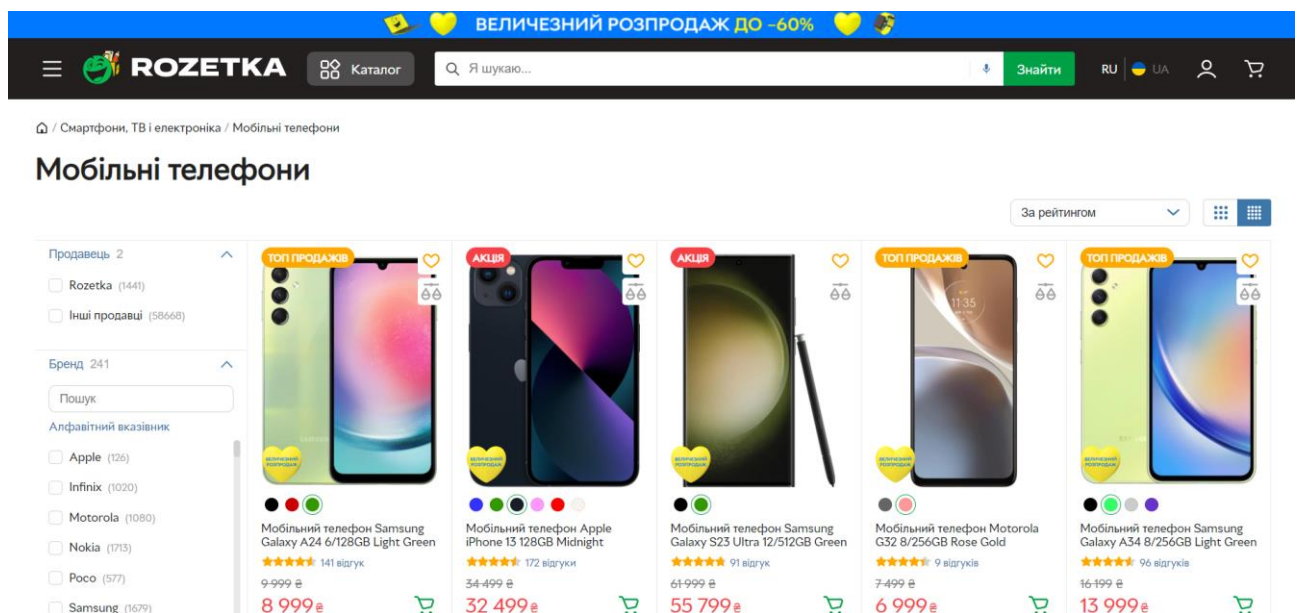


Рисунок 1.1 – Сторінка «Мобільні телефони» сайту rozetka.com.ua

Один з ключових аспектів, який вирізняє Rozetka, це потужний механізм пошуку та фільтрації товарів. Клієнти можуть швидко знаходити потрібні товари за допомогою різноманітних фільтрів, таких як бренд, ціновий діапазон, характеристики тощо. Це значно полегшує процес вибору та зменшує час, необхідний для знаходження підходящого товару.

Ще однією вагомою перевагою є наявність докладних оглядів та рецензій для багатьох товарів. Це дозволяє клієнтам отримати більш докладну інформацію про продукт перед покупкою. Реальні відгуки від інших покупців допомагають у прийнятті обґрунтованих рішень та врахуванні плюсів та мінусів товару.

Сайт ретельно організований з логічною структурою та зручною навігацією. Категорії товарів легко знаходяться на головній сторінці, а основні функції, такі як кошик та особистий кабінет, доступні на видному місці.

Rozetka часто надає акції, знижки та спеціальні пропозиції на певні товари, включаючи мобільні телефони та аксесуари. Це створює стимул для клієнтів здійснювати покупки та отримувати вигоду від купівлі.

Розуміючи важливість мобільності, Rozetka пропонує зручну та функціональну мобільну версію свого сайту. Це дозволяє клієнтам здійснювати покупки зі своїх смартфонів та планшетів, забезпечуючи комфортний досвід.

Команда підтримки Rozetka готова допомогти клієнтам з будь-якими питаннями або проблемами, що можуть виникнути під час покупки. Це підвищує рівень задоволеності клієнтів та робить їхні покупки більш приємними.

Rozetka.com.ua є прикладом успішного інтернет-магазину з великим досвідом у продажу мобільних телефонів та аксесуарів. Їхній акцент на зручному пошуку, якісних оглядах, а також забезпеченням зв'язку з клієнтами робить їх одним з лідерів електронної комерції в Україні.

Моюо.ua є відомим та популярним магазином електроніки та техніки в Україні [2], і вони також активно займаються продажем мобільних пристроїв та аксесуарів (рис. 1.2).

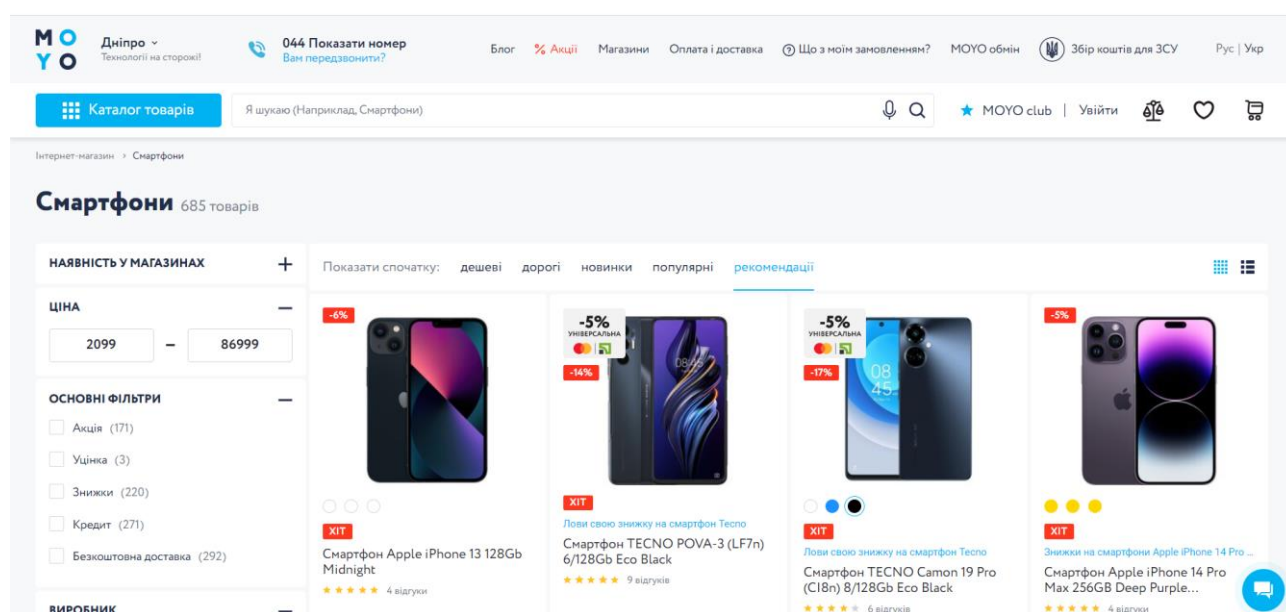


Рисунок 1.2 – Сторінка «Смартфони» сайту моюо.ua

Однією з важливих особливостей Мою є можливість створення особистого кабінету для зареєстрованих користувачів. Цей функціонал дозволяє клієнтам зберігати свої налаштування, адреси доставки, історію замовлень та інші дані. Це сприяє персоналізації досвіду покупця та полегшує процес оформлення замовлення.

Мою.ua використовує чітко структурований категорійний підхід для організації товарів. Це допомагає покупцям швидко знаходити потрібні товари, так як вони можуть переходити від однієї категорії до іншої за допомогою зручного меню.

Мою.ua регулярно пропонує акції, знижки та спеціальні пропозиції для різних категорій товарів, включаючи мобільні пристрої та аксесуари. Це збільшує інтерес споживачів до покупок та стимулює їхні вибори.

Мою.ua славиться якістю свого обслуговування та підтримки клієнтів. Вони надають можливість зв'язку через різні канали: телефон, електронну пошту, чат тощо. Це робить покупки більш впевненими та забезпечує додатковий рівень довіри.

Мою.ua також дбає про зручність мобільних користувачів, пропонуючи оптимізовану мобільну версію свого сайту. Це дозволяє клієнтам здійснювати покупки на ходу зі своїх мобільних пристроїв.

Мою.ua відомий своїм акцентом на особистий підхід до клієнтів, зручною навігацією та якісним сервісом. Їхні практики забезпечують високий рівень задоволеності клієнтів та роблять їхні покупки ефективними та комфортними.

Zhuk.ua є визначним представником спеціалізованих інтернет-магазинів, який зосереджується на продажі мобільних телефонів та гаджетів [3].

Однією з ключових рис Zhuk є його спеціалізація на мобільних пристроях. Це означає, що їхня команда має глибше розуміння продуктів, які вони продають, та може надати експертні поради клієнтам. Такий підхід сприяє збільшенню довіри споживачів.

Zhuk.ua відомий своєю активною політикою стосовно акцій та знижок. Вони регулярно пропонують спеціальні пропозиції на певні товари, що стимулює покупців до здійснення покупок та виявляє їхній інтерес.

Оскільки Zhuk спеціалізується на мобільних телефонах та гаджетах, вони можуть запропонувати широкий вибір товарів. Це дозволяє клієнтам вибирати серед різних брендів, моделей та характеристик (рис.1.3).

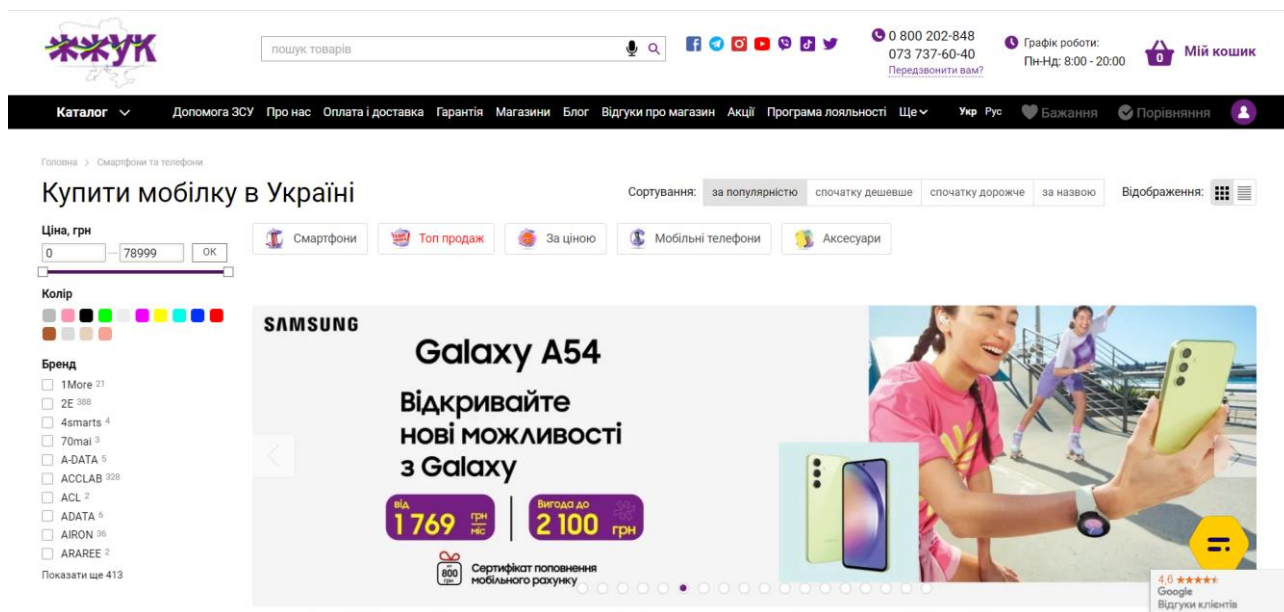


Рисунок 1.3 – Сторінка «Смартфони та телефони» сайту zhuk.ua

Сайт Zhuk.ua пропонує зручний пошук та навігацію, які допомагають клієнтам швидко знаходити потрібні товари. Це особливо важливо, оскільки при великому асортименті легкість знаходження є ключовою для збільшення конверсії.

Важливою рисою є також якість обслуговування. Zhuk.ua стежить за якістю комунікації з клієнтами, відповідає на їхні запитання та надає необхідну інформацію.

Zhuk.ua встановлює зв'язок з клієнтами через різні канали, включаючи телефон, чат та електронну пошту. Це забезпечує зручність для споживачів та можливість отримати допомогу.

Zhuk.ua вирізняється своєю спеціалізованою спрямованістю, акційними пропозиціями та глибоким асортиментом. Їхній фокус на мобільних пристроях роблять їх успішним гравцем у сегменті мобільної електроніки.

Порівняння ключових характеристик інтернет-магазинів rozetka.com.ua, moyo.ua та zhuk.ua наведено в таблиці 1.1.

Таблиця 1.1

Порівняння аналогічних програмних продуктів

Характеристика	Rozetka.com.ua	Moyo.ua	Zhuk.ua
Тип магазину	Загальний асортимент	Електроніка	Мобільні телефони
Зручна фільтрація	Так	Так	Так
Огляди та рецензії	Так	Так	Відсутність
Особистий кабінет	Так	Так	Так
Акції та знижки	Так	Так	Так
Спеціалізація	Загальний асортимент	Електроніка	Мобільні телефони
Широкий асортимент	Так	Так	Так
Якість обслуговування	Висока	Висока	Висока
Мобільна версія сайту	Так	Так	Так

З порівняльної таблиці 1.1 видно, що Rozetka.com.ua, Moyo.ua та Zhuk.ua мають свої унікальні риси та переваги. Rozetka.com.ua відзначається широким асортиментом та зручним пошуком, Moyo.ua - спеціалізацією на електроніці та акціями, а Zhuk.ua - на спеціалізованій пропозиції мобільних пристроїв та знижках. Важливо взяти з цих прикладів зручний пошук та широкий асортимент, а також вдосконалити якість обслуговування, забезпечивши особистий кабінет для клієнтів. Враховуючи потреби клієнтів та практики конкурентів, наш сайт повинен забезпечити зручну навігацію, інформативні огляди, та можливість використовувати акції для привернення уваги та збільшення продажів.

Згідно з результатів порівняння аналогів, ми спрямуємо свої зусилля на створення зручного механізму фільтрації, що допоможе нашим клієнтам швидко знаходити необхідні товари. Ми також впровадимо особистий кабінет для зареєстрованих користувачів, де вони зможуть зберігати свої налаштування та переглядати історію замовлень.

1.3 Визначення архітектури веб-додатку

Архітектура веб-сайту є важливим кроком у процесі розробки інтернет-магазину для продажу мобільних телефонів та аксесуарів. Вона визначає структуру та організацію всього веб-додатку, забезпечуючи його ефективну функціональність та зручний досвід для користувачів [4].

Для досягнення оптимальної архітектури веб-сайту, розглянемо наступні аспекти:

Модель клієнт-сервер. Встановлення відношень між клієнтами (користувачами) та сервером, який забезпечує обробку запитів та надсилання відповідей. Врахування швидкості обробки та передачі даних є ключовим для забезпечення швидкої реакції сайту.

Структура бази даних. Визначення структури бази даних для зберігання інформації про товари, користувачів, замовлення та інші дані. Оптимізована структура бази даних допомагає забезпечити швидкий доступ до інформації.

Архітектура backend. Вибір технологій та фреймворків для реалізації серверної частини веб-додатку. Важливо обрати такі інструменти, які забезпечать надійність, безпеку та ефективність серверної частини.

Архітектура frontend. Розробка структури користувацького інтерфейсу, включаючи розташування елементів, навігаційні шляхи та інші деталі. Забезпечення зручного та привабливого інтерфейсу сприяє позитивному досвіду користувачів.

Масштабованість. Розробка архітектури, яка може легко масштабуватися з ростом кількості користувачів та обсягу даних. Використання методів масштабування допоможе уникнути проблем при збільшенні завантаження сайту.

Безпека. Врахування заходів безпеки, таких як шифрування даних, захист від SQL-ін'єкцій, аутентифікація та авторизація користувачів, щоб забезпечити захищеність даних та конфіденційність.

Враховуючи ці аспекти, ми створимо архітектуру веб-сайту, яка буде забезпечувати оптимальну продуктивність, надійність та зручність використання для наших користувачів.

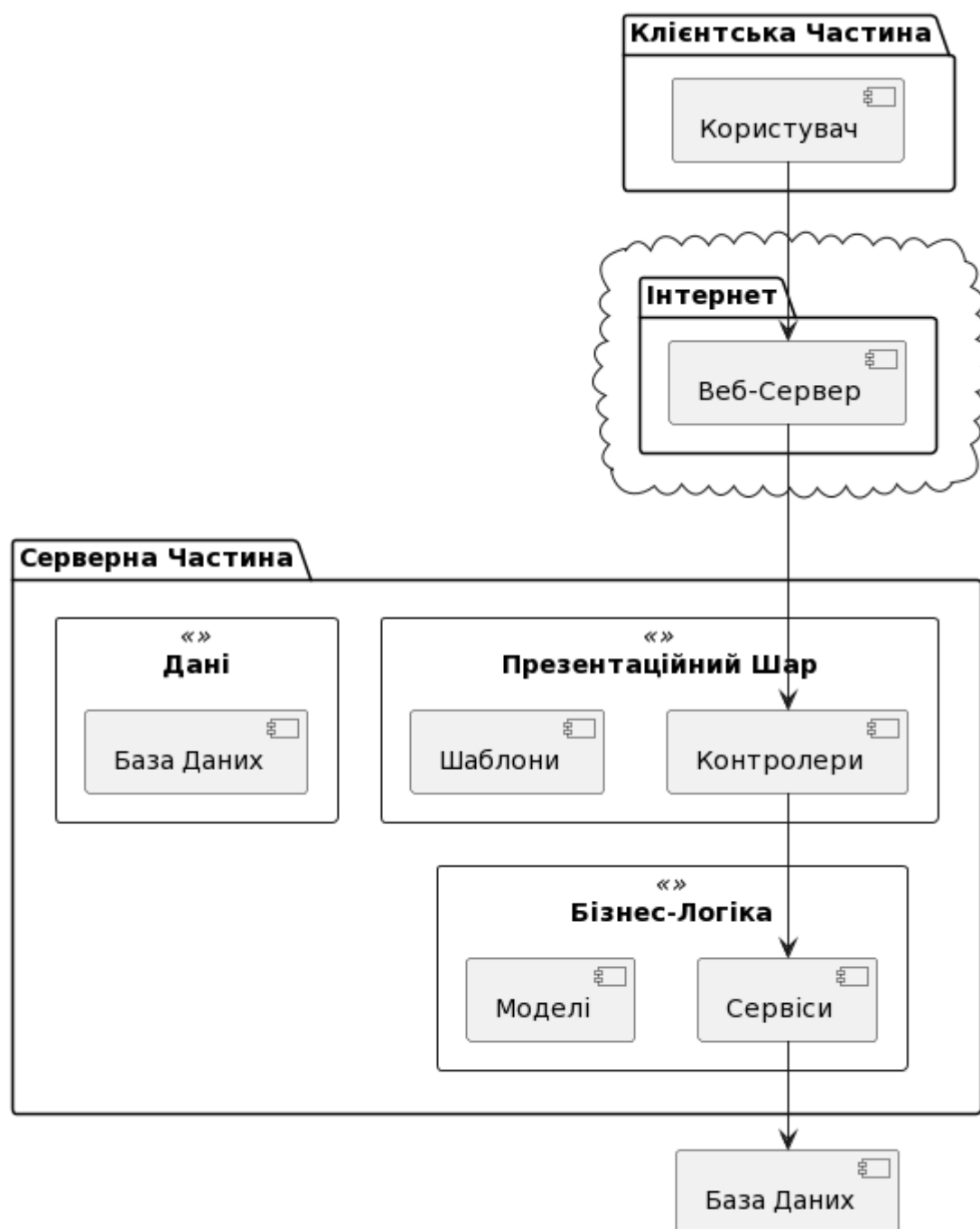


Рисунок 1.5 – Архітектура сайту з продажу мобільних телефонів та аксесуарів

1.4 Аналіз інструментів та технологій для створення веб-сайту

При розробці інтернет-магазину для продажу мобільних телефонів та аксесуарів важливо ретельно підібрати інструменти та технології, які забезпечать ефективну та стабільну роботу веб-сайту.

1.4.1 Вибір мови програмування

Вибір мови програмування є ключовим кроком для розробки веб-сайту. Ми розглянемо такі мови, як Ruby, Python, PHP, а також JavaScript для клієнтської сторони.



Рисунок 1.6 – Логотипи Ruby, Python, JavaScript, PHP

Важливим фактором буде наявність ресурсів для розробки, підтримки та розширення магазину на обраній мові. Також для розробки інтернет-магазину будуть використані HTML, CSS та фреймворк Bootstrap. HTML використовуватиметься для структури та розмітки контенту, CSS - для стилізації та оформлення елементів, а Bootstrap - для швидкого створення адаптивного та привабливого дизайну сайту. Ці технології допоможуть забезпечити зручний та привабливий користувацький досвід на сайті.

Ruby - динамічна, об'єктно-орієнтована мова програмування з акцентом на простоту та читабельність коду. Вона була створена в Японії в середині 1990-х років, і відразу привернула увагу програмістів завдяки своєму зручному та елегантному синтаксису [5].

Однією з основних філософій мови Ruby є "Матерія - це об'єкт, об'єкт - це матерія". Це означає, що в Ruby майже все є об'єктом, включаючи числа, рядки, навіть функції. Це сприяє високому рівню абстракції та дозволяє писати більш зрозумілий та ефективний код.

Однією з переваг Ruby є його гнучкість та динамічність. Ви можете змінювати типи даних об'єктів під час виконання програми, що робить його дуже потужним для розробки великої кількості різноманітних застосунків.

Ruby також має велику спільноту та багато сторонніх бібліотек, що дозволяє розробникам використовувати готові рішення для різних завдань. Один з найвідоміших фреймворків для веб-розробки, Ruby on Rails, розроблений на базі мови Ruby, і відомий своєю продуктивністю та зручністю для створення веб-додатків.

Ruby відомий своєю елегантністю, простотою вивчення та здатністю забезпечувати продуктивність та гнучкість у різних галузях програмування.

Python - високорівнева, інтерпретована мова програмування, яка відзначається простотою синтаксису та читабельністю коду. Вона була створена у 1991 році Гвідо ван Россумом і здобула популярність завдяки своїй легкості в освоєнні, розширюваності та широкому спектру застосувань [6].

Основні характеристики Python:

1. Синтаксис Python дуже чіткий та лаконічний, що дозволяє програмістам писати більш зрозумілий та зрозумілий код.
2. Python має широкий вибір стандартних бібліотек та сторонніх модулів, які допомагають вирішувати різноманітні завдання, від обробки даних до веб-розробки.
3. Python підтримується на різних операційних системах, що дозволяє розробникам створювати програми, які працюють на різних платформах.
4. Python застосовується в різних галузях, таких як веб-розробка, наукові дослідження, машинне навчання, штучний інтелект, автоматизація та багато інших.
5. Python не вимагає складних декларацій чи великої кількості коду. Це дозволяє програмістам зосередитися на логіці програми, а не на синтаксичних деталях.
6. Код Python виконується рядок за рядком інтерпретатором, що спрощує відлагодження програми.

У зв'язку зі своєю простотою та розширюваністю, Python є популярним вибором для веб-розробки, наукових досліджень, аналізу даних, створення прототипів та багатьох інших завдань.

PHP (Hypertext Preprocessor) - популярна мова програмування загального призначення, спеціально створена для розробки веб-додатків та динамічних веб-сайтів. Вона використовується для створення серверної частини веб-додатків, що оброблюють запити від клієнтів та генерують відповіді для них [7].

Основні риси мови PHP:

1. PHP має синтаксис, схожий на C, що робить його зрозумілим для програмістів з досвідом в інших мовах. Він також відомий своєю легкістю вивчення, що робить його доступним для новачків.
2. Однією з ключових особливостей PHP є його здатність працювати на серверній стороні. Він використовується для обробки форм, генерації динамічного контенту, доступу до баз даних та багатьох інших завдань, пов'язаних з серверною розробкою.
3. PHP дуже популярний в сфері веб-розробки. Він використовується для створення різних типів веб-додатків, від невеликих блогів до великих веб-платформ.
4. Однією з основних переваг PHP є його здатність генерувати динамічний контент на льоту. Це дозволяє створювати інтерактивні та персоналізовані веб-сайти.
5. У PHP є велика та активна спільнота розробників, яка надає підтримку, документацію та різноманітні ресурси для вивчення та розробки.
6. PHP має багато розширень та бібліотек, що дозволяють вам виконувати різноманітні завдання, такі як робота з зображеннями, робота зі звуком, робота з XML та інше.

В загальному, PHP є потужним інструментом для розробки веб-додатків, особливо в сфері e-commerce, включаючи інтернет-магазини.

JavaScript - це популярна мова програмування, яка використовується для розробки динамічних та інтерактивних веб-сайтів [8]. Вона є однією з ключових

технологій для фронтенд-розробки і дозволяє створювати веб-додатки з взаємодією між користувачем та сторінкою без необхідності перезавантаження.

Основні особливості JavaScript:

1. JavaScript дозволяє змінювати та оновлювати вміст сторінки без перезавантаження. Це дозволяє створювати більш плавні та зручні користувацькі інтерфейси.
2. Мова надає можливість реагувати на події користувача, такі як кліки, наведення курсору, введення тексту тощо. Це дозволяє створювати інтерактивні елементи на сторінці.
3. JavaScript дозволяє змінювати структуру та вигляд HTML-документу, що називається Document Object Model (DOM). Це дозволяє динамічно створювати, видаляти та змінювати елементи сторінки.
4. JavaScript підтримує асинхронну обробку подій і запитів, що дозволяє виконувати дії без блокування інтерфейсу користувача. Це особливо корисно при завантаженні даних з сервера.
5. JavaScript має широку спільноту розробників, яка активно співпрацює над розробкою бібліотек, фреймворків та інструментів для розширення можливостей мови.

JavaScript використовується як для фронтенд-розробки (веб-сторінки), так і для бекенд-розробки (серверні додатки). Ця мова є невід'ємною частиною сучасної веб-розробки і допомагає забезпечити більш динамічний та інтерактивний досвід для користувачів.

Грунтовно проаналізувавши, ми обираємо мови програмування Python і JavaScript для розробки нашого інтернет-магазину з продажу мобільних телефонів та аксесуарів.

1.4.2 Вибір системи керування базами даних

Система керування базами даних (СКБД) впливає на зберігання, обробку та доступ до даних веб-сайту. Ми розглянемо такі СКБД, як MySQL, PostgreSQL,

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						18

SQLite. Вибір СКБД повинен забезпечити швидкий доступ до інформації, надійність та безпеку даних.



Рисунок 1.7 – Логотипи MySQL, PostgreSQL, SQLite

MySQL є однією з найпопулярніших та найрозповсюдженіших систем керування базами даних у світі. Вона відноситься до реляційних баз даних і здатна ефективно зберігати, управляти та обробляти великі об'єми даних, що робить її привабливою для веб-розробки, зокрема для інтернет-магазину з продажу мобільних телефонів та аксесуарів [9].

Деякі ключові характеристики MySQL:

- MySQL відома своєю надійністю та стабільністю, що робить її відмінним вибором для веб-додатків, які потребують постійної доступності бази даних.
- MySQL демонструє високу швидкодію завдяки оптимізаціям та ефективному використанню ресурсів. Це особливо важливо для веб-сайтів з великою кількістю запитів до бази даних.
- MySQL надає різні методи оптимізації запитів та індексації даних, що допомагає покращити продуктивність бази даних.
- Завдяки можливостям реплікації та кластеризації, MySQL може бути масштабована для роботи з великою кількістю користувачів та обсягом даних.
- MySQL має велику активну спільноту розробників, яка надає підтримку, допомогу та розв'язання проблем.
- MySQL надає можливості для захисту даних, включаючи шифрування та управління правами доступу.

- MySQL дотримується стандартів SQL, що робить її сумісною з більшістю інших СКБД.

З усіма цими перевагами MySQL може бути ефективним рішенням для забезпечення надійної та ефективної роботи бази даних для нашого інтернет-магазину з продажу мобільних телефонів та аксесуарів.

PostgreSQL є потужною та розширюваною відкритою системою керування базами даних, яка відноситься до реляційних баз даних. Її висока функціональність, розширюваність та здатність до обробки складних запитів роблять її відмінним вибором для веб-додатків, таких як інтернет-магазин мобільних телефонів та аксесуарів [10].

Опишемо ключові характеристики PostgreSQL:

- PostgreSQL славиться своєю здатністю до розширення та кастомізації, завдяки чому можна створювати власні функції, типи даних та розширення, а також інтегрувати їх у базу даних.
- PostgreSQL надає різні можливості для складних та аналітичних запитів, включаючи віконні функції та агрегацію.
- Для веб-додатків, де потрібна робота з геоданими, PostgreSQL має розширені можливості географічного аналізу та операцій з геоданими.
- PostgreSQL підтримує ACID-властивості (Атомарність, Консистентність, Ізольованість, Довіреність) та має потужний механізм керування конкурентністю для одночасного доступу до даних.
- У PostgreSQL є вбудована підтримка для роботи з документоорієнтованими даними у форматі JSON.
- Є активна спільнота розробників, яка надає допомогу та підтримку, а також регулярно випускає оновлення та патчі для системи.
- PostgreSQL показує добрі результати в роботі з великими об'ємами даних та запитами завдяки оптимізаціям та ефективному використанню пам'яті.
- PostgreSQL надає широкі можливості для налаштування прав доступу та шифрування даних.

З усіма цими перевагами PostgreSQL стає потужним інструментом для забезпечення надійного та ефективного зберігання та обробки даних і її можна використовувати в нашому інтернет-магазині з продажу мобільних телефонів та аксесуарів.

SQLite - це легка та вбудована система керування базами даних, яка також відноситься до реляційних баз даних. Основною особливістю SQLite є те, що вона працює в межах одного файлу, не вимагаючи окремого сервера або конфігурації. Це робить її привабливою для простих веб-додатків та проектів з обмеженими ресурсами [11].

Ключові характеристики SQLite включають:

- SQLite може бути вбудована безпосередньо в програму або веб-додаток, що спрощує розгортання та управління базою даних.
- Запити та операції з базою даних виконуються в рамках SQL, що робить її доступною для більшості розробників з досвідом у роботі з SQL.
- SQLite має динамічну типізацію даних, що дозволяє зберігати різні типи даних у одному стовпці.
- Бази даних SQLite можуть бути легко перенесені між різними платформами та операційними системами.
- Є широка спільнота розробників, яка надає підтримку та допомогу при роботі з SQLite.
- SQLite має добре документовану та зрозумілу документацію, яка полегшує вивчення та використання системи.

Враховуючи легкість використання, вбудованість та інші описані характеристики обираємо SQLite для нашого інтернет-магазину мобільних телефонів та аксесуарів, де не потрібно великих обсягів даних та високої швидкодії обробки.

1.4.3 Вибір фреймворку

Використання веб-фреймворку спростить розробку та допоможе забезпечити структуровану архітектуру. Ми дослідимо такі фреймворки, як Django, Flask, Pyramid. Оптимальний фреймворк повинен прискорити розробку, забезпечити безпеку та масштабованість веб-додатку.



Рисунок 1.8 – Найпопулярніші веб-фреймворки для Python

Django - це потужний та високорівневий веб-фреймворк, розроблений на мові програмування Python. Він спроектований для прискорення процесу розробки веб-додатків шляхом надання готових рішень для багатьох типових задач. Django сприяє забезпеченню швидкості, безпеки, якості коду та структурованості проєктів, що робить його відмінним вибором для розробки інтернет-магазину з продажу мобільних телефонів та аксесуарів [12].

Основні характеристики Django:

- Django використовує концепцію модулів (пакетів) та механізм розширення, що дозволяє створювати функціональні блоки і повторно використовувати їх у різних проєктах.
- Django надає ORM, яке дозволяє працювати з базою даних за допомогою об'єктів та класів, що спрощує роботу з даними та зменшує кількість SQL-запитів.
- Django надає готовий адміністративний інтерфейс, який автоматично генерується на основі моделей даних. Це дозволяє швидко створити інтерфейс для керування даними без додаткових зусиль.
- Завдяки готовим рішенням, автоматизованим процесам та широким набором бібліотек, Django забезпечує швидкість розробки проєктів.

- Django має вбудовані механізми для захисту від різних видів атак, таких як SQL-ін'єкції, XSS та інші.
- Django використовує власну мову шаблонів, що дозволяє відокремити логіку та представлення.
- Django надає інструменти для зручного розгортання на сервері, що допомагає прискорити процес публікації веб-додатку.

Завдяки цим перевагам Django є відмінним фреймворком для створення інтернет-магазину з продажу мобільних телефонів та аксесуарів. Він допоможе створити добре структурований, безпечний та продуктивний веб-додаток зі зручним інтерфейсом для адміністрування та користувачів.

Flask - це легкий та гнучкий мікрофреймворк для розробки веб-додатків на мові програмування Python. Він спроектований з мінімальним набором компонентів, що дозволяє розробникам вибирати самостійно бібліотеки та інструменти для своїх проектів. Flask покликаний забезпечити простоту та свободу у розробці, що робить його привабливим вибором для створення інтернет-магазину з продажу мобільних телефонів та аксесуарів [13].

Ключові характеристики Flask:

- Flask надає лише основні компоненти, такі як маршрутизація та обробка запитів, що дозволяє розробникам будувати свій власний стек технологій.
- Flask підтримує розширення (extensions), які додають додаткові функціональності, коли вони потрібні, зберігаючи при цьому простоту базового фреймворку.
- Flask підтримує різні движки шаблонів, що дозволяє виділяти логіку та представлення відокремлено.
- Flask може бути легко інтегрований з іншими бібліотеками та інструментами Python.
- Flask має добре написану та доступну документацію, що допомагає розробникам швидко розібратися з фреймворком.
- Є активна спільнота розробників, яка надає підтримку та допомогу при роботі з Flask.

- Завдяки своєму мінімалістичному підходу, Flask дозволяє швидко освоїти основи та почати розробку.
- Flask підтримує реалізацію RESTful API, що дозволяє створювати веб-сервіси.

Flask відмінно підходить для розробки простих веб-додатків та прототипів, де важлива легкість використання та можливість вибору технологій. Він дозволяє створювати зручні та ефективні веб-додатки без надмірної складності.

Pyramid - це високорівневий фреймворк для розробки веб-додатків на мові програмування Python. Він спроектований з акцентом на гнучкість та розширюваність, надаючи розробникам можливість побудувати власну архітектуру додатку відповідно до їх потреб. Pyramid підходить для проектів різного розміру та складності, забезпечуючи потужний набір інструментів та підходів [14].

Основні характеристики Pyramid:

- Pyramid підтримує різні підходи до розробки, що дозволяє розробникам вибирати, як організувати код та структуру проекту.
- Фреймворк побудований на принципах модульності, що дозволяє використовувати лише необхідні компоненти та розширювати їх за потребою.
- Pyramid не накладає обмежень на вибір ORM або бази даних, дозволяючи використовувати той варіант, який вам зручний.
- Фреймворк не обмежує вас в виборі технологій, що дозволяє використовувати різні бібліотеки та інструменти для різних аспектів проекту.
- Pyramid може бути легко інтегрований з іншими бібліотеками та інструментами Python.
- Є велика кількість розширень (extensions) та бібліотек, які можна використовувати з Pyramid для різноманітних задач.
- Є активна спільнота розробників, яка надає підтримку та допомогу при роботі з Pyramid

Pyramid підходить для розробки веб-додатків різної складності, від невеликих проектів до великих та складних систем. Він надає розробникам велику вільність у виборі технологій та підходів, що допомагає створити проект з оптимальною архітектурою та функціональністю.

Для реалізації нашого проекту, який передбачає створення інтернет-магазину для продажу мобільних телефонів та аксесуарів, ми обираємо фреймворк Django.

Висновки до розділу 1

В першому розділі кваліфікаційної роботи було проведено теоретичний огляд та аналіз сучасних рішень у галузі розробки інтернет-магазину для продажу мобільних телефонів та аксесуарів. Акцент був зроблений на визначенні постановки завдання, детальному аналізі аналогічних програмних рішень, виборі архітектури веб-сайту, аналізі методів вирішення проблем, виборі мов програмування, системи керування базами даних та фреймворку.

На основі аналізу було виявлено, що інтернет-магазини, які пропонують продаж мобільних телефонів та аксесуарів, мають ряд спільних рис, таких як зручний пошук та фільтрація товарів, докладні огляди, особисті кабінети користувачів, акції та знижки. Переваги та недоліки кожного аналога були проаналізовані з метою використання найкращих практик у реалізації нашого проекту.

Під час вибору технологічних рішень для нашого проекту було прийнято рішення використовувати фреймворк Django. Цей вибір обґрунтований його потужністю, широким набором функцій, високим рівнем безпеки та швидкістю розробки. Мова програмування Python також була обрана через її читабельність, зрозумілість та розширюваність.

Вибір системи керування базами даних припав SQLite бо це легка, вбудована та сервер-незалежна система керування базами даних, яка зберігає всі дані в одному файлі. Вона має кілька переваг та особливостей, які роблять її ідеальним варіантом для нашого проекту

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						25

Загалом, проведений аналіз та вибір технологій підтверджують, що наш підхід до розробки інтернет-магазину має великий потенціал для успішної реалізації проекту. У наступних розділах ми розглянемо деталі реалізації нашого інтернет-магазину з урахуванням обраної архітектури та технологій.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА ЕТАПИ РОЗРОБКИ САЙТУ З ПРОДАЖУ МОБІЛЬНИХ ТЕЛЕФОНІВ ТА АКСЕСУАРІВ

2.1. Діаграма варіантів використання

Діаграма варіантів використання (Use Case Diagram) є графічним інструментом [15], який допомагає моделювати функціональність системи через взаємодії між акторами (користувачами чи зовнішніми сутностями) та різними можливими сценаріями їх взаємодії з системою (рис. 2.1).

Діаграма варіантів використання дозволяє зрозуміти, які дії можуть бути виконані акторами у системі, які взаємодії між ними існують, та які функціональні можливості пропонує система. Це корисний інструмент для забезпечення зрозумілості вимог та специфікацій системи для всіх учасників проекту [16].

В нашій діаграмі варіантів використання для інтернет-магазину з продажу смартфонів та аксесуарів відбувається наступне:

1. Користувачі:

- **Незареєстрований користувач.** Може реєструватися на сайті, переглядати товари та аксесуари.
- **Користувач.** Має можливість входу в систему, переглядати товари та аксесуари, обирає товари та аксесуари для замовлення, оформляє замовлення.

2. Адміністратор:

- Має можливість керування даними користувачів (створення, редагування, видалення), керування товарами (додавання, редагування, видалення) та керування аксесуарами (додавання, редагування, видалення).

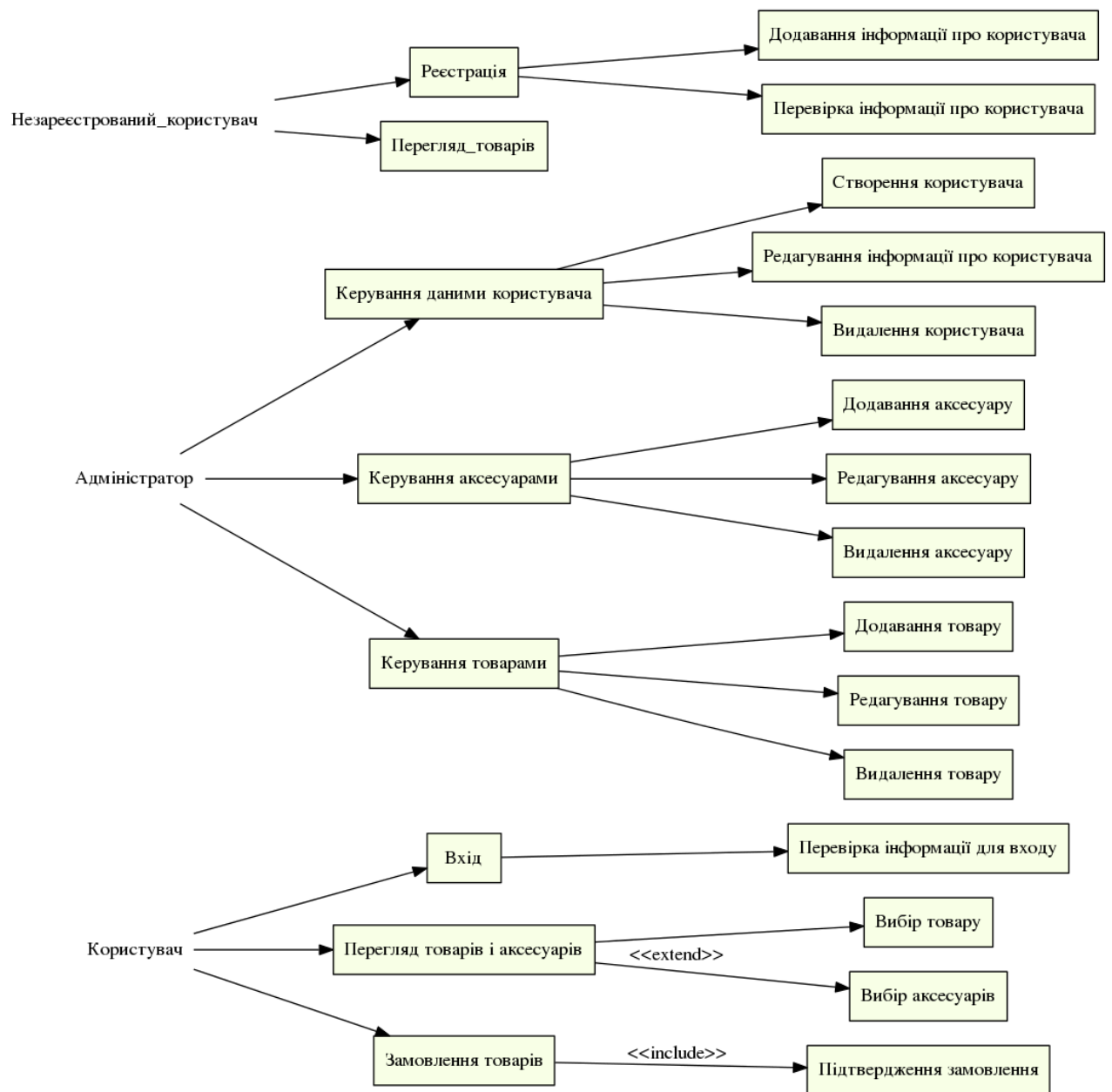


Рисунок 2.1 – Діаграма варіантів використання інтернет-магазину

3. Сценарії взаємодії:

- **Реєстрація.** Незареєстрований користувач може зареєструватися, вводячи необхідну інформацію.
- **Вхід.** Користувач може увійти в систему, вводячи свої облікові дані.
- **Перегляд товарів і аксесуарів.** Користувач може переглядати список доступних товарів та аксесуарів.
- **Вибір товару та аксесуарів.** Користувач може обирати товари та аксесуари для замовлення.

- **Керування даними користувача, товарами і аксесуарами.** Адміністратор може виконувати різні операції керування даними користувачів, товарів та аксесуарів.
- **Замовлення товарів.** Користувач може оформити замовлення на обрані товари та аксесуари.
- **Підтвердження замовлення.** Система підтверджує та обробляє замовлення користувача.

Ця діаграма допомагає чітко визначити функціональні можливості системи, ролі користувачів та їх взаємодії з системою, що сприяє зрозумінню і уточненню вимог до проекту.

2.1.2 Діаграма діяльності (activitydiagram)

Діаграма діяльності (Activity Diagram) є графічним інструментом моделювання, який дозволяє зобразити послідовність дій, процесів, рішень та взаємодій між об'єктами, що здійснюють активності [17]. Цей тип діаграми використовується для візуального опису поведінки системи, процесу чи алгоритму (рис. 2.2).

Ця діаграма діяльності відображає послідовність дій користувача на інтернет-магазині для продажу мобільних телефонів та аксесуарів:

1. Користувач заходить на сайт та переглядає головну сторінку.
2. Користувач може вибрати опцію фільтрації для зручного пошуку товарів (якщо бажає).
3. Користувач обирає товари та аксесуари, які його цікавлять, і вони відображаються.
4. Користувач вибирає опцію "Замовити", щоб перейти до оформлення замовлення.
5. Якщо користувач зареєстрований, відбувається оформлення замовлення і відображення підтвердження замовлення.
6. Якщо користувач не зареєстрований, він перенаправляється на сторінку реєстрації, де він може заповнити реєстраційну форму.

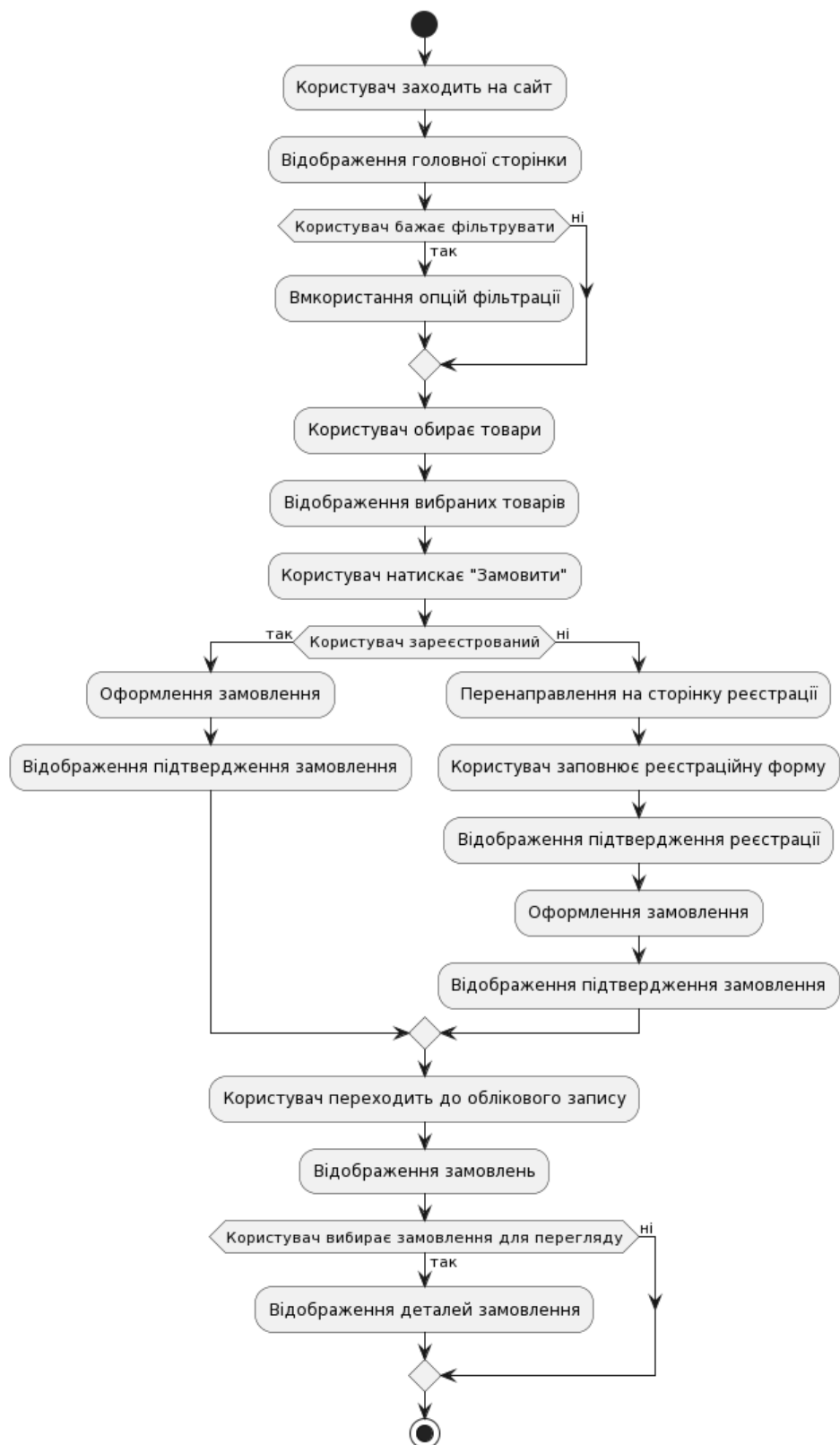


Рисунок 2.2 – Діаграма діяльності інтернет-магазину

7. Після реєстрації користувач також оформляє замовлення та отримує підтвердження.

8. Після оформлення замовлення, користувач має можливість перейти до свого облікового запису.
9. На сторінці облікового запису він може переглянути список своїх замовлень.
10. Користувач може вибрати конкретне замовлення для перегляду його деталей.

2.1.3 Діаграма послідовності (Sequencediagram)

Діаграма послідовності - це графічний інструмент в моделюванні програмного забезпечення, який використовується для візуалізації послідовності взаємодій між різними об'єктами або компонентами системи в певному часовому порядку [18]. Вона дозволяє зобразити, як об'єкти спілкуються між собою шляхом відправлення повідомлень, запитів та відповідей (рис. 2.3).

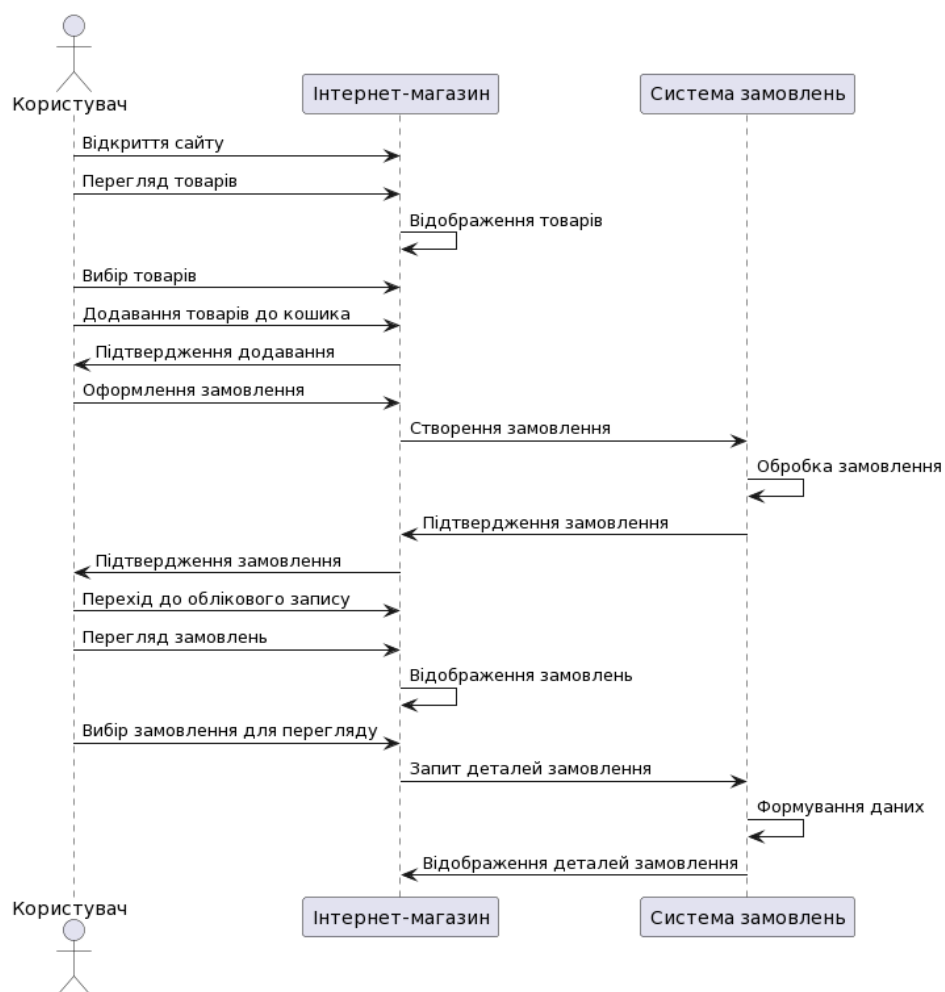


Рисунок 2.3 – Діаграма послідовності інтернет-магазину

Діаграми послідовності допомагають розібратися у взаємодії різних компонентів системи, визначити послідовність дій, виявити можливі проблеми та взаємодіючі компоненти. Вони корисні для аналізу та проектування взаємодій у складних процесах, таких як замовлення товарів, реєстрація користувачів, обробка даних тощо.

У нашому випадку, діаграма послідовності відображає взаємодію між користувачем, інтернет-магазином та системою замовлень під час різних дій: перегляду товарів, додавання їх до кошика, оформлення та перегляду замовлень. Це допомагає нам краще зрозуміти, як взаємодіють різні компоненти системи під час виконання конкретних сценаріїв та виявити можливі покращення в процесах.

2.1.4 Діаграма класів

Діаграма класів сутностей (рис. 2.4) створена для візуалізації структури та взаємодії між двома основними класами: "Product" (продукт) та "Order" (замовлення). Кожен клас представляє сутність, яка має свої атрибути (поля) та методи [19].

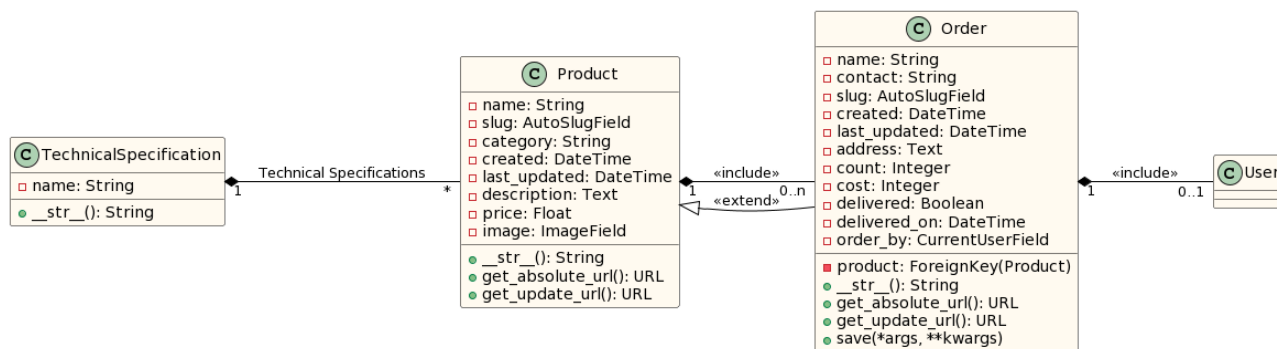


Рисунок 2.4 – Діаграма класів сутностей інтернет-магазину

Клас "Product" визначає характеристики продукту, такі як назва, категорія, ціна, зображення тощо. Він також має методи для отримання URL-адреси продукту та редагування.

Клас "Order" представляє замовлення, яке може містити один продукт. Він включає атрибути, такі як ім'я замовника, контактна інформація, адреса, кількість, вартість тощо. Важливий атрибут - "product", який вказує на зв'язок із

класом "Product". Клас "Order" має методи для обробки замовлення, включаючи можливість підтвердження та збереження.

У діаграмі також використані відношення "include" та "extend". "Include" вказує на включення функціональності одного класу в інший, як наприклад, замовник може включати додаткові дані користувача. "Extend" вказує на розширення функціональності одного класу за допомогою функцій іншого класу, наприклад, замовлення може розширювати функціональність продукту.

У класа TechnicalSpecification є відношення "один до багатьох" з класом Product через поле technical_specifications

Ця діаграма надає візуальне подання структури та взаємодії між класами "Product" та "Order", що допомагає зрозуміти, як вони пов'язані та як функціонує основна логіка між ними у вашому інтернет-магазині мобільних телефонів та аксесуарів.

2.2. Розробка бази даних

Розробка бази даних є важливим етапом у створенні програмного продукту, особливо веб-додатку чи інтернет-магазину. Основна мета розробки бази даних полягає в організації зберігання, оновлення та отримання інформації для ефективної роботи програми (рис. 2.5).

База даних містить такі таблиці:

1. Таблиця TechnicalSpecification:

- Поля:
 - name (Текстове поле, максимальна довжина 255 символів) - назва технічної характеристики.

2. Таблиця Product:

- Поля:
 - name (Текстове поле, максимальна довжина 255 символів) - назва товару.
 - slug (Текстове поле, автозаповнюване значенням name для унікальності) - унікальний ідентифікатор товару.

- category (Текстове поле, максимальна довжина 255 символів) - категорія товару.
- created (Дата та час) - дата та час створення товару.
- last_updated (Дата та час) - дата та час останнього оновлення товару.
- description (Текстове поле) - опис товару.
- price (Дійсне число) - ціна товару.
- image (Зображення) - зображення товару.
- **Зв'язки:**
 - technical_specifications (Багато-до-багатьох з TechnicalSpecification) - технічні характеристики товару.

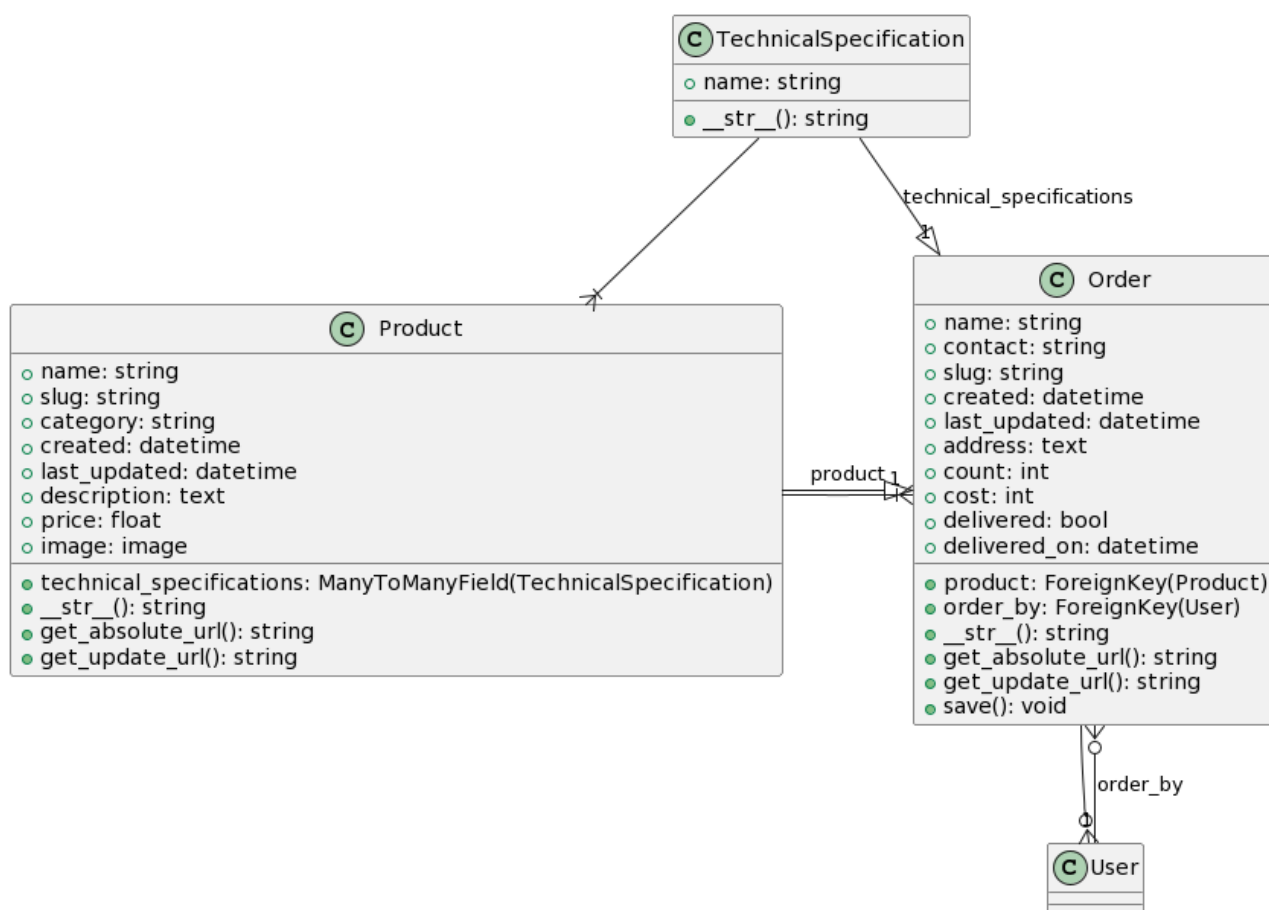


Рисунок 2.5 – Розробка бази даних інтернет-магазину

3. Таблиця Order:

- **Поля:**

- name (Текстове поле, максимальна довжина 255 символів) - назва замовлення.
- contact (Текстове поле, максимальна довжина 10 символів) - контактна інформація.
- slug (Текстове поле, автозаповнюване значенням name для унікальності) - унікальний ідентифікатор замовлення.
- created (Дата та час) - дата та час створення замовлення.
- last_updated (Дата та час) - дата та час останнього оновлення замовлення.
- address (Текстове поле) - адреса для доставки.
- count (Ціле число, значення за замовчуванням 1) - кількість товарів у замовленні.
- cost (Ціле число, значення за замовчуванням 0) - вартість замовлення.
- delivered (Логічний тип, значення за замовчуванням False) - чи замовлення доставлене.
- delivered_on (Дата та час) - дата та час доставки замовлення.

- **Зв'язки:**

- product (Зовнішній ключ до Product, зв'язок один-до-багатьох) - товари в замовленні.
- order_by (Зовнішній ключ до користувача, зв'язок один-до-багатьох) - користувач, який зробив замовлення.

Ця структура допомагає організувати та зберігати інформацію про продукти, замовлення та користувачів у нашому інтернет-магазині мобільних телефонів та аксесуарів.

2.3 Розробка дизайну інтернет-магазину

Веб-дизайн - це процес створення зовнішнього вигляду та структури веб-сайту. Він об'єднує різні аспекти дизайну, розробки та взаємодії з метою створення привабливого, естетичного та функціонального середовища для відвідувачів [20].

При створенні дизайну веб-додатку, спочатку розробляють його макет. Під час макетування вебсайту зазвичай першим кроком є створення блоків або контейнерів, в які впроваджуватиметься контент і функціональність вебсайту. Потім додають окремі елементи всередині цих блоків. Такий підхід дозволяє забезпечити логічний і структурований макет для всього вебсайту, а також полегшує внесення змін і коригування дизайну за потреби (рис. 2.6).

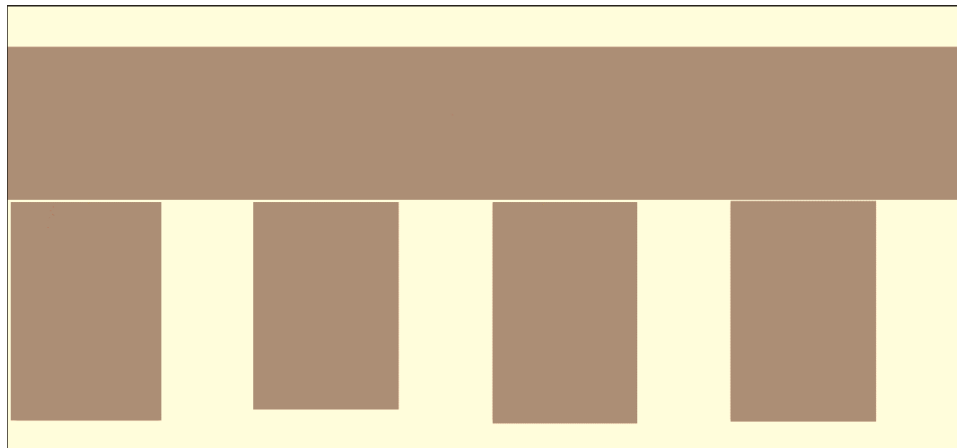


Рисунок 2.6 – Макетування головних блоків головної сторінки

На початковому етапі цього процесу зазвичай розпочинається з створення каркасу або макета веб-сайту, який визначає основну структуру макета та сприяє визначенню розташування ключових компонентів, таких як заголовки, верхні панелі і навігаційні меню (рис. 2.7).

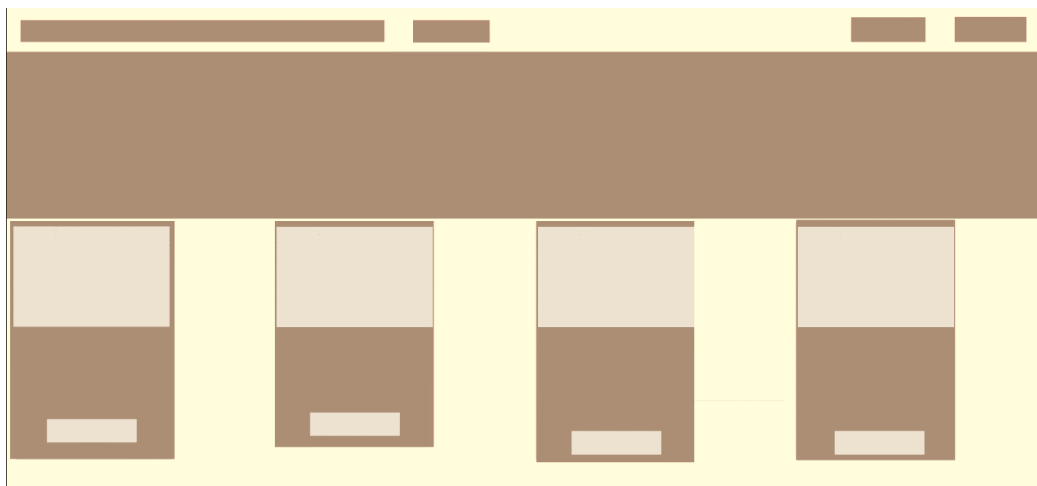


Рисунок 2.7 – Макетування ключових елементів головної сторінки

Далі в ці контейнери можна додавати окремі елементи, такі як текст, зображення, кнопки та форми. Працюючи таким чином, можна гарантувати, що макет залишається послідовним і добре організованим, а також можна легко додавати, видаляти або змінювати окремі елементи, не впливаючи на загальну структуру сайту (рис. 2.8).

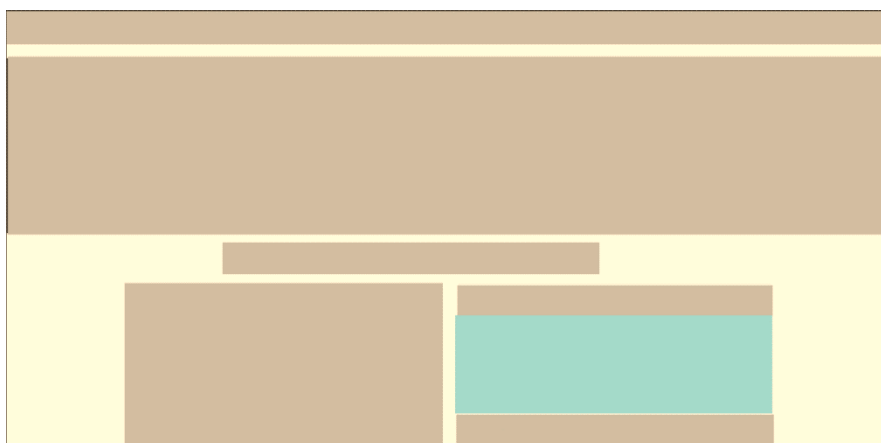


Рисунок 2.8 – Макетування сторінки з обраним продуктом

Макетування вебсайту є важливою частиною процесу дизайну, оскільки воно допомагає забезпечити зручність навігації по сайту, його візуальну привабливість і функціональність. Починаючи з чіткого плану і будуючи з нуля, дизайнери можуть створити макет, який відповідає потребам їхніх клієнтів та їхньої аудиторії, а також дозволяє легко підтримувати і оновлювати його в майбутньому (рис. 2.9).

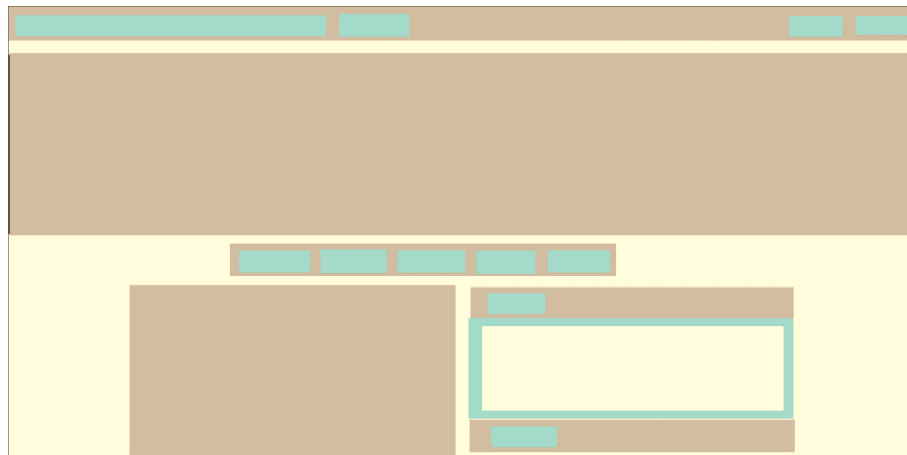


Рисунок 2.9 – Макетування ключових елементів сторінки з обраним продуктом

Для дизайну веб-сайтів, включаючи створення графіки, макетів та елементів дизайну, можна використовувати різноманітні інструменти та програми. Один із найпоширеніших і популярних інструментів для цього є Adobe Photoshop.

Adobe Photoshop - це професійний графічний редактор, який надає широкий спектр можливостей для створення та редагування різних видів графіки [21]. Скористаємося Photoshop для дизайну нашого сайту.

Навігаційна панель (Navbar). Це верхній блок, де розташована навігаційна панель. Навігаційна панель містить бренд (назву сайту), кнопку "бургер" для розкриття/згортання меню на маленьких екранах, елементи списку для навігації, а також інші елементи навігації для входу та реєстрації (рис. 2.10).



Рисунок 2.10 – Дизайн навігаційної панелі (Navbar)

Карусель (Carousel). Це блок із слайдером, який демонструє зображення. Використовується для показу рекламних банерів (рис. 2.11).

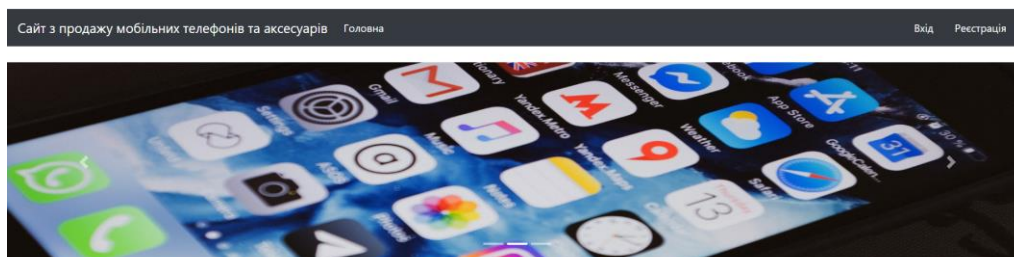


Рисунок 2.11 – Дизайн Каруселі

Кнопки фільтрації. Блок з кнопками, які використовуються для фільтрації контенту. Кожна кнопка буде викликати JavaScript-функцію, яка змінить URL сторінки, передаючи тип фільтра (рис. 2.12).

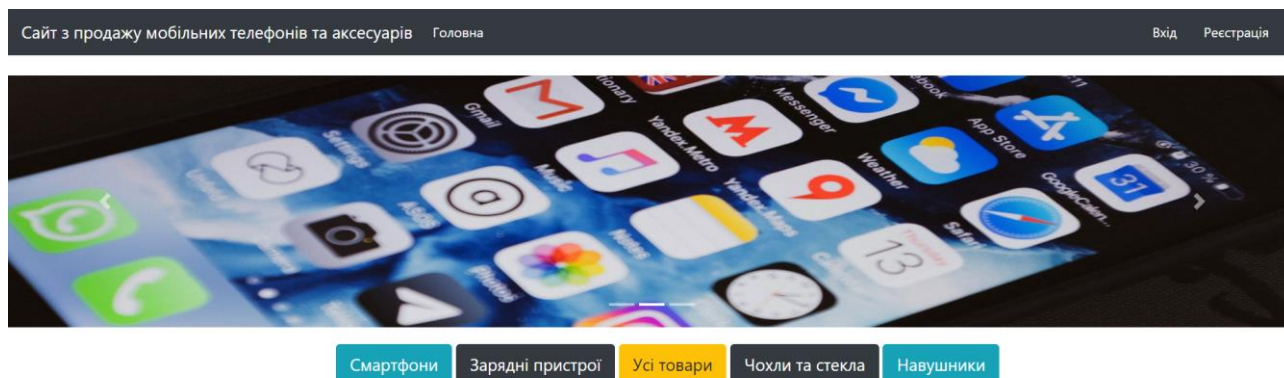


Рисунок 2.12 – Дизайн кнопок фільтрації

Основний контент (Content Block): Це блок, який відводить місце для основного вмісту веб-сайту. Він призначений для використання в інших HTML-шаблонах для вставки конкретного вмісту. Створимо дизайн для одного окремого товару (рис. 2.13).



Рисунок 2.13 – Дизайн для одного окремого товару

Та для сторінки з деталями обраного товару (рис. 2.14).

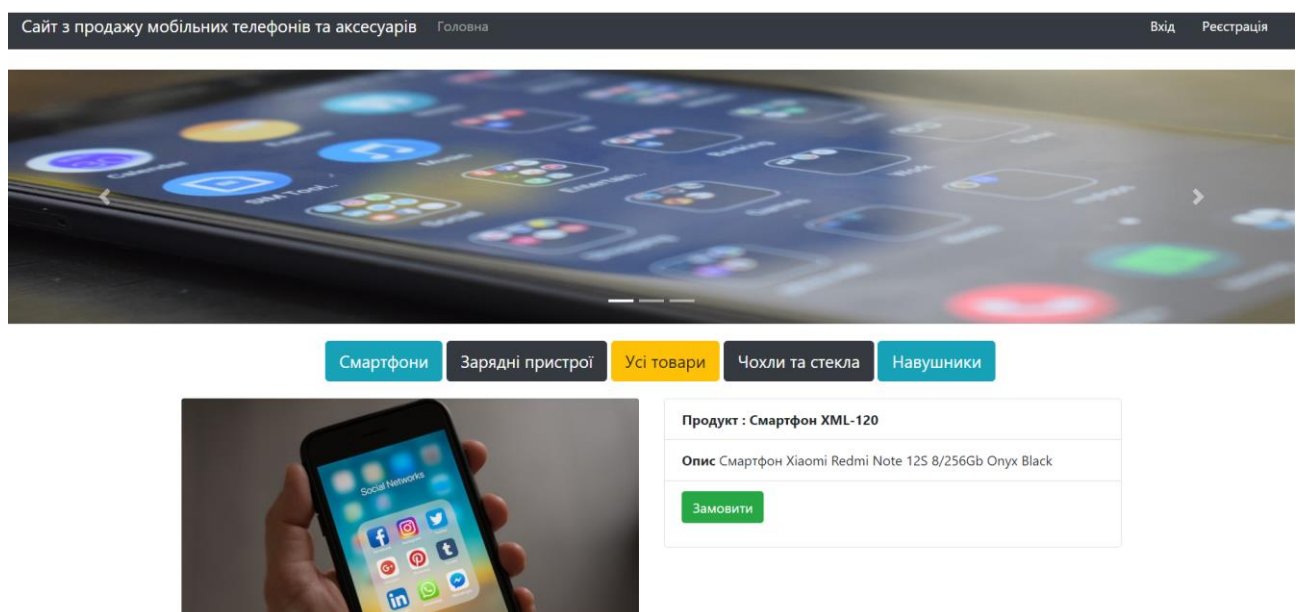


Рисунок 2.14 – Дизайн сторінки з деталями обраного товару

У наступному етапі ми перейдемо до створення верстки відповідно до розробленого дизайну.

2.4. Верстка сайту по створеному дизайну

Верстка - це процес створення веб-сторінок шляхом об'єднання графічного дизайну і структурного коду. Вона включає в себе розташування тексту, зображень, відео та інших елементів на веб-сторінці з використанням мов розмітки (зазвичай HTML) та стилів (зазвичай CSS). Цей процес також забезпечує адаптацію сторінки для різних розмірів екранів, забезпечуючи респонсивний дизайн для користувачів з різних пристроїв, таких як комп'ютери, планшети та смартфони. Верстка допомагає перетворити графічний дизайн на функціонуючу веб-сторінку, забезпечуючи зручний та естетично привабливий досвід для користувачів.

Для верстки ми будемо використовувати фреймворк Bootstrap. Bootstrap - це популярний набір інструментів, який допомагає зручно і швидко розробляти адаптивні та естетично привабливі веб-сайти [22].

Використовуючи Bootstrap, ми зможемо забезпечити респонсивний дизайн, використовувати готові стилізовані компоненти, включаючи кнопки, меню, форми, картки та навігаційні панелі, використовувати гнучку сітку для розташування елементів на сторінці, інтегрувати інтерактивні елементи JavaScript, такі як модальні вікна та каруселі, а також спростити стилізацію за допомогою заздалегідь визначених класів, що дозволить зменшити необхідність у написанні власного CSS.

Файл "base.html" є базовим шаблоном для веб-сторінок і містить загальну структуру та елементи, які будуть спільними для багатьох сторінок сайту (додаток А) . Основні елементи та їх опис:

- **{% load static %}**: Ця директива завантажує статичні файли, такі як CSS і JavaScript, з використанням підтримки Django.
- **<html lang="en">**: Визначає мову HTML документа (англійська у даному випадку).
- **<head>**: Частина документа, де розміщуються мета-теги і посилання на зовнішні ресурси.

- Bootstrap CSS: Підключає стилі Bootstrap з офіційного CDN (Content Delivery Network), що забезпечує вигляд сторінки згідно з Bootstrap.
- **<title>**: Встановлює заголовок веб-сторінки, який буде відображатися в заголовку вкладки браузера.
- **<nav class="navbar">**: Створює навігаційну панель (Navbar) з темним фоном.
- ****: Відображає бренд (назву сайту) в навігаційній панелі.
- **<button class="navbar-toggler">**: Створює кнопку "бургер" для розкривання/згортання меню на малих екранах.
- **{% block nav_home %}{% endblock nav_home %}**: Динамічний блок, який може бути перезаповнений у конкретних сторінках.
- **{% if user.is_authenticated %}**: Умова перевірки, чи користувач аутентифікований.
- **<ul class="navbar-nav">**: Список елементів навігації.
- **<ul class="nav justify-content-end">**: Список елементів навігації, вирівнюваний вправо.
- **<div id="carouselExampleIndicators">**: Слайдер (карусель) зображень, який може автоматично переключати зображення.
- **<div style="margin: 20px 20px 0 auto; display: block;" class="container">**: Контейнер для рядка кнопок з фільтрами.
- **{% block content %}{% endblock content %}**: Динамічний блок для контенту, який буде перезаповнений у конкретних сторінках.
- **<script>**: JavaScript код для фільтрації товарів на основі кнопок.

Цей файл є базовим шаблоном, що використовується для створення інших сторінок сайту. Він включає загальну структуру, навігаційну панель, слайдер зображень, кнопки фільтрів та контейнер для основного контенту, що може бути динамічно заповнений у конкретних сторінках.

Розглянемо детальніше верстку на прикладі каруселі (рис. 2.14).

```

base.html x
<div class="carousel-item active">
  
</div>
<div class="carousel-item">
  
</div>
<div class="carousel-item">
  
</div>
</div>
<a class="carousel-control-prev" href="#carouselExampleIndicators" role="button" data-slide="prev">
  <span class="carousel-control-prev-icon" aria-hidden="true"></span>
  <span class="sr-only">Попередній</span>
</a>
<a class="carousel-control-next" href="#carouselExampleIndicators" role="button" data-slide="next">
  <span class="carousel-control-next-icon" aria-hidden="true"></span>
  <span class="sr-only">Наступний</span>
</a>

```

Рисунок 2.14 – Верстка каруселі (слайдера) зображень

Цей код відображає карусель (слайдер) зображень на веб-сторінці:

- **<div id="carouselExampleIndicators" class="carousel slide" data-ride="carousel">**: Створює контейнер для каруселі з унікальним ідентифікатором та класами для використання Bootstrap каруселі.
- **<ol class="carousel-indicators">**: Визначає список індикаторів для слайдів.
- **<li data-target="#carouselExampleIndicators" data-slide-to="0" class="active">**: Кожен пункт відповідає одному слайду. **data-target** вказує на цільовий елемент (карусель), а **data-slide-to** визначає індекс слайда. Клас **active** позначає перший активний слайд.
- **<div class="carousel-inner">**: Визначає контейнер для слайдів каруселі.
- **<div class="carousel-item active">**: Створює слайд із класом **active**, який відображається за замовчуванням.
- ****: Вставляє зображення для слайда. **style** визначає стилі для зображення (у цьому випадку висота 300px), а **src** показує шлях до зображення.
- ****: Створює кнопку "попередній" для перемикавання до попереднього слайда.

- ****: Визначає піктограму для кнопки "попередній".
- **Попередній**: Текст, який виводиться при наведенні на кнопку "попередній" для доступності.
- ****: Створює кнопку "наступний" для перемикання до наступного слайда.
- ****: Визначає піктограму для кнопки "наступний".
- **Наступний**: Текст, який виводиться при наведенні на кнопку "наступний" для доступності.

Цей код визначає карусель, яка може переключати зображення на веб-сторінці та надає зручну інтерактивну функціональність для користувачів. Всі інші елементи створюються аналогічно.

Висновки до розділу 2

У розділі 2 було розглянуто методологію та етапи розробки веб-сайту з продажу мобільних телефонів та аксесуарів. В цьому контексті були використані методи та засоби об'єктно-орієнтованого аналізу і проектування, що дозволило здійснити ретельний аналіз та визначити основні вимоги до сайту.

Діаграми варіантів використання, діяльності та послідовності відображають взаємодії користувачів з сайтом, дії та послідовності подій, що допомогли зрозуміти взаємодію між різними складовими системи. Діаграма класів надала можливість структурувати компоненти системи та їх зв'язки, що сприяло зрозумінню архітектури.

Розробка бази даних була важливим етапом, який дозволив забезпечити зберігання та організацію інформації про товари, замовлення та користувачів. Це створило основу для функціональності інтернет-магазину.

Розробка дизайну інтернет-магазину дозволила визначити вигляд та взаєморозміщення елементів на сторінках, забезпечивши зручну та привабливу візуальну інтеракцію з користувачем.

Завершальним етапом була верстка сайту за створеним дизайном. Використання Bootstrap дозволило створити респонсивний дизайн, використовувати готові компоненти та структурувати сторінку відповідно до задуму.

В цілому, розділ 2 надав вичерпний огляд методології та кроків, що були виконані для розробки імплементації інтернет-магазину з продажу мобільних телефонів та аксесуарів.

РОЗДІЛ 3. РОЗРОБКА ТА ТЕСТУВАННЯ ВЕБ-САЙТУ НА ОСНОВІ DJANGO

3.1. Структура веб-сайту

Створимо веб-сайт на основі фреймворку Django. Ми будемо використовувати Django для створення різних функціональних частин сайту, таких як керування товарами та замовленнями, аутентифікація та авторизація користувачів, візуальний дизайн та багато іншого. Завдяки Django, розробка буде більш структурованою, а забезпечення безпеки та ефективності додатку буде здійснюватися на високому рівні (рис. 3.1).

У корені проекту:

- **.gitignore**: Файл ігнорування для Git.
- **db.sqlite3**: Файл бази даних SQLite.
- **manage.py**: Скрипт для керування Django-проектом.

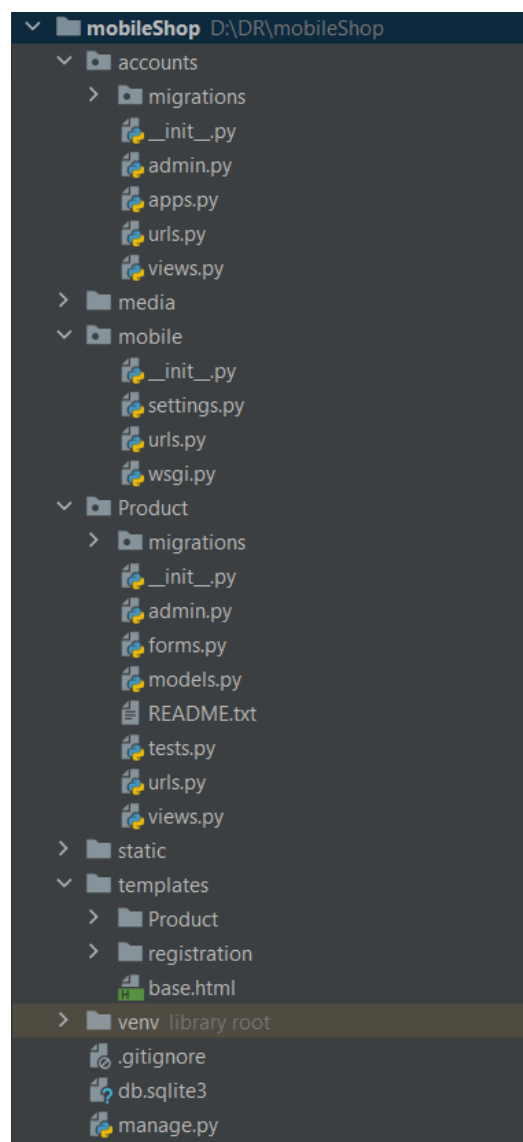


Рисунок 3.1 – Структура веб-сайту

Додаткові теки та файли вказують на такі частини проекту:

- **accounts**: додаток для обробки аутентифікації та авторизації користувачів.
- **media/Product-images**: зображення товарів для інтернет-магазину.
- **mobile**: налаштування проекту Django.
- **Product**: додаток для управління товарами та замовленнями.
- **static**: статичні файли (CSS, JavaScript, зображення) для візуального оформлення.

- **templates:** HTML-шаблони для рендерінгу сторінок.

Ця структура демонструє організацію нашого Django-проекту, розділяючи логіку на різні додатки та ресурси для зручності розробки та підтримки.

3.2. Програмування серверної частини веб-сайту

Програмування серверної частини веб-сайту - це процес створення логіки та функціональності, яка оброблює запити від клієнтів, взаємодіє з базою даних, оброблює дані та надає користувачам відповіді через браузер. Серверна частина веб-сайту відповідає за обробку бізнес-логіки, збереження та отримання даних, здійснення аутентифікації користувачів та забезпечення безпеки [23].

У випадку використання фреймворку Django для розробки серверної частини, програмування включає такі етапи:

1. **Створення моделей:** Визначення структури даних за допомогою класів моделей, які описують таблиці бази даних та взаємозв'язки між ними.

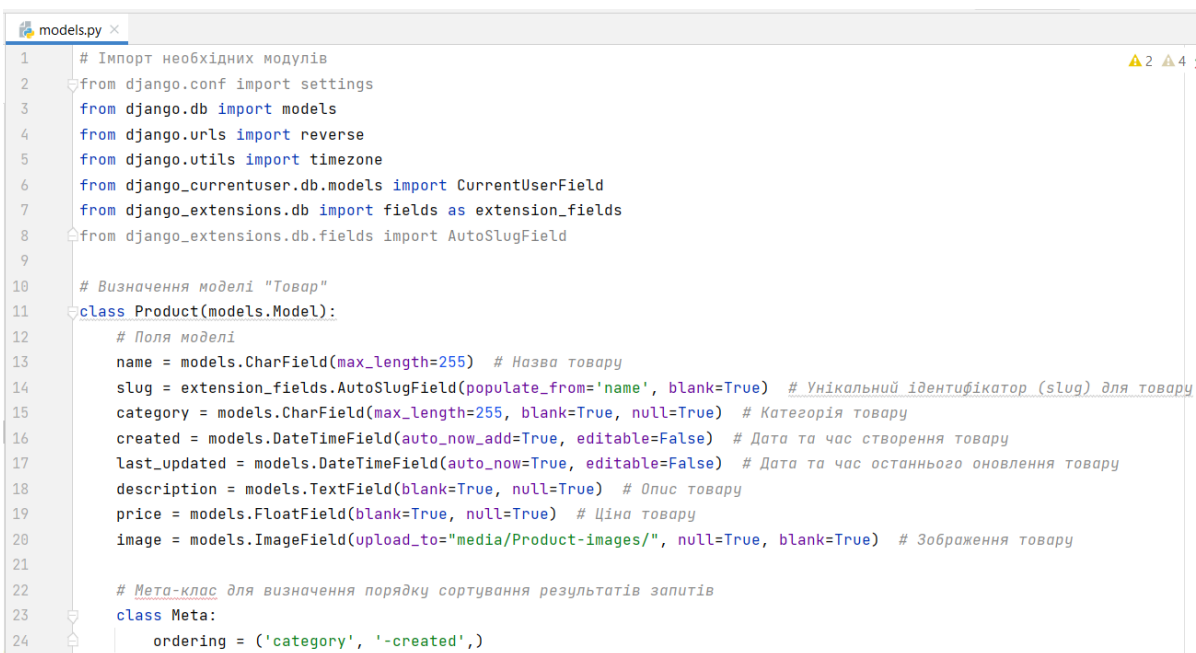
Створення моделей у фреймворку Django - це процес визначення структури даних, які будуть зберігатися в базі даних. Моделі визначаються за допомогою класів, які успадковуються від базового класу **models.Model**. Кожен атрибут цього класу представляє поле в таблиці бази даних, а взаємозв'язки між моделями визначаються за допомогою різних типів полів.

У вашому коді визначені дві моделі: **Product** та **Order** (рис.3.2). Давайте розглянемо кожну модель та її поля більш детально:

1) Модель **Product**:

- Поля:
 - **name:** Назва товару (тип поля **CharField**).
 - **slug:** Унікальний ідентифікатор для товару (тип поля **AutoSlugField**).
 - **category:** Категорія товару (тип поля **CharField**).
 - **created:** Дата та час створення товару (тип поля **DateTimeField**).

- **last_updated:** Дата та час останнього оновлення товару (тип поля **DateTimeField**).
- **description:** Опис товару (тип поля **TextField**).
- **price:** Ціна товару (тип поля **FloatField**).
- **image:** Зображення товару (тип поля **ImageField**).
- **Методи:**
 - **__str__(self):** Повертає рядок, який представляє товар (використовується для виведення в адміністративному інтерфейсі).
 - **get_absolute_url(self):** Генерує URL для перегляду деталей товару.
 - **get_update_url(self):** Генерує URL для редагування товару.



```

1  # Імпорт необхідних модулів
2  from django.conf import settings
3  from django.db import models
4  from django.urls import reverse
5  from django.utils import timezone
6  from django_currentuser.db.models import CurrentUserField
7  from django_extensions.db import fields as extension_fields
8  from django_extensions.db.fields import AutoSlugField
9
10 # Визначення моделі "Товар"
11 class Product(models.Model):
12     # Поля моделі
13     name = models.CharField(max_length=255) # Назва товару
14     slug = extension_fields.AutoSlugField(populate_from='name', blank=True) # Унікальний ідентифікатор (slug) для товару
15     category = models.CharField(max_length=255, blank=True, null=True) # Категорія товару
16     created = models.DateTimeField(auto_now_add=True, editable=False) # Дата та час створення товару
17     last_updated = models.DateTimeField(auto_now=True, editable=False) # Дата та час останнього оновлення товару
18     description = models.TextField(blank=True, null=True) # Опис товару
19     price = models.FloatField(blank=True, null=True) # Ціна товару
20     image = models.ImageField(upload_to="media/Product-images/", null=True, blank=True) # Зображення товару
21
22 # Мета-клас для визначення порядку сортування результатів запитів
23 class Meta:
24     ordering = ('category', '-created',)
  
```

Рисунок 3.2 – Створення моделей інтернет-магазину

2) Модель Order:

- **Поля:**
 - **name:** Назва замовлення (тип поля **CharField**).
 - **contact:** Контактна інформація (тип поля **CharField**).
 - **slug:** Унікальний ідентифікатор для замовлення (тип поля **AutoSlugField**).

- **created:** Дата та час створення замовлення (тип поля **DateTimeField**).
- **last_updated:** Дата та час останнього оновлення замовлення (тип поля **DateTimeField**).
- **address:** Адреса для доставки (тип поля **TextField**).
- **count:** Кількість товарів у замовленні (тип поля **IntegerField**).
- **cost:** Вартість замовлення (тип поля **IntegerField**).
- **delivered:** Прапорець, що позначає, чи замовлення доставлене (тип поля **BooleanField**).
- **delivered_on:** Дата та час доставки замовлення (тип поля **DateTimeField**).
- Взаємозв'язки:
 - **product:** Зовнішній ключ до моделі **Product** (тип поля **ForeignKey**).
 - **order_by:** Зовнішній ключ до користувача, який зробив замовлення (тип поля **CurrentUserField**).
- Методи:
 - **__str__(self):** Повертає рядок, який представляє замовлення.
 - **get_absolute_url(self):** Генерує URL для перегляду деталей замовлення.
 - **get_update_url(self):** Генерує URL для редагування замовлення.
 - **save(self, *args, **kwargs):** Перевизначений метод для збереження замовлення. Якщо замовлення доставлене, встановлює дату доставки.

Ці моделі визначають структуру даних, яка буде зберігатися у базі даних. Кожен об'єкт цих класів відповідає рядку у відповідній таблиці бази даних. Також вони дозволяють описати взаємозв'язки між даними, що допомагає забезпечити цілісність даних та зручний доступ до них.

2. **Роутинг та відображення:** Визначення URL-шляхів та функцій-відображень, які обробляють запити, виконують бізнес-логіку та генерують відповіді на основі шаблонів та даних (рис. 3.2).

```
urlpatterns = [
    path('admin/', admin.site.urls), # URL для адміністративної панелі Django
    path('', include('Product.urls')), # Включення роутінгу з Product/urls.py
    path('accounts/', include('accounts.urls')), # URL для ваших користувачів та облікових записів
    path('accounts/', include('django.contrib.auth.urls')), # Включення стандартних URL Django для облікових записів
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT) + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Рисунок 3.2 – Створення роутінгу інтернет-магазину

Цей файл **mobile/urls.py** визначає загальний роутінг для вашого веб-сайту. Давайте розглянемо кожен рядок детальніше:

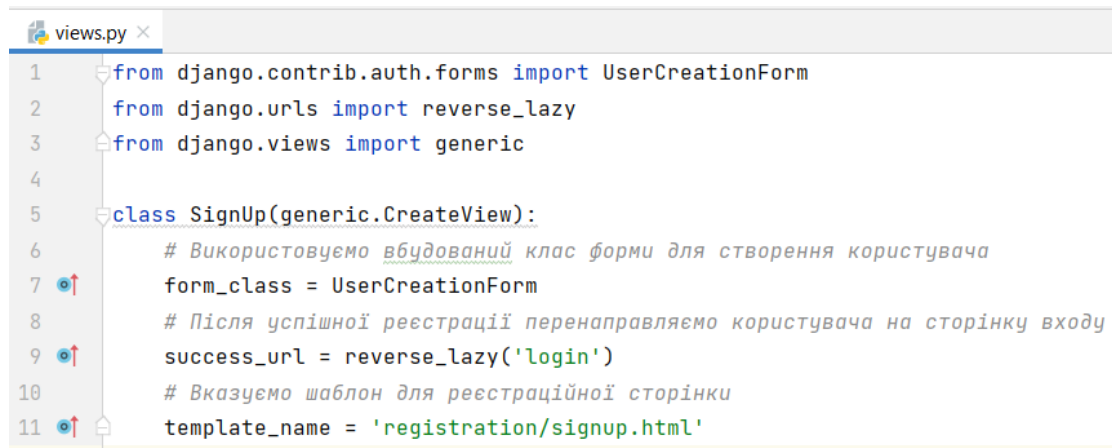
1. **path('admin/', admin.site.urls):** Цей шлях веде до адміністративної панелі Django, де можна керувати вашими моделями, даними тощо.
2. **path('', include('Product.urls')):** Цей шлях включає роутінг з файлу **Product/urls.py**, що дозволяє вам розподілити роутінг на більші менші модулі для легшого керування.
3. **path('accounts/', include('accounts.urls')):** Цей шлях веде до URL для ваших користувачів та облікових записів, він призначений для управління користувачами.
4. **path('accounts/', include('django.contrib.auth.urls')):** Цей шлях включає стандартні URL Django для облікових записів, такі як вхід, вихід, відновлення пароля тощо.
5. **+static(settings.STATIC_URL, document_root=settings.STATIC_ROOT):** Цей рядок додає URL та шлях до статичних файлів, таких як CSS, JavaScript.
6. **+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT):** Цей рядок додає URL та шлях до медіа-файлів, таких як зображення, завантажені користувачами.

У файлі **Product/urls.py**:

3. **Створення шаблонів:** Розробка HTML-шаблонів, які відображають дані користувачам і забезпечують візуальний дизайн веб-сторінок.

Створення шаблонів ми детально розглянули під час верстки сайту.

4. **Аутентифікація та авторизація:** Включення системи аутентифікації користувачів та визначення рівнів доступу до різних ресурсів сайту (рис. 3.3).



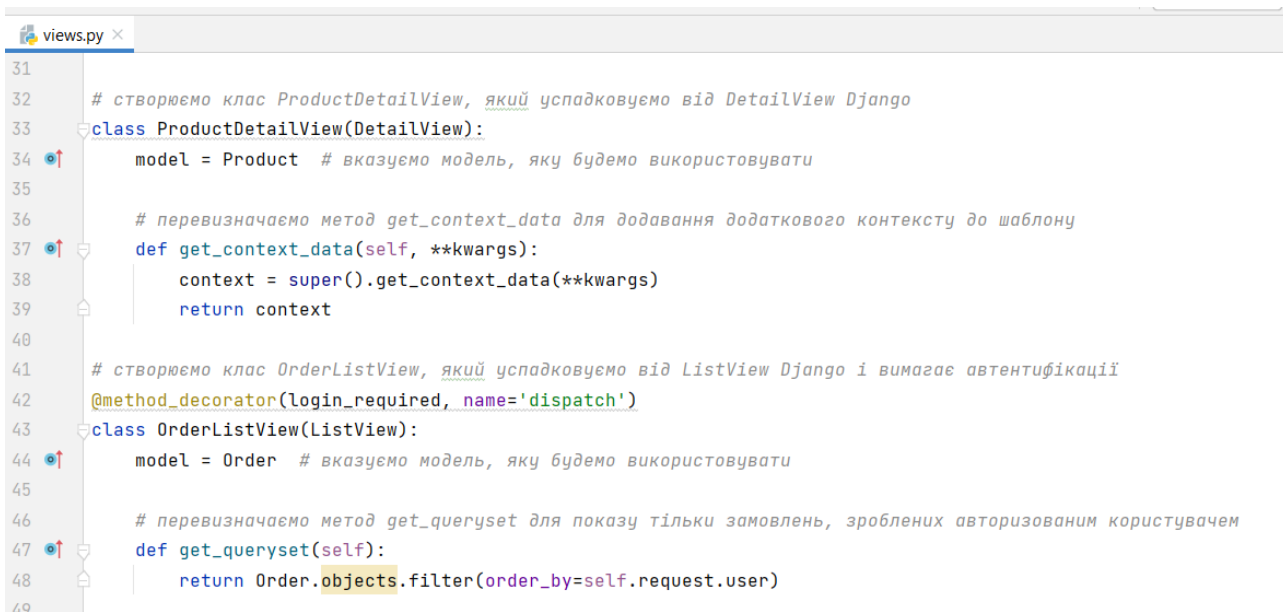
```
views.py
1 from django.contrib.auth.forms import UserCreationForm
2 from django.urls import reverse_lazy
3 from django.views import generic
4
5 class SignUp(generic.CreateView):
6     # Використовуємо вбудований клас форми для створення користувача
7     form_class = UserCreationForm
8     # Після успішної реєстрації перенаправляємо користувача на сторінку входу
9     success_url = reverse_lazy('login')
10    # Вказуємо шаблон для реєстраційної сторінки
11    template_name = 'registration/signup.html'
```

Рисунок 3.3 – Створення аутентифікації інтернет-магазину

- **UserCreationForm:** Це вбудована форма, яка використовується для створення нового користувача. Вона містить поля для введення ім'я користувача, паролю та підтвердження паролю.
- **success_url:** Після успішної реєстрації користувача, він перенаправляється на сторінку входу. **reverse_lazy** використовується для зворотного визначення URL за допомогою імені URLconf, тобто **'login'** вказує на URL для входу.
- **template_name:** Вказує шлях до шаблону, який відображатиме сторінку реєстрації. У цьому шаблоні зазвичай виводяться поля форми та вказуються додаткові стилі чи скрипти для відображення форми реєстрації.

Коли користувач відкриває сторінку реєстрації, відображається форма **UserCreationForm**. Користувач вводить необхідні дані (ім'я користувача, пароль тощо), натискає кнопку "Зареєструватися" і, якщо дані валідні, відбувається реєстрація користувача і його перенаправлення на сторінку входу за допомогою **success url**.

5. Логіка бізнес-процесів: Реалізація специфічних функцій та бізнес-логіки, пов'язаної з конкретною діяльністю сайту (рис. 3.4).



```
31
32 # створюємо клас ProductDetailView, який успадковуємо від DetailView Django
33 class ProductDetailView(DetailView):
34     model = Product # вказуємо модель, яку будемо використовувати
35
36     # перевизначаємо метод get_context_data для додавання додаткового контексту до шаблону
37     def get_context_data(self, **kwargs):
38         context = super().get_context_data(**kwargs)
39         return context
40
41 # створюємо клас OrderListView, який успадковуємо від ListView Django і вимагає автентифікації
42 @method_decorator(login_required, name='dispatch')
43 class OrderListView(ListView):
44     model = Order # вказуємо модель, яку будемо використовувати
45
46     # перевизначаємо метод get_queryset для показу тільки замовлень, зроблених авторизованим користувачем
47     def get_queryset(self):
48         return Order.objects.filter(order_by=self.request.user)
49
```

Рисунок 3.4 – Створення логіки бізнес-процесів інтернет-магазину

1. ProductListView:

- Цей клас успадковується від класу **ListView** з Django і використовує модель **Product**.
- Метод **get_context_data** додає додатковий контекст до шаблону, який буде відображений на сторінці. Залежно від переданого параметра **category**, фільтруються об'єкти моделі **Product** за вказаною категорією.
- Контекст, який містить список об'єктів **Product**, передається в шаблон для відображення.

2. ProductDetailView:

- Цей клас успадковується від класу **DetailView** з Django і використовує модель **Product**.
- Він дозволяє відображати деталі окремого об'єкта моделі **Product**.

3. OrderListView:

- Цей клас успадковується від класу **ListView** з Django і використовує модель **Order**.

- Для відображення цієї сторінки вимагається автентифікація користувача. Метод **get_queryset** повертає тільки замовлення, зроблені користувачем, який зараз авторизований.

4. **OrderCreateView:**

- Цей клас успадковується від класу **CreateView** з Django і використовує модель **Order** та форму **OrderForm**.
- Для створення замовлення користувача вимагається автентифікація. Шаблон **OrderForm** дозволяє користувачу вибрати кількість товару.
- Метод **get_context_data** додає додатковий контекст до шаблону, включаючи інформацію про обраний продукт.
- Метод **form_valid** перевіряє дані форми, встановлює об'єкту замовлення відповідний продукт і розраховує вартість замовлення на основі кількості та ціни продукту.

5. **OrderDetailView:**

- Цей клас успадковується від класу **DetailView** з Django і використовує модель **Order**.
- Для перегляду деталей окремого замовлення користувача вимагається автентифікація.

6. Функція **order_conform:**

- Ця функція відображає сторінку з повідомленням "Дякуємо за замовлення" після підтвердження замовлення.

Цей код забезпечує функціональність для відображення списку продуктів, деталей продукту, списку замовлень користувача та можливості створення та перегляду замовлень. Він також додає функціональність автентифікації для деяких сторінок, щоб забезпечити доступ тільки авторизованим користувачам.

В результаті програмування серверної частини веб-сайту ми отримаємо функціональний та безпечний додаток, який забезпечує зручну та ефективну взаємодію користувачів з даними та функціональністю сайту. Повний код розглянутих прикладів наведений в додатку Б.

3.3. Тестування роботи веб-сайту

Під час тестування роботи веб-додатку ми переконалися, що наш веб-додаток на платформі Django працює належним чином, перевіrivши різні сторінки та функціональності, які ми реалізували. Ми вручну тестували кожен з наступних сторінок і функцій:

1. **Список продуктів (ProductListView):** Ми переконалися, що список продуктів відображається на веб-сторінці, і що кожен продукт має вірну інформацію, таку як назва, категорія, опис та зображення.

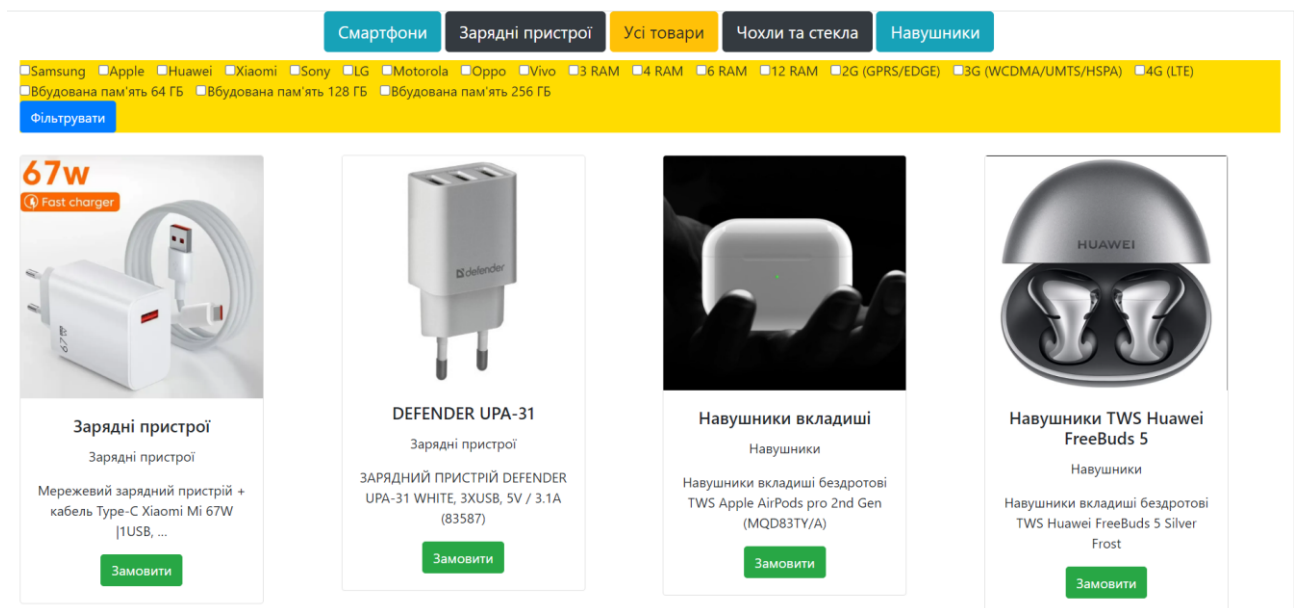


Рисунок 3.5 – Тестування сторінки список продуктів

Також ми перевірили, чи вірно фільтруються продукти за категоріями.

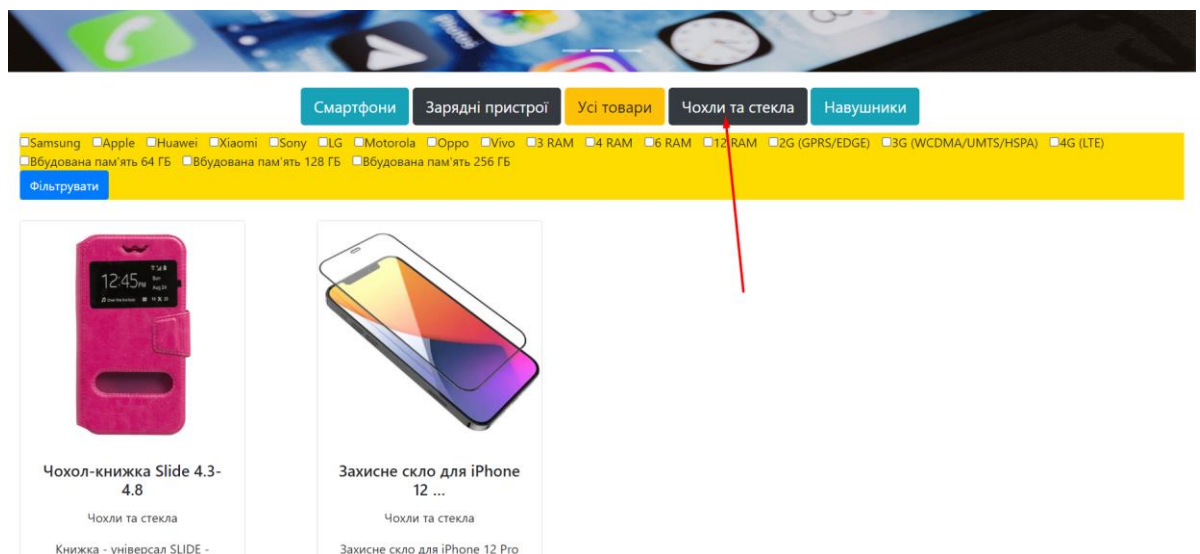


Рисунок 3.6 – Тестування фільтрації за категоріями

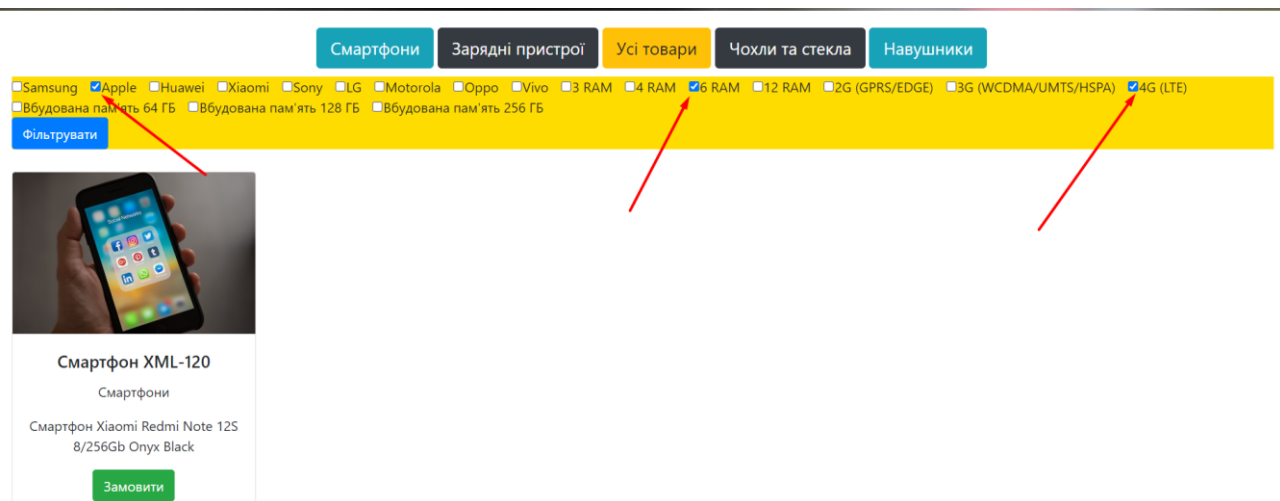


Рисунок 3.6 – Тестування фільтрації за технічними характеристиками

2. **Деталі продукту (ProductDetailView):** Ми переконалися, що при переході на сторінку деталей продукту відображається коректна інформація про продукт, така як назва, категорія, опис та зображення.

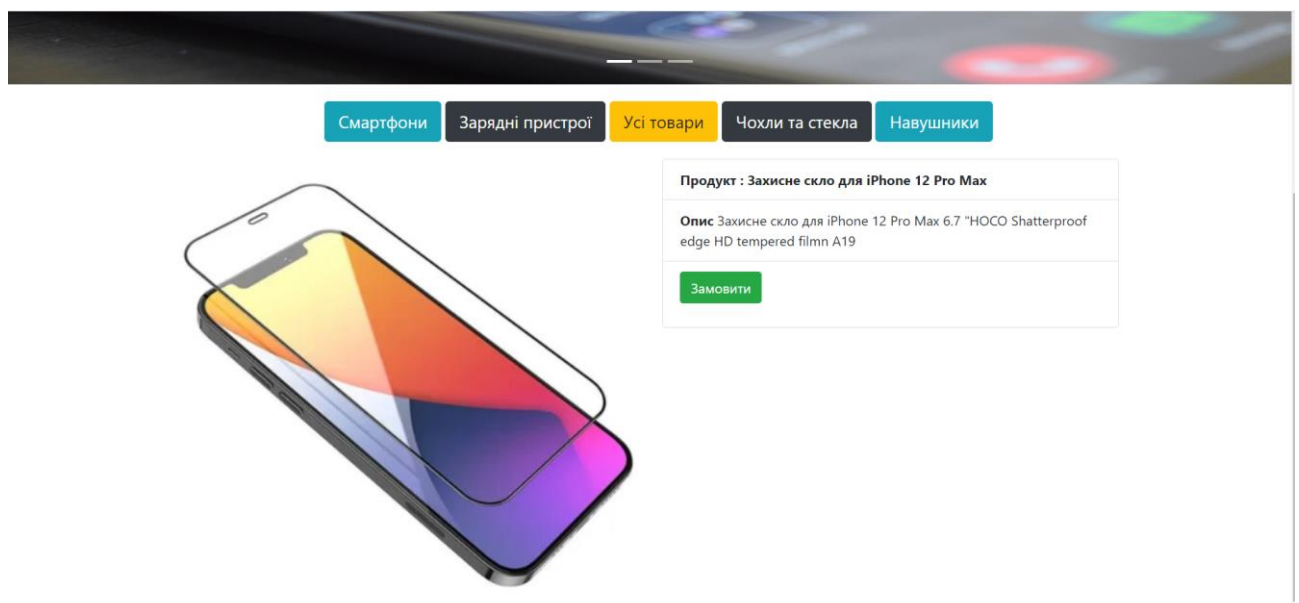


Рисунок 3.8 – Тестування сторінки з обраним товаром

3. **Створення замовлення (OrderCreateView):** Ми перевірили, що можемо створити нове замовлення для вибраного продукту, вводячи необхідну кількість. Ми впевнилися, що вартість замовлення правильно обчислюється на основі ціни продукту та кількості одиниць.

ЗАХИСНЕ СКЛО ДЛЯ IPHONE 12 PRO MAX



Count*

Name*

Contact

Address

Order

Рисунок 3.9 – Тестування сторінки створення замовлення

4. **Список замовлень (OrderListView):** Після автентифікації ми впевнилися, що список замовлень відображається коректно та показує лише замовлення, зроблені авторизованим користувачем.

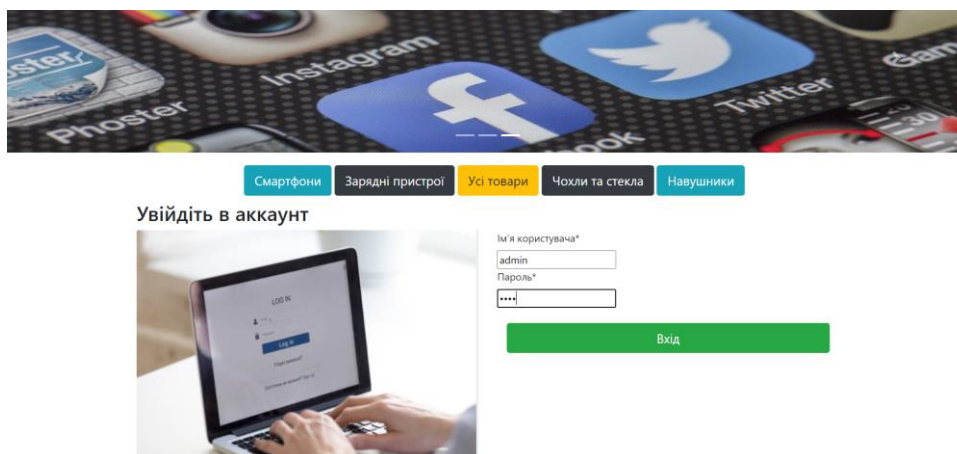


Рисунок 3.10 – Тестування сторінки авторизації

5. **Деталі замовлення (OrderDetailView):** Ми переконалися, що при переході на сторінку деталей замовлення відображається правильна інформація про замовлення (рис. 3.11).

Список замовлень

#	Замовлення	Кількість	Вартість	Замовлення відкрито	Останнє оновлення	Ім'я	Доставка на адресу	Доставлено	Виконано
1	ЗАХИСНЕ СКЛО ДЛЯ IPHONE 12 PRO MAX	3	414	24 серпня 2023 р. 18:50	24 серпня 2023 р. 18:50	Олексій	Житомир	Буде доставлено	Незабаром

Рисунок 3.11 – Тестування сторінки з деталями замовлення

6. **Підтвердження замовлення (order_conform):** Ми перевірили, що після підтвердження замовлення відображається сторінка з повідомленням "Дякуємо за замовлення" (рис. 3.12).

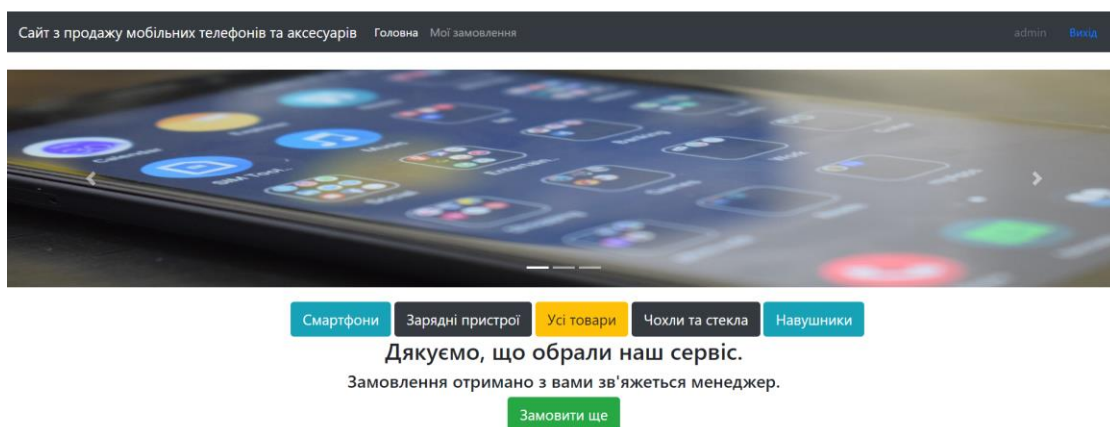


Рисунок 3.12 – Тестування сторінки Підтвердження замовлення
Тестування сторінок адмінки (рис. 3.13):

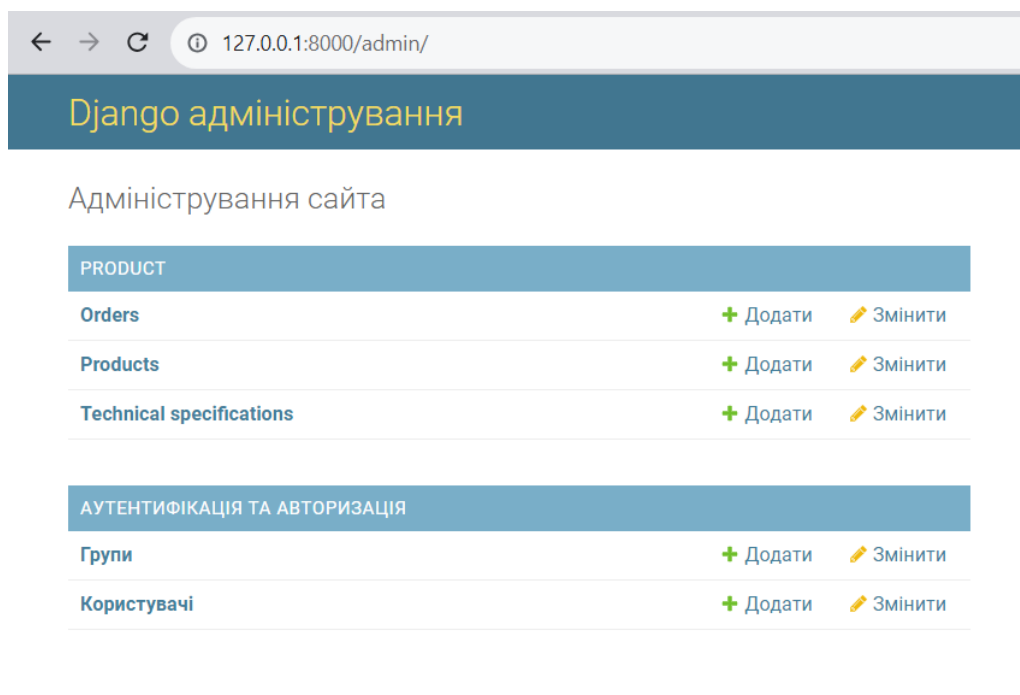


Рисунок 3.13 – Тестування сторінки адміністрування сайта

1. **Створення товару (ProductCreateView):**
 - а. Увійшовши до панелі адміністратора, ми перевірили, чи можемо знайти сторінку додавання нового товару.
 - б. На сторінці створення товару, ми ввели усі необхідні дані, такі як назва, категорія, опис, ціна та зображення.
 - с. Після збереження товару, ми переконалися, що дані правильно додалися до бази даних.
 - д. Ми також впевнилися, що після створення товару перенаправлення

відбувається на відповідну сторінку адмінки (список товарів або деталі нового товару).

2. **Редагування товару (ProductUpdateView):** а. Для перевірки редагування товару, ми обрали конкретний товар зі списку і перейшли на сторінку редагування. б. На сторінці редагування товару, ми змінили одне або декілька полів і зберегли зміни. с. Ми переконалися, що зміни правильно відобразилися на сторінці деталей товару адмінки та в базі даних (рис. 3.14).
3. **Видалення товару (ProductDeleteView):** а. Ми видалили один із створених товарів, перейшовши на сторінку видалення і підтвердивши видалення. б. Після видалення, ми переконалися, що відповідний товар більше не відображається на сторінці списку товарів адмінки та в базі даних.

Рисунок 3.14 – Тестування сторінки редагування товару

Під час ручного тестування ми впевнилися, що всі сторінки та функції працюють належним чином і надають користувачам очікуваний функціонал. Це дозволило нам забезпечити якісну роботу веб-додатку та впевненість в його функціональності перед впровадженням в реальному середовищі, а також

переконалися, що функціональність створення, редагування та видалення товарів в адмінці належним чином працює, а також що дані коректно зберігаються в базі даних.

Висновки до розділу 3

В розділі 3 роботи детально розглянуто процес розробки веб-сайту з використанням фреймворку Django. Розділ починається з опису структури веб-сайту, де наведено перелік необхідних файлів та папок, що використовуються для побудови функціональності веб-додатку. Ця інформація допомагає зрозуміти організацію проекту та розміщення його компонентів.

У пункті 3.2 "Програмування серверної частини веб-сайту" докладно описано процес створення моделей бази даних, які відображають товари та замовлення. Визначені відносини між ними та реалізована логіка зберігання даних. Також описано створення URL-адрес, класів-представників та методів для відображення списків, деталей та створення об'єктів.

Під час тестування роботи веб-сайту відзначено важливість тестування функціональності веб-додатку. Описано ручне тестування сторінок та функцій для переконання у їх правильному функціонуванні. Окреслено процес перевірки роботи різних аспектів, включаючи створення, редагування та видалення товарів та замовлень, а також аутентифікацію користувачів.

В підсумку, цей розділ надав глибокого розуміння процесу розробки веб-додатку з використанням Django. Описані кроки забезпечують стійку основу для подальшого розвитку, додавання нової функціональності та покращення користувацького досвіду.

ВИСНОВКИ

У даній кваліфікаційній роботі була розглянута розробка веб-додатку для продажу мобільних телефонів та аксесуарів на основі фреймворку Django. Під час виконання роботи було проведено аналіз сучасних рішень у галузі розробки подібних веб-сайтів, визначено постановку задачі та здійснено аналіз архітектури веб-додатку. Також були розглянуті різні інструменти та технології, включаючи вибір мови програмування, системи керування базами даних та фреймворку для створення веб-сайту.

У результаті роботи було розроблено структуру веб-сайту, включаючи моделі даних, сторінки продуктів та замовлень, а також здійснено програмування серверної частини веб-додатку. Використовуючи фреймворк Django, було створено класи моделей, представлені таблиці бази даних, і реалізовані різні функціональності, такі як додавання та перегляд продуктів, створення та керування замовленнями.

Для забезпечення якості та надійності веб-додатку, було проведено тестування роботи різних сторінок та функцій, включаючи перевірку роботи сторінок продуктів, замовлень та авторизації користувачів. Також було виконано ручне тестування, щоб переконатися у правильності функцій та коректному відображенні інформації на веб-сайті.

В цілому, розробка веб-додатку на основі Django дозволила створити функціональний інтернет-магазин для продажу мобільних телефонів та аксесуарів. Реалізована структура, функціональність та тестування підтверджують ефективне використання зазначеного фреймворку для розробки веб-додатків даного типу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rozetka / Інтернет-магазин Rozetka [Електронний ресурс] – Режим доступу до ресурсу: <https://rozetka.com.ua/>
2. Моюо.ua / Інтернет-магазин Моюо [Електронний ресурс] – Режим доступу до ресурсу: <https://mooyo.ua/>
3. Zhuk.ua / Інтернет-магазин Жжук [Електронний ресурс] – Режим доступу до ресурсу: [https:// zhuk.ua/](https://zhuk.ua/)
4. Що таке архітектура веб-сайтів? - визначення з техопедії - Розвиток 2023 [Електронний ресурс] // Icy Science. – Режим доступу: <https://uk.theastrologypage.com/website-architecture>
5. Ruby Programming Language [Електронний ресурс] // Ruby Programming Language. – Режим доступу: <https://www.ruby-lang.org/en/>
6. Welcome to Python.org [Electronic resource] // Python.org. – Mode of access: <https://www.python.org/> (date of access: 24.08.2023). – Title from screen.
7. PHP: Hypertext Preprocessor [Електронний ресурс] // PHP: Hypertext Preprocessor. – Режим доступу: <https://www.php.net/>
8. JavaScript | MDN [Електронний ресурс] // MDN Web Docs. – Режим доступу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
9. MySQL [Електронний ресурс] // mysql.com/. – Режим доступу: <https://www.mysql.com/>
10. PostgreSQL [Електронний ресурс] // PostgreSQL. – Режим доступу: <https://www.postgresql.org/>
11. SQLite Home Page [Електронний ресурс] // SQLite Home Page. – Режим доступу: <https://www.sqlite.org/index.html>
12. Django [Електронний ресурс] // Django Project. – Режим доступу: <https://www.djangoproject.com/>
13. Welcome to Flask – Flask Documentation (2.3.x) [Електронний ресурс] // Welcome to Flask. – Режим доступу: <https://flask.palletsprojects.com/en/2.3.x/>
14. Welcome to Pyramid, a Python Web Framework [Електронний ресурс] // Welcome to Pyramid. – Режим доступу: <https://trypyramid.com/>

15. Застосування UML в дипломних роботах :: Державний університет інформаційно-комунікаційних технологій [Електронний ресурс] // Головна :: Державний університет інформаційно-комунікаційних технологій. – Режим доступу: https://dut.edu.ua/ua/news-1-626-7758-zastosuvannya-uml-v-diplomnih-robotah_kafedra-kompyuternih-nauk-ta-informaciyних-tehnologiy
16. Діаграми UML для моделювання процесів і архітектури проекту [Електронний ресурс] // Evergreen - web розробка і діджиталізація бізнесу за допомогою AI продуктів. – Режим доступу: <https://evergreens.com.ua/ua/articles/uml-diagrams.html>
17. Що таке діаграма діяльності? - визначення з техопедії - Розвиток 2023 [Електронний ресурс] // Icy Science. – Режим доступу: <https://uk.theastrologypage.com/activity-diagram>
18. Махум Z. Діаграма послідовності (Sequence Diagrams) [Електронний ресурс] / Zosym Махум // Махум Zosym. – Режим доступу: <https://www.maxzosim.com/sequence-diagrams/>
19. Діаграми класів – UA5.org [Електронний ресурс] // UA5.org – Матеріали з інформаційних технологій. – Режим доступу: <https://ua5.org/oop/392-diagrami-klasiv.html>
20. Онлайн курс графічного дизайну «Професія дизайнер: вхід у графічний та веб-дизайн» - Навчання графічному WEB дизайну з нуля - Курси для цифрових дизайнерів - Wizeclub Education [Електронний ресурс] // Wizeclub Education. – Режим доступу: <https://wizeclub.education/courses/design-start/>
21. Photoshop [Електронний ресурс]. – Режим доступу: <https://www.adobe.com/ua/products/photoshop.html>
22. Bootstrap [Електронний ресурс] // Bootstrap · The most popular HTML, CSS, and JS library in the world. – Режим доступу: <https://getbootstrap.com>
23. Що таке backend-технологія? Необхідні навички та знання [Електронний ресурс] // MC.today, Media for Creators. – Режим доступу: <https://mc.today/uk/shho-take-backend-rozrobka/>

ДОДАТКИ

					ПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						3

Відредагований під вимоги Django product_detail.html код сайту

```

{% extends "base.html" %}
{% load static %}
{% block content %}

<div style="margin-top: 20px" class="container">
  <div class="row">
    <div class="col-md-6">
      <!-- Відображення зображення продукту -->
      
    </div>
    <div class="col-md-6">
      <div class="card">
        <ul class="list-group list-group-flush">
          <li class="list-group-item"><b>Продукт:
{{object.name}}</b></li>
          <li class="list-group-item"><b> Опис: </b>
{{object.description}}</li>

          <li class="list-group-item">
            <form method="post" action="{% url
'Product_order_create' object.slug %}" id="myForm">
              {% csrf_token %}
              {% if item %}
                <!-- Відображення ціни продукту -->
                <b>Ціна</b> <input id="price"
name="price" value="{{object.price}}">
              {% endif %}
              <!-- Кнопка для замовлення -->
              <button type="submit" class="btn btn-
success" onclick="submitForm()">Замовити</button>
            </form>
            <script>
              // Функція для відправки форми
              function submitForm() {
                var form =
document.getElementById('myForm');
                console.log(form);
              }

              // Функція для додавання ціни
              function addPrice() {
                price2 =
document.getElementById("price").value

                console.log(price2);
              }

```


Лістинг основних методів та класів

views.py

```

# імпортуємо необхідні класи і моделі з Django
from django.views.generic import DetailView, ListView, CreateView
from .models import Product, Order, TechnicalSpecification #
імпорт моделей
from .forms import OrderForm # імпорт форми OrderForm
from django.shortcuts import get_object_or_404, render, redirect
from django.contrib.auth.decorators import login_required
from django.utils.decorators import method_decorator
from django.views.generic import ListView
from .models import Product
from .forms import TechnicalFilterForm # Передбачаючи, що ваша
форма називається TechnicalFilterForm
class ProductListView(ListView):
    model = Product
    template_name = 'product_list.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['filter_form'] =
TechnicalFilterForm(self.request.GET)

        selected_specifications =
self.request.GET.getlist('technical_specifications')

        filter_set = Product.objects.all()

        if self.kwargs.get('category'):
            category = self.kwargs.get('category')
            if category == 1:
                filter_set =
filter_set.filter(category="Смартфони")
            elif category == 2:
                filter_set = filter_set.filter(category="Зарядні
пристрої")
            elif category == 4:
                filter_set = filter_set.filter(category="Чохли та
стекла")
            elif category == 5:
                filter_set =
filter_set.filter(category="Навушники")

        for spec_id in selected_specifications:
            filter_set =
filter_set.filter(technical_specifications__id=spec_id)

        context["products"] = filter_set
        return context

```

```

# створюємо клас ProductDetailView, який успадковуємо від
DetailView Django
class ProductDetailView(DetailView):
    model = Product # вказуємо модель, яку будемо використовувати

    # перевизначаємо метод get_context_data для додавання
    додаткового контексту до шаблону
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        return context

# створюємо клас OrderListView, який успадковуємо від ListView
Django і вимагає автентифікації
@method_decorator(login_required, name='dispatch')
class OrderListView(ListView):
    model = Order # вказуємо модель, яку будемо використовувати

    # перевизначаємо метод get_queryset для показу тільки
    замовлень, зроблених авторизованим користувачем
    def get_queryset(self):
        return Order.objects.filter(order_by=self.request.user)

# створюємо клас OrderCreateView, який успадковуємо від CreateView
Django і вимагає автентифікації
@method_decorator(login_required, name='dispatch')
class OrderCreateView(CreateView):
    model = Order # вказуємо модель, яку будемо використовувати
    form_class = OrderForm # вказуємо клас форми, який будемо
    використовувати
    success_url = '/order/conformed/'

    # перевизначаємо метод get_context_data для додавання
    додаткового контексту до шаблону
    def get_context_data(self, object_list=None, **kwargs):
        context = super(OrderCreateView,
self).get_context_data(**kwargs)
        context['product'] = get_object_or_404(Product,
slug=self.kwargs['slug']) # отримуємо вибраний продукт
        context['item'] = 'item'
        return context

    # перевизначаємо метод form_valid для встановлення значень
    продукту та вартості на основі введення з форми
    def form_valid(self, form):
        slug = self.kwargs.get('slug')
        product = Product.objects.get(slug__iexact=slug)
        form.instance.product = product
        form.instance.cost = int(form.instance.count) *
int(product.price)

        return super(OrderCreateView, self).form_valid(form)

# створюємо клас OrderDetailView, який успадковуємо від DetailView

```

					ІІЗ.КР.Б = 121 = 23 = ІІЗ	Арк.
						67

```

Django і вимагає автентифікації
@method_decorator(login_required, name='dispatch')
class OrderDetailView(DetailView):
    model = Order # вказуємо модель, яку будемо використовувати

# створюємо функцію для відображення сторінки "Дякуємо за
замовлення" після підтвердження замовлення
def order_conform(self):
    return render(self, 'Product/thanks.html')

```

models.py

```

# Імпорт необхідних модулів
# Імпорт необхідних модулів
from django.conf import settings
from django.db import models
from django.urls import reverse
from django.utils import timezone
from django_currentuser.db.models import CurrentUserField
from django_extensions.db import fields as extension_fields
from django_extensions.db.fields import AutoSlugField

class TechnicalSpecification(models.Model):
    # Поля моделі
    name = models.CharField(max_length=255)

    def __str__(self):
        return self.name

class Product(models.Model):
    # Поля моделі
    name = models.CharField(max_length=255) # Назва товару
    slug = extension_fields.AutoSlugField(populate_from='name',
blank=True) # Унікальний ідентифікатор (slug) для товару
    category = models.CharField(max_length=255, blank=True,
null=True) # Категорія товару
    created = models.DateTimeField(auto_now_add=True,
editable=False) # Дата та час створення товару
    last_updated = models.DateTimeField(auto_now=True,
editable=False) # Дата та час останнього оновлення товару
    description = models.TextField(blank=True, null=True) # Опис
товару
    price = models.FloatField(blank=True, null=True) # Ціна
товару
    image = models.ImageField(upload_to="media/Product-images/",
null=True, blank=True) # Зображення товару
    technical_specifications =
models.ManyToManyField(TechnicalSpecification, blank=True) #
Технічні характеристики товару

# Мета-клас для визначення порядку сортування результатів
запитів
class Meta:
    ordering = ('category', '-created',)

```

```

        # Метод, який повертає рядок для відображення об'єкту у
адміністративному інтерфейсі
    def __str__(self):
        return u'%s' % self.slug

    # Метод для отримання URL для перегляду деталей товару
    def get_absolute_url(self):
        return reverse('Product_product_detail',
args=(self.slug,))

    # Метод для отримання URL для редагування товару
    def get_update_url(self):
        return reverse('Product_product_update',
args=(self.slug,))

# Визначення моделі "Замовлення"
class Order(models.Model):
    # Поля моделі
    name = models.CharField(max_length=255) # Назва замовлення
    contact = models.CharField(max_length=10, null=True,
blank=True) # Контактна інформація
    slug = extension_fields.AutoSlugField(populate_from='name',
blank=True) # Унікальний ідентифікатор (slug) для замовлення
    created = models.DateTimeField(auto_now_add=True,
editable=False) # Дата та час створення замовлення
    last_updated = models.DateTimeField(auto_now=True,
editable=False) # Дата та час останнього оновлення замовлення
    address = models.TextField(blank=True, null=True) # Адреса
для доставки
    count = models.IntegerField(default=1) # Кількість товарів у
замовленні
    cost = models.IntegerField(default=0) # Вартість замовлення
    delivered = models.BooleanField(default=False) # Прапорець,
чи замовлення доставлене
    delivered_on = models.DateTimeField(blank=True, null=True) #
Дата та час доставки замовлення

    # Відносини
    product = models.ForeignKey(
        Product,
        on_delete=models.CASCADE, related_name="orders",
null=True, blank=True
    ) # Зовнішній ключ до моделі "Товар"
    order_by = CurrentUserField(blank=True, null=True,
related_name="orders_user", on_delete=models.CASCADE)
    # Зовнішній ключ до користувача, який зробив замовлення
(використовується додатковий пакет CurrentUserField)

    # Мета-клас для визначення порядку сортування результатів
запитів
    class Meta:
        ordering = ('delivered', '-created',)

```

						Арк.
					ІІЗ.КР.Б = 121 = 23 = ІІЗ	69

```

# Метод, який повертає рядок для відображення об'єкту у
адміністративному інтерфейсі
def __str__(self):
    return u'%s' % self.slug

# Метод для отримання URL для перегляду деталей замовлення
def get_absolute_url(self):
    return reverse('Product_order_detail', args=(self.slug,))

# Метод для отримання URL для редагування замовлення
def get_update_url(self):
    return reverse('Product_order_update', args=(self.slug,))

# Перевизначений метод збереження замовлення
def save(self, *args, **kwargs):
    if self.delivered:
        if not self.delivered_on:
            self.delivered_on = timezone.now()
    super(Order, self).save(*args, **kwargs)

```

					ПЗ.КР.Б = 121 = 23 = ПЗ	Арк.
						70