

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРНЕТ-МАГАЗИНУ НА БАЗІ
ФРЕЙМБОРКУ DJANGO
SOFTWARE IMPLEMENTATION OF ONLINE STORE BASED ON
DJANGO FRAMEWORK

Виконав: студент(ка) 4 курсу, групи КНТ-227
Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

Сопін В.С

(прізвище та ініціали)

Керівник Колпакова Т.О

(прізвище та ініціали)

Рецензент Голуб Т.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Сопіна Василя Сергійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація інтернет-магазину на базі фреймворку Django. Software Implementation of an Online Store Based on Django Framework.

керівник проєкту (роботи) Колпакова Тетяна Олексіївна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “03” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 26 травня 2021 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Розробка програми 4. Експлуатація та тестування програми дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Колпакова Т.О., доцент		
Нормоконтролер	Липовець М.В., асистент		

7. Дата видачі завдання “ 30 ” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Експлуатація та тестування програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент(ка)

(підпис) Сопін В.С.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Колпакова Т.О.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 107 с., 3 табл., 43 рис., 3 дод., 18 джерел.

ІНТЕРНЕТ-МАГАЗИН, КОМП'ЮТЕРНА ПРОГРАМА, PYTHON, ФРЕЙМВОРК, DJANGO.

Об'єкт дослідження – програмна реалізація Інтернет-магазину на базі фреймворку Django.

Предмет дослідження – вебсервіси на базі фреймворку Django.

Мета роботи – розробка Інтернет-магазину бази на фреймворку Django з використанням бази даних Sqlite.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування Python, мова тегів Html, фреймворк Django, база даних Sqlite, персональний комп'ютер з процесором Intel Core i5, під управлінням операційної системи Microsoft Windows 10.

Результати. Створено Інтернет-магазин, в якому користувач легко та швидко може придбати або виставити на продаж будь-який товар. Магазин було реалізовано на базі популярного фреймворку Django.

Висновки. Розроблено Інтернет-магазин, що спрощує та прискорює процес купівлі товару та автоматизує процес розміщення нового товару на вітрині.

Галузь використання – електронна комерція комерційних фірм та підприємств.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 107 pages, 3 tables, 43 figures, 3 appendices, 18 sources.

INTERNET-SHOP, COMPUTER SOFTWARE, PYTHON, FRAMEWORK, DJANGO.

The object of research is the software implementation of the online store based on the Django framework.

The subject of research – web services based on the Django framework.

The purpose of the work is to develop an online store based on the Django framework using the Sqlite database.

Materials, methods and technical means: object-oriented programming, Python programming language, Html tag language, Django framework, Sqlite database, personal computer with Intel Core i5 processor, running Microsoft Windows 10 operating system.

Results. Created an online store where the user can easily and quickly buy or sell any product. The store was implemented on the basis of the popular Django framework.

Conclusions. An online store has been developed that simplifies and speeds up the process of buying goods and automates the process of placing new goods on display.

Field of use – e-commerce of commercial firms and entrepreneurs.

ЗМІСТ

С.

Переклік скорочень та умовних познач	7
Вступ	8
1 Аналіз предметної області та огляд аналогів	9
1.1 Огляд проблеми створення Інтернет - магазину	9
1.2 Огляд програмних аналогів	19
1.3 Постановка задачі	27
1.4 Висновки за розділом	28
2 Розробка архітектури програми	29
2.1 Вибір фреймворку	29
2.2 Основні вимоги до Інтернет-магазину	33
2.3 Структурна схема Інтернет-магазину	33
2.4 Висновки за розділом	36
3 Розробка програми	37
3.1 Опис моделей реалізованої програми	37
3.1.1 Модель category	37
3.1.2 Модель product	38
3.1.3 Модель customer	39
3.1.4 Модель order	41
3.2 Структура проєкту	42
3.3 Висновки за розділом	45
4 Експлуатація та тестування програми	46
4.1 Опис основного функціоналу для клієнта	46
4.2 Опис основного функціоналу для продавця	51
4.3 Висновки за розділом	52
Висновки	53
Перелік джерел посилання	54
Додаток А Технічне завдання	56
Додаток Б Текст програми	59
Додаток В Слайди презентації	101

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

CMS – Content Management System;

FS – Full Stack;

HTML – Hypertext Markup Language;

MVC – Model-View-Controller

MVT – Model -View-Templates;

OC – OpenCart;

ORM – Object-Relational Mapping;

SEO – Search Engine Optimization;

WP – WordPress.

ВСТУП

Сьогодні уявити своє подальше існування без Інтернету, майже не можливо. Він стрімко увійшов в наше життя та швидко став невід'ємною його частиною. На сьогоднішній день всесвітня павутина вже перестала бути лише бібліотекою людських знань чи місцем для спілкування та обміну інформацією. Сьогодні Інтернет – це величезний торговий майданчик. На ньому можна продати чи, навпаки, купити майже будь-які послуги та товари. Саме тому для багатьох людей Інтернет це робоче місце чи універсальна торгівельна точка, що завжди знаходиться під рукою. Тому не дивно, що в наш час величезною популярністю користуються так звані Інтернет-магазини, що є в цілому цифровими аналогами звичних нам у повсякденному житті магазинів. Звісно, багато хто має бажання мати свою частинку Інтернету, в якій він міг би бути керівником маленької торгівельної компанії, мати ненапружений графік, працювати в своє задоволення. На тлі шаленої популярності Інтернет-комерції зросла і зацікавленість в процесі створення таких сайтів, адже у кожного свої вимоги.

Саме через це дана робота присвячена створенню Інтернет-магазину. Цей веб-сайт буде вирізнятись серед інших своєю простотою та зрозумілістю роботи як для продавця, так і для покупця. Оскільки особистий контакт продавця з покупцем не потрібен, цей вебресурс можна вважати актуальним у наш час.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД АНАЛОГІВ

1.1 Огляд проблеми створення Інтернет-магазину

Інтернет-магазин – це вебресурс, який дозволяє продавати послуги або товари через мережу Інтернет, а також надає можливість полегшити процес управління. Такі магазини, зазвичай, поєднують в собі деякі елементи торгівлі, в традиційному її вигляді, та прямого маркетингу. Головними відмінностями Інтернет-магазину, порівняно з традиційним є велика кількість товару, та інформації про нього, що подається користувачу в зручній формі, інтерактивність веб-ресурсу, доступність майже в будь-якому куточку планети та у будь-який час. Одним з найвагоміших недоліків електронних магазинів полягає в відсутності можливості оцінити товар тактильно та візуально. В основі Інтернет-торгівлі лежить структура, що ідентична структурі традиційної торгівлі. До основних функцій Інтернет-магазину відносять:

- надання користувачу повної інформації про доступні послуги та товари;
- прийом та обробка замовлень;
- обробка платежів (при умові підключення платіжної системи);
- збір та аналіз статистичної інформації.

Інтернет-магазин – це вебресурс, з допомогою якого споживачі безпосередньо купують товари чи послуги у продавця без посередницьких послуг через Інтернет. Покупці можуть відвідувати вебмагазин, не виходячи з дому та робити покупки, сидячі перед комп'ютером. Інтернет-магазини, як правило, працюють цілодобово, і багато споживачів мають доступ до Інтернету як на роботі, так і вдома. Тож їм дуже зручно робити покупки в Інтернеті.

Інтернет-магазин найкраще підійде тим, в кого є бажання швидко та ефективно почати продажі та недорого створивши online-вітрину з функцією обробки замовлень через Інтернет. Завдяки стрімкому зростанню технологій,

багато бізнес-організацій перейшли від традиційного способу продажу товарів до електронного. Такі організації використовують Інтернет як основний засіб здійснення комерційних операцій. Інтернет-магазини не мають обмеженого простору, і на веб-сайтах можна відображати найрізноманітніші товари. Це допомагає покупцям придбати товар після гарного пошуку. Від купівлі та продажу в Інтернеті є багато переваг. Найпоширеніші переваги це зручність, заощадження коштів, заощадження часу, різноманітність, відсутність тиску, легке порівняння.

Дуже зручно робити покупки з того місця, де ви знаходитесь. Сервіс доступний цілодобово, та в усіх країнах світу. Магазин працює на свята та вихідні. Завдяки постійно зростаючим цінам на бензин, покупки в Інтернеті економлять ваші витрати на проїзд до магазинів. Також більшість товарів ви можете придбати за зниженою ціною. Це, насамперед, обумовлено відсутністю плати за оренду приміщення, послуги охорони та посередників. Клієнти не повинні стояти в чергах до кас, щоб оплатити придбану ними продукцію. Вони можуть робити покупки з дому чи з місця роботи, і їм не доведеться витрачати час на поїздки. Клієнти також можуть шукати потрібні їм товари, вводячи ключові слова або використовуючи пошукові системи. Інтернет надає продавцям необмежену полицю, тому вони, швидше за все, пропонують широкий асортимент товарів, ніж у цегляних магазинах. Жоден продавець не знаходиться навколо і не вимагає від вас покупки у віртуальному або Інтернет-магазині. Покупки в Інтернеті позбавляють потреби блукати від магазину до магазину, порівнюючи ціни.

Проте, слід не забувати, що Інтернет-магазин для покупця має і ряд недоліків. До них відносять: відсутність можливості перевірити якість товару перед покупкою, відсутність можливості, як правило, повернення товару, загроза особистим даним, електронна реклама.

Іноді покупки в Інтернеті нагадають нам лотерею, адже якість товару чи його функціональність може відрізнятись від зазначених на сайті. Тож якщо ви виявили не відповідність якості чи функціональності замовленого

вами товару, то зазвичай, повернути його чи ваші гроші стає проблематично. Також слід мати на увазі, що Інтернеті існує безліч вразливостей, ваші особисті дані, які ви вказували при реєстрації, чи платіжні реквізити можуть бути розкриті хакерами та використані проти вас. Також можна стикнутись з електронною рекламою, яку Інтернет-магазини використовують для інформування своїх клієнтів щодо нової продукції чи акцій.

Як можна бачити, переваг Інтернет-покупок більше, аніж недоліків. Однак недоліки та підводні камені Інтернету не слід залишати без уваги. Так найважливішим недоліком покупців є ризик розкриття особистих даних. Також слід зазначити, що покупець не завжди може повернути товари, придбані в Інтернеті, навіть якщо має на це законне право. Згідно з законом «Захист прав споживача» кожен має право повернути товар в термін, який рівний двом тижням, незалежно від того де він був придбаний, в звичайному чи Інтернет-магазині.

У сучасних ділових умовах покупець має свободу вибору. Він може обрати консервативний погляд на купівлю товарів та послуг чи зробити вибір на користь Інтернет-магазину. Кожен має для себе оцінити недоліки та переваги Інтернет-покупок і обрати підходящий для себе варіант.

Збільшення обсягів продажу товарів через Інтернет, зробило життя людей з особливими потребами значно легше. Адже вони іноді не мають можливості відвідати традиційний магазин. Те саме стосується людей похилого віку.

Перейдемо до переваг та недоліків Інтернет-продажу для продавців. До переваг відносять: збільшення кількості клієнтів, без затрат на модернізацію приміщення, не обмежений асортимент, Інтернет-реклама, полегшення взаємодії з обслуговуючими банками і партнерами.

Оскільки Інтернет-магазин не використовує торгову площу, продавець має змогу покращити його привабливість для покупців за рахунок розширення асортименту чи покращення функціональності, затративши на це значно менше часу та ресурсів, аніж на модернізацію торгового приміщення.

Також слід пам'ятати, що з розвитком Інтернету стають доступні все нові види оплати товару, серед них можна виділити: розрахунок банківською картою, через електронний гаманець та розрахунок з мобільного телефону. Усе це значно облегшує процес розрахунку, який є дуже важливим для продавця. Оскільки Інтернет-магазин надає продавцю необмежену онлайн-вітрину, продавець може значно розширити свій асортимент товарів та послуг. Якщо звичайні магазини, зазвичай спеціалізуються на певній групі товарів, будь то споживчі, побутові чи будівельні товари. Інтернет-магазин позбавлений цього недоліку. Він може бути одночасно місцем де продаються різні групи товарів. За рахунок цього Інтернет-магазин отримує додаткових покупців, а продавець – додатковий дохід. Оскільки Інтернет-магазин – це вебресурс, для нього стає не ефективною звичайна реклама, яка як правило зустрічається в газетах, телевізорі, білбордах. Замість цього в Інтернет-комерції використовують особливі види реклам. Можна виділити 5 основних видів Інтернет-реклами : SEO, контекстна реклама, банерна реклама, тизерна реклама.

SEO або оптимізація пошукової системи – це органічний набір кроків, здійснених експертами в цій галузі, які забезпечують збільшення відвідування сайту за рахунок використання ключових слів, що роблять сайт більш видимими для пошукових систем. Головною перевагою SEO є довгострокова перспектива підвищення видимості сайту для пошукових систем [1]. Правила оптимізації пошукової системи структуровані пошуковими системами, кожна з яких використовує власні алгоритми та регулярно оновлюються, що складаються з безлічі елементів. І, хоча формула тримається в таємниці, експерти не знають, які елементи становлять основну частину. Дотримуючись їх, ви можете покращити свою позицію щодо ключових питань у результатах пошуку. Поведінка споживачів, зручність користувача та швидкість завантаження сайту мають важливе значення для результатів розвитку. З розвитком та складністю алгоритму значення вищезазначеного фактора постійно зростає, що в підсумку призводить до високоякісного пошуку.

Пошукові системи намагаються доставити найкорисніший вміст за запитом одного користувача. Як результат, власникам ресурсів потрібно буде публікувати вміст, який є найбільш цікавим для їх аудиторії. Якщо розглядати контекстну рекламу то такий вид реклами отримав свою назву від механізму її дії. Цей вид реклами підлаштовується під користувача та його інтереси. Спочатку відстежується активність користувача потім на основі цих даних користувачу демонструють рекламу, що відповідає його активності. Оголошення відображаються на необхідних сторінках через наявність ключових слів у налаштуваннях. Ось чому вони здаються актуальними в кожному конкретному випадку. Клацнувши на посилання на веб-сайті рекламодавця, користувач може побачити оголошення, яке більш-менш пов'язане з його інтересами. Відвідувачі можуть затримуватися допізна, ознайомитися з пропонованими товарами, оформити замовлення або повернутися пізніше. У будь-якому випадку, потенційні клієнти дізналися про цей сайт, і власник натиснув з якихось причин. Це означає, що однією з функцій контекстної реклами є інформування споживача про новий бренд (ресурси, товари тощо), тобто залучення потенційних клієнтів. Це особливо важливо при визначенні ефективності проекту. З цієї причини в Інтернеті є попит на продаж текстових повідомлень - тих, що впливають на аудиторію та спонукають її купувати. Лояльність клієнтів, як правило, залежить від цільової сторінки - сайту, де користувач робить покупку. Якщо це легко побачити, легко скласти каталог із матеріалами та відповідною темою реклами, це збільшить ваші шанси на цілеспрямовані дії. Показ оголошень на мобільних пристроях буде працювати лише у випадку, якщо у користувача є мобільна версія сайту. Це дуже простий і привабливий інструмент для просування вашого бізнесу. Банерну рекламу, Інтернет-магазини використовують найчастіше, бо таким чином можна легко виділити та приманити цільову аудиторію. Окрім прямого розміщення графічні оголошення також доступні для використання в Google AdWords. Ви можете побачити їх на будь-якому веб-сайті, вони включають в себе спливаючі

банери в нижній частині сайту чи в порожньому місці. Часто це виглядає як зображення та провокаційний, захоплюючий текст. Зображення ,як правило в форматі GIF абр JPG. Не так давно в Інтернеті з'явився новий вид банерів – rich media. Особливість цього виду баннеру полягає в можливості використовувати звуковий супровід, а також «реагувати» на рух мишки. В даному випадку оплата здійснюється за кількість показів. Головною проблемою цього виду реклами є блокири реклами, що досить часто встановлюється користувачами. Однак, слід зазначити, що цей вид реклами гарно підходить для вирішення багатьох маркетингових проблем таких як: підвищення впізнаваності про бренд, інформування про запуск продажу нових продуктів, або представлення нового дизайну старих товарів, загострення уваги на певних послугах чи товарах [2]. Про тизерну рекламу можна сказати, що суть цього специфічного виду реклами полягає в демонстрації цільовій аудиторії спеціальних оголошень з певними тригерами, щоб привернути увагу користувачів до певної послуги чи товару. Основною метою в даному випадку є привернення уваги та залучення покупців, що стануть новими клієнтами. Звичайно не можна забувати про Рекламу в соціальних мережах або так звану таргетовану рекламу, що задається відповідними стандартами. В даному випадку використовується особиста інформація про користувача на основі його профілю в соціальній мережі чи мережах. І вже на основі цих даних користувачу демонструють відповідну рекламу. В такому виді реклами оплата, як правило, відбувається за кліки або цільову дію, так як підписку, вподобання, тощо. Соціальні мережі, особливо Facebook пропонує дуже вдосконалений інструмент перегляду та аналізу статистичних даних [3] .

Зазначені вище переваги створюють комфортні умови для продавців Інтернет-магазинів, та роблять їх привабливими для тих хто хоче себе спробувати в цій області. Проте з недоліками Інтернет-магазинів необхідно бути знайомим, щоб вони не стали несподіванкою, для починаючого Інтернет-продавця. До головних недоліків відносять: витрати на впровадження системи, хакерська загроза, несплата за покупки.

Звичайно витрати на впровадження системи зовсім не відповідають витратам на відкриття звичайного магазину, проте зараз відкриття Інтернет-магазину потребує вкладання певних коштів. Наразі цей недолік потроху сходить на ні, адже кожного року збільшується кількість веб – програмістів. Також слід відмітити появу так званих «конструкторів сайтів», більше відомих як CMS (Content Management System) – система керування контентом. Говорячи професійною мовою, CMS, це «двигун», що являє собою збірку скриптів, що відповідають за керування та створення контенту на сайті. Найвідоміші CMS це WP. (WordPress), OC. (OpenCart). Не слід також забувати про те, що в наш час захисту інформації в Інтернеті приділяється велика увага, а нові версії антивірусних програм виходять майже кожного тижня, хакерська загроза не втрачає своєї небезпеки. Тому кожен продавець повинен остерігатись цієї загрози, та приймати усі необхідні дії для її запобігання. Ще одним страшним сном кожного Інтернет-продавця є несплата за покупку. На жаль у всі часи були люди, які прагнули отримати вигоду безоплатно, навіть якщо для цього необхідно буде обманути іншу людину. І в наші ці люди нікуди, на жаль, не поділись. Тому несплата за покупку це не так рідкісна річ. Це часто відбувається коли використовується спосіб оплати готівкою. У цьому випадку оплата відбувається по факту отримання товару після його доставки. Саме на цій стадії і відбувається ця неприємна несплата покупки. На жаль, продавцю не завжди вдається повернути товар, чи отримати за нього кошти.

Інтернет – це середовище, яке постійно змінюється, практично щодня відкривається щось нове. Інтернет-магазини стали для нас чудовим місцем з можливістю зручних покупок та місцем для реалізації наших комерційних талантів. Робота в Інтернеті більше не викликає неоднозначної реакції, це тепер повноцінна зайнятість, що приносить непоганий дохід, та має ряд переваг в порівнянні із звичайним робочим місцем. Існують два головні способи реалізації Інтернет-магазинів, це на базі CMS чи на базі фреймворку. Кожне з цих рішень має свої переваги та недоліки. CMS – це

середовище інструментів для створення та адміністрування сайту без знання мов програмування. Система управління сайтом, або CMS-движок, містить готові шаблони структур, набір функцій для проєктування та наповнення даними. CMS сайту дозволяє зручно підтримувати ресурс - контролювати доступ до вмісту, змінювати дані на сторінках, надсилати пошту. Сучасний ринок побудови сайтів пропонує користувачам різні типи CMS, орієнтовані на створення сайтів будь-якої складності – від простої візитної картки компанії до високонавантажених порталів [4].

До найбільш відомих CMS відносять WordPress, Open Cart та Joomla.

WordPress – система керування вмістом з відкритим кодом, яка через свою простоту в установленні та використанні широко застосовується для створення веб-сайтів. Сфера використання – від блогів до складних веб-сайтів. Вбудована система тем і плагінів у поєднанні з вдалою архітектурою дозволяє конструювати на основі WP. практично будь-які веб-проєкти [5].

Open Cart – це популярна система управління вмістом для Інтернет-магазинів, яка легко встановлюється на будь-якому хостингу з підтримкою PHP та MySQL. CMS знаходиться у вільному доступі через сховище GitHub і підтримується відомим розробником Даніелем Керром. Двигун побудований на сучасній архітектурі MVC і повністю безкоштовний. OC. – це швидка і надійна система для розробки інтернет-магазинів, яка містить широкий спектр функцій, які зроблять веб-ресурс привабливим для користувачів. Серед основних переваг двигуна – можливості пошуку та хороша технічна оптимізація. Крім того, сайт на OC. буде добре взаємодіяти з пошуковими системами та просувати за допомогою "доступних з коробки" SEO-інструментів. CMS OC. оптимальна для веб-проєктів малого та середнього бізнесу. Розробка веб-сайтів на OC., макет веб-сторінок на OC. доступні та зручні [6].

Joomla – це безкоштовна система для створення веб-сайтів. Це проєкт з відкритим вихідним кодом, який, як і більшість подібних проєктів, не стоїть на місці. Він дуже успішно розвивається, протягом ось уже семи років, і

користується популярністю у мільйонів користувачів по всьому світу. Слово «Joomla» є похідним від слова «Jumla» з африканської мови суахілі і означає "всі разом". Проєкт Joomla – результат палких дискусій між компанією Mambo Foundation, заснованої в серпні 2005 року, і її тодішньої команди розробників. Joomla використовується у всьому світі як для створення простих домашніх сторінок, так і складних корпоративних веб-сайтів. Вона проста в установці, управлінні і дуже надійна. Команда Joomla вдосконалюється з моменту свого створення, щоб краще задовольняти потреби користувачів [7].

Фреймворк – це інфраструктура програмного рішення, що полегшує розробку складних систем. Зручно, що цю інфраструктуру можна розглядати як комплексний тип бібліотеки. Програмне забезпечення - це набір готових до використання програмних рішень, що включає основні системи, логіку та функціональність для роботи з єдиною системою чи підсистемою. Відповідно, програмна база може включати утиліти, деякі бібліотеки коду, скрипти та взагалі все, що полегшує поєднання та інтеграцію різних програмних компонентів або готове та необхідне. Не швидко створення масштабних програмних продуктів. Загальна конструкція виробу зазвичай виконується на місцях API (Application Programming Interface) [8].

Фреймворки будуються на базі різних мов програмування, до найпопулярніших відносять :

- Java. Найвідоміші фреймворки на цій мові програмування – Spring Framework, PrimeFaces, Blade;
- Python. Найвідоміші фреймворки на цій мові програмування – Django, Flask, Tornado;
- PHP. Найвідоміші фреймворки на цій мові програмування це – Yii, Laravel.

Як бачимо, кожен спосіб має безліч рішень. Однак, щоб обрати, який саме спосіб для створення Інтернет-магазину треба розглянути які переваги та недоліки ці способи мають відносно один одного.

Порівняння переваги CMS та фреймворків наведено в табл. 1.1.

Таблиця 1.1 – Переваги CMS та фреймворків

Переваги CMS	Переваги фреймворків
Вже створено структуру Інтернет магазину, що економить час та бюджет.	Дозволяє реалізувати всі вимоги та задумки клієнтів.
Базова функціональність надається відразу «з коробки».	Дозволяє створити унікальний, з точки зору функціональності, проєкт.
Якщо вимоги прості та стандартні, реалізація буде легкою результатом – передбачуваним.	Додаючи нові функції можна адаптувати сайт таким чином, щоб втрата продуктивності була мінімальна.
У будь-який час ви можете легко змінити шаблон чи дизайн сайту.	Сайти, розроблені на базі Framework, працюють значно швидше, ніж аналоги які розроблені на базі CMS.

Розглянувши переваги, слід також розглянути і недоліки, адже часто саме недоліки грають вирішальну роль у виборі способу реалізації Інтернет-магазину

Порівняння недоліки CMS та фреймворків наведено в табл. 1.2.

Таблиця 1.2 – Недоліки CMS та фреймворків

Недоліки CMS	Недоліки фреймворків
Відсутність унікальності. Кожен може використати таку ж тему CMS.	Фаза розробки, як правило, займає більше часу ніж у випадку з CMS.
Продуктивність обмежена обраною CMS.	Необхідні високоосвічені спеціалісти, оскільки розробка значно складніша ніж побудова вебсайту на CMS.
Поновлення ліцензії має відбуватись щороку.	Усі зміни слід замовляти у розробників, адже ви не можете їх впровадити самостійно.
Стандартне рішення, як правило, включає в себе непотрібні функціональні можливості	Створення сайту, який по функціональності буде рівний аналогу на CMS, займе більше часу .

Технологія – це головний фактор, який впливає на ціну розробки. Загальновизнана думка, що створення Інтернет-магазину на CMS, завжди дешевше, ніж на фреймворк. Однак неможливо сказати, який варіант виявиться більш бюджетним. Для кожного проєкту є індивідуальне рішення.

Наприклад, для магазину, який не пропонує адміністративної панелі, але керується 1С, буде більш доцільно використовувати розробку на фреймворки, адже в цьому випадку мінімізуються витрати часу розробника на створення центру управління Інтернет-магазину. Також у випадку, якщо вам потрібний багатосторінковий Інтернет-магазин, який має складний функціонал так має унікальний дизайн, також буде більш доцільно використати розробку на фреймворку.

Незалежно від того, обрали ви CMS чи фреймворк, завжди слід досліджувати різні можливості для реалізації. Якщо вам знадобиться змінити розробника, важливо розуміти що на ринку є експерти, які зможуть довести ваш проєкт до кінця. Також поширені випадки, коли власники підприємств звертаються за технічною підтримкою до веб-сайтів, написаних на власній CMS. У цьому випадку унікальність коду CMS стає майже невирішеною проблемою, а власник – заручником вмінь одного розробника.

1.2 Огляд програмних аналогів

У наш час в Інтернеті є безліч Інтернет-магазинів найрізноманітніших тематик. Від магазинів машин до магазинів аксесуарів, в Інтернеті можна купити геть усе. Інтернет-магазини популярні також у сфері бджільництва. Оскільки в нашій країні цей вид діяльності активно розвивається, то з кожним роком попит на товари для бджолярів росте. Зараз в Україні функціонує близько тисячі різних Інтернет-магазинів даної тематики. Кожен з них має унікальний дизайн та по своєму приваблює покупців. Розглянемо деякі з них.

На рис. 1.1 зображена головна сторінка Бджоломагазину «Вулик».

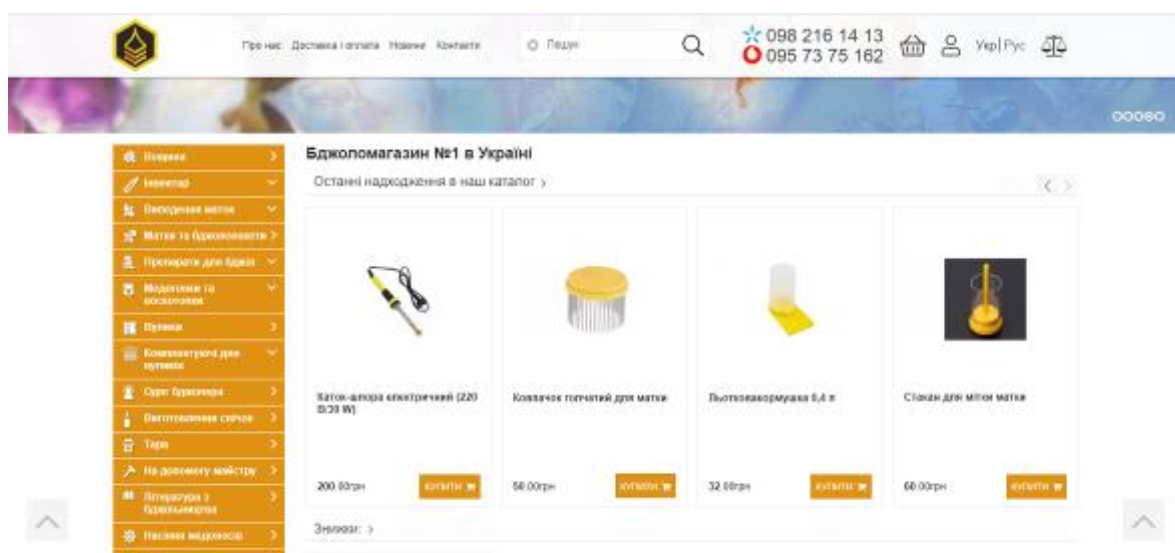


Рисунок 1.1 – Головна сторінка Бджоломагазину «Вулик»

На цій сторінці ми можемо бачити зручне меню категорій товару, що є найголовнішою перевагою цього Інтернет-магазину. Також в шапці сайту є зручна функція зміна мови, та наявність телефонів технічної підтримки. Слід також відмітити наявність відгуків на головній сторінці, що допомагають завоювати довіру покупців. Також можна відмітити сторінку «Контакти», на якій окрім телефону та адреси, знаходиться зручна інтерактивна мапа, що значно спрощує процес пошуку відділення магазину в реальному світі.

Сторінка «Контакти» зображена на рисунку 1.2.

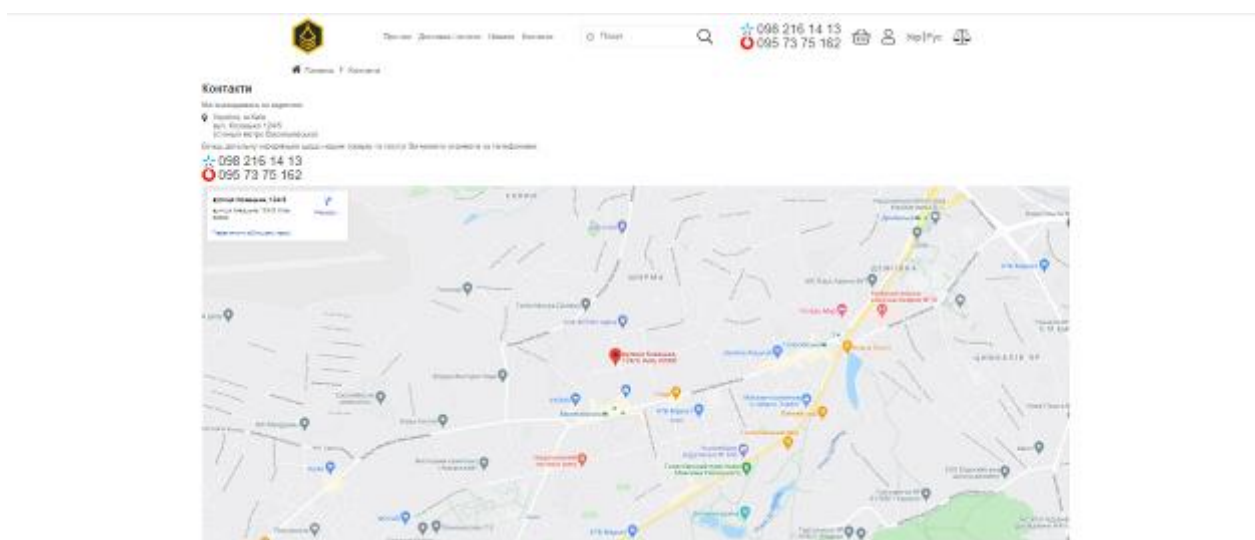


Рисунок 1.2 – Сторінка «Контакти»

Однак однією з найголовніших деталей будь-якого Інтернет-магазину, є сторінка товару.

Сторінка товару Бджоломагазину «Вулик» зображена на рисунку 1.3.

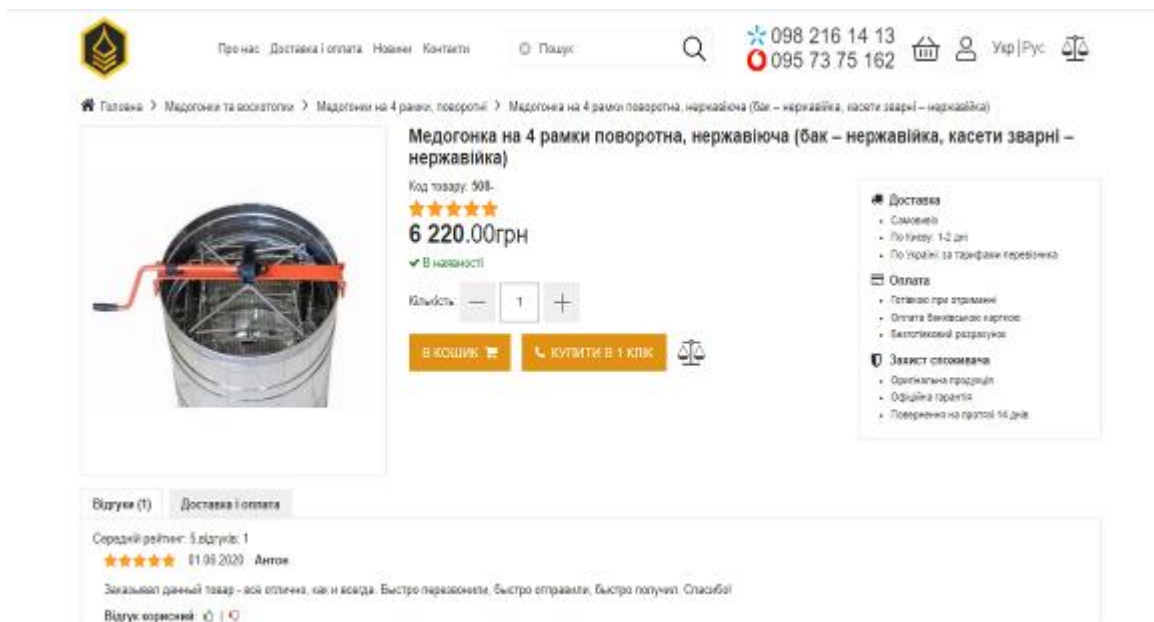


Рисунок 1.3 – Сторінка товару Бджоломагазину «Вулик»

На цій сторінці залишається шапка сайту, однак ця сторінка не містить зайвої інформації. В наявності також функція відгуків, щоб кожен покупець мав можливість прокоментувати товар, та детальна інформація про доставку товару та його оплату.

На сайті присутня сторінка «Доставка і оплата». В ній наявна інформація щодо реквізитів, доставки по містами та по Україні, доставки за кордон. Звичайно, в структурі вебсайту така сторінка не є обов'язковою, однак наявність такої сторінки значно підвищує шанси того, що користувач зможе чітко ознайомитись з можливостями Інтернет-магазину, що в свою чергу тягне за собою те, що користувач з більшою вірогідністю захоче щось придбати на сайті.

Сторінку «Доставка і оплата» зображено на рисунку 1.4.

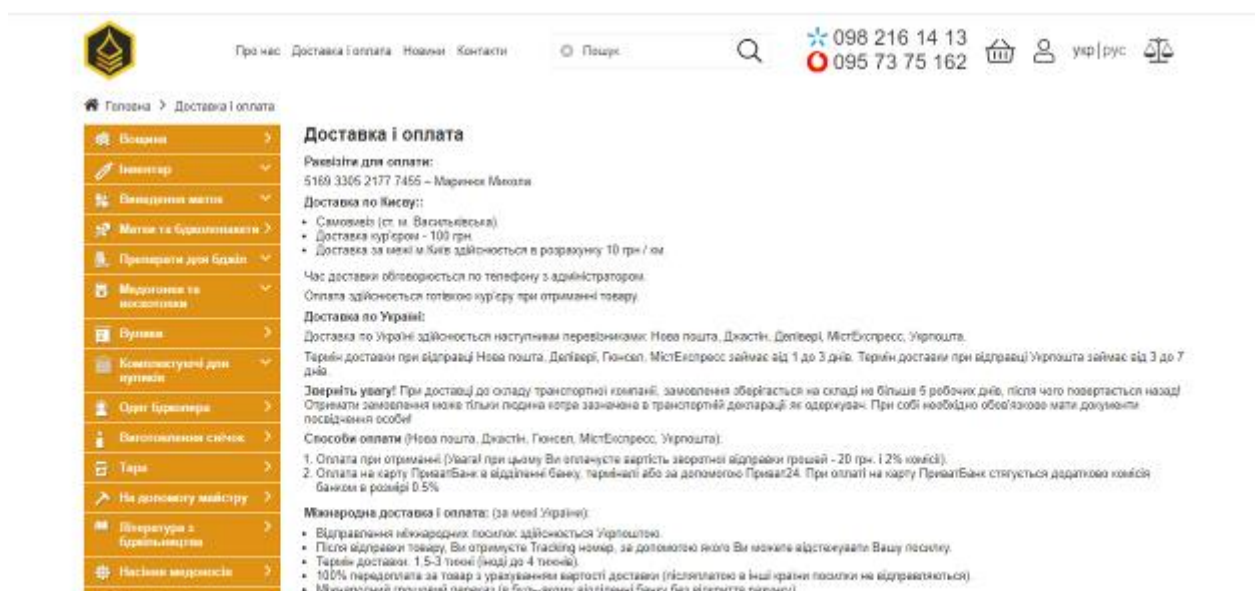


Рисунок 1.4 – Сторінка «Доставка і оплата»

І звичайно ж, розглядаючи будь-який Інтернет магазин потрібно розглянути його корзину. Як правило, на такій сторінці окрім знайомої нам шапки, наявна форма, яку слід заповнити для підтвердження та обробки замовлення.

Сторінку замовлення Бджоломагазину «Вулик» зображено на рисунку 1.5.

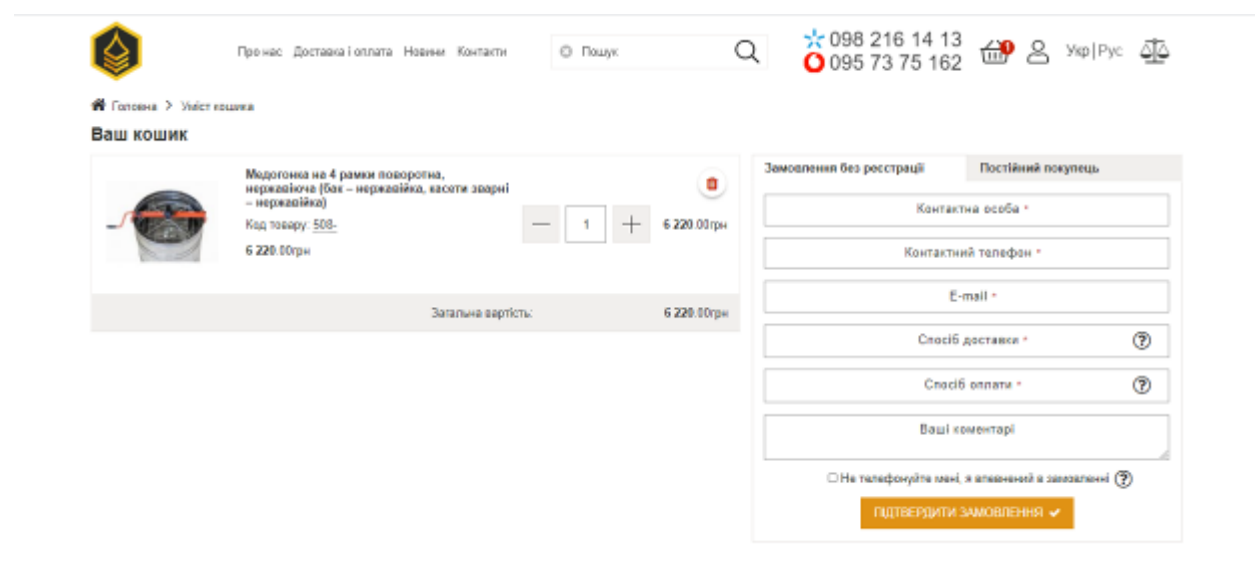


Рисунок 1.5 – Сторінка замовлення Бджоломагазину «Вулик»

На головній сторінці, на відмінну від Бджоломагазину «Медок», є лише меню категорій товару, невеличка реклама товарів. Це дуже вигідно виризняє цей Інтернет-магазин від інших. Відсутність зайвої інформації позитивно відображається на залученні нових покупців. Також на цій сторінці присутня інформація про магазин. Така головна сторінка, на мою думку, у порівнянні з головною сторінкою «Вулика» значно краще, адже на ній немає зайвої інформації, і покупець за потреби може відразу легко перейти до категорії, що йому цікава.

Розглядаючи сторінку товару, можна сказати що стилістично вона виконана дуже вдало. Тут немає нічного зайвого, лише товар, інформація про нього, так кнопка «додати до кошика». Мінімалістичний дизайн не відлякує покупців. Також, якщо більш детально розглянути інформацію про товар, то вона тут наведена досить розгорнуто.

Детальна сторінка товару зображена на рисунках 1.8 , 1.9 та 1.10.

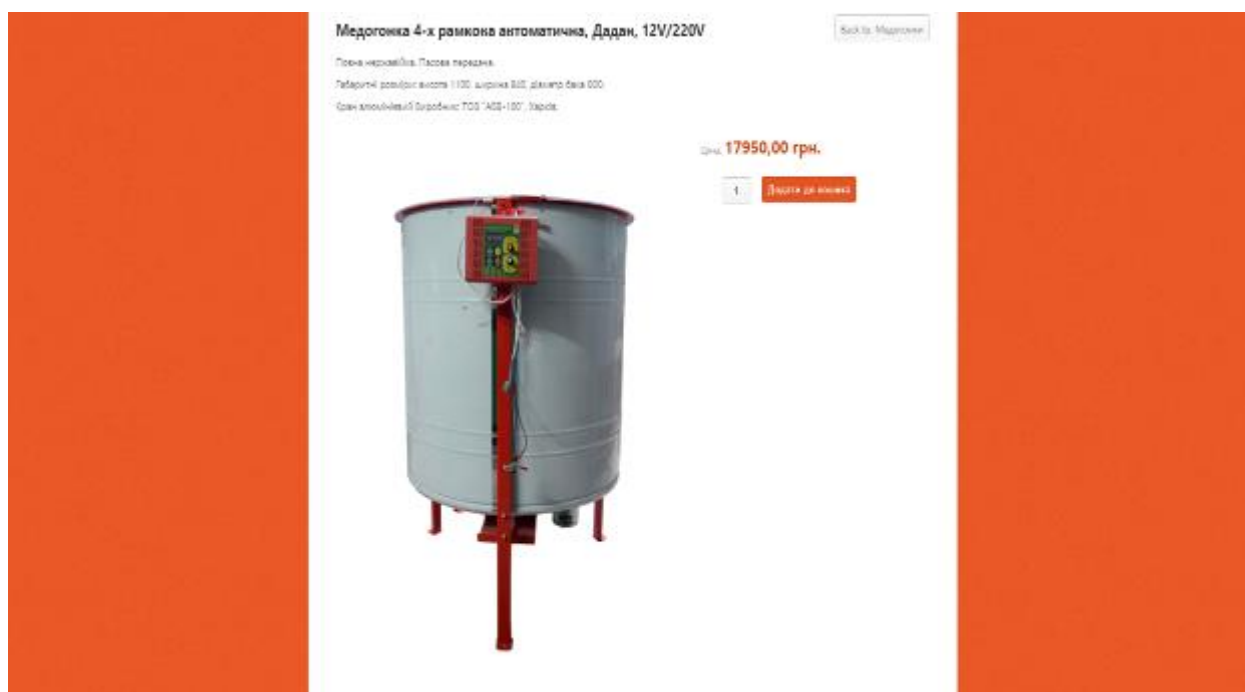


Рисунок 1.8 – Сторінка товару Бджоломагазину «Медок»



Рисунок 1.9 – Продовження сторінки товару Бджоломагазину «Медок»



Рисунок 1.10 – Продовження сторінки товару Бджоломагазину «Медок»

На останок розглянемо сторінку замовлення Бджоломагазину «Медок». Отож першим кроком є перелік обраних товарів, щоб покупець краще бачив, які товари будуть формувати його замовлення. Другим кроком є підтвердження форми, яка автоматично формує замовлення. В ній інформація схожа на інформацію з першого кроку, проте є більш лаконічною та короткою. Третій і останній крок являє собою форму особистих даних, заповнивши яку ваше замовлення почне оброблятися.

Три ступені підтвердження замовлення зображено на рисунках 1.11 – 1.13 .

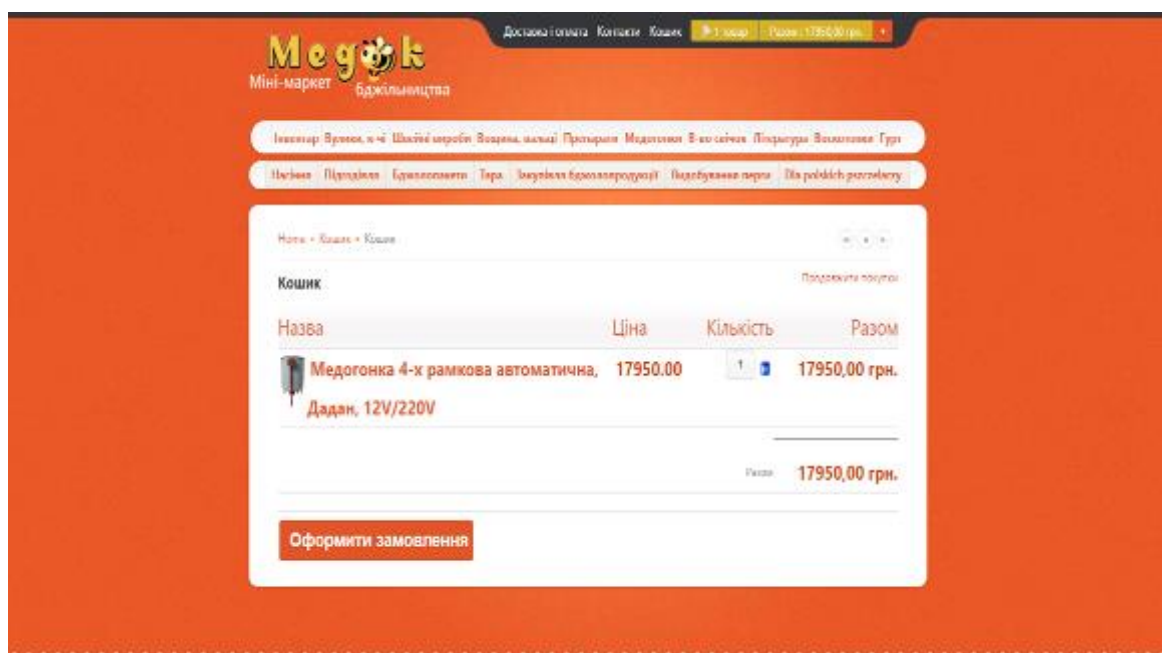


Рисунок 1.11 – Сторінка замовлення Бджоломагазину «Медок»

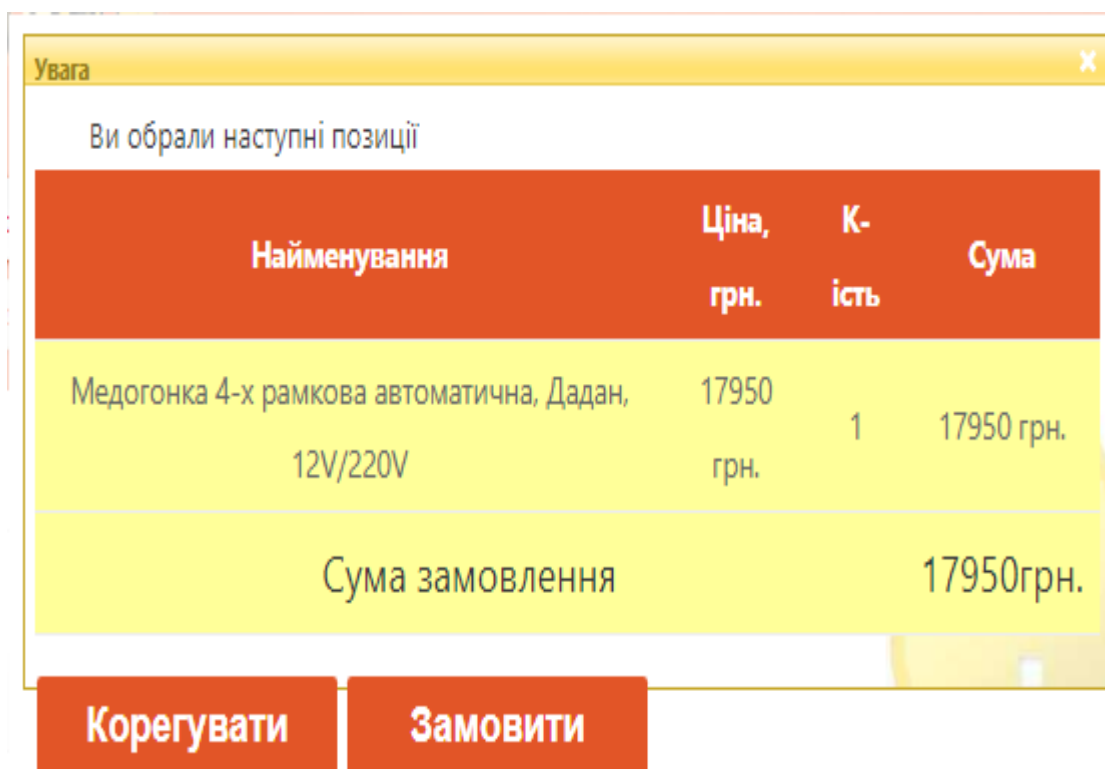


Рисунок 1.12 – Сторінка перегляду замовлення Бджоломагазину «Медок»

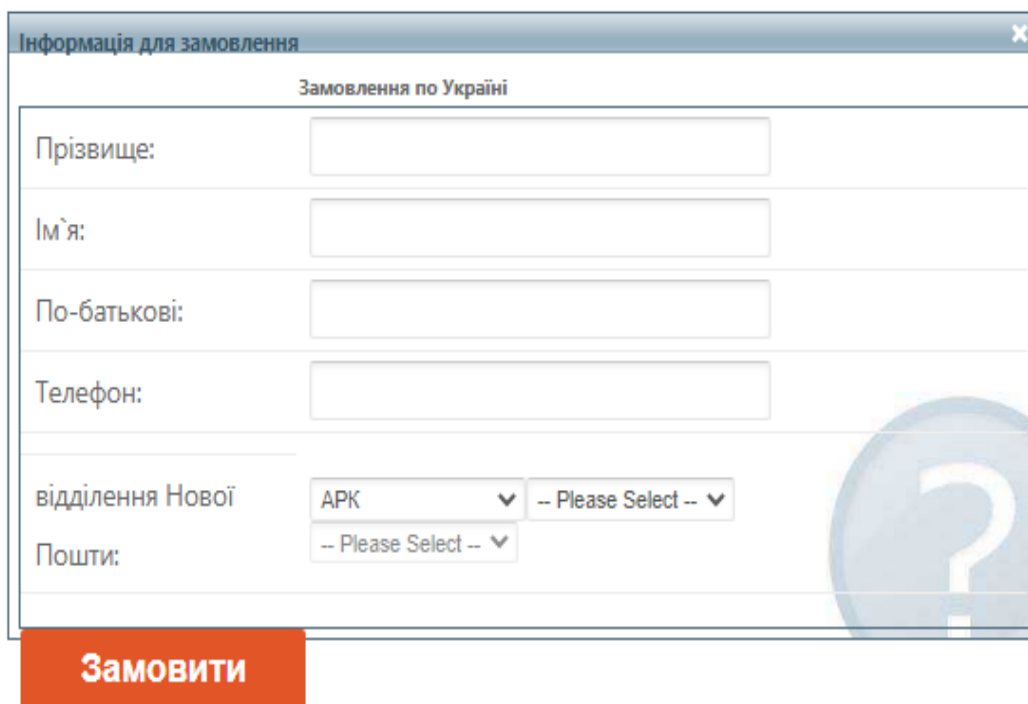


Рисунок 1.13 – Сторінка замовлення Бджоломагазину «Медок»

Розглянувши два Інтернет-магазину, ми побачили, що кожен з них має свій унікальний дизайн, функції та погляд на те, який саме має бути вебресурс у цій сфері.

1.3 Постановка задачі

Метою роботи є розробка Інтернет-магазину, який допоможе людям, що хочуть почати онлайн – торгівлю, надавши зручний функціонал для продажу товару через цей сервіс. Основною ідеєю сайту – спростити процес продажу та купівлі товарів як для продавців так і для покупців. Саме тому цей – ресурс буде користуватися популярністю. Для досягнення мети необхідно :

- розглянути популярні фреймворки для створення Інтернет-магазину, та обрати фреймворк для розробки;
- створити прототип проекту, що буде відрізнятись від відомих рішень;

- виконати розробку frontend та backend частини;
- протестувати розроблений програмний продукт.

1.4 Висновки за розділом

Було розглянуто переваги та недоліки Інтернет-магазинів для покупців та для продавців, а також основні способи їх реалізації. Були розглянуті програмні аналоги, а саме Бджоломагазини «Вулик» та «Медок». Виконано постановку задачі роботи.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вибір фреймворку

На сьогоднішній день для створення веб-сайту підходять багато мов програмування. Найпоширенішими в цій галузі є Java, PHP, Python та інші. Кожна з них має свої особливості, та переваги.

Для створення торгового майданчика була обрана мова програмування Python. Однак, чому було обрано саме Python? Python – це потужна мова програмування, якою легко оволодіти. Вона має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Легкий синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять його ідеальним для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ [9].

В порівнянні Java та PHP, Python демонструє високу ефективність, та досить легкий для розуміння код. Саме це, а також простий синтаксис, допомагає новачку в світі програмування легко вивчити Python.

Для Python існує досить багато фреймворків. Однак всіх їх можна поділити на дві категорії: мікрофреймворки та Full-stack фреймворки.

До переваг мікрофреймворків можна віднести те, що вони компактні та нагадують конструктор, де ви можете налаштувати все за вашим бажанням.

Однак вони в собі містять лише мінімальний, найнеобхідніший функціонал для роботи. Найпопулярнішим мікрофреймворком є Flask.

Flask є мікрофреймворком, які мають модульний дизайн. Він дозволяє розробляти безліч додатків, при цьому швидкість розробки велика за рахунок коробкових рішень. Це зручний фреймворк Python, в якому є можливість вибору розширень [10].

FS (Full Stack) фреймворки виділяються тим, що мають велику кількість вбудованого функціоналу, зокрема ORM (Object-Relational-

Mapping), систему, що спрощує роботу з функціоналом та базами даних. Найкращим Full-stack фреймворком є Django.

Django – є надзвичайно популярним і повнофункціональним серверним веб-фреймворком, що написаний на Python. Даний модуль показує вам, чому Django є одним з найпопулярніших структур веб-сервера, а також як налаштувати середовище розробки, і як приступити до роботи щоб створювати власні веб-додатки [11]. То ж вибір був між Flask та Django.

В таблиці 2.1 наведені головні розбіжності цих двох фреймворків.

Таблиці 2.1 – Головні розбіжності Flask та Django

Критерії	Фреймворк Flask	Фреймворк Django
1	2	3
Об'єкт Request	Використовує локальні потоки даних.	Відображає запит лише в тому місці, де це необхідно.
Бази даних	SQLAlchemy, функціонал, що спрощує управління БД	Вбудована ORM система має чудову систему міграцій, що значно спрощує управління БД.
Форми	Відсутні вбудовані форми, однак для подолання цієї проблеми використовують WTForms.	Django доступний відразу з вбудованими формами, що чудово співпрацюють з ORM системою
Аутентифікація	В якості інструмента для роботи з реалізації функції аутентифікації, Flask надає безпечні cookie файли.	Django по замовчуванню надає додаток, аутентифікації, що значно спрощує розробку всього проєкту.

Продовження таблиці 2.1

1	2	3
Панель адміністратора	В Flask не вбудована адмін – панель. Проте існує достеменно поширене та популярне розширення для Flask, що має назву Flask-Admin.	Django включає в себе повністю інтегровану в систему адмін – панель, для управління даними додатка.
Відомі проєкти	Red Hat, Netflix, Reddit, RackSpace, Airbnb, Lyft.	Instagram, Spotify, Dropbox, Mozilla, Pinterest.

Для роботи було обрано саме Django. Адже Django, на відміну від Flask, має вбудовані форми, що легко інтегруються в систему ORM, та адмін – панеллю. Ще в цей фреймворк вбудовано функції аутентифікації, що значно спрощує розробку сайту. Django – це веб-фреймворк з відкритим кодом, написаний на Python, який є безкоштовним. Веб-фреймворк, це в свою чергу набір компонентів, що значно полегшують розробку сайтів, сервісів, чи мобільних додатків.

При створенні сайту, як правило, завжди необхідні деякі однакові компоненти, такі як панель керування сайтом, реєстрація, авторизація та форми. Люди з часом помітили це, тому вони об'єднали їх в фреймворки. Django, чудовий приклад. Django використовує структуру MVT, яка розшифровується як Model View Templates. В MVT структурі: Model – шар доступний для даних, View – шар бізнес-логіки, Templates – шар представлення даних [12].

– Model – модель буде виступати в якості інтерфейсу ваших даних. Він відповідає за ведення даних. Це логічна структура даних, що лежить в основі всього програми, яка представлена базою даних (зазвичай це реляційні бази даних, такі як MySQL, Postgres) [13];

– View – це означає представлення даних в певному форматі, ініційоване рішенням контролера представити дані. Це системи шаблонів на основі популярних сценаріїв, такі як JSP, ASP, PHP [14];

– Templates – містить статичний компонент бажаного Html-виводу, а також конкретний синтаксис, який описує спосіб вставлення динамічного змісту [15] .

Схема роботи MVT зображена на рисунку 2.1 [16].

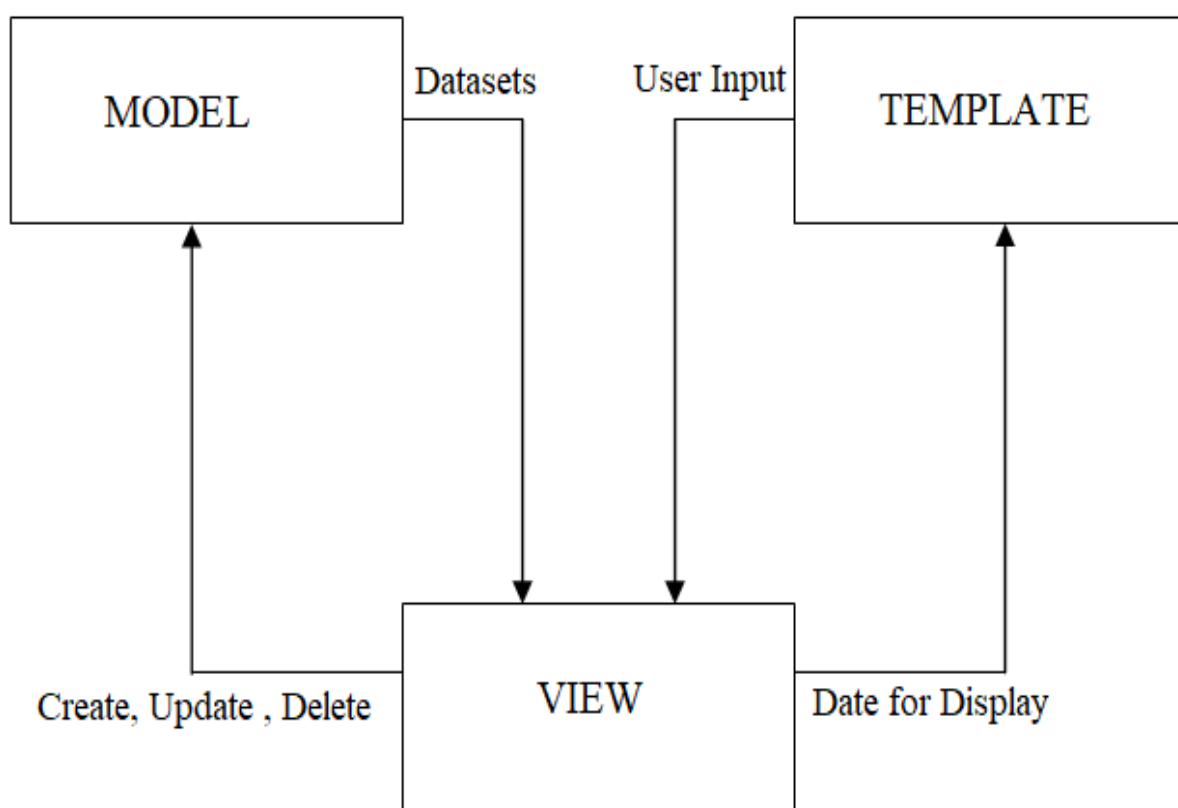


Рисунок 2.1 – Схема роботи MVT [16]

При створенні проєкт на базі Django має такі основні файли як:

– Manage.py – цей файл використовується для взаємодії з проєктом через командний рядок. Для прикладу можна навести команду запуску серверу, проведення міграцій баз даних;

– View.py – даний файл відповідає за відображення html – шаблонів, форм, віджетів;

- Models.py – у даному файлі зберігаються моделі програм;
- Setting.py – як видно з назви, цей файл містить в собі усі налаштування проєкту. Саме тут ми реєструємо всі створені нами додатки, саме тут зберігається конфігурація бази даних, та саме тут ми вказуємо розташування статичних файлів;
- urls.py – цей файл містить усі посилання проєкту, а також усі функції для виклику;
- __init__.py - файл містить в собі пакет Python.

На останок, можна виділити найголовніші переваги Django не тільки над Flask, а й над іншими фреймворками. До цих переваг відносять:

- дуже швидко роботу;
- безпеку проєкту, та його надійність;
- легкість масштабування, та гнучкість.

2.2 Основні вимоги до Інтернет-магазину

Для того, аби магазин користувався популярністю необхідно виконати певні вимоги. Для покупця цими вимогами є: перегляд товару за категоріями, можливість формування корзини, функція аутентифікації, великий асортимент товару. Однак магазин також має бути зручним і для протилежної сторони, тож основними вимогами Інтернет-магазину для продавця є: зручне адміністрування товарів та систему оброблення замовлень. Існують також загальні вимоги до Інтернет-магазину, а саме: зручне використання та стабільна робота веб-сайту.

2.3 Структурна схема Інтернет-магазину

Для інтернет-магазину міцний фундамент - це налагоджена, правильно структурована структура. Про це потрібно подумати спочатку - перед тим, як

спочатку малювати прототип та елементи дизайну. Якщо ви цього ще не зробили, попереду можуть бути серйозні проблеми. Наприклад, не буде місця для посилань на сторінки, або це може бути обмежено, якщо деякі пункти у меню вище обмежені.

Чим менше часу і сил користувач витрачає на роботу з сайтом, тим більш лояльним він стає до компанії. Існують сотні Інтернет-магазинів, і ніхто не буде «пересилювати себе» і терпіти незручності, щоб скористатися саме вашою пропозицією. Користувачеві простіше закрити вкладку і знайти більш «доброзичливий» портал. Для покупця це заощаджений час, для вас – збитки. Щоб користувачеві було зручно працювати з сайтом, потрібно розробити правильну структуру. Це «скелет» порталу, який грає ключову роль в зручності використання. Як приклад візьмемо текст. Він може бути дуже корисним, але відсутність абзаців і підрозділів доведе читача до знемоги. Те ж саме відбувається з сайтами: нікому не подобаються портали, що нагадують лабіринт. Структура сайту – це «путівник» покупця. Без нього людина «заблукає» і піде в інший магазин. Конкуренція на мережевому ринку колосальна, і кожна дрібниця може стати фатальною для бізнесу [17].

Сьогодні доволі важко знайти сферу Інтернет-комерції, де намагає конкуренції. Це безсумнівно мінус для нового вебресурсу. Однак, з цього можна також отримати певну користь. Наприклад, при розробці дизайну можна певною мірою скопіювати вдалі дизайнерські рішення, що позитивно вплине на вебсайт. Тому наявність конкурентів, несе з собою як очевидні мінуси так і не очевидні плюси.

При розробці Інтернет-магазину слід велику увагу приділити детальному розподілу товару на категорії з допомогою яких покупець може швидко знайти необхідний йому товар. Не поділяти товар на категорії – це серйозна помилка. Користувач часто не може переглянути увесь асортимент товару, для нього значно краще було б швидко та легко знайти необхідний товар, переглянути його опис та характеристики, а потім замовити його в декілька кліків без зайвих дій та втрат часу.

Схему структури розроблюваного Інтернет-магазину наведено на рис.

2.2.

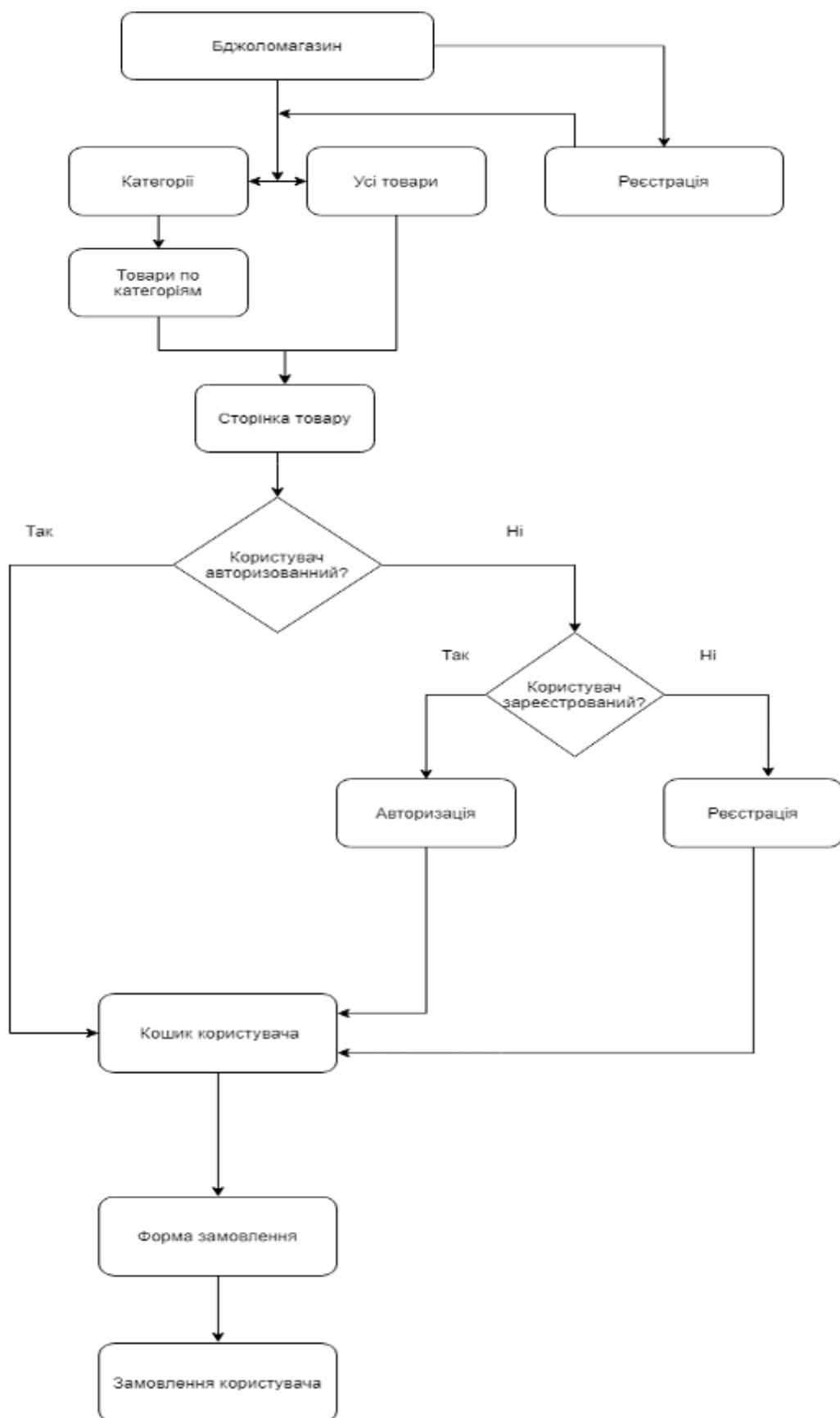


Рисунок 2.2 – Структура Інтернет - магазину

На ньому можна бачити що на веб-сайті відсутня зайва інформація, а також присутній розподіл товарів на категорії. Процес обрання та замовлення товару дуже простий і потребує лише реєстрації на сайті та заповнення простої форми, після чого замовлення передається на сервер, та оброблюється продавцем.

2.4 Висновки за розділом

У цьому розділі було розглянуто можливі рішення щодо реалізації проєкту та обрано фреймворк для розробки. Також було висунуто основні вимоги до Інтернет-магазину, як з боку покупця, так і з боку продавця. Було розглянуто усі моделі, що будуть використовуватись в проєкті, а також поля що характеризують моделі. В кінці розділу було приведено структуру Інтернет-магазину.

3 РОЗРОБКА ПРОГРАМИ

3.1 Опис моделей реалізованої програми

Перед початком розробки Інтернет-магазину було підключено необхідні бібліотеки. Також при розробці було використано деякі основні пакети, а саме :

- Django 3.1 – фреймворк на базі якого було вирішено розроблять Інтернет-магазин;
- Django-bootstrap4 – пакет, що надає можливість використовувати при розробці bootstrap4 за допомогою шаблонних тегів;
- Python-slugify – пакет що дозволяє створювати унікальну та зрозумілу url-адресу.

При розробці проєкту на Django було розроблено декілька моделей даних, які використовуються для роботи Інтернет-магазину, саме модель категорій – category, модель продукту – product, модель замовлення – order, та модель користувача – customer.

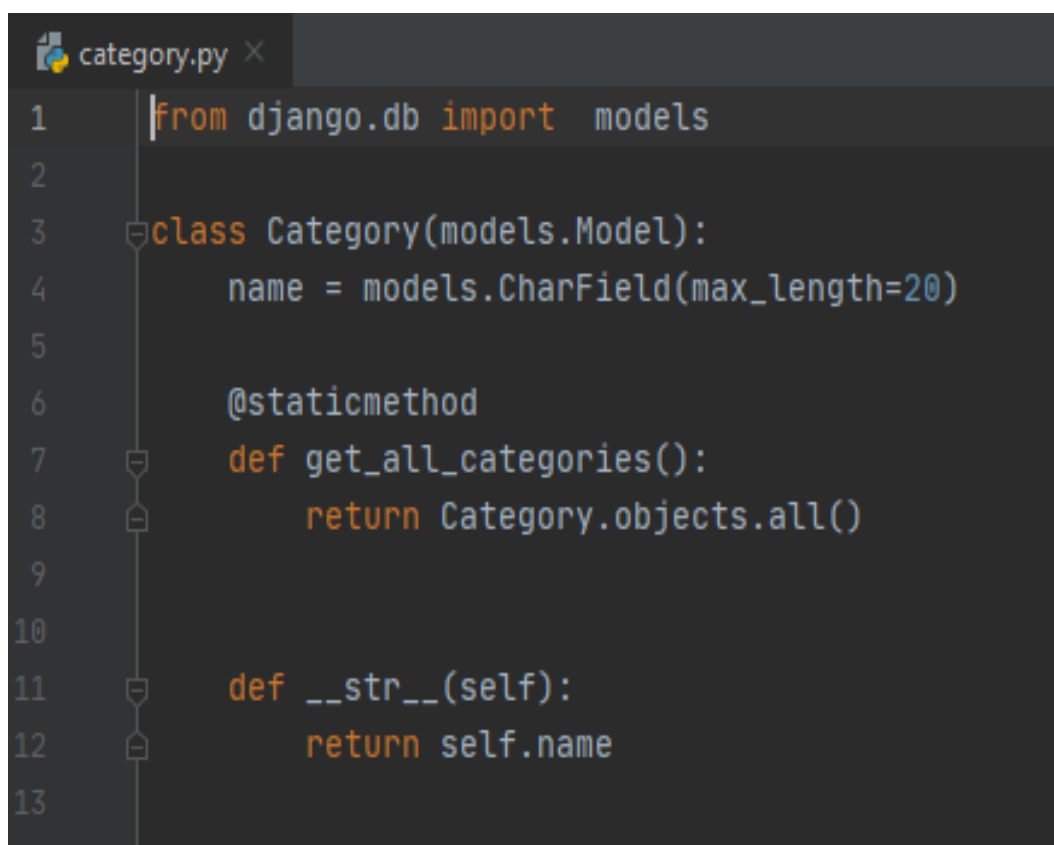
Модель є єдиним джерелом інформації про ваших даних. Вона містить основні поля і поведінку даних, які ви зберігаєте. Як правило, кожна модель відображається в одну таблицю бази даних [18].

У файлі кожної моделі, окрім власне опису моделі наведено також функції, що відповідають за обробку цих моделей. Розглянемо ці моделі більш детально.

3.1.1 Модель category

Ця модель була розроблена для можливості додавання нової категорії товарів для продавця. Ця модель, має лише одне поле CharField, що дозволяє вводити назву категорії. Не дивлячись на малу кількість полів, ця модель є дуже важливою для роботи усього проєкту.

Програмний код моделі category зображений на рисунку 3.1.



```

1  | from django.db import models
2
3  | class Category(models.Model):
4  |     name = models.CharField(max_length=20)
5
6  |     @staticmethod
7  |     def get_all_categories():
8  |         return Category.objects.all()
9
10
11 |     def __str__(self):
12 |         return self.name
13

```

Рисунок 3.1 – Модель category

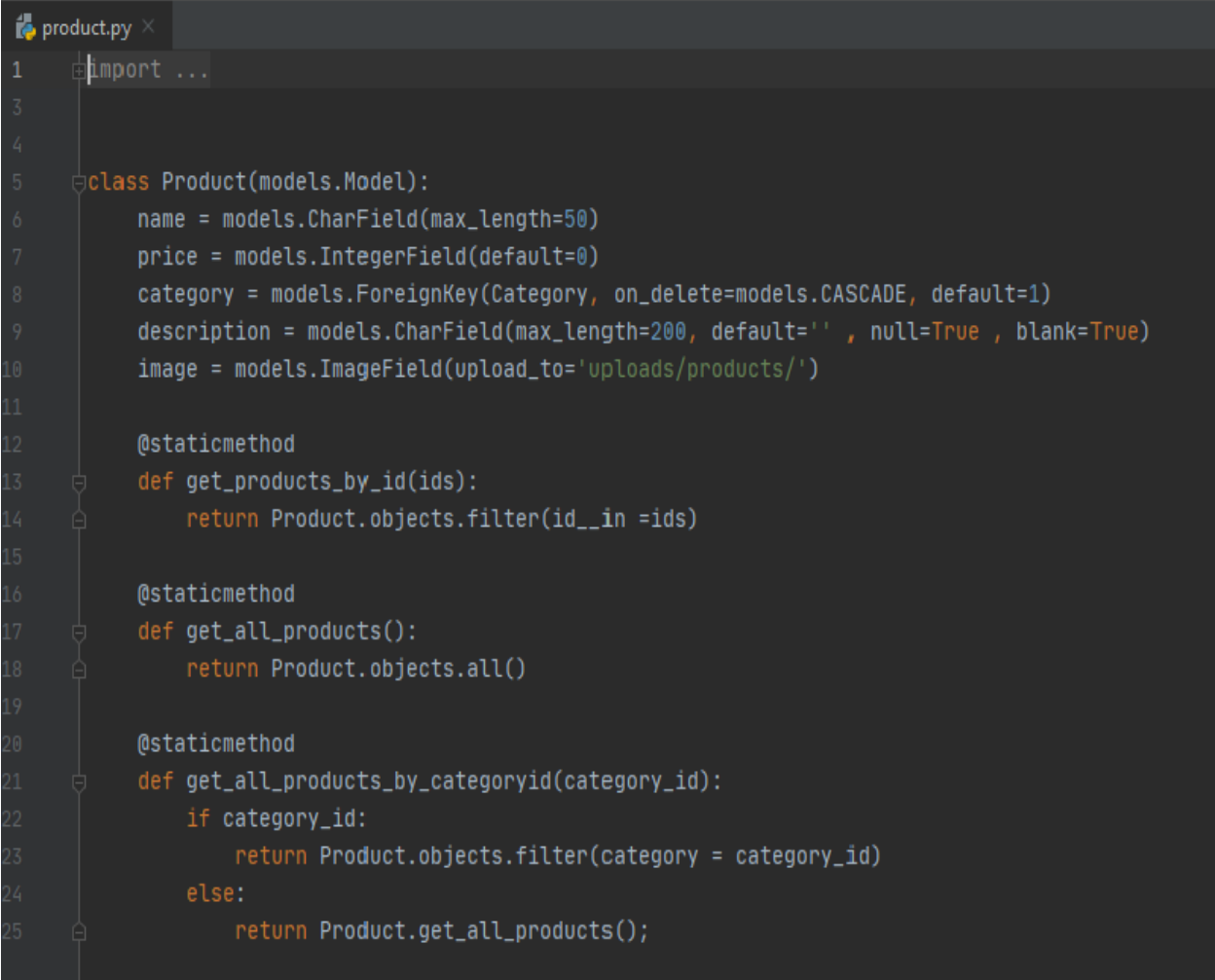
3.1.2 Модель product

Ця модель вже виконує більш складну функцію, оскільки описує структуру продукту, що буде додано на сайт. В даній моделі представлені наступні поля:

- рядок `name` – що відповідає назві товару, представлено полем `CharField`, що дозволяє зберігати рядок тексту та цифр;
- рядок `price` – що відповідає ціні товару, представлене полем `IntegerField`, що дозволяє зберігати рядок цілих чисел;
- рядок `category` – що відповідає категорії товару, переставлено полем `ForeignKey`, оскільки цей рядок використовує назву категорії з моделі `category`;
- рядок `description` – що відповідає опису товару, являє собою поле `CharField`, що дозволяє зберігати рядок тексту та цифр;

– рядок `image` – що відповідає зображенню товару, представлене полем `ImageField`, та дозволяє зберігати фотоматеріали.

Програмний код моделі `product` зображений на рисунку 3.2.



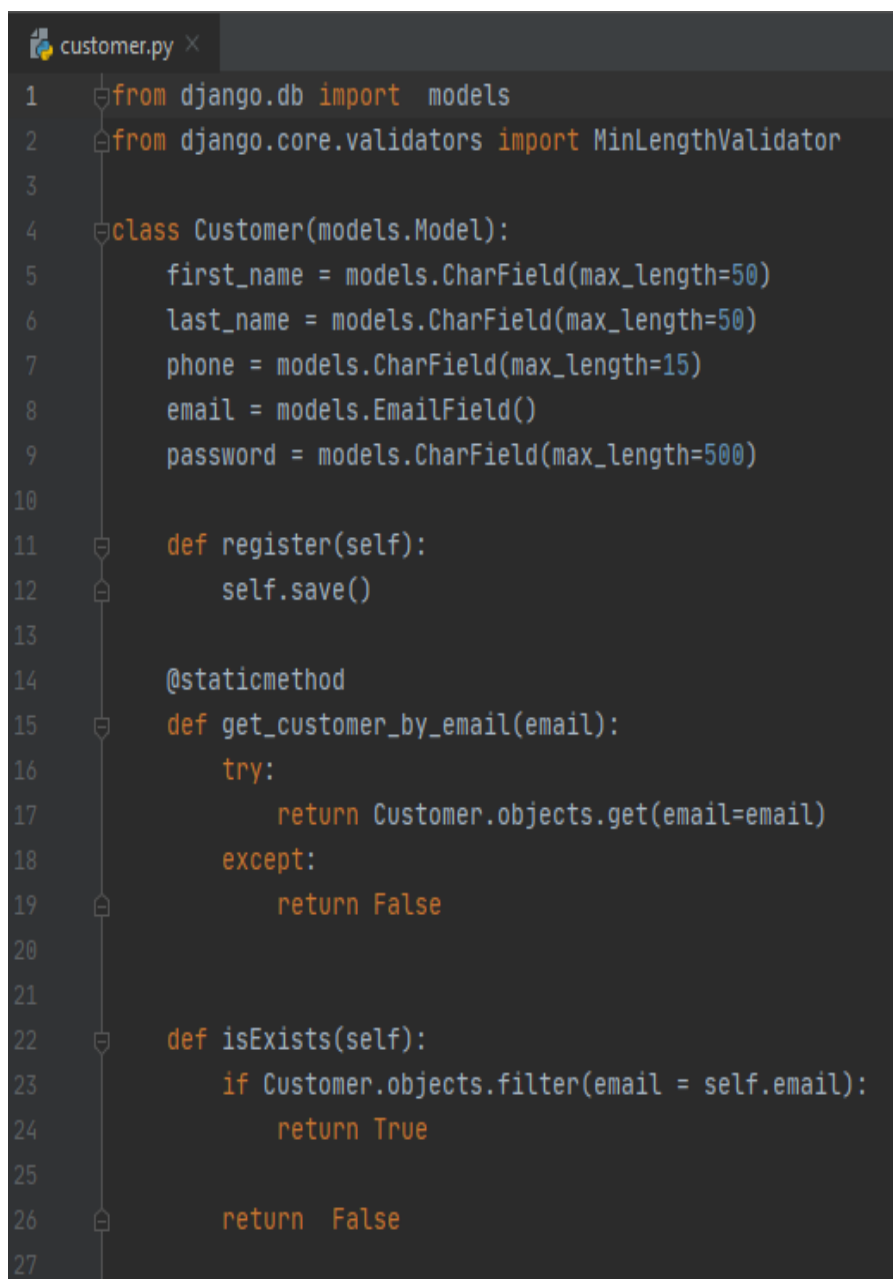
```
1  import ...
3
4
5  class Product(models.Model):
6      name = models.CharField(max_length=50)
7      price = models.IntegerField(default=0)
8      category = models.ForeignKey(Category, on_delete=models.CASCADE, default=1)
9      description = models.CharField(max_length=200, default='', null=True, blank=True)
10     image = models.ImageField(upload_to='uploads/products/')
11
12     @staticmethod
13     def get_products_by_id(ids):
14         return Product.objects.filter(id__in =ids)
15
16     @staticmethod
17     def get_all_products():
18         return Product.objects.all()
19
20     @staticmethod
21     def get_all_products_by_categoryid(category_id):
22         if category_id:
23             return Product.objects.filter(category = category_id)
24         else:
25             return Product.get_all_products();
```

Рисунок 3.2 – Модель `Product`

3.1.3 Модель `customer`

Ця модель описує структуру даних користувача, що буде реалізовано на сайті.

Програмний код моделі `customer` зображений на рисунку 3.3.



```

1  from django.db import models
2  from django.core.validators import MinLengthValidator
3
4  class Customer(models.Model):
5      first_name = models.CharField(max_length=50)
6      last_name = models.CharField(max_length=50)
7      phone = models.CharField(max_length=15)
8      email = models.EmailField()
9      password = models.CharField(max_length=500)
10
11     def register(self):
12         self.save()
13
14     @staticmethod
15     def get_customer_by_email(email):
16         try:
17             return Customer.objects.get(email=email)
18         except:
19             return False
20
21
22     def isExists(self):
23         if Customer.objects.filter(email = self.email):
24             return True
25
26         return False
27

```

Рисунок 3.3 – Модель Customer

Ця модель представлена такими полями:

- рядок `first_name` – що являє собою ім'я користувача, представлено полем `CharField`, що дозволяє зберігати рядок тексту та цифр;
- рядок `last_name` – що являє собою прізвище користувача, представлено полем `CharField`, що дозволяє зберігати рядок тексту та цифр;
- рядок `phone` – що являє собою телефон замовника, представлений полем `CharField`, що дозволяє зберігати рядок тексту та цифр;

- рядок email – що являє собою електронну пошту користувача, представлено полем EmailField, що дозволяє зберігати рядок цілих чисел;
- рядок password – що являє собою пароль користувача, представлений полем CharField, що дозволяє зберігати рядок тексту та цифр.

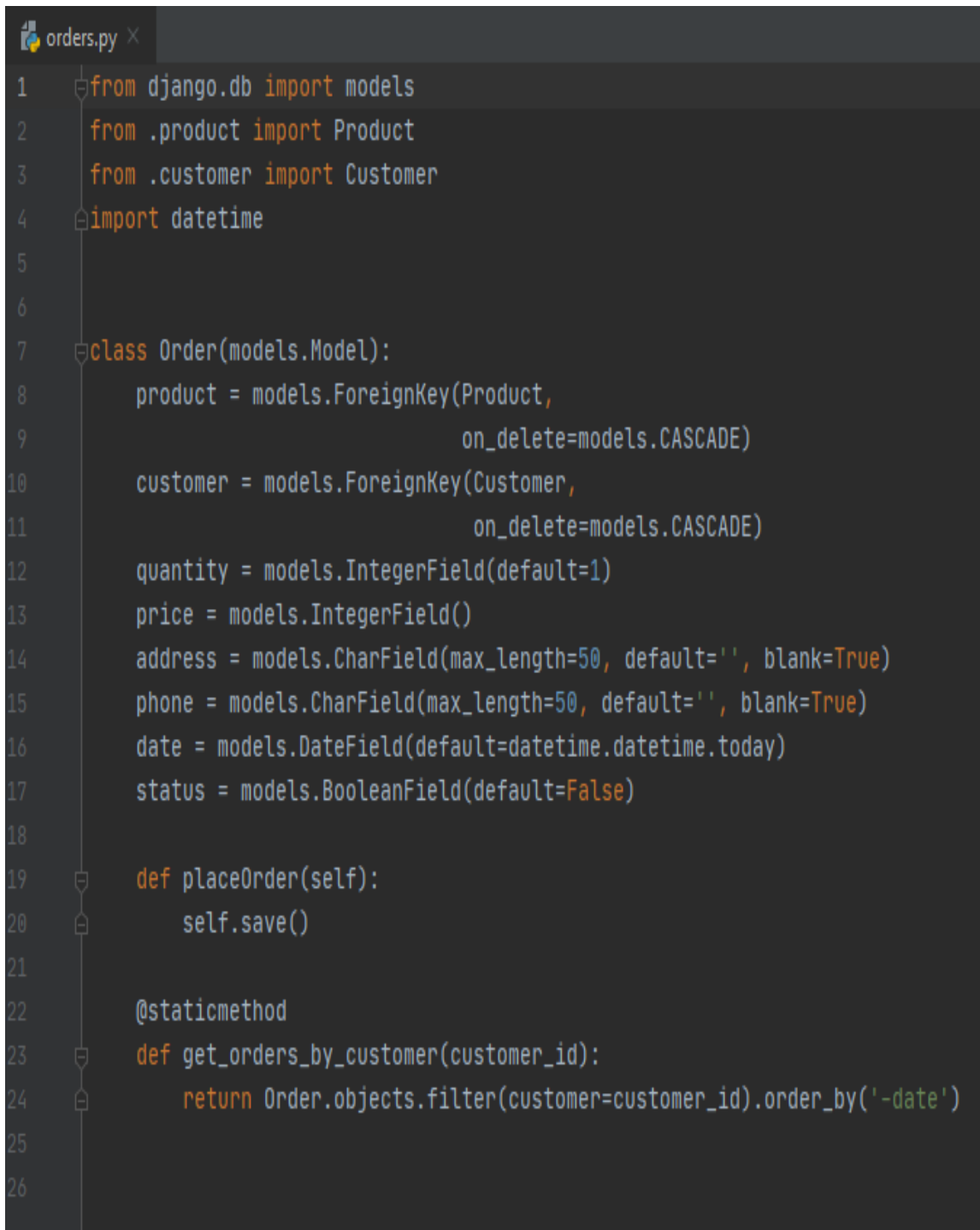
3.1.4 Модель order

Ця модель описує структуру замовлення, що буде реалізовано на сайті.

Ця модель використовує моделі category та product. Ця модель представлена такими полями:

- рядок product – що представляє в замовленні продукт, обраний користувачем представлено полем ForeignKey, оскільки для цього поля необхідні дані моделі product;
- рядок customer – що представляє в замовленні користувача, який робить замовлення представлено полем ForeignKey, оскільки для цього поля необхідні дані моделі customer;
- рядок quantity – що в замовленні відповідає за кількість товару, представлено полем IntegerField, що дозволяє зберігати рядок цілих чисел;
- рядок price – що віддвітає ціні усього замовлення також представлений полем IntegerField;
- рядок address - рядок, який, як зрозуміло з назви відповідає за адрес користувача, представлений полем CharField, що дозволяє зберігати рядок тексту та цифр;
- рядок phone – що являє собою телефон замовника, представлений полем CharField, що дозволяє зберігати рядок тексту та цифр;
- рядок date – що відповідає даті замовлення, представлений полем DateField, що дозволяє зберігати дату;
- рядок status – що відповідає за статус замовлення, представлено полем BooleanField, що може бути рівним True або False.

Програмний код моделі `order` зображений на рисунку 3.4.



```
1 from django.db import models
2 from .product import Product
3 from .customer import Customer
4 import datetime
5
6
7 class Order(models.Model):
8     product = models.ForeignKey(Product,
9                                 on_delete=models.CASCADE)
10    customer = models.ForeignKey(Customer,
11                                on_delete=models.CASCADE)
12    quantity = models.IntegerField(default=1)
13    price = models.IntegerField()
14    address = models.CharField(max_length=50, default='', blank=True)
15    phone = models.CharField(max_length=50, default='', blank=True)
16    date = models.DateField(default=datetime.datetime.today)
17    status = models.BooleanField(default=False)
18
19    def placeOrder(self):
20        self.save()
21
22    @staticmethod
23    def get_orders_by_customer(customer_id):
24        return Order.objects.filter(customer=customer_id).order_by('-date')
```

Рисунок 3.4 – Модель `order`

3.2 Структура проєкту

Проєкт на Django має дуже зручну та зрозумілу структуру. Як правило це головний додаток, в якому містяться головні файли проєкту, необхідні для

роботи проєкту та папки додатків, кожен яких розробник може налаштовувати окремо та зв'язувати з головним додатком. Така структура для розробника легкою та зрозумілою, що значно облегшує розробку проєкту на цьому фреймворку. Слід відмітити, що в проєкті на Django, дуже легко налаштовувати додатки, адже при створенні в них вже наявні всі необхідні файли і розробнику залишається лише змінити їх, підключити додаток до головного додатку через файли `settings.py` та `urls.py` і додаток буде включений в структуру проєкту.

Опис структури деяких файлів:

- `settings.py` – один з найголовніших файлів усього проєкту. Саме тут реєструються нові додатки проєкту, визначається мова та часовий пояс, налаштовується база даних та задається розміщення статичних об'єктів проєкту;

- `manage.py` – файл, який використовується для створення нових додатків та для роботи з базою даних проєкту;

- `urls.py` – цей файл також виконує дуже важливу для проєкту функцію, а саме задає асоціації `url`-адреси з уявленням;

- `wsgi.py` – даний файл використовують для налагодження зв'язку між вебсервісом та проєктом Django;

- `test.py` – цей файл відповідає за тестування проєкту;

- `admin.py` – у цьому файлі відбувається налаштування адміністративної частини вебсайту;

- `apps.py` – у цьому файлі відбувається реєстрація додатків;

- `__init__.py` – це пустий файл, необхідний для коректної роботи проєкту.

Під час реалізації проєкту, була визначена структура, наведена на рис. 3.5.

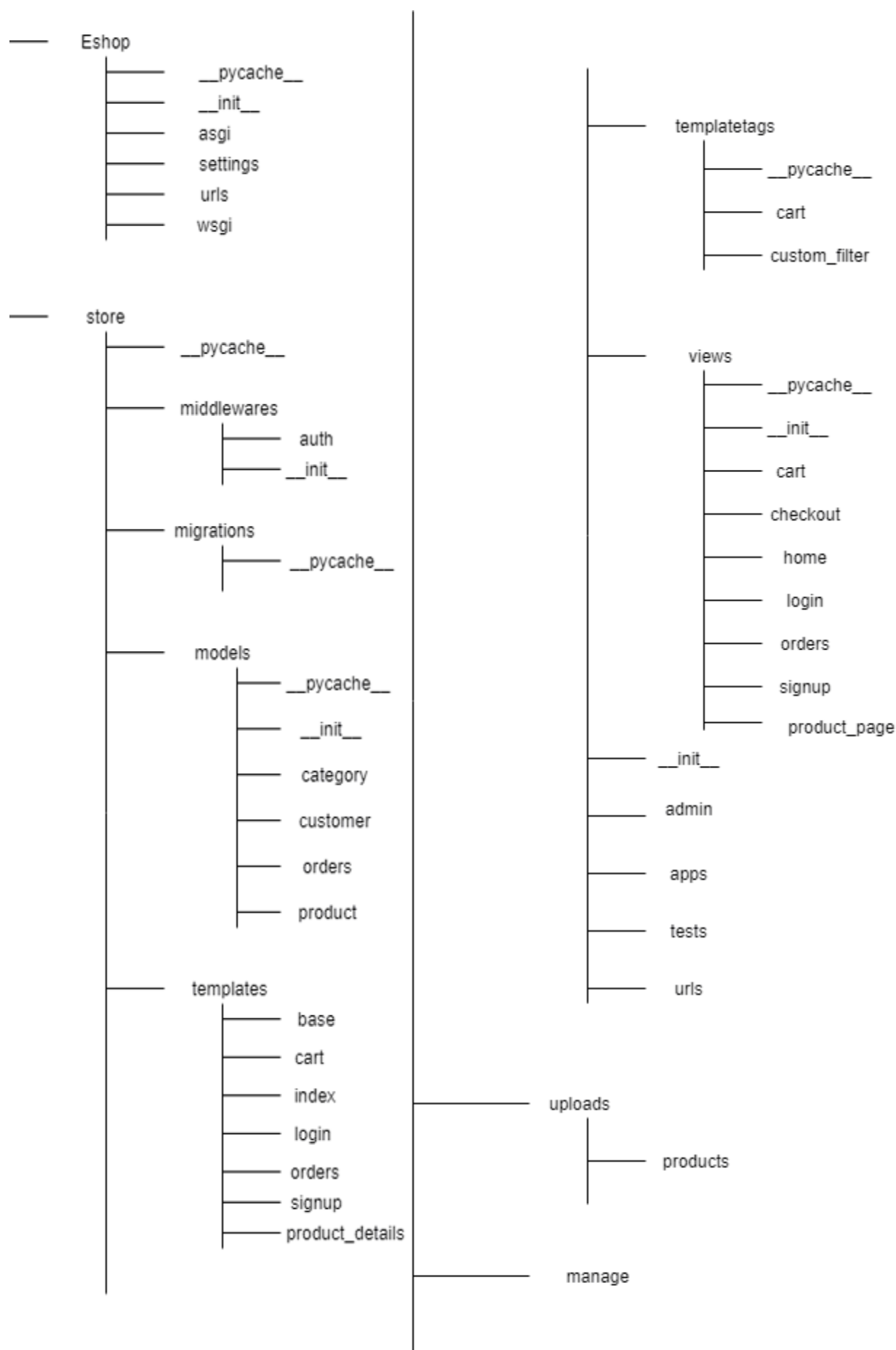


Рисунок 3.5 – Структура проекта

3.3 Висновки за розділом

У цьому розділі було розглянуто моделі, розроблені під час програмної реалізації проєкту. Було детально розглянуто, якими полями представлені ці моделі. Також було розглянуто структуру проєкту, та описано деякі важливі файли програми.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Опис основного функціоналу для клієнта

Будь-який покупець, який зайшов на наш сайт, має змогу переглядати доступні для покупки товари, а також обирати для перегляду товари певної категорії, що представлені на сайті.

Головну сторінку Інтернет-магазину зображено на рисунку 4.1.

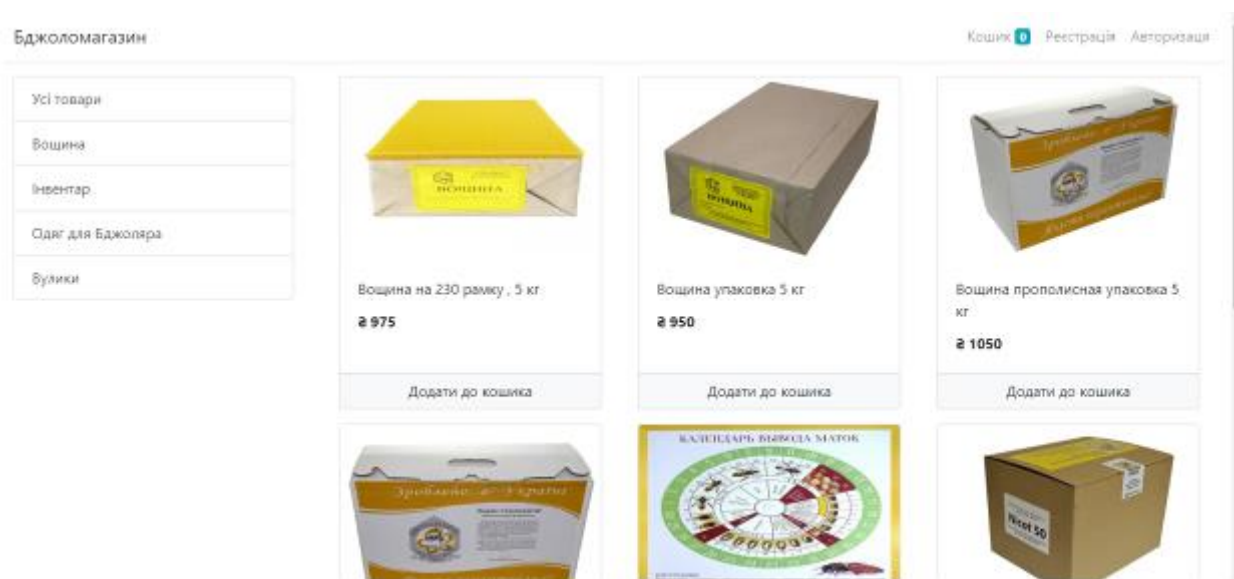


Рисунок 4.1 – Головна сторінка Інтернет-магазину

Категорія «Вощина» зображена на рисунку 4.2.



Рисунок 4.2 – Товари категорії «Вощина»

Як можна бачити, товар представлено в зручному вигляді, а саме у вигляді картки. На кожній картці є кнопка «Додати до кошика», що виконує відповідну функцію.

Картку продукту зображено на рисунку 4.3.



Рисунок 4.3 – Картка товару

Доволі зручною є функція збільшення чи зменшення кількості товару в кошику прямо з головної сторінки магазину. Ці дії виконують відповідні кнопки «+» та «-».

Можливість зміни кількості товару у кошику зображена на рисунку 4.4.

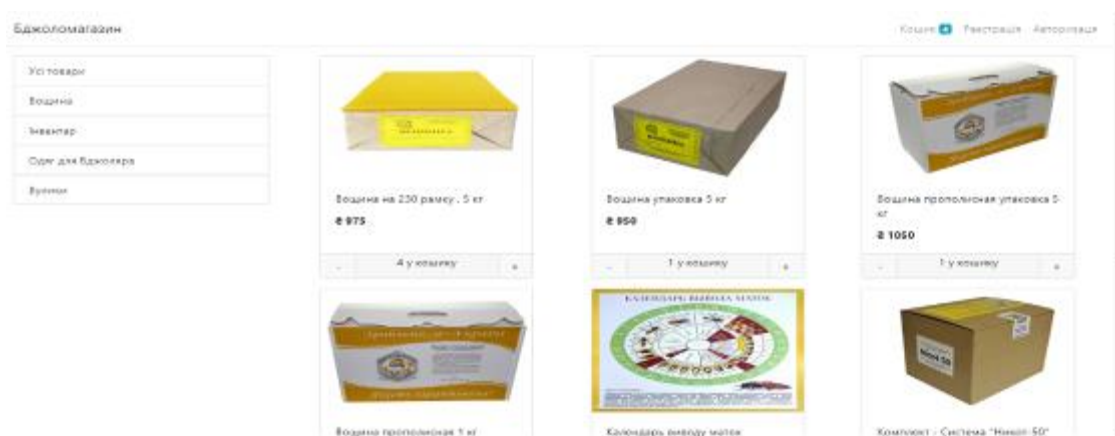


Рисунок 4.4 – Зміна кількості товару у кошику

На кожній картці товару, назва являє собою посилання на окрему сторінку товару, на якій міститься більш детальна інформація про товар.

Сторінка товару зображена на рисунку 4.5.



Рисунок 4.5 – Сторінка товару

Усі додані до кошика товари зберігаються у ньому. Також у кошику можна побачити суму усього замовлення.

Сторінка кошику зображена на рисунку 4.6.

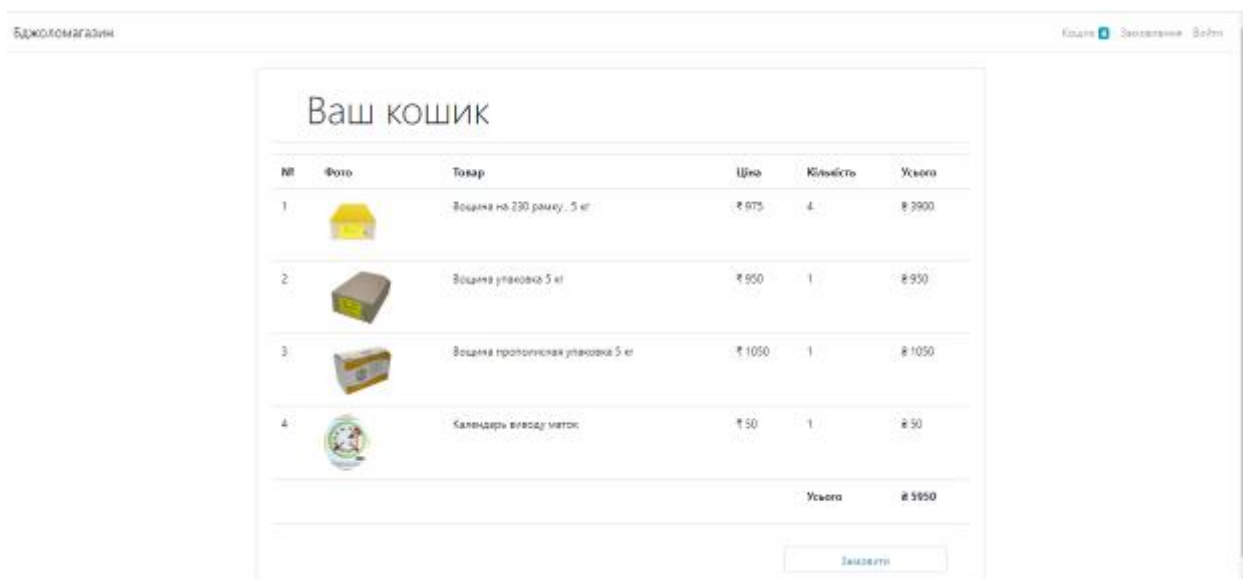


Рисунок 4.6 – Сторінка кошику Інтернет-магазину

Для того щоб замовити товар, необхідно у кошику натиснути відповідну кнопку «замовлення» та заповнити просту форму.

Форма замовлення зображена на рисунку 4.7

№	Фото	Товар	Ціна	Кількість	Усього
1		Водичка на 250 мл, 5 шт	195	4	780
2		Водичка упаковка 5 шт	95	1	95
3		Водичка прополісний упаковка 5 шт	105	1	105
4		Календар виходу маток	50	1	50
				Усього	990

Рисунок 4.7 – Форма замовлення

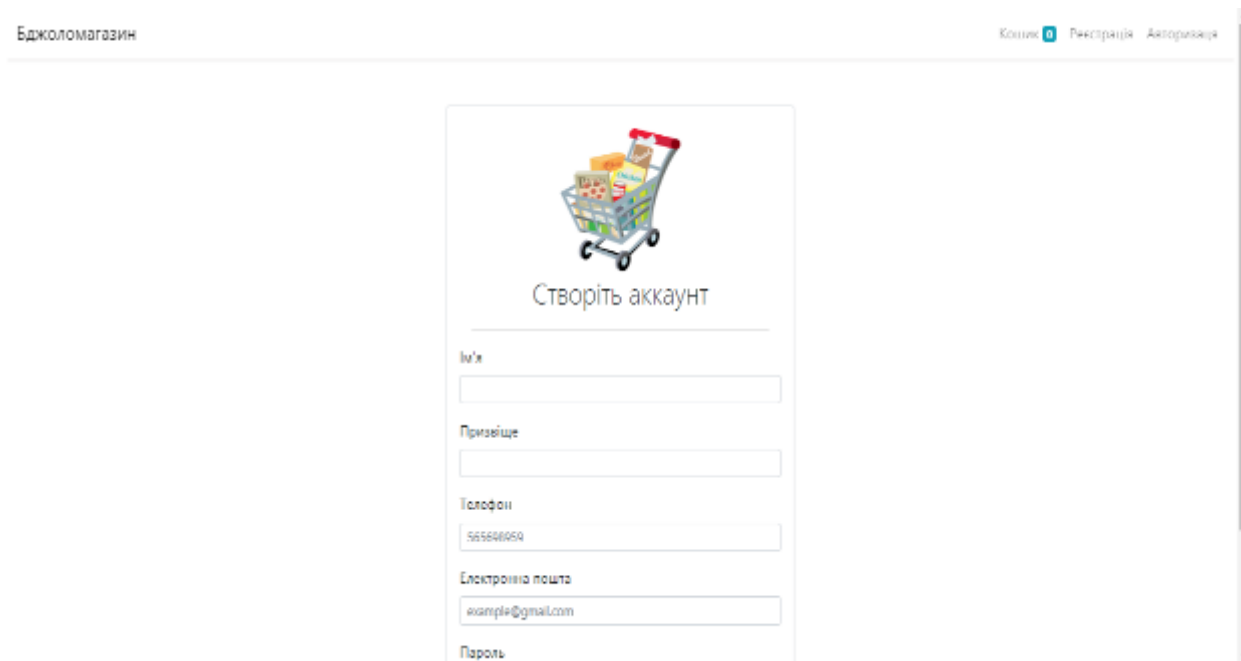
Після оформлення замовлення користувач має змогу перейти у вкладку «Замовлення», та відстежити статус свого замовлення.

Сторінка «Замовлення» зображена на рисунку 4.8.

№	Фото	Товар	Ціна	Кількість	Усього	Статус
1		Водичка на 250 мл, 5 шт	26 апреля 2021 г. 195	4	780	Успішно
2		Водичка упаковка 5 шт	26 апреля 2021 г. 95	1	95	Успішно
3		Водичка прополісний упаковка 5 шт	26 апреля 2021 г. 105	1	105	Успішно
4		Календар виходу маток	26 апреля 2021 г. 50	1	50	Успішно

Рисунок 4.8 – Сторінка «Замовлення»

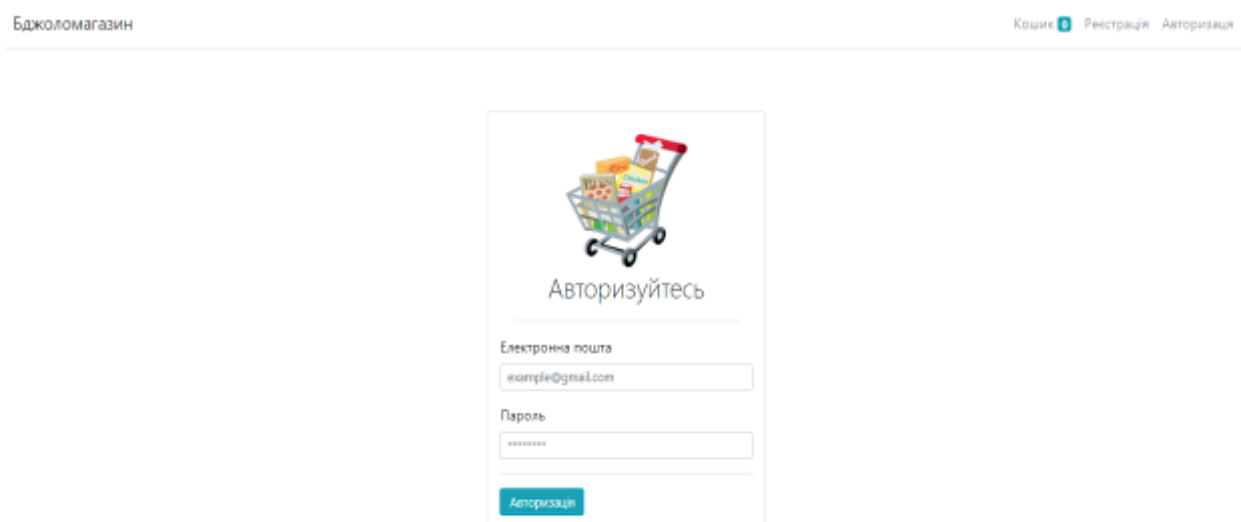
На нашому сайті також присутні функції Реєстрації та Авторизації.
Сторінка реєстрації зображена на рисунку 4.9.



The screenshot shows the registration page of an online store. At the top left is the text "Бджоломагазин". At the top right are links: "Кошик", "Реєстрація" (highlighted with a blue square), and "Авторизація". The main content area is a white box with a shopping cart icon and the title "Створіть аккаунт". Below the title are five input fields: "Ім'я", "Прізвище", "Телефон" (with the placeholder "555555555"), "Електронна пошта" (with the placeholder "example@gmail.com"), and "Пароль".

Рисунок 4.9 – Сторінка Реєстрації Інтернет-магазину

Сторінка авторизації зображена на рисунку 4.10.



The screenshot shows the authorization page of an online store. At the top left is the text "Бджоломагазин". At the top right are links: "Кошик", "Реєстрація", and "Авторизація" (highlighted with a blue square). The main content area is a white box with a shopping cart icon and the title "Авторизуйтесь". Below the title are two input fields: "Електронна пошта" (with the placeholder "example@gmail.com") and "Пароль" (with masked characters). At the bottom of the box is a blue button labeled "Авторизація".

Рисунок 4.10 – Сторінка Авторизації Інтернет-магазину

4.2 Опис основного функціоналу для продавця

Звичайно, наш Інтернет-магазин має бути також і зручним робочим місцем з легким та зрозумілим функціоналом. На нашому сайті, продавець може легко, використовуючи вбудовану admin-панель, додати до магазину нову категорію товарів, або змінити наявні.

Сторінка створення нової категорії товарів зображена на рисунку 4.11.

Начало · Store · Categories · Добавить category

STORE

- Categories + Добавить
- Customers + Добавить
- Orders + Добавить
- Products + Добавить

ПОЛЬЗОВАТЕЛИ И ГРУППЫ

- Группы + Добавить
- Пользователи + Добавить

Добавить category

Name:

Сохранить и добавить другой объект Сохранить и продолжить редактирование СОХРАНИТЬ

Рисунок 4.11 – Сторінка створення нової категорії товарів

Також продавець має змогу додати, чи редагувати товари в Інтернет-магазині, з допомогою все тієї ж admin-панелі.

Сторінку створення нового товару зображено на рисунку 4.12.

Администрирование Django

Начало · Store · Products · Добавить product

STORE

- Categories + Добавить
- Customers + Добавить
- Orders + Добавить
- Products + Добавить

ПОЛЬЗОВАТЕЛИ И ГРУППЫ

- Группы + Добавить
- Пользователи + Добавить

Добавить product

Name:

Price:

Category: +

Description:

Image: Файл не выбран

Сохранить и добавить другой объект Сохранить и продолжить редактирование СОХРАНИТЬ

Рисунок 4.12 – Сторінка створення нового товару

Як можна бачити, продавець може встановити назву, ціну, категорію, опис, та фотографію товару.

Також у продавця є можливість відстежувати замовлення, зроблені користувачами та змінювати їх статус. При натисканні на будь-яке замовлення з'являється розширена інформація про замовлення.

Сторінка замовлень, де показані усі замовлення, що зроблені в магазині зображена на рисунку 4.13.

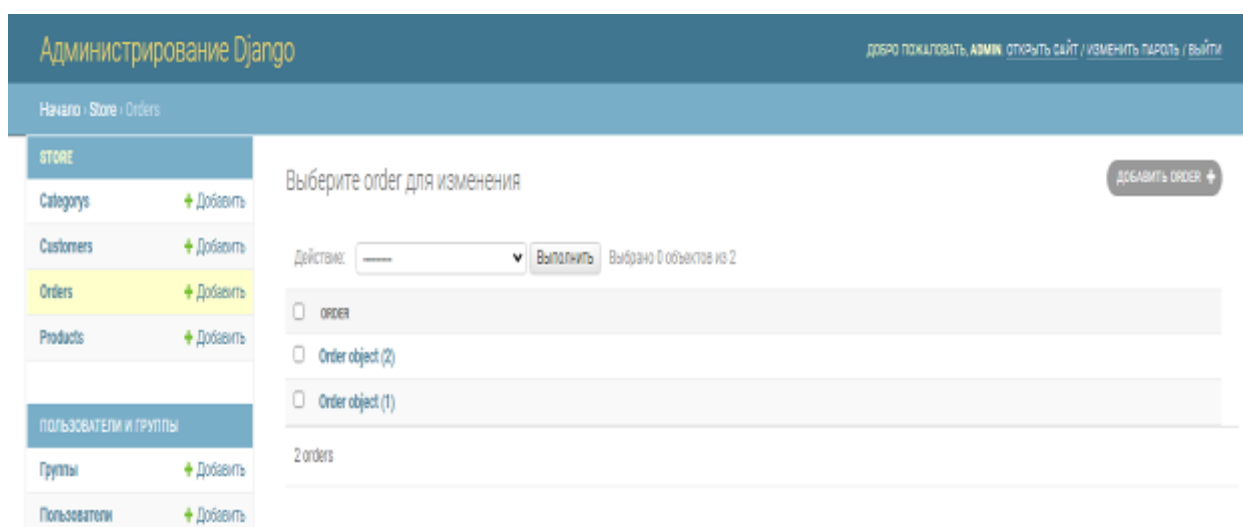


Рисунок 4.13 – Сторінка замовлення Інтернет-магазину

4.3 Висновки за розділом

Було розглянуто інтерфейс розроблюваного Інтернет-магазину, а саме головну сторінку та сторінку товару. Було протестовано функції покупця, а саме формування кошику та процес замовлення товару. Також було розглянуто можливості продавців, а саме функції додавання та редагування товару чи категорії, а також сторінку відстежування замовлень.

ВИСНОВКИ

На сьогоднішній день, електронна комерція є важливою складовою економічного розвитку будь-якої країни. Це, безсумніно, справедливо і для нашої держави, де електронна комерція буде продовжувати швидко зростати протягом наступних кількох років. Інтернет швидко розвивається, тож і функцій та можливостей стає більше з кожним роком. Підприємства, що зробили ставку на електронну комерцію, отримують перевагу над тими, хто працює лише в традиційній формі.

В ході роботи проведено аналіз предметної області, у результаті якого було сформовано ряд вимог до проектування власного вебресурсу. Також було проведено огляд сучасних технологій та методів розробки вебплатформ у результаті чого було обрано фреймворк Django. Це сучасний та прогресивний фреймворк, що користується популярністю серед розробників вебсайтів та має ряд переваг над іншими фреймворками. У роботі було досягнуто усіх вимог, що були висунуті, а саме зручну та легку процедуру покупки товару для клієнтів та зручну систему адміністрування Інтернет-магазину для продавців.

Кінцевий проект є якісним продуктом, що не містить зайвої інформації, анімації, та функціоналу. Він швидко завантажується на різних швидкостях Інтернету, завдяки прогресивній мові програмування Python, та простого дизайну та сучасним технологіям веброботи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. McDonald, J. SEO Fitness Workbook: The Seven Steps to Search Engine Optimization Success on Google / J. McDonald. – Charleston : Independent Publishing, 2016. – 350 p.
2. Lee, N. Google It: Total Information Awareness / N. Lee. – Berlin : Springer, 2016. – 587 p.
3. Smith, M. Targeted: How Technology Is Revolutionizing Advertising and the Way Companies Reach Consumers / M. Smith. – New York : AMACOM, 2014. – 224 p.
4. Що таке CMS и Які види CMS для сайтів бувають [Електрон. ресурс]. – Режим доступу: <https://cityhost.ua/uk/blog/chto-takoe-cms-i-kakie-vidy-cms-dlya-saytov-byvayut.html/>.
5. WordPress [Електрон. ресурс]. – Режим доступу: [https://uk.wikipedia.org/wiki/WordPress /](https://uk.wikipedia.org/wiki/WordPress/).
6. OpenCart [Електрон. ресурс]. – Режим доступу: <https://astwellsoft.com/uk/blog/cms/opencart.html>.
7. Что такое Joomla? [Электрон. ресурс]. – Режим доступа: <https://joomla.ru/docs/administrator/joomla3-start/1742-chto-takoe-joomla/>.
8. Vincent, W. Django for APIs: Build web APIs with Python & Django / W. Vinsent. – Charleston : Independent Publishing, 2018. – 190 p.
9. Matthes, E. Python Crash Course: A Hands-On, Project-Based Introduction to Programming / E. Matthes. – San Francisco : No Starch Press, 2015. – 544 p.
10. 11 лучших фреймворков Python для веб-разработки в 2021 году [Электрон. ресурс]. – Режим доступа: <https://wezom.com.ua/blog/python-frameworks#flask/>.
11. Django Web Framework (Python) [Electronic resource]. – Access mode: <https://developer.mozilla.org/uk/docs/Learn/Server-side/Django>.

12. Django документация [Электрон. ресурс]. – Режим доступа: <https://djbook.ru/>.

13. DJANGO PROJECT MVT СТРУКТУРА [Электрон. ресурс]. – Режим доступа: <http://espressocode.top/django-project-mvt-structure/>.

14. Alchin, M. Pro Django (Expert's Voice in Web Development) / М. Alchin. – New York : Apress, 2013. – 320 p.

15. Pinkham, A. Django Unleashed / A. Pinkham. – Indianapolis : Sams Publishing, 2015. – 840 p.

16. Почему Django — лучший фреймворк для разработки сайтов [Электрон. ресурс]. – Режим доступа: <https://ru.hexlet.io/blog/posts/pochemu-django-luchshiy-freymvork-dlya-razrabotki-saytov>.

17. Розвиваємо бізнесі та бренді с помощью кращих інструментів Digital [Електрон. ресурс]. – Режим доступа: <https://ilion.digital/ua>.

18. Документация по фреймворку Django [Электрон. ресурс]. – Режим доступа: <https://django.fun/docs/django/ru/3.1/topics/db/models/>.

ДОДАТОК А
Технічне завдання

Вступ

Програма призначена для зручної купівлі та продажу товару в Інтернеті.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмна реалізація інтернет-магазину на базі фреймворку Django», затверджене наказом по Національному університету «Запорізька політехніка».

A.2 Призначення розробки

Програма призначена для використання людьми, що прагнуть швидко та легко купувати або продавати товари в Інтернеті.

A.3 Вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Функції, які мають виконуватись програмним продуктом:

- надавати доступ до каталогу товарів.
- виконувати функцію додавання товару до кошика.
- виконувати функцію оформлення замовлення.
- виконувати функцію додавання нової категорії та товару.
- виконувати функцію відстеження замовлень.

A.3.2 Вимоги до надійності

Програма має забезпечувати коректну роботу своїх модулів.

A.3.3 Умови експлуатації

Для роботи з програмою користувач має володіти достатнім рівнем користування персональним комп'ютером.

A.3.4 Вимоги до складу й параметрів технічних засобів

Для роботи необхідний персональний комп'ютер з версією операційної системи Microsoft Windows 10.

A.4 Вимоги до програмної документації

Програмна документація включає в себе технічне завдання і повинна бути написана українською мовою.

A.5 Порядок контролю й приймання

Робота допускається до захисту керівником, нормоконтролером, затверджується завідувачем кафедри і захищається в присутності членів екзаменаційної комісії.

ДОДАТОК Б
Текст програми

Б.1 Файл settings.py

```

import os
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
import os
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
SECRET_KEY = '^_g%33qd(g8bjc+*40&uh(ptgkb$&-*+0!i3$lu7xj1u166cbb'
DEBUG = True
ALLOWED_HOSTS = []
# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'store'
]
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    # 'store.middlewares.auth.auth_middleware'
]
ROOT_URLCONF = 'Eshop.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',

```

```

'DIRS': [],
'APP_DIRS': True,
'OPTIONS': {
    'context_processors': [
        'django.template.context_processors.debug',
        'django.template.context_processors.request',
        'django.contrib.auth.context_processors.auth',
        'django.contrib.messages.context_processors.messages',
    ],
},
],
WSGI_APPLICATION = 'Eshop.wsgi.application'
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

```

```

LANGUAGE_CODE = 'ru'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True
STATIC_URL = '/static/'
MEDIA_URL = "/image/download/"
MEDIA_ROOT = BASE_DIR
SECRET_KEY = '^_g%33qd(g8bjc+*40&uh(ptgkb$&-*+0!i3$lu7xj1u166cbb'
DEBUG = True
ALLOWED_HOSTS = []
# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'store'
]
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    # 'store.middlewares.auth.auth_middleware'
]
ROOT_URLCONF = 'Eshop.urls'
TEMPLATES = [
    {

```

```

'BACKEND': 'django.template.backends.django.DjangoTemplates',
'DIRS': [],
'APP_DIRS': True,
'OPTIONS': {
    'context_processors': [
        'django.template.context_processors.debug',
        'django.template.context_processors.request',
        'django.contrib.auth.context_processors.auth',
        'django.contrib.messages.context_processors.messages',
    ],
},
]
WSGI_APPLICATION = 'Eshop.wsgi.application'

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },

```

```
]
LANGUAGE_CODE = 'ru'
```

```
TIME_ZONE = 'UTC'
```

```
USE_I18N = True
```

```
USE_L10N = True
```

```
USE_TZ = True
```

```
STATIC_URL = '/static/'
MEDIA_URL = "/image/download/"
MEDIA_ROOT = BASE_DIR
```

Б.2 Файл urls.py

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path("", include('store.urls'))
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Б.3 Файл auth.py

```
from django.shortcuts import redirect
def auth_middleware(get_response):
    def middleware(request):
        print(request.session.get('customer'))
```

```

returnUrl = request.META['PATH_INFO']
print(request.META['PATH_INFO'])
if not request.session.get('customer'):
    return redirect(f'login?return_url={returnUrl}')
response = get_response(request)
return response
return middleware

```

Б.4 Файл 0001_initial.py

```

from django.db import migrations, models

class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Product',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
                ('name', models.CharField(max_length=50)),
                ('price', models.IntegerField(default=0)),
                ('description', models.CharField(default="", max_length=200)),
                ('image', models.ImageField(upload_to='products/')),
            ],
        ),
    ]

```

Б.5 Файл 0002_auto_20200528_1402.py

```

from django.db import migrations, models
class Migration(migrations.Migration):
    dependencies = [
        ('store', '0001_initial'),
    ]
    operations = [
        migrations.CreateModel(
            name='Category',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
                ('name', models.CharField(max_length=20)),
            ],
        ),
        migrations.AlterField(
            model_name='product',
            name='image',
            field=models.ImageField(upload_to='uploads/products/'),
        ),
    ]

```

Б.6 Файл 0003_product_category.py

```

from django.db import migrations, models
import django.db.models.deletion
class Migration(migrations.Migration):
    dependencies = [
        ('store', '0002_auto_20200528_1402'), ]
    operations = [
        migrations.AddField(
            model_name='product',
            name='category',

```

```

        field=models.ForeignKey(default=1,
on_delete=django.db.models.deletion.CASCADE, to='store.Category'),
    ),
]

```

Б.7 Файл 0004_auto_20200528_1430.py

```

from django.db import migrations, models
class Migration(migrations.Migration):
    dependencies = [
        ('store', '0003_product_category'),
    ]
    operations = [
        migrations.AlterField(
            model_name='product',
            name='description',
            field=models.CharField(default="", max_length=200, null=True),
        ),
    ]

```

Б.8 Файл 0005_auto_20200528_1732.py

```

from django.db import migrations, models
class Migration(migrations.Migration):
    dependencies = [
        ('store', '0004_auto_20200528_1430'),
    ]
    operations = [
        migrations.AlterField(
            model_name='product',
            name='description',
            field=models.CharField(blank=True, default="", max_length=200, null=True),
        ),
    ]

```

Б.9 Файл 0006_customer.py

```

from django.db import migrations, models
class Migration(migrations.Migration):
    dependencies = [
        ('store', '0005_auto_20200528_1732'),
    ]
    operations = [
        migrations.CreateModel(
            name='Customer',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
                ('first_name', models.CharField(max_length=50)),
                ('last_name', models.CharField(max_length=50)),
                ('phone', models.CharField(max_length=15)),
                ('email', models.EmailField(max_length=254)),
                ('password', models.CharField(max_length=500)),
            ],
        ),
    ]

```

Б.10 Файл 0007_auto_20200529_2246.py

```

import django.core.validators
from django.db import migrations, models
class Migration(migrations.Migration):
    dependencies = [
        ('store', '0006_customer'),
    ]
    operations = [
        migrations.AlterField(
            model_name='customer',
            name='email',

```

```

        field=models.EmailField(max_length=254,
validators=[django.core.validators.MinLengthValidator(limit_value=2)]),
    ),
]

```

Б.11 Файл 0008_auto_20200531_2229.py

```

from django.db import migrations, models
class Migration(migrations.Migration):
    dependencies = [
        ('store', '0007_auto_20200529_2246'),
    ]
    operations = [
        migrations.AlterField(
            model_name='customer',
            name='email',
            field=models.EmailField(max_length=254),
        ),
    ]

```

Б.12 Файл 0009_order.py

```

import datetime
from django.db import migrations, models
import django.db.models.deletion
class Migration(migrations.Migration):
    dependencies = [
        ('store', '0008_auto_20200531_2229'),
    ]
    operations = [
        migrations.CreateModel(
            name='Order',
            fields=[

```

```

        ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
        ('quantity', models.IntegerField(default=1)),
        ('price', models.IntegerField()),
        ('date', models.DateField(default=datetime.datetime.today)),
        ('customer',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='store.Customer')),
        ('product',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='store.Product')),
    ],
),
]

```

Б.13 Файл 0010_auto_20200531_2234.py

```

from django.db import migrations, models
class Migration(migrations.Migration):
    dependencies = [
        ('store', '0009_order'),
    ]
    operations = [
        migrations.AddField(
            model_name='order',
            name='address',
            field=models.CharField(blank=True, default="", max_length=50),
        ),
        migrations.AddField(
            model_name='order',
            name='phone',
            field=models.CharField(blank=True, default="", max_length=50),
        ),
    ]

```

Б.14 Файл 0011_order_status.py

```

from django.db import migrations, models
class Migration(migrations.Migration):
    dependencies = [
        ('store', '0010_auto_20200531_2234'),
    ]
    operations = [
        migrations.AddField(
            model_name='order',
            name='status',
            field=models.BooleanField(default=False),
        ),
    ]

```

Б.15 Файл category.py

```

from django.db import models

class Category(models.Model):
    name = models.CharField(max_length=20)

    @staticmethod
    def get_all_categories():
        return Category.objects.all()

    def __str__(self):
        return self.name

```

Б.16 Файл customer.py

```

from django.db import models
from django.core.validators import MinLengthValidator
class Customer(models.Model):
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)

```

```

phone = models.CharField(max_length=15)
email = models.EmailField()
password = models.CharField(max_length=500)
def register(self):
    self.save()
@staticmethod
def get_customer_by_email(email):
    try:
        return Customer.objects.get(email=email)
    except:
        return False
def isExists(self):
    if Customer.objects.filter(email = self.email):
        return True
    return False

```

Б.17 Файл orders.py

```

from django.db import models
from. product import Product
from. customer import Customer
import datetime

class Order(models.Model):
    product = models.ForeignKey(Product,
                                on_delete=models.CASCADE)
    customer = models.ForeignKey(Customer,
                                on_delete=models.CASCADE)
    quantity = models.IntegerField(default=1)
    price = models.IntegerField()
    address = models.CharField(max_length=50, default="", blank=True)
    phone = models.CharField(max_length=50, default="", blank=True)
    date = models.DateField(default=datetime.datetime.today)

```

```
status = models.BooleanField(default=False)
```

```
def placeOrder(self):
```

```
    self.save()
```

```
@staticmethod
```

```
def get_orders_by_customer(customer_id):
```

```
    return Order.objects.filter(customer=customer_id).order_by('-date')
```

Б.18 Файл product.py

```
from django.db import models
```

```
from .category import Category
```

```
class Product(models.Model):
```

```
    name = models.CharField(max_length=50)
```

```
    price = models.IntegerField(default=0)
```

```
    category = models.ForeignKey(Category, on_delete=models.CASCADE, default=1)
```

```
    description = models.CharField(max_length=200, default="", null=True, blank=True)
```

```
    image = models.ImageField(upload_to='uploads/products/')
```

```
@staticmethod
```

```
def get_products_by_id(ids):
```

```
    return Product.objects.filter(id__in =ids)
```

```
@staticmethod
```

```
def get_all_products():
```

```
    return Product.objects.all()
```

```
@staticmethod
```

```
def get_all_products_by_categoryid(category_id):
```

```
    if category_id:
```

```
        return Product.objects.filter(category = category_id)
```

```
    else:
```

```
        return Product.get_all_products();
```

Б.19 Файл base.html

```

<!doctype html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">

  <!-- Bootstrap CSS -->
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
  <style>
    nav{
      background-color: white;
    }
    .one-edge-shadow {
      box-shadow: 0 8px 2px -5px rgb(246, 245, 245);
    }
    .display-8{
      font-weight: 200;
      font-size: 30px;
    }
  </style>
</head>

<body>

  <!-- navbar -->
  <nav class="one-edge-shadow

```

```

sticky-top navbar
navbar-expand-lg
navbar-light bg-white">
  <a class="navbar-brand" href="/">Бджоломагазин</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent"
  aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
      </li>
    </ul>
    <ul class=" navbar-nav my-2 my-lg-0">
      <li class="nav-item ">
        <a class="nav-link" href="/cart">Кошик
          <span class="badge badge-info">{{ request.session.cart.keys|length }}</span>
          <span class="sr-only">(current)</span></a>
        </li>
        {% if request.session.customer %}

        <li class="nav-item ">
          <a class="nav-link" href="/orders">Замовлення<span class="sr-
only">(current)</span></a>
        </li>
        <li class="nav-item ">
          <a class="nav-link" href="/logout">Вийти<span class="sr-
only">(current)</span></a>
        </li>
        {% else %}

        <li class="nav-item ">

```

```

        <a      class="nav-link"      href="/signup">Реєстрація<span      class="sr-
only">(current)</span></a>
    </li>
    <li class="nav-item ">
        <a      class="nav-link"      href="/login">Авторизація<span      class="sr-
only">(current)</span></a>
    </li>
    {% endif %}
</ul>

```

```

</div>

```

```

</nav>

```

```

    {% block content %} {% endblock %}

```

```

<!-- Optional JavaScript -->

```

```

<!-- jQuery first, then Popper.js, then Bootstrap JS -->

```

```

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"

```

```

    integrity="sha384-

```

```

KJ3o2DKtIkvYIK3UENzmM7KCKR/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"

```

```

    crossorigin="anonymous"></script>

```

```

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"

```

```

    integrity="sha384-

```

```

ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"

```

```

    crossorigin="anonymous"></script>

```

```

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"

```

```

    integrity="sha384-

```

```

JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYI"

```

```

    crossorigin="anonymous"></script>

```

```

</body>

```

```

</html>

```

Б.20 Файл cart.html

```
{% extends 'base.html' %}
{% block content %}
<title>Кошик</title>
{% load cart %}
{% load custom_filter %}
<div class="container">
  <div class="border rounded p-4 m-4">
    <p class="display-4 pl-4 ml-4">Ваш кошик</p>
    <hr>
    <table class="table">
      <thead>
        <tr>
          <th>№</th>
          <th>Фото</th>
          <th>Товар</th>
          <th>Ціна</th>
          <th>Кількість</th>
          <th>Усього</th>
        </tr>
      </thead>
      <tbody>

        {% for product in products %}
        <tr>
          <td>{{ forloop.counter }}</td>
          <td></td>
          <td>{{ product.name }}</td>
          <td>{{ product.price|currency }}</td>
          <td>{{ product|cart_quantity:request.session.cart }}</td>
          <td>₾ {{ product|price_total:request.session.cart }}</td>
        </tr>
```

```

        {% endfor %}

</tbody>

<tfoot>
    <tr>
        <th colspan="4"></th>
        <th class="" colspan="">Усього</th>
        <th>{{ products|total_cart_price:request.session.cart }}</th>
    </tr>
</tfoot>
</table>
<hr>
<div class="m-3 p-3">
    <a href="#" data-toggle="modal" data-target="#exampleModal" class="btn btn-
outline-info border rounded col-lg-3 float-right">Замовити</a>
</div>
</div>
</div>

<!-- modal -->

<!-- Modal -->
<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel">
                    Форма замовлення
                </h5>
            <hr>
            <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                <span aria-hidden="true">&times;</span>

```

```

        </button>
    </div>
    <div class="modal-body">
        <div class="m-2 p-3">
            <form action="/check-out" method="POST">
                {% csrf_token %}
                <div class="form-group">
                    <label for="">Адреса</label>
                    <input type="text" name="address" id="" class="form-control"
placeholder="" aria-describedby="helpId">

                </div>
                <div class="form-group">
                    <label for="">Телефон</label>
                    <input type="text" name="phone" id="" class="form-control"
placeholder="" aria-describedby="helpId">
                </div>

                <input type="submit" class="btn float-right btn-outline-info col-lg-6"
value="Замовити">
            </form>
        </div>
    </div>
</div>
</div>

```

```
{% endblock %}
```

Б.21 Файл index.html

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

```

{% block content %}
<title>Бджоломагазин</title>
{% load cart %}
{% load custom_filter %}
<!-- body -->
<div class="container-fluid mt-3">
    <div class="row">
        <!-- filter -->
        <div class="col-lg-3 mx-auto">
            <div class="list-group">
                <a href="/" class="list-group-item list-group-item-
action">Усі товари</a>

                {% for category in categories %}
                <a href="/?category={{ category.id }}"
                    class="list-group-item list-group-item-
action">{{ category.name }}</a>

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

    <!-- all products -->
    <div id='products' class="col-lg-9 mx-auto">
        <div class="row mx-auto">
            {% for product in products %}
            <div class="card mx-auto mb-3" id={{ product.id }}
style="width: 18rem;">
                
                <div class="card-body">
                    <p class="card-title"><a href="{% url
'ProductDetailView' pk=product.pk %}">{{ product.name }}</a></p>
                    <p class="card-text"><b>
{{ product.price }}</b></p>

```

```

}} -->

</div>

<div class="card-footer p-0 no-gutters">

    {% if product|is_in_cart:request.session.cart

%}

<div class="row no-gutters">
    <form      action="/#{product.id}"
class="col-2 " method="post">

        {% csrf_token %}
        <input  hidden  type="text"
name='product' value='{{product.id}}'>

        <input  hidden  type="text"
name='remove' value='True'>

        <input type="submit" value=" -
" class="btn btn-block btn-light border-right">

        </form>
        <div      class="text-center
col">{{product|cart_quantity:request.session.cart}} у корзины</div>
        <form      action="/#{product.id}"
class="col-2 " method="post">

            {% csrf_token %}
            <input  hidden  type="text"
name='product' value='{{product.id}}'>

            <input type="submit" value="
+ " class="btn btn-block btn-light border-left">

        </form>
    </div>
    {% else %}
    <form      action="/#{product.id}"
method="POST" class="btn-block">

        {% csrf_token %}

```

```

<input      hidden      type="text"
name='product' value='{ {product.id} }'>

<input  type="submit"  class="float-
right btn btn-light form-control"

value="Додати до кошика">

</form>
{% endif %}

</div>

</div>
{% endfor %}

</div>

</div>

</div>

</div>

{% endblock %}

```

Б.22 Файл login.html

```

{% extends 'base.html' %}
{% block content %}
<title>Авторизація</title>
<div class="container">
  <div class="p-4 m-4">
    <div class="col-lg-4 mt-4 border mb-4 mx-auto rounded pt-4">
      <div class="text-center col">

        
<!-- <hr> -->
<div class="display-8">Авторизуйтесь</div>
<hr>
</div>
<h3 hidden class="alert alert-light border mt-4 mb-4 rounded">Увійти</h3>
<form action="/login" method="POST">

    {% csrf_token %}
    {% if error%}
    <div class="alert alert-danger" role="alert">
        {{error}}
    </div>
    {% endif %}
    <!-- email -->
    <div class="form-group">
        <label for="">Електронна пошта</label>
        <input required type="email" name="email" id=""
            value="{{ values.email }}"
            class="                form-control-sm                form-control"
placeholder="example@gmail.com">
    </div>

    <!-- password -->
    <div class="form-group">
        <label for="">Пароль</label>
        <input type="password"
            name="password"
            id=""
            class="form-control form-control-sm" placeholder="*****">
    </div>

    <hr>
    <div class="form-group">

```

```

        <input class="btn btn-sm btn-info col-lg-4" type="submit"
value="Авторизація">
    </div>

```

```

    </form>
</div>
</div>
</div>

```

```
{% endblock %}
```

Б.23 Файл orders.html

```

{% extends 'base.html' %}
{% block content %}
<title>Ваші замовлення</title>
{% load cart %}
{% load custom_filter %}
<div class="container">
    <div class="border rounded p-4 m-4">
        <p class="display-4 pl-4 ml-4">Ваші замовлення</p>
        <hr>
        <table class="table">
            <thead>
                <tr>
                    <th>№</th>
                    <th>Фото</th>
                    <th>Товар</th>
                    <th>Ціна</th>
                    <th>Кількість</th>
                    <th>Усього</th>
                    <th>Статус</th>
                </tr>
            </thead>

```

```

<tbody>

    {% for order in orders %}
    <tr>
        <td>{{ forloop.counter }}</td>
        <td></td>
        <td>{{ order.product.name }}</td>
        <td>{{ order.date }}</td>
        <td>{{ order.price|currency }}</td>
        <td>{{ order.quantity }}</td>
        <td> ₺ {{ order.quantity|multiply:order.price }}</td>
        {% if order.status %}
        <td><small class="badge badge-success">Completed</small></td>
        {% else %}
        <td><small class="badge badge-warning">Pending</small></td>
        {% endif %}
    </tr>

    {% endfor %}

</tbody>

</table>

</div>
</div>
{% endblock %}

```

Б.24 Файл product_details.html

```

{% extends "./base.html" %}
{% load cart %}
{% load custom_filter %}

```

```

{% block content %}
<div class="container">
  <div class="row">
    <div class="col">
      <div class="p-4 m-3">
        <div class="w-75 text-center mx-auto">
          
        </div>
      </div>
    </div>
    <div class="col p-3 m-2">
      <br>
      <br>
      <div class="">
        <div class="display-4">{{product.name}}</div>
        <hr>
      </div>

      <br>
      <div class="p-3 display-8">
        <span class="text-danger"> € {{ product.price }}</span>
        &nbsp;
        <h4><span>{{product.description}}</span></h4>
      </div>

      {% if product|is_in_cart:request.session.cart %}
        <div class="row no-gutters">
          <form class="col-2 " method="post">
            <input type="text"
name='product' value='{{product.id}}'>
            </input
name='product' value='{{product.id}}'>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>

```

```

name='remove' value='True'>
    <input hidden type="text"

    <input type="submit" value=" -

" class="btn btn-block btn-light border-right">

</form>
<div class="text-center
col">{{product|cart_quantity:request.session.cart}} у кошику</div>
<form action="/#{product.id}"
class="col-2 " method="post">

    {% csrf_token %}
    <input hidden type="text"

    name='product' value='{{product.id}}'>

    <input type="submit" value="

+ " class="btn btn-block btn-light border-left">

</form>
</div>
{% else %}
<form action="/#{product.id}"

    {% csrf_token %}
    <input hidden type="text"

    name='product' value='{{product.id}}'>

    <input type="submit" class="float-

    right btn btn-light form-control"

    value="Додати до кошика">

</form>
{% endif %}

</div>
</div>
</div>
</div>
</div>
</div>
{% endblock %}

```

Б.25 Файл signup.html

```
{% extends 'base.html' %}
{% block content %}
<title>Рєєстрація</title>
<div class="container mt-4">
  <div class="p-4 m-4">
    <div class="col-lg-5 rounded mx-auto border pt-4">
      <div class="text-center col">
        
      <!-- <hr> -->
      <div class="display-8">Створіть аккаунт</div>
      <hr>
    </div>
    <h3 hidden class="alert alert-light border rounded">Створіть аккаунт</h3>
    <form action="/signup" method="POST">

      {% csrf_token %}

      {% if error%}
      <div class="alert alert-danger" role="alert">
        {{error}}
      </div>
      {% endif %}
      <!-- firstname -->
      <div class="form-group">
        <label for="">Ім'я</label>
        <input type="text" name="firstname"
id="" value="{{ values.first_name }}" class="form-control form-control-sm"
placeholder="">
      </div>
```

```

<!-- last name -->
<div class="form-group">
  <label for="">Прізвище</label>
  <input type="text" name="lastname"
    id="" value="{{ values.last_name }}" class="form-control form-control-sm"
  >
</div>

```

```

<!-- phone -->
<div class="form-group">
  <label for="">Телефон</label>
  <input type="text" name="phone"
    id="" class="form-control form-control-sm"
    value="{{ values.phone }}"
    placeholder="565698959"
  >
</div>

```

```

<!-- email -->
<div class="form-group">
  <label for="">Електронна пошта</label>
  <input required type="email" name="email" id=""
    value="{{ values.email }}"
    class="form-control-sm form-control"
    placeholder="example@gmail.com">
</div>

```

```

<!-- password -->
<div class="form-group">
  <label for="">Пароль</label>
  <input type="password"
    name="password"
    id=""
    class="form-control form-control-sm" placeholder="*****">

```

```

        </div>

        <hr>
        <div class="form-group">
            <input class="btn btn-sm btn-info" type="submit" value="Створити
аккаунт ">
        </div>

```

```

        </form>
    </div>
</div>
</div>

```

```
{% endblock %}
```

Б.26 Файл cart.py

```

from django import template
register = template.Library()

```

```

@register.filter(name='is_in_cart')
def is_in_cart(product, cart):
    keys = cart.keys()
    for id in keys:
        if int(id) == product.id:
            return True
    return False;

```

```

@register.filter(name='cart_quantity')
def cart_quantity(product, cart):
    keys = cart.keys()
    for id in keys:
        if int(id) == product.id:

```

```

        return cart.get(id)
    return 0;

```

```

@register.filter(name='price_total')
def price_total(product, cart):
    return product.price * cart_quantity(product, cart)

```

```

@register.filter(name='total_cart_price')
def total_cart_price(products, cart):
    sum = 0 ;
    for p in products:
        sum += price_total(p, cart)

    return sum

```

Б.27 Файл custom_filter.py

```

from django import template
register = template.Library()
@register.filter(name='currency')
def currency(number):
    return "₹ "+str(number)

```

```

@register.filter(name='multiply')
def multiply(number, number1):
    return number * number1
from django import template
register = template.Library()

```

```

@register.filter(name='currency')
def currency(number):
    return "₹ "+str(number)

```

```
@register.filter(name='multiply')
def multiply(number, number1):
    return number * number1
```

Б.28 Файл cart.py

```
from django.shortcuts import render, redirect
from django.contrib.auth.hashers import check_password
from store.models.customer import Customer
from django.views import View
from store.models.product import Product
class Cart(View):
    def get(self, request):
        ids = list(request.session.get('cart').keys())
        products = Product.get_products_by_id(ids)
        print(products)
        return render(request, 'cart.html', {'products': products} )
```

Б.29 Файл checkout.py

```
from django.shortcuts import render, redirect
from django.contrib.auth.hashers import check_password
from store.models.customer import Customer
from django.views import View
from store.models.product import Product
from store.models.orders import Order

class CheckOut(View):
    def post(self, request):
        address = request.POST.get('address')
        phone = request.POST.get('phone')
        customer = request.session.get('customer')
```

```

cart = request.session.get('cart')
products = Product.get_products_by_id(list(cart.keys()))
print(address, phone, customer, cart, products)

for product in products:
    print(cart.get(str(product.id)))
    order = Order(customer=Customer(id=customer),
                  product=product,
                  price=product.price,
                  address=address,
                  phone=phone,
                  quantity=cart.get(str(product.id)))
    order.save()
request.session['cart'] = {}

return redirect('cart')

```

Б.30 Файл **home.py**

```

from django.shortcuts import render, redirect, HttpResponseRedirect
from store.models.product import Product
from store.models.category import Category
from django.views import View

class Index(View):

    def post(self, request):
        product = request.POST.get('product')
        remove = request.POST.get('remove')
        cart = request.session.get('cart')
        if cart:
            quantity = cart.get(product)
            if quantity:
                if remove:
                    if quantity >= 1:

```

```

        cart.pop(product)
    else:
        cart[product] = quantity-1
    else:
        cart[product] = quantity+1

    else:
        cart[product] = 1
    else:
        cart = {}
        cart[product] = 1

    request.session['cart'] = cart
    print('cart', request.session['cart'])
    return redirect('homepage')

```

```

def get(self, request):
    # print()
    return HttpResponseRedirect(f'/store{request.get_full_path()[1:]}')

```

```

def store(request):
    cart = request.session.get('cart')
    if not cart:
        request.session['cart'] = {}
    products = None
    categories = Category.get_all_categories()
    categoryID = request.GET.get('category')
    if categoryID:
        products = Product.get_all_products_by_categoryid(categoryID)
    else:
        products = Product.get_all_products();

    data = {}

```

```

data['products'] = products
data['categories'] = categories

print('you are : ', request.session.get('email'))
return render(request, 'index.html', data)

```

Б.31 Файл login.py

```

from django.shortcuts import render, redirect, HttpResponseRedirect
from django.contrib.auth.hashers import check_password
from store.models.customer import Customer
from django.views import View

class Login(View):
    return_url = None

    def get(self, request):
        Login.return_url = request.GET.get('return_url')
        return render(request, 'login.html')

    def post(self, request):
        email = request.POST.get('email')
        password = request.POST.get('password')
        customer = Customer.get_customer_by_email(email)
        error_message = None
        if customer:
            flag = check_password(password, customer.password)
            if flag:
                request.session['customer'] = customer.id

            if Login.return_url:
                return HttpResponseRedirect(Login.return_url)
            else:
                Login.return_url = None
                return redirect('homepage')
        else:

```

```

        error_message = 'Email or Password invalid !!'
    else:
        error_message = 'Email or Password invalid !!'

    print(email, password)
    return render(request, 'login.html', {'error': error_message})

def logout(request):
    request.session.clear()
    return redirect('login')
```

Б.32 Файл orders.py

```

from django.shortcuts import render, redirect
from django.contrib.auth.hashers import check_password
from store.models.customer import Customer
from django.views import View
from store.models.product import Product
from store.models.orders import Order
from store.middlewares.auth import auth_middleware
class OrderView(View):
    def get(self, request ):
        customer = request.session.get('customer')
        orders = Order.get_orders_by_customer(customer)
        print(orders)
        return render(request, 'orders.html', {'orders': orders})
```

Б.33 Файл product_page.py

```

from store.models.product import Product
from store.models.orders import Order
from django.shortcuts import render, HttpResponseRedirect, redirect
from math import floor
from django.views.generic.detail import DetailView
```

```

class ProductDetailView(DetailView):
    template_name = 'product_details.html'
    model = Product

    def product_detail(request, id, slug=None):
        product = get_object_or_404(Product,
                                     id=id,
                                     slug=slug)

        return render(request, 'store/product_details.html', {'product': product})

    def post_detail(request, pk):
        post = get_object_or_404(Product, pk=pk)
        return render(request, 'store/product_details.html', {'post': post})

```

Б.34 Файл `signup.py`

```

from django.shortcuts import render, redirect
from django.contrib.auth.hashers import make_password
from store.models.customer import Customer
from django.views import View

class Signup(View):
    def get(self, request):
        return render(request, 'signup.html')

    def post(self, request):
        postData = request.POST
        first_name = postData.get('firstname')
        last_name = postData.get('lastname')
        phone = postData.get('phone')
        email = postData.get('email')
        password = postData.get('password')
        # validation

```

```

value = {
    'first_name': first_name,
    'last_name': last_name,
    'phone': phone,
    'email': email
}
error_message = None

customer = Customer(first_name=first_name,
                    last_name=last_name,
                    phone=phone,
                    email=email,
                    password=password)
error_message = self.validateCustomer(customer)

if not error_message:
    print(first_name, last_name, phone, email, password)
    customer.password = make_password(customer.password)
    customer.register()
    return redirect('homepage')
else:
    data = {
        'error': error_message,
        'values': value
    }
    return render(request, 'signup.html', data)

def validateCustomer(self, customer):
    error_message = None;
    if (not customer.first_name):
        error_message = "First Name Required !!"
    elif len(customer.first_name) < 4:
        error_message = 'First Name must be 4 char long or more'
    elif not customer.last_name:
        error_message = 'Last Name Required'

```

```

elif len(customer.last_name) < 4:
    error_message = 'Last Name must be 4 char long or more'
elif not customer.phone:
    error_message = 'Phone Number required'
elif len(customer.phone) < 10:
    error_message = 'Phone Number must be 10 char Long'
elif len(customer.password) < 6:
    error_message = 'Password must be 6 char long'
elif len(customer.email) < 5:
    error_message = 'Email must be 5 char long'
elif customer.isExists():
    error_message = 'Email Address Already Registered..'
return error_message

```

Б.35 Файл `urls.py`

```

from django.contrib import admin
from django.urls import path
from . views.home import Index, store
from . views.signup import Signup
from . views.login import Login, logout
from . views.cart import Cart
from . views.checkout import CheckOut
from . views.orders import OrderView
from . middlewares.auth import auth_middleware
from . views.product_page import ProductDetailView

urlpatterns = [
    path("", Index.as_view(), name='homepage'),
    path('store', store, name='store'),

    path('signup', Signup.as_view(), name='signup'),
    path('signup/', Signup.as_view(), name='signup'),
    path('login/', Login.as_view(), name='login'),
    path('login', Login.as_view(), name='login'),

```

```
path('logout', logout, name='logout'),
path('logout/', logout, name='logout'),
path('cart',auth_middleware(Cart.as_view()), name='cart'),
path('cart/', auth_middleware(Cart.as_view()), name='cart'),
path('check-out', CheckOut.as_view(), name='checkout'),
path('check-out/', CheckOut.as_view(), name='checkout'),
path('orders', auth_middleware(OrderView.as_view()), name='orders'),
path('orders/', auth_middleware(OrderView.as_view()), name='orders'),
path('product/<int:pk>', ProductDetailView.as_view(), name = "ProductDetailView"),
]
```

ДОДАТОК В
Слайди презентації



Рисунок В.1 – Слайд № 1



Рисунок В.2 – Слайд № 2

Способи створення Інтернет-магазинів

Переваги

Переваги CMS	Переваги фреймворків
Вже створено структуру Інтернет-магазину, що економить час та бюджет.	Дозволяє реалізувати всі вимоги та задумки клієнтів.
Базова функціональність надається відразу «з коробки»	Дозволяє створити унікальний точковий функціональності, проект
Якщо вимоги прості та стандартні, реалізація буде легкою результатом – передбачуваним.	Додаючи нові функції можна адаптувати сайт таким чином, щоб втрата продуктивності була мінімальна.
У будь-який час ви можете легко змінити шаблон чи дизайн сайту.	Сайти, розроблені на базі Framework, працюють значно швидше, ніж аналоги які розроблені на базі CMS.

5

Рисунок В.3 – Слайд № 3

Способи створення Інтернет-магазинів

Недоліки

Недоліки CMS	Недоліки фреймворків
Відсутність унікальності. Кожен може використати таку ж тему CMS.	Фаза розробки, як правило, займає більше часу ніж у випадку з CMS.
Продуктивність обмежена обраною CMS. Впровадження додаткових функцій, базується, як правило, на додаткових витратах, та не завжди можливо.	Необхідні високоосвічені спеціалісти, оскільки розробка значно складніша ніж побудова Інтернет-магазину на CMS.
Поновлення ліцензії має відбуватись щороку.	Усі зміни слід замовляти у розробників, адже ви не можете їх впровадити самостійно.
Стандартне рішення, як правило, виключає в себе непотрібні функціональні можливості, які накладають навантаження на сайт.	Створення сайту, який по функціональності буде рівний аналогу на CMS, займе більше часу та ресурсів.

6

Рисунок В.4 – Слайд № 4

Розробка програми

Під час розробки програми було розроблено декілька необхідних моделі, а саме:

МОДЕЛЬ CATEGORY

```
1 from django.db import models
2
3 class Category(models.Model):
4     name = models.CharField(max_length=20)
5
6     @staticmethod
7     def get_all_categories():
8         return Category.objects.all()
9
10
11 def __str__(self):
12     return self.name
13
```

МОДЕЛЬ PRODUCT

```
1 import ...
2
3 class Product(models.Model):
4     name = models.CharField(max_length=50)
5     price = models.IntegerField(default=0)
6     category = models.ForeignKey(Category, on_delete=models.CASCADE, default=1)
7     description = models.CharField(max_length=200, default='', null=True, blank=True)
8     image = models.ImageField(upload_to='uploads/products/')
9
10     @staticmethod
11     def get_products_by_id(ids):
12         return Product.objects.filter(id__in=ids)
13
14     @staticmethod
15     def get_all_products():
16         return Product.objects.all()
17
18     @staticmethod
19     def get_all_products_by_category_id(category_id):
20         if category_id:
21             return Product.objects.filter(category=category_id)
22         else:
23             return Product.get_all_products()
24
```

5

Рисунок В.5 – Слайд № 5

Розробка програми

МОДЕЛЬ CUSTOMER

```
1 from django.db import models
2 from django.core.validators import MinLengthValidator
3
4 class Customer(models.Model):
5     first_name = models.CharField(max_length=30)
6     last_name = models.CharField(max_length=30)
7     phone = models.CharField(max_length=15)
8     email = models.EmailField()
9     password = models.CharField(max_length=100)
10
11     def register(self):
12         self.save()
13
14     @staticmethod
15     def get_customer_by_email(email):
16         try:
17             return Customer.objects.get(email=email)
18         except:
19             return False
20
21     def authenticate(self):
22         if Customer.objects.filter(email=self.email).exists():
23             return True
24         return False
25
```

МОДЕЛЬ ORDER

```
1 from django.db import models
2 from django.conf import settings
3 from django.db import connection
4
5 class Order(models.Model):
6     product = models.ForeignKey(Product,
7                                   on_delete=models.CASCADE)
8     customer = models.ForeignKey(Customer,
9                                   on_delete=models.CASCADE)
10     quantity = models.IntegerField(default=1)
11     price = models.IntegerField()
12     address = models.CharField(max_length=100, default='', blank=True)
13     phone = models.CharField(max_length=15, default='', blank=True)
14     date = models.DateTimeField(auto_now_add=True)
15     status = models.BooleanField(default=False)
16
17     def create_order(self):
18         self.save()
19
20     @staticmethod
21     def get_orders_by_customer(customer_id):
22         return Order.objects.filter(customer=customer_id)
23
```

6

Рисунок В.6 – Слайд № 6



Рисунок В.7 – Слайд № 7

Вибір мови програмування

Java	Python
Код java схожий на C++, його більш важко читати.	Код Python значно простіше тому його легше читати.
Більш складний синтаксис	Простий синтаксис
Статична типізація	Динамічна типізація
Більш складна компіляція в два етапи	Проста компіляція

8

Рисунок В.8 – Слайд № 8

Розробка архітектури програми

Вибір фреймворку

Фреймворк Flask	Фреймворк Django
Використовує локальні потоки даних	Відображає запит лише в тому місці, де це необхідно.
SQLAlchemy функціонал, що спрощує управління БД	Вбудована ORM система має чудову систему міграцій, що значно спрощує управління БД
Відсутні вбудовані форми, однак для подолання цієї проблеми використовують WTForgm.	Django доступний виразу з вбудованими формами, що чудово співпрацюють з ORM системою
В якості інструмента для роботи з реалізацій функції аутентифікації, Flask надає безпечні cookie файли.	Django по замовчуванню надає додаток аутентифікації, що значно спрощує розробку всього проекту.

9

Рисунок В.9 – Слайд № 9

Висновки

- ❑ Проведено аналіз предметної області
- ❑ Сформовано ряд вимог до проектування власного вебресурсу
- ❑ Проведено огляд сучасних технологій та методів розробки вебплатформ
- ❑ Обгрунтовано вибір фреймворку Django
- ❑ Спроектовано зручну процедуру покупки товару для клієнтів та зручну систему адміністрування Інтернет-магазину для продавців
- ❑ Виконано тестування розробленого програмного продукту

10

Рисунок В.10 – Слайд №10