**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**
**ІМЕНІ ІВАНА ФРАНКА**

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Дискретного аналізу та інтелектуальних систем
(повна назва кафедри)

# ДИПЛОМНА РОБОТА

на тему:
Розробка веб-застосунку засобами Python та фреймворку Django

Виконав: студент 4 курсу, групи ПМі-45 ,
спеціальності
122 «Комп'ютерні науки»
(шифр та назва спеціальності)
Підгорецький А. В
(прізвище та ініціали)

Керівник доц. Олійник Р. М.
(підпис) (прізвище та ініціали)
Рецензент
(підпис) (прізвище та ініціали)

Львів – 2023
***ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА***

**Факультет** Прикладної математики та інформатики

**Кафедра** Дискретного аналізу та інтелектуальних систем

**Спеціальність** 122 «Комп'ютерні науки»

(шифр і назва)

**«ЗАТВЕРДЖУЮ»**

**Завідувач кафедри**_____

**"31" серпня 2022 року**

# ЗАВДАННЯ

## НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Підгорецький Арсен

( прізвище, ім'я, по батькові)

1. Тема роботи Розробка веб-застосунку засобами Python та фреймворку Django

керівник роботи Олійник Роман Миколайович, доцент кафедри дискретного аналізу на інтелектуальних систем,

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

затверджені Вченою радою факультету від **"13" вересня 2022 року №15**

2. Строк подання студентом роботи **13.06.2023р.**

3. Вихідні дані до роботи мова програмування Python, середовщие розробки PyCharm

4. Зміст дипломної роботи (перелік питань, які потрібно розробити) огляд рішень, вибір мови програмування, вибір бази даних, огляд доступних фреймворків, контейнеризація веб-застосунку, огляд розробки за допомогою Django, розробка веб-застосунку

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Популярність мов програмування, архітектура MVC, зображення мов програмування, зображення фреймворків, зображення баз даних, зображення сервісів для контейнеризації, архітектура MVT в Django, сумісність версій Django, приклад розробки базового додатку, діаграма бази даних

6. Консультанти розділів роботи

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

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання **31 серпня 2022 р.**

**КАЛЕНДАРНИЙ ПЛАН**

| № з/п | Назва етапів дипломної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| **1** | **Вивчення предметної області** | **01.09.2022 – 01.10.2022** | |
| **2** | **Огляд доступних мов програмування** | **02.10.2022 – 25.10.2022** | |
| **3** | **Огляд можливих фреймворків із вибраною мовою програмування** | **26.10.2022 – 17.11.2022** | |
| **4** | **Вибір бази даних** | **18.11.2022 – 10.12.2022** | |
| **5** | **Аналіз можливих сервісів для контейнеризації веб-застосунку** | **11.12.2022 – 13.01.2023** | |
| **6** | **Вивчення вибраного фреймворку** | **13.01.2023 – 13.02.2023** | |
| **7** | **Розробка серверної частини веб-застосунку** | **14.02.2023 – 28.02.2023** | |
| **8** | **Розробка клієнтської частини веб-застосунку** | **01.03.2023 – 28.03.2023** | |
| **9** | **Вивчення технології для контейнеризації** | **29.03.2023 – 11.04.2023** | |
| **10** | **Контейнеризація веб-застосунку** | **12.04.2023 – 19.04.2023** | |
| **11** | **Оформлення дипломної роботи** | **20.04.2023 – 30.05.2023** | |

**Керівник роботи** _____

(підпис) (прізвище та ініціали)

# ЗМІСТ

# ВСТУП

В сучасному світі більшість задач оптимізується і спрощується. За допомогою інтернету багато задач стало робити простіше. Щодня в мережі Інтернет з'являється велика кількість різних сайтів. За допомогою таких сайтів люди дізнаються безліч цікавих та корисних звісток, не виходячи з дому.

Тема інтернет-торгівлі є надзвичайно актуальною в сьогодення, тому що на сьогодні багато людей віддає перевагу покупкам в інтернеті. Це набагато зручніше, оскільки купуючи різні товари в інтернет магазинах не потрібно виходити з дому та витрачати багато часу. В інтернеті все набагато простіше, достатньо клікнути кнопкою мишки. Люди, які мають свої власні магазини, часто стараються відобразити їх в інтернеті, для них це не лишень зручно, а й приносить нових клієнтів. Звісно, що інтернет- магазини не замінять традиційні магазини, а просто розширять сферу застосування та ринку. У деяких сферах торгівлі відсутність інтернет-магазинів є стратегічним недоліком. Інтернет-магазини використовують фізичні магазини, як частину свого ланцюжка постачань, щоб скоротити витрати на управління запасами і прискорити доставку, тим самим збільшити прибуток. Також це дає багато інших переваг користувачам: вони з легкістю можуть дізнатись додаткову інформацію про магазин, подивитись наявний каталог товарів та їх ціни. У світі кожного дня стає все більше користувачів інтернету і по цій причині збільшується кількість нових клієнтів у інтернет магазинах. Ось тому інтернет-магазин завжди буде залишатися актуальним.

Метою роботи є дослідження можливостей створення веб-застосунку засобами Python та фреймворку Django.

Потрібно реалізувати такі завдання:

- дослідити особливості мов програмування
- дослідити доступні фреймворки вибраної мови
- дослідити бази даних
- описати використання Django в проекті
- розробити веб-застосунок

# РОЗДІЛ 1. ОГЛЯД РІШЕНЬ

## 1.1 Вибір реалізації

Сьогодні існує чимало мов програмування. Для кожної з них є свої плюси і мінуси. Вибір мови залежить від поставлених задач, від уподобань і знань самого розробника.
Для реалізації певного завдання потрібно правильно підібрати інструменти – мову програмування, фреймворк.

### 1.1.1 Огляд мов програмування

Мови програмування — це формальні комп'ютерні мови. Звичайно, в реальному житті на них ніхто не розмовляє. Мови програмування використовуються для створення програм, які контролюють поведінку машин, та для запису алгоритмів. На різних мовах програмування пишуться різні програми, які висловлюють певні алгоритми або контролюють поведінку комп'ютера. Від природних мов вони разюче відрізняються. І, щоб почати вчити якусь мову, спочатку треба зрозуміти принцип його роботи**[1]**.

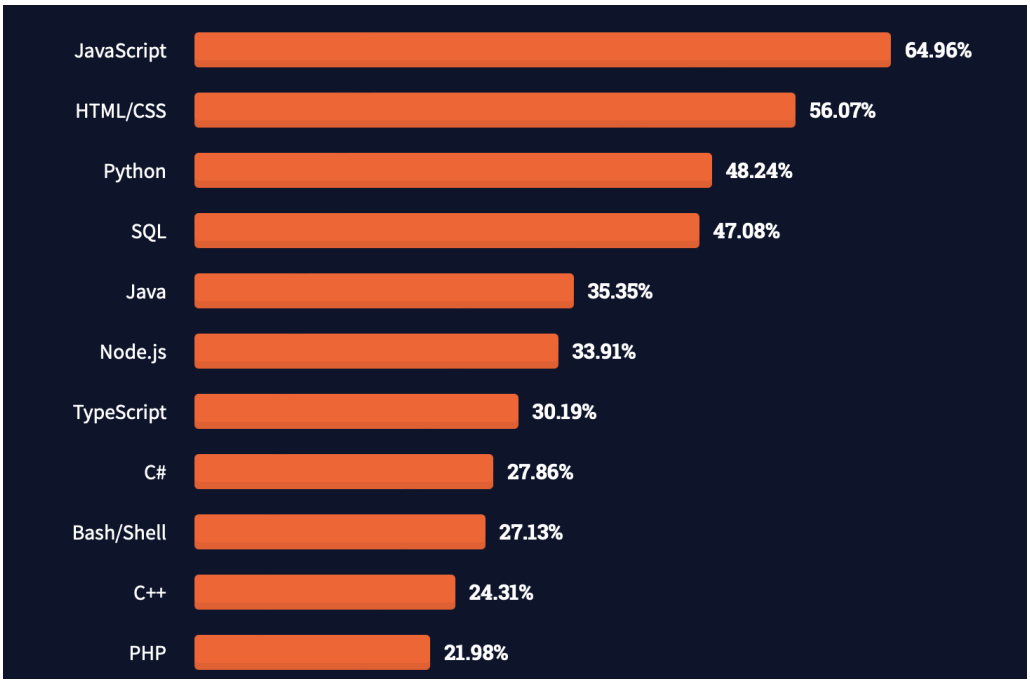



Рисунок 1.1 – Популярність мов програмування**[2]**.

Мови веб-програмування поділяються на клієнтські та серверні мови. Клієнтські мови програмування(Front End) використовуються на стороні клієнта, тобто веб-браузера. Серверні мови програмування(Back End) використовуються на веб сервері, для обробки запитів клієнта. JavaScript(JS) – найпопулярніша клієнтська мова програмування, яку розробили у 1995 році. Майже у будь-якому популярному веб-сайті використовується JavaScript. JS допомагає зробити сторінки сайту більш інтерактивними, додати нові ефекти, які будуть взаємодіяти з користувачем, обробляти дії користувачів сайту. Це об'єктно-орієнтована клієнтська мова, яка підтримується додатками, що працюють з дизайном сайту**[3]**. Великою перевагою JavaScript є те що він простий і легко піддається вивченню. На основі JS побудовано чимало популярних фреймворків. JS з кожним роком удосконалюється. Деякі фахівці вважають що у самій логіці

JS є деякі недопрацювання та поступово усувають їх. Порівняно з минулими роками до розробників JS все менше і менше надходить скарг**[4]**.
HTML, CSS – невід'ємна частина під час розробки веб-сайту.
HTML (англ. HyperText Markup Language – мова розмітки гіпертексту) – стандартизована мова розмітки документів для перегляду веб-сторінок у браузері. Веб-браузери отримують HTML документ від сервера за протоколами HTTP/HTTPS або відкривають з локального диска, далі інтерпретують код в інтерфейс, який відображатиметься на екрані монітора**[5]**.
CSS (англ. Cascading Style Sheets, укр. Каскадні таблиці стилів) — це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних**[5]**. За допомогою CSS можна підкреслити певні елементи сайту, зробити їх більш лаконічними.
Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня, одна з найпопулярніших серверних мов програмування, універсальна, динамічна та проста у використанні та розробці. Синтаксис мови є простим, легким для читання і простим у вивченні. Python використовують у багатьох різних напрямках: веб-розробка, машинне навчання, автоматизоване тестування, і тд. На мою думку, для новачків в сфері комп'ютерних наук мова програмування Python одна з найкращих для вступу у сферу IT**[8, 9]**.
SQL (англ. Structured query language — мова структурованих запитів) — декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних**[6]**.
Часто в людей виникає думка, що SQL не потрібний, оскільки ми можемо створити програму, яка буде робити аналогічні прості операції(CRUD) як і SQL і повертатиме такий же результат. Але все не так просто, оскільки коли почнуть надходити складні і часозатратні запити, потрібно буде написати чимало рядків коду, щоб це подати кількома рядками SQL, і не завжди велика кількість коду буде працювати швидше ніж SQL запит**[7]**.
Java – найпопулярніша мова програмування на даний час. Вона створювалась для програмування побутових пристроїв у США, але з часом компанія, що її створила, змінила спрямування і мова набула ширшого кола застосування. Мова була названа в честь сорту кави – Java. У неї навіть логотип відповідний – силует горнятка кави з парою**[10]**.
Для розробки інтернет-магазину обрав мову програмування Python, так як у Python простий синтаксис, динамічна та нестрога типізація.

### 1.1.2 Огляд Python фреймворків

При створенні веб-сайту, завжди, потрібен схожий функціонал, набір компонентів: панель управління, форми, валідація даних, створення акаунту, вхід, вихід, і багато інших. Зіткнувшись з цією проблемою і прагнучи відповідати потребам ринку IT-технологій, веб-фахівці об'єдналися в команду і створили фреймворки, які містять готові до використання компоненти**[17]**.
Фреймворк – набір компонентів, які полегшують процес написання, та допомагають розробляти веб-додатки швидше. Стандартна архітектура фреймворків грунтується на трьох шарах MVC – модель, уявлення, контроллер.

- Модель – відповідає за формування структури, бізнес логіки.
- Уявлення – графічне відображення даних
- Контроллер – відповідає за зв'язок з користувачем.

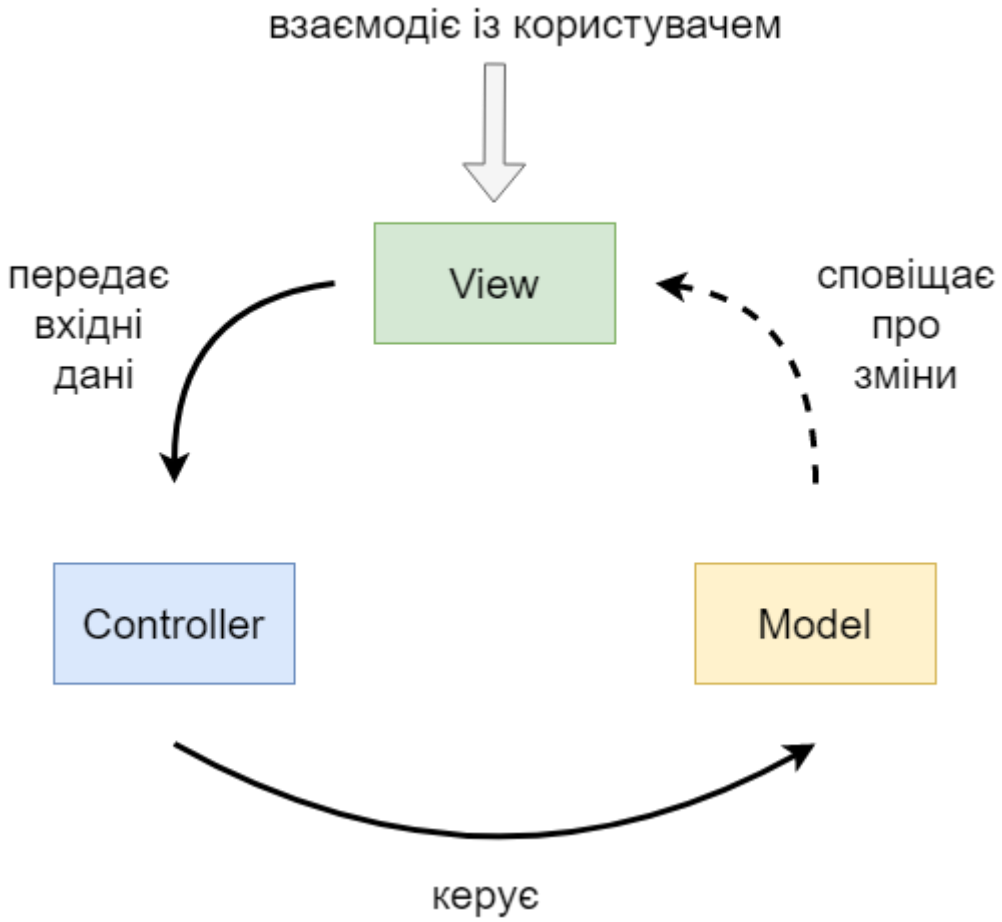



Рисунок 1.2 – Архітектура MVC

Розробка сайті з використання фреймворків отримала чимало переваг:

- висока швидкість створення додатків
- застосування фреймворка забезпечує сайт швидким реагуванням і здатністю витримувати великі навантаження
- велика кількість готових компонентів

Фреймворки поділяються на такі типи: Full-Stack Framework, мікрофреймворки(Microframework), асинхронні фреймворки(Asynchronous Framework).

Full-Stack Framework – фреймворки цього типу є хорошим рішенням для всіх вимог розробників. Макети шаблонів, генератори форм, валідація даних зазвичай це все є доступно в даному типі фреймворків.

Microframework – цей тип фреймворків немає додаткових функцій і особливостей, таких як перевірка форм, рівень абстракцій бази даних, додактових інструментів та бібліотек. Розробники які використовують цей тип фреймворків потрібно вручну реалізовувати поставлені задачі, додавши більше коду, ніж це б було потрібно за допомогою Full-Stack фреймворків.

Asynchronous Framework – останнім часом стають все більш популярніше. Будь який асинхронний фреймворк це мікрофреймворк який дозволяє обробляти багато з'єднань одночасно. Як правило, асинхронна структура на основі використовує бібліотеку asyncio.для Python

Найпопулярніші Python фреймворки: Django, Flask, Tornado, FastAPI, і тд.


Рисунок 1.3 – Логотип Django

Django – Full-Stack фреймворк, є одним найпопулярніших фреймворків 2022 року. Безкоштовний у використанні та з відкритим вихідним кодом. Містить велику кількість вбудованих функцій, без використання інших бібліотек. Для роботи з базами даних Django використовує власний ORM, це дозволяє вільно працювати з будь-якою базою даних, також цим він полегшує перехід з однієї бази даних на іншу. Django має підтримку MySQL, PostgreSQL, SQLite та Oracle Database і додаткови може підтримувати будь яку іншу базу даних, використовуючи сторонні драйвера. Ключові особливості використання фреймворку: підтримка аутентифікації, міграції схеми бази даних, підтримка шаблонів, маршрутизація URL-адрес, наявність власного ORM, безліч готових до використання бібліотек, підтримка веб серверів.


Рисунок 1.4 – Логотип Flask

Flask – відноситься до типу мікрофреймворків. Для налаштування та встановлення потрібно набагато менше часу, ніж з іншими фреймворками. Основна ідея Flask полягає в тому, щоб допомогти створити міцну основу веб-застосунків. Також Flask відрізняється мінімальною кількістю базового функціонал, але в будь-який момент його додатково можна розширити і додати потрібний функціонал. Але все ж містить чимало корисних функцій і особливостей: показує гарну продуктивність, вбудований сервер розробки, підтримує роботу з шаблонами, можливість підключити будь-яку ORM, підтримка безпечних файлів cookie, запобіганню витоку інформації та запобіганню веб-атак.


Рисунок 1.5 – Логотип Tornado

Tornado – ідеальний асинхронний фреймворк, з відкритим кодом, за допомогою якого можна створити програми які потребують високої продуктивності. Головна особливість цього фреймворку, це те що він запросто вирішує проблеми 10000 з'єднань(C10k), що при правильній настройці дає йому змогу обробляти десятки тисяч з'єднань одночасно. Фреймворк Tornado це щось середнє між Flask і Django. Головні переваги і особливості: дозволяє реалізувати сторонні схеми авторизації та аутентифікації, сервіси в режимі реального часу, підтримує локалізацію та переклад, хороша структура, неблокуючий HTTP-клієнт.


Рисунок 1.6 – Логотип FaskAPI

FastAPI – це є відносно новий асинхронний фреймворк, випущений у 2018 році. Цей фреймворк це є гібрид Starlett та Pydantic. Starlett — асинхронний веб-фреймворк, Pydantic — бібліотека для серіалізації, валідації даних, тощо. Переваги даного фреймворку:

- Висока продуктивність
- Можна швидко створити потрібний додаток – розробникам не потрібно створювати все з нуля, і завдяки цьому створення додатку не займає багато часу, порівняно з іншими фреймворками
- Легкий для вивчення – кожен хто добре розуміє Python може з легкістю працювати з даним фреймворком, фреймворк не має особливого синтаксису, це просто розширена версія Python.
- Сумісний з OpenAPI(Swagger)
- Проста інтеграція з базою даних**[13-15]**.

Вирішив використовувати фреймворк Django, так як зараз він є надзвичайно популярним на ринку, у нього є власний ORM, проста система аутентифікації, і багато різного корисного функціоналу.

Для розробки клієнтської частини інтернет магазину використовуємо Django Templates для генерації контенту. Також для основи використовуємо HTML, CSS та JavaScript так як ці мови є найпопулярніші у сфері Front End та легкі для вивчення.

### 1.1.3 Огляд баз даних

База даних (БД) – це електронна система управління інформацією, яка повинна постійно і ефективно обробляти великі обсяги даних, без протиріч і помилок і може цифровим чином відображати логічні відносини**[24]**. Бази даних призначені для зберігання, змін, обробки інформації. Бази даних використовують для проектів, де динамічно змінюється наповнення самого сайту. Головною перевагою баз даних є швидкість використання та додавання потрібної інформації. Завдяки спеціальним оптимізованим алгоритмам, які використовуються в базах даних, можна запросто знайти всі потрібні дані всього за декілька секунд. Для формування запиту в базу даних, використовують мову програмування SQL, яку ми розглянули вище. Бази даних поділяють на такі типи: ієрархічні, мережеві, SQL, NoSQL.

Найпопулярніші і найнадійніші серед них це є SQL і NoSQL бази даних.

Нереляційні(NoSQL) бази даних – це бази даних побудовані на основі нереляційної моделі, тобто не є структуровані в формі таблиць. Такий вид баз даних використовують з величезними наборами даних. NoSQL бази даних не найкращий вибір для важких запитів. Вставлення рядків в такий тип баз даних потребує менше часу порівняно з SQL, але більше часу для читання з бази. Популярні NoSQL бази даних: MongoDB, Azure Cosmos DB, Cassandra, і тд.

Реляційні(SQL) бази даних – це бази даних побудовані на основі реляційної моделі, тобто має табличний спосіб зображення даних. Кожний об’єкт записується рядком у таблиці. Рядок називається записом. Кожен стовпець у таблиці має ім'я поля і його тип. Реляційні бази даних в основу використовуються в тих випадках, коли ми маємо взаємозв'язок між різними сутностями. Популярні реляційні бази даних: MySQL, PostgreSQL, SQLite, Microsoft Server SQL.

SQL містить оператори різного типу, серед них:

- DDL (мова визначення даних): Це Запити, які використовуються для створення чи модифікації схеми. Загальні команди: CREATE, ALTER і DROP.
- DML (мова обробки даних): Цей запит використовується для виконання операцій вибору, вставки, оновлення та видалення в базі даних. Загальні команди: SELECT, INSERT, UPDATE та DELETE.
- DCL (мова контролю даних): Такі запити використовуються для контролю доступу та надання дозволу на базу даних. Загальні команди: GRANT та REVOKE.
- TCL (Мова контролю транзакцій): Ці запити використовуються для контролю та управління транзакціями для підтримки цілісності даних. Загальні команди: BEGIN, COMMIT і ROLLBACK**[25-27]**.

Для інтернет-магазину нам цілком достатньо використовувати реляційні бази даних.

Розглянемо плюси і мінуси кожної з реляційних баз даних:


Рисунок 1.7 – Логотип MySQL

MySQL – реляційна база даних розроблена в 1994 році.

Плюси:

- Може використовуватись на різних платформах, таких як Linux, Windows, MacOS, і тд. Це є хороший вибір для проектів, які будуть працювати на декількох платформах
- Має підтримку від великої спільноти
- Може використовуватися для роботи з великими масивами
- Безпека даних – MySQL визнали найбезпечнішою базою даних. Дані захищаються паролем, і паролі зберігаються в зашифрованому вигляді

Мінуси:

- MySQL не дуже ефективно справляється з великим розміром бази даних
- Виникають проблеми із стабільністю
- Функціональність сильно залежить від доповнень
- Не дуже ефективно обробляє транзакції, і може пошкодити дані**[28]**.


Рисунок 1.8 – Логотип PostgreSQL

PostgreSQL – є однією з найстаріших, але одночасно і найдосконаліших систем управління базами даних з відкритим кодом. Його можна запускати на багатьох різних платформах.

Завдяки його стабільності він не потребує високого обслуговування. Дана база даних з легкістю працює з багатьма різними фреймворками. Підтримка синхронної та асинхронної реплікації полегшує розподіл збережених даних між декількома серверами для мінімального часу доступу до критичних даних та високої стійкості.

Плюси:

- Транзакції
- Мультипроцессинг, що дає йому більшу швидкість, порівняно з MySQL
- Легко розширюваний – якщо нам потрібна додаткова функція, ми можемо запросто її додати самостійно. Це дозволяє нам придумати функції, тригери, типи даних і з легкістю встановити їх у базі даних.
- Високий рівень безпеки
- Підтримує багато мов програмування(Python, C, Java, C++, та інші)

Мінуси:

- Повільна швидкість читання
- Розширена документація доступна тільки на англійській
- Не доступно на всіх хостах за замовчуванням**[29]**.


Рисунок 1.9 – Логотип SQLite

SQLite – популярна реляційних база даних яка не використовує окремого серверного процесу, всі дані пишуться і читаються з файлу на диску.

Плюси:

- Портативний – SQLite може працювати майже на будь якій платформі
- Не потребує окремого серверного процесора
- Встановлення не потрібно – просто потрібно завантажити бібліотеки SQLite на пристрій
- Одразу надходить з нульовою конфігурацією, немає потреби в настройці

- База даних зберігається в одному файлі
- Підтримує транзакції
- Доступні додаткові розширення

Мінуси:

- Розмір бази даних є обмеженим 2GB
- Не підтримує багато різних функцій
- SQLite не підтримує багато різних типів: Date, Time, DateTime**[30,31]**.


Рисунок 1.10 – Логотип Microsoft SQL Server

Microsoft SQL Server – система управління реляційними базами даних розроблена компанією Microsoft. Для запитів використовує мову Transact-SQL

Плюси:

- Простота налаштування – на відміну від іншої системи установка та налаштування є одними з найпростішими.
- Підвищення безпеки даних – забезпечує безпеку бази даних
- Оптимізоване сховище даних – нам не буде потрібно жодного іншого сховища даних для тієї ж бази даних під час використання різних пристроїв

- Підтримка відновлення даних: у разі перерви живлення або зупинки сервера дані можуть бути пошкоджені, тому Microsoft SQL Server усуває ризик втрати даних, маючи функції відновлення та відновлення даних.

Мінуси:

- Якщо потрібні додаткові функції та програми за це потрібно буде платити
- Нові версії потребують покращених технологій, і в такому випадку може знадобитись новий комп'ютер
- Microsoft SQL Server може не працювати на вашій платформі**[32]**.

Вирішив використовувати для інтернет-магазину PostgreSQL, оскільки він має високий рівень безпеки, має порівняно більшу швидкість ніж MySQL, працює на всіх платформах порівняно з Microsoft SQL Server, та підтримує більше типів даних ніж SQLite.

### 1.1.4 Контейнеризація веб застосунку

Контейнерізація — це упаковка програмного коду лише з бібліотеками операційної системи (ОС) і програмними залежностями, необхідними для запуску коду для створення єдиного легкого виконуваного файлу, який називається контейнером, який працює послідовно в будь-якій інфраструктурі. Більш портативні та ресурсоефективні, ніж віртуальні машини (ВМ), контейнери стали обчислювальними одиницями сучасних хмарних програм.

За допомогою контейнеризації розробники можуть створювати та розгортати програми значно простіше, безпечніше та швидше. При використанні традиційних методів програма розробляється в спеціальних середовищах і при переносі на на новий пристрій часто трапляються помилки і баги в програмі. Наприклад, коли розробник переносить код із настільного комп’ютера або на віртуальну машину, або коли переносить із операційної системи Linux на Windows. Завдяки контейнеризації цю проблему можна передбачити та подолати, оскільки цей метод об’єднує код програми разом із відповідними конфігураційними файлами, бібліотеками та залежностями, необхідними для роботи програми. Цей пакет програмного забезпечення, або іншими словами “контейнер” залежить від головної операційної системи, і, отже, він стоїть окремо та стає портативним — може працювати на будь-якій платформі чи хмарі без проблем.

Концепції контейнеризації та ізоляції процесів насправді існує десятиліття, але поява в 2013 році Docker Engine з відкритим кодом — галузевого стандарту для контейнерів із простими інструментами розробника та універсальним підходом до пакування — прискорила впровадження цієї технології. Сьогодні організації все частіше використовують контейнеризацію для створення нових програм і модернізації існуючих програм для хмари. В опитуванні IBM 61% користувачів контейнерів повідомили, що використовували контейнери в 50% або більше нових програм, які вони створили протягом попередніх двох років; 64% користувачів очікують, що 50% або більше їхніх існуючих додатків будуть поміщені в контейнери протягом наступних двох років**[34]**.

Отже, що ж таке взагалі контейнер? Контейнер — це портативне обчислювальне середовище. Він містить усе, що потрібно програмі для роботи, від двійкових файлів до залежностей і конфігураційних файлів.

Контейнери працюють на абстрактному рівні над основною операційною системою. Як і віртуальні машини (VM), вони ізольовані та мають ретельно обмежений доступ до системних ресурсів. Контейнери не споживають ресурси віртуального обладнання, віртуального ядра або

віртуальної операційної системи для запуску програм. Таким чином, контейнеризація є набагато меншим і ефективнішим методом віртуалізації.
Контейнери є ізольованими та самодостатніми, і хост може запускати один або декілька контейнерів одночасно, більше того кількість контейнерів на хості обмежена лише наявністю обчислювальних ресурсів.
Контейнерізація дозволяє розробникам створювати та розгортати програми швидше та безпечніше, незалежно від того, чи є вона традиційною монолітною (однорівневою програмною програмою) чи модульною програмою, побудованою на архітектурі мікросервісів . Нові хмарні програми можна створювати з нуля як контейнерні мікросервіси, розбиваючи складну програму на низку менших спеціалізованих і керованих служб. Існуючі програми можна упакувати в контейнери (або контейнерні мікросервіси), які ефективніше використовують обчислювальні ресурси**[33]**.
Плюси контейнеризації:

- Портативність – контейнер створює виконуваний пакет програмного забезпечення, який не є прив'язаним до чи залежним від основної операційної системи і здатний працювати на будь якій оперативній системі.
- Ефективність – контейнерізація є одним із найефективніших методів віртуалізації, доступних розробникам. Контейнери підвищують ефективність двома способами: вони використовують усі доступні ресурси та мінімізують накладні витрати. Якщо зробити правильну конфігурацію то контейнери дозволяють хосту використовувати всі доступні ресурси. Також ізольовані контейнери можуть виконувати свої операції не заважаючи іншим контейнерам.
- Швидкість – можете швидко створювати контейнери, розгортати їх у будь-якому середовищі, де їх можна використовувати для вирішення багатьох різноманітних завдань. Коли постає завдання, ви можете швидко розробити контейнер для виконання цього завдання. Якщо він більше не потрібен, ви можете автоматично вимкнути його, поки він знову не знадобиться.
- Покращена безпека – ізоляція, яку створює контейнеризація, також забезпечує додатковий рівень безпеки. Так як контейнери ізольовані один від одного, то ви можете бути впевнені, що ваші програми працюють у власному автономному середовищі. Це означає що якщо навіть щось з одним контейнером стається і його безпека порушена, то інші контейнери на цьому ж хості залишаються в безпеці
- Порівняно швидший запуск програми – якщо порівнювати з іншими методами віртуалізації контейнери є досить легкі, а через те що вони легкі то і їх запуск набагато швидший
- Простота керування – якщо взяти наприклад платформу Kubernetes, то вона пропонує різноманітні інструменти, які спрощують керування контейнерами, як-от відкат і оновлення. Він також займається установкою. Існують функції самовідновлення, які можна використовувати, щоб спробувати відновити несправні контейнери, припинити роботу контейнерів, які не пройшли перевірку працездатності, і постійно стежити за працездатністю та статусом своїх контейнерів.
- Гнучкість – контейнерізація дає розробникам універсальність для роботи зі своїм кодом як у віртуалізованому середовищі, так і в початковому середовищі компютера. Якими б не були вимоги розгортання, контейнеризація може піднятися, щоб їх задовольнити. Якщо раптом виникне потреба переобладнати ваше середовище з металевого на віртуальне або навпаки, ваші контейнерні програми вже готові до переходу**[35]**.

Розглянемо деякі популярні платформи для контейнеризації веб-застосунків:

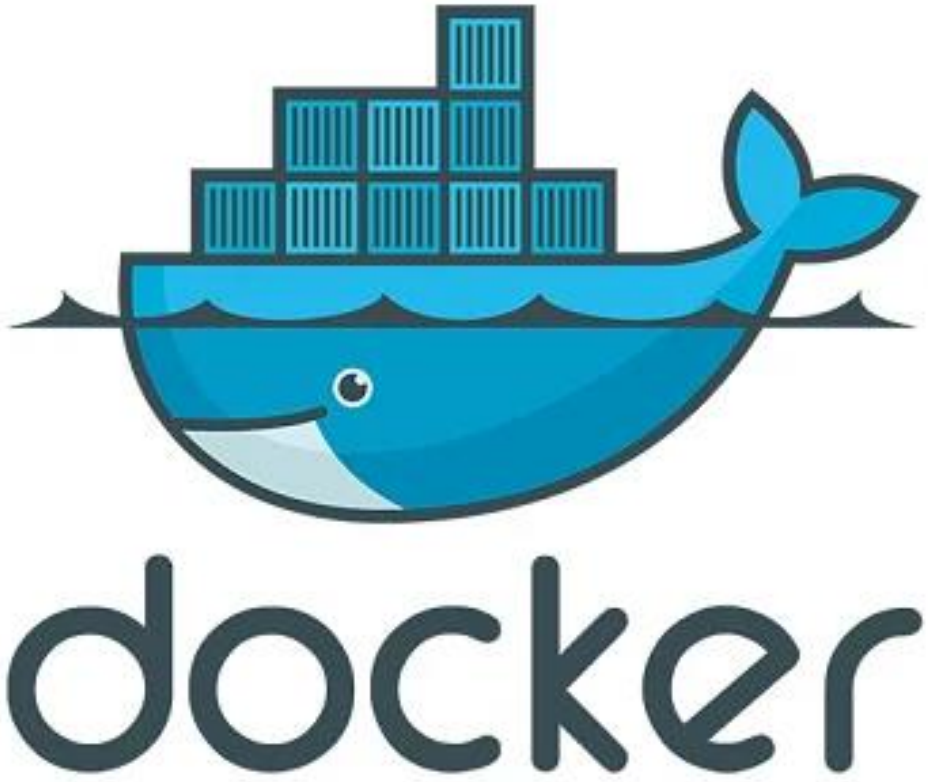



Рисунок 1.11 – Логотип Docker

Docker — це платформа контейнеризації з відкритим кодом, яка допомагає створювати, розгортати та керувати контейнерами. За допомогою Docker розробники можуть пакувати та запускати додатки разом зі своїми залежностями в слабко ізольованих середовищах, відомих як контейнери. Docker спрощує доставку програм, ізолюючи їх від інфраструктури. Методологія доставки, тестування та швидкого розгортання коду Docker допомагає зменшити затримки між написанням коду та його запуском у виробництві. Docker забезпечує високий ступінь переносимості, щоб користувачі могли реєструвати та ділитися контейнерами на різних хостах у приватному та публічному середовищах**[36]**.

Переваги використання Docker:

- Послідовне та ізольоване середовище – контейнери Docker допомагають розробникам створювати ізольовані та передбачувані середовища, що забезпечує послідовне та ефективне масштабування.
- Економічно ефективним – Docker допомагає скоротити час розгортання зображень до секунд, що може бути значною перевагою продуктивності. Усі контейнери Docker базуються на зображенні. Такі процеси, як ініціалізація, яка включає розподіл серверів і ресурсів і налаштування інфраструктури, традиційно займають багато часу. Проте, при поміщені кожного процесу у контейнер, ви можете ділитися процесами з новими програмами. Таким чином, процес розгортання прискорюється.
- Портативність – програма завжди працюватиме без всяких ускладнень.
- Масштабованість – можна створювати контейнери відповідно до потреб програми за допомогою Docker. Легкий і портативний характер Docker дозволяє масштабувати або демонтувати програми відповідно до вимог і потреб. Завдяки цьому можна легко керувати робочим навантаженням. Існує також широкий спектр можливостей керування контейнерами при використанні кількох контейнерів**[37]**.


Рисунок 1.12 – Логотип Kubernetes

Kubernetes (також відомий як k8s або «kube») — це платформа оркестровки контейнерів з відкритим кодом , яка автоматизує багато ручних процесів, пов'язаних із розгортанням, керуванням і масштабуванням контейнерних програм. K8s як абревіатура є результатом підрахунку восьми літер між «K» і «s». Google відкрила проект Kubernetes у 2014 році. Kubernetes поєднує в собі понад 15 років досвіду Google у створенні робочих навантажень у великих масштабах із найкращими у своєму класі ідеями та практиками спільноти.

Kubernetes дотримується архітектури клієнт-сервер із панеллю керування, яка керує загальним станом системи, і набором вузлів, які запускають контейнери. Площина керування включає кілька компонентів, таких як сервер API тощо, планувальник і менеджер контролера. Ці компоненти працюють разом, щоб керувати життєвим циклом програм, що працюють на Kubernetes**[38]**.

Kubernetes надає:

- Виявлення служб і балансування навантаження
- Оркестровка сховищ
- Автоматичне розгортання та відкат

- Автоматичне пакування в контейнер
- Kubernetes із самовідновленням
- Управління секретом і конфігурацією

Переваги використання Kubernetes:

- Автоматизовані операції – Kubernetes постачається з потужним API та інструментом командного рядка під назвою kubectl, який виконує основну частину важкої роботи, пов'язаної з керуванням контейнером, дозволяючи автоматизувати свої операції. Шаблон контролера в Kubernetes гарантує, що програми/контейнери працюють точно так, як зазначено.
- Абстракція інфраструктури – Kubernetes керує доступними ресурсами від запущеного імені. Це дозволяє розробникам зосереджуватися на написанні програмного коду, а не на базовій обчислювальній, мережевій інфраструктурі чи інфраструктурі зберігання.
- Моніторинг працездатності служби – Kubernetes відстежує робоче середовище та порівнює його з бажаним станом. Він виконує автоматичні перевірки працездатності служб і перезапускає контейнери, які вийшли з ладу або зупинилися. Kubernetes робить служби доступними лише тоді, коли вони запущені та готові.
- Масштабованість – Kubernetes дозволяє масштабувати додатки по горизонталі та вертикалі залежно від використання ресурсів і попиту. Іншими словами, еластичність є основною особливістю кластерів Kubernetes.
- Наявність – Kubernetes має високу доступність, допомагаючи захистити програму від будь-якої єдиної точки збою. За допомогою Kubernetes можна створити кілька вузлів площини керування, а це означає, що якщо будь-який із головних вузлів вийде з ладу, інші підтримуватимуть роботу кластера.
- Можливість кількох хмар – Kubernetes має кілька хмарних можливостей. Завдяки своїй портативності він може розміщувати робочі навантаження в одній хмарі, а робочі навантаження розподіляються між кількома хмарами. Крім того, він може масштабувати своє середовище від однієї хмари до іншої.
- Гнучкість – Kubernetes дуже гнучкий, тобто він може працювати практично з будь-яким середовищем виконання контейнера. Середовище виконання контейнера — це програмний компонент, який допомагає запускати контейнер у головній операційній системі. Крім того, він може працювати майже з будь-яким типом базової інфраструктури, будь то публічна хмара, приватна хмара або локальний сервер**[39]**.

Основна відмінність Docker та Kubernetes полягає в тому, що Docker — це середовище виконання контейнерів, Kubernetes — це платформа для запуску та керування контейнерами з багатьох середовищ виконання контейнерів. Kubernetes підтримує численні середовища виконання контейнерів, включаючи Docker, container, CRI-O та будь-яку реалізацію

Kubernetes CRI (Container Runtime Interface). Гарною метафорою є Kubernetes як «операційна система», а контейнери Docker — це «додатки», які ви встановлюєте в «операційну систему». Сам по собі Docker дуже корисний для розробки сучасних програм. Це вирішує класичну проблему «працює на моїй машині», але ніде більше. Інструмент оркестровки контейнерів Docker Swarm здатний обробляти робоче навантаження виробничого контейнера, розгортаючи кілька контейнерів. Коли система розростається й потребує додавання багатьох контейнерів, підключених один до одного, автономний Docker може зіткнутися з проблемами зростання, які Kubernetes допомагає вирішити**[40]**.

# 1.2 Постановка задачі

Метою роботи є розробка та реалізація веб-застосунку, а саме інтернет-магазину. Інтернет-магазин повинен бути реалізований як веб-сайт. Для досягнення результату необхідно було виконати такі завдання:

1. Дослідження мов програмування
2. Дослідження фреймворків вибраної мови
3. Дослідження баз даних
4. Планування функціоналу
5. Проектування бази даних
6. Налаштування фреймворку
7. Розробка backend частини
8. Розробка frontend частини
9. Тестування
10. Обгортання проекту в контейнер для простого розгортання на сервері

Даний додаток повинен мати наступний функціонал:

- ➔ Пошук товару за схожою назвою
- ➔ Пагінація
- ➔ Фільтрування та сортування товарів за ціною, наявністю,
- ➔ Кошик
- ➔ Історія покупок
- ➔ Авторизація
- ➔ Особистий профіль
- ➔ Адміністративна панель для додавання товарів та знижок
- ➔ Онлайн чат для зв'язку із підтримкою, в разі проблем
- ➔ Можливість розсилки при не наявності продукту

Сайт буде складатись з двох частин:

Адміністративний інтерфейс – для створення, редагування, видалення непотрібних товарів, знижок

Користувацький інтерфейс – повинен бути зрозумілим, легким для користування відвідувачів сайту

# РОЗДІЛ 2. ОГЛЯД РОЗРОБКИ НА DJANGO

## 2.1 Переваги та недоліки розробки на Django

Хоч і ринок переповнений фреймворками, але Django – хороший варіант для того щоб почати розбиратись у веб-розробці, оскільки він має найточнішу документацію, із всіма можливими аспектами, написаний на Python, що робить його простим у розумінні, і швидкості написання коду.

Перша версія Django була випущена в 2005 році, і з часом він увійшов до рейтингу найкращих фреймворків. За допомогою Django тисячам веб-розробникам виконувати роботу стало в декілька разів простіше і швидше. Цей фреймворк використовують архітектуру MVT, який по факту є модифікацією MVC.

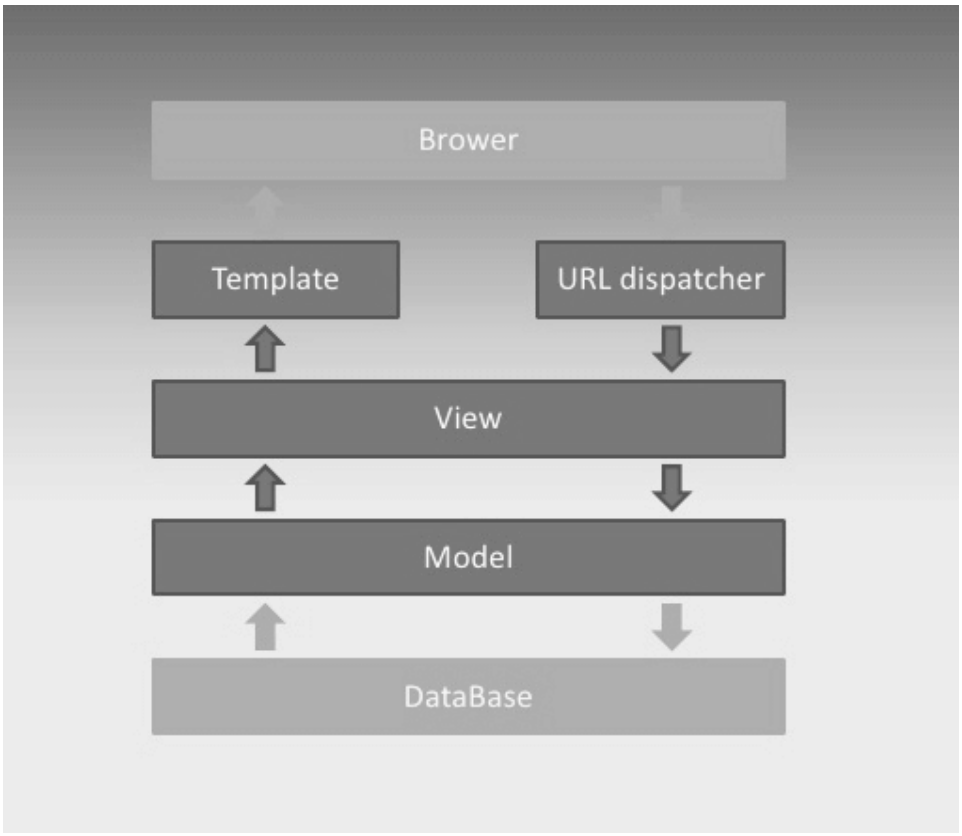



Рисунок 2.1 – Архітектура MVT в Django

Основні елементи патерна:

- URL dispatcher – так званий маршрутизатор. При надходженні запиту, на основі URL визначає який ресурс повинен опрацювати даний запит
- View – відпрацьовує запит і надсилає відповідь користувачеві. Якщо при опрацювання запиту потрібне звернення до моделей та бази даних, то View взаємодіє з ними. При надсиланні відповіді можуть використовуватись як шаблони, так і звичайні JsonResponse. В архітектурі MVC цьому відповідає Controller.

- Model – описує самі об'єкти, які будуть використовуватись в самому додатку та базі даних.
- Template – представляє логіки view у вигляді згенерованого HTML документа. В архітектурі MVC цьому відповідає View**[18].**

Переваги Django:

- Швидкість – одна з головних переваг Django це те що він забезпечує швидку розробку. Django був розроблений, щоб допомогти розробникам створити додаток настільки швидко, наскільки це можливо. При правильній конфігурації веб-розробка використовуючи Django забезпечує оптимізацію веб-додатків. Конфігурація та архітектурний дизайн фреймворка полягають у тому, що він дозволяє використовувати кілька компонентів одночасно, що забезпечує швидкий розвиток. Розробники можуть працювати паралельно без втрати швидкості розробки. Його можна назвати ідеальним рішенням для розробників, для яких питання дедлайну є в пріоритеті.
- Масштабований – Django постійно розвивається, що дає змогу розробникам створювати кращі та сучасні веб-додатки. Розробники на Django можуть з легкістю змінювати та розширювати фукціонал, впроваджуючи покращення в його окремі компоненти.
- Великий набір пакетів та доповнень – Django працює з багатьма додатковими функціями, які помітно допомагають з адмініструванням вмісту, аутентифікацією користувача, та багато чим іншим.
- Добре підходить для пошукової оптимізації(SEO) – Django пропонує багато різних корисних інструментів SEO. У Django є додатковий модуль під назвою Django SEO Framework, за допомогою якого розробники можуть зменшити час завантаження сторінки, використовуючи кешовані сторінки(шаблони), стисненні CSS та JavaScript
- Надійні функції безпеки – Django забезпечує високий рівень безпеки і містить механізми запобігання загальним атакам, як-от SQL Injection (XSS) і підробка міжсайтових запитів (CSRF). Для ефективної роботи з іменами користувачів і паролями є система аутентифікації користувачів.
- Гнучкий – оскільки він написаний на Python, універсальній мові програмування, Django надає розробникам більше гнучкості і динамізму, в порівнянні з іншими мовами. Вони можуть налаштувати додаток відповідно до своїх потреб. Django забезпечує чудову підтримку пакетів і зовнішніх бібліотек. У Django більше уваги приділяється явному програмуванню, а не неявному, що робить його однією з ідеальних фреймворків для програм, які потребують швидких змін.

Недоліки Django

- Покладається на систему ORM – система ORM, надана Django дозволяє простіше розробникам працювати з декількома базами даних і виконувати загальні операції з базою даних. Система ORM, яку використовує веб-фреймворк не має всіх функцій, наданих іншими більш використовуваним ORM системами
- Занадто монолітний – для деяких розробників це є особливість, але для деяких недолік. Він має деяку кількість залежностей, що можуть ускладнювати його використання.
- Не найкращий вибір для малих проектів – велика функціональність Django надходить з великою кількістю коду. Це витрачає більше часу на обробку сервера та пропускну здатність під час розробки, що може створювати деякі проблеми для недорогих та малих проектів.
- Неможливо обробляти декілька запитів одночасно – на відмінну від інших фреймворків, які можуть обробляти кілька запитів одночасно, то Django не може виконувати такі дії. Це запити для окремих процесів, і для обробки кожного з ним потрібен час
- Необхідно володіти всією системою для якісної розробки – Django має надзвичайно великий функціонал і конфігурації, які не завжди можуть бути зрозумілими розробникам**[20-22]**.

Для проектів з дедлайном найкращим вибором буде Django. Більш цього завжди є можливість встановити свою конфігурацію, що прискорить робочий процес. Якщо за допомогою інших фреймворків приходилось тратити чимало часу для написання коду, то цей фреймворк надав можливість спростити цей процес.

## 2.2 Розробка на Django

Перед тим, як встановлювати django потрібно налаштувати віртуальне середовище(virtualenv), це дозволить вільно видаляти або встановлювати додаткові пакети, при цьому не змінюючи залежності інших проектів. Щоб створити віртуальне середовище виконаємо команду:

```
python3 -m venv path_to_venv
```

Опісля створення віртуального середовища активуємо його:

Для Windows – `path_to_venv\Scripts\activate`

Для MacOS і Linux – `source path_to_venv/bin/activate`

Наступним кроком нам потрібно буде встановити Django:

```
python -m pip install Django
```

Це встановить останню доступну версію Django сумісну з версією вашого Python інтерпретатора, для встановлення специфічної версії Django можна скористатись командою:

```
python -m pip install Django==x.x.x
```

Де x.x.x версія яка вам потрібна. У таблиці 2.1 наведено сумісність версій Django та Python.

Таблиця 2.1 – Сумісність версій Django

| Версія Django | Версії Python |
|---|---|
| 2.2 | 3.5, 3.6, 3.7, 3.8 (добавлена в 2.2.8), 3.9 (добавлена в 2.2.17) |

| | |
|---|---|
| 3.0 | 3.6, 3.7, 3.8, 3.9 (добавлена в 3.0.11) |
| 3.1 | 3.6, 3.7, 3.8, 3.9 (добавлена в 3.1.3) |
| 3.2 | 3.6, 3.7, 3.8, 3.9, 3.10 (added in 3.2.9) |
| 4.0 | 3.8, 3.9, 3.10 |

Для того щоб переконатись що все правильно встановлено скористаємось командним рядком:

```
(venv) admin@MacBook-Pro-Arsen ~ % python
Python 3.9.10 (v3.9.10:f2f3f53782, Jan 13 2022, 17:02:14)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> django.get_version()
'3.2.7'
```

Остання команда поверне встановлену версію Django.

Наступним кроком потрібно буде створити проект. За допомогою командного рядка це можна зробити так:

```
django-admin startproject example
```

Після успішного створення проекту, ви будете бачити наступну структуру проекту:


manage.py – скрипт, який допомагає управляти сайтом. За його допомогою ми можемо виконувати всі дії пов'язані з нашим проектом.

example/__init__.py – даний файл вказує, що директорія являється пакетом

example/asgi.py – точка входу для обслуговування проекту за допомогою ASGI веб-серверів

example/settings.py – конфігурація і налаштування проекту

example/urls.py – url dispatcher, відповідає за те куди нам потрібно надіслати запит на обробку

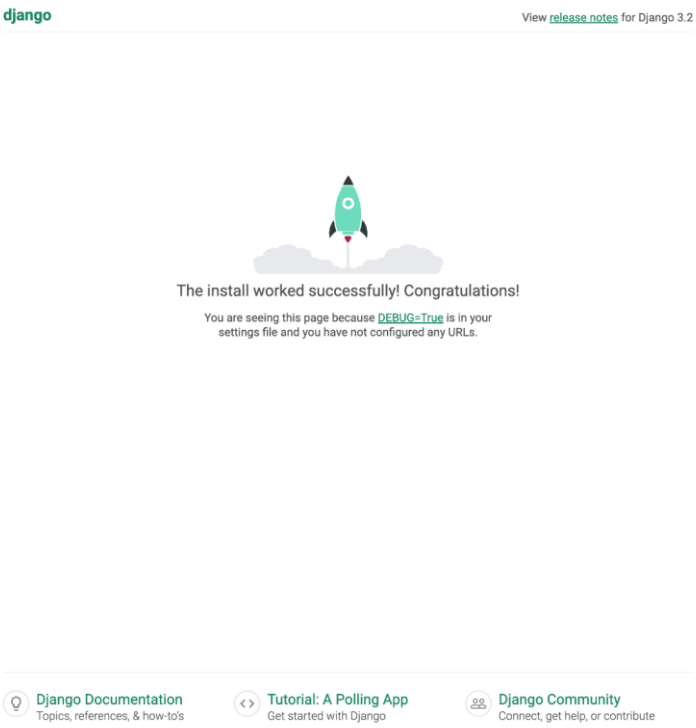
example/wsgi.py – точка входу для обслуговування проекту за допомогою WSGI веб-серверів

Для запуску додатка потрібно написати в командному рядку наступне:

```
python manage.py runserver
```

В результаті в відповідь ми отримаємо інформацію про запуск нашого проекту. Щоб перевірити чи все гаразд перейдемо за посиланням отриманим від запуску команди “Starting development server at *”, де * це адреса де працює наш сервер, зазвичай це http://127.0.0.1:8000/

Якщо все зроблено правильно то в браузері ми побачимо таку сторінку:




Як тільки все успішно працює, створимо першу сторінку, для цього створимо файл в папці example під назвою views.py, і додаємо наступний код:

```
from django.http import HttpResponse, HttpRequest


def home(request: HttpRequest) -> HttpResponse:
    return HttpResponse("Hello in our site!")

```

Для того щоб прив'язати наш вид(контролер) до маршрутизації, ми повинні вказати йому інтернет адресу, по якій ми будемо до цього контролера звертатись, для цього додамо до urlpatterns в файлі example/urls.py новий шлях, в результаті отримуємо:

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


urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home, name='home')
]
```

І коли ми запустимо сайт і перейдемо до головної сторінки, побачимо напис “Hello in our site!”

Наступним нашим кроком створимо додаток:
`python manage.py startapp animal`

Папка створеного додатку виглядає ось так:

```
.
├── __init__.py
├── admin.py
├── apps.py
├── migrations
│   └── __init__.py
├── models.py
├── tests.py
└── views.py
```

Розглянемо структуру додатку:

admin.py – Задає параметри для адміністрації додатка
apps.py – Відбувається реєстрація додатку
migrations – Папка в якій будуть зберігатись всі міграційні файли до бази даних, які дозволять нам автоматично оновити базу даних, при оновленні моделей

models.py – Тут ми описуваимемо наші обєкти
tests.py – У цьому файлі прийнято зберігати тести

views.py – Так звані контроллери

При необхідності, ми можемо створювати додаткові модулі, і навіть модифікувати їх до вигляду пакетів, Django в цьому не обмежує розробників.

Після створення додатку, це ще не означає, що він використовується на сайті. Щоб ми змогли використовувати додаток нам потрібно зареєструвати його в нашій системі, для цього перейдемо в файл конфігурацій, а саме `example/settings.py`, і додаємо назву нашого додатку до переліку в `INSTALLED_APPS`. В результаті ми отримуємо наступне:

```python
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'animals'
]
```

Створимо модель тварина:

```python
from django.db import models


class Animal(models.Model):
    id = models.BigAutoField(primary_key=True)

    birthday = models.DateTimeField()
    name = models.CharField(max_length=256, blank=True)
    animal_class = models.CharField(max_length=64, blank=True)
    animal_type = models.CharField(max_length=64, blank=True)

    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

models.Model – ми вказуємо, що class Animal є Django моделлю

Тепер пройдемось по типам полів:

models.BigAutoField – за замовчуванням Django автоматично генерує кожній моделі первинний ключ, але його можна вказати явно для моделі

models.DateTimeField – поле з датою та часом, використали два параметра, auto_now_add – при створенні нового об'єкта Animal цьому полю буде встановлено значення текучої дати та часу, та auto_now – при кожному оновленні об'єкта поле буде перезаписане текучою датою та часом.

models.CharField – використовуємо для текстових полів, з обмеженою кількістю символів, для цього використовуємо властивість max_length, якщо потрібно необмежений текст, то використовуємо замість цього models.TextField.

Згенеруємо міграцію для бази даних за допомогою команди:

`python manage.py makemigrations`

Отримуємо наступне:

```
(.venv) admin@MacBook-Pro-Arsen example % python manage.py makemigrations
Migrations for 'animals':
  animals/migrations/0001_initial.py
    - Create model Animal
```

Перш ніж прийняти наші міграції, давайте налаштуємо саму базу даних, для прикладу використаємо sqlite3, при необхідності можна використати будь-яку іншу базу даних, Django не обмежує нас в цьому. Налаштування бази даних можна змінити в файлі конфігурації проекту

```python
# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Тепер накотимо всі наші міграції:

```
python manage.py migrate
```

При успішному виконанні команди побачимо наступне, також бачимо що деякі міграції згенеровані автоматично, це міграції додатків, які були підключені до проекту:

```
Applying contenttypes.0001_initial... OK
Applying auth.0001_initial... OK
Applying admin.0001_initial... OK
Applying admin.0002_logentry_remove_auto_add... OK
Applying admin.0003_logentry_add_action_flag_choices... OK
Applying animals.0001_initial... OK
Applying contenttypes.0002_remove_content_type_name... OK
Applying auth.0002_alter_permission_name_max_length... OK
Applying auth.0003_alter_user_email_max_length... OK
Applying auth.0004_alter_user_username_opts... OK
```

# РОЗДІЛ 3. РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ

## 3.1 Розробка моделей бази

Розробив діаграму бази даних, і після цього створив відповідні Django моделі, згенерував міграції за допомогою команди:

```
python manage.py makemigrations
```

Після цього накотив міграції за допомогою команди:

```
python manage.py migrate
```

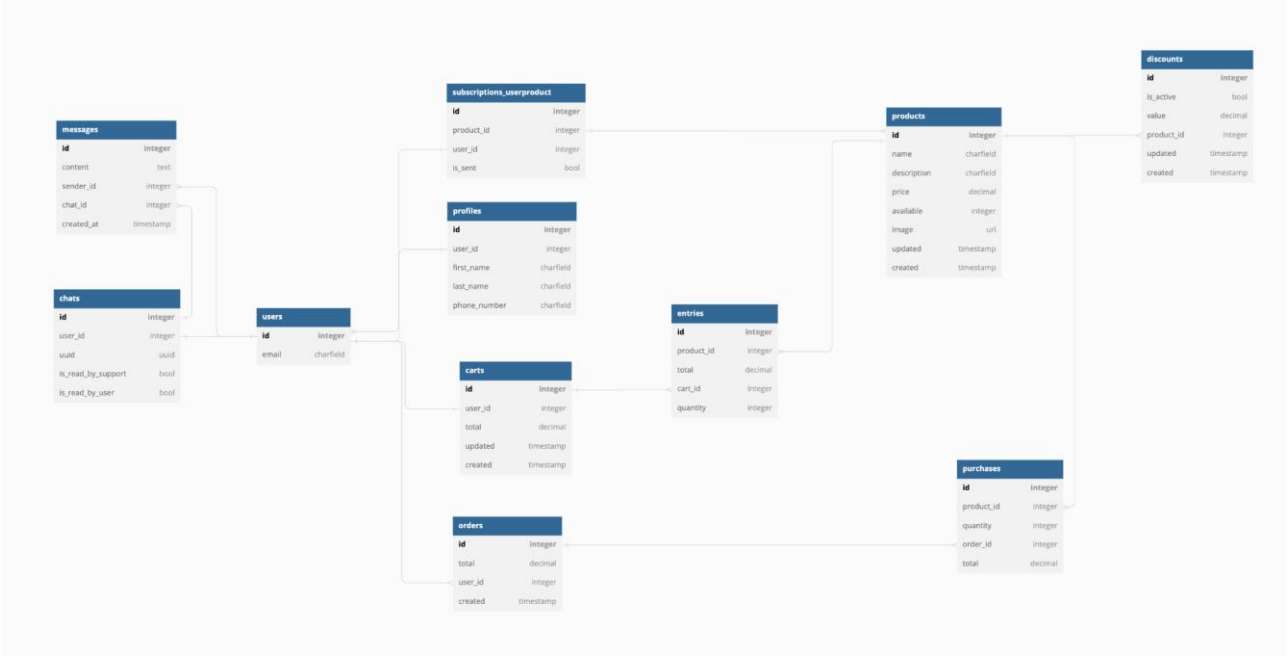



Рисунок 3.1 – Діаграма бази даних

## 3.2 Опис функціоналу

Створив сторінки для логування та реєстрації, для цього використав стандартну Django аутентифікації, також перезавантажив форму для реєстрації щоб додати додаткові поля.

Також створив сторінки для додавання продукту та знижок, це доступно лише для адміністратора. Для кожного продукту є можливість редагування, видалення. Для знижок передбачена можливість вимикати їх.

Додав можливість зробити підписку на певний продукт, якщо його немає в наявності. Раз в годину за допомогою celery відбувається розсилка email для того, щоб повідомити користувачу чи з'явився продукт в наявності, при тому перевіряється чи потрібно робити розсилку, чи очікує її користувач і чи була вона відправлена.

На головній сторінці є можливість додавання в кошик товару, фільтрація по наявності ціні та імені.

Реалізована історія покупок, список очікування та редагування профілю.

Також є чат який працює через веб-сокети, для онлайн листування з підтримкою.

Весь проект додатково огорнуто в docker контейнери.

## 3.3 Як запустити проект

Для запуску проекту потрібно виконати лишень дві послідовні команди, при цьому має бути доступний порт 5432 для того, щоб підняти базу даних. Щоб запустити проект потрібно виконати наступні команди, при цьому має бути встановлений Docker:

```
docker-compose build
docker-compose up
```

# ВИСНОВКИ

Було розглянуто доступні найпопулярніші мови програмування, для створення веб сайтів, і було обрано Python, HTML, CSS, JavaScript як основні мови програмування, так як вони прості у вивченні, і досить популярні. Також було досліджено особливості деяких фреймворків на мові програмування Python, та обрано Django, тому що за допомогою нього можна з легкістю розробити наш Інтернет-магазин, добавити в нього все необхідне. Як базу даних було обрано PostgreSQL. Для контейнеризації обрав Docker, так як він потребує менше налаштувань, ніж Kubernetes. Він пропонує ті самі переваги, що й Kubernetes, як-от розгортання програми за допомогою декларативних файлів YAML, автоматичне масштабування служб до бажаного стану, балансування навантаження між контейнерами в кластері, а також безпека та контроль доступу до служб.

Було розглянуто роботу фреймворку Django, його структуру, детально пройшлись по перевагам і недолікам цього фреймворку. Створили перший простий проекту за допомогою Django. Також розібрались із створення за допомогою Django: новий об'єктів в нашій базі даних, створення та підключення додатків до нашого проекту, розглянули додавання нових маршрутів.

У процесі розробки використовувалися різноманітні інструменти та технології, що дозволило створити практичну та ефективну веб-програму.

Основні переваги використання Python і Django для створення веб-додатків включають їх швидкість, легке використання синтаксису та наявність попередньо зібраних компонентів, які спрощують роботу розробників і забезпечують швидку інтеграцію необхідної функціональності.

Розроблений веб-додаток включає реєстрацію користувача, каталог товарів, можливість додавання товарів у кошик, виконання замовлень, додавання товару в список спостережень, та інші необхідні для роботи інтернет-магазину. З урахуванням вимог до дизайну та зручності використання було забезпечено приємний досвід користувача.

Робота з базою даних, забезпечення безпеки та оптимізація продуктивності були одними з труднощів, які виникли під час розробки веб-додатку. Однак завдяки знанням і досвіду використання Python і Django вдалося успішно подолати ці труднощі та гарантувати належний рівень функціональності та надійності веб-додатку.

В результаті, за допомогою даних технологій, ми отримали інтернет-магазин, на якому відвідувач сайту може замовляти товари, переглядати історію своїх покупок не виходячи із дому.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Stack Overflow Developer Survey 2021 [Electronic resource] // Stack Overflow. — Available from : https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language.

2. Що таке мови програмування [Електронний ресурс] // Интересные новости из мира науки и техники на IPKey.com.ua. — Режим доступу : http://ipkey.com.ua/uk/faq/925-programming-languages.html.

3. JavaScript - Клієнтською мова програмування, що робить сторінки сайту інтерактивними [Електронний ресурс] // Astwellsoft - A software development company that's easy to work with. — Режим доступу : https://astwellsoft.com/uk/blog/tehnology/javascript.html.

4. 7 популярных языков программирования в 2022 - merehead [Електронний ресурс] // Merehead. — Режим доступу : https://merehead.com/ru/blog/top-7-programming-languages-learn-in-2022/.

5. HTML & CSS - W3C [Електронний ресурс] // Wayback Machine. — Режим доступу : https://web.archive.org/web/20101129081921/http://www.w3.org/standards/webdesign/htmlcss.

6. SQL шпаргалка на співбесіду [Електронний ресурс] // QualityAssuranceGroup. — Режим доступу : https://qagroup.com.ua/publications/sql-info/.

7. Як працює двигун бази даних SQL [Електронний ресурс] // Только самое интересное. Форум программистов. Ит Новости. Ит юмор. Ит статьи. Новости It компаний, отзывы. senior.ua. — Режим доступу : https://senior.ua/articles/yak-pracyu-dvigun-bazi-danih-sql.

8. Basel K. Python pros and cons (2021 update) [Electronic resource] / Krzysztof Basel, Netguru // Custom Software Development | Netguru. — Available from : https://www.netguru.com/blog/python-pros-and-cons.

9. Джава для початківців, чи варто починати навчання на програміста з мови java? - Logos [Електронний ресурс] // Logos IT Academy | Комп`ютерні курси Львів | Курси програмування. — Режим доступу : https://lviv.logos-academy.com/chy-varto-pochynaty-navchannya-na-programista-z-movy-java.

10. 10 Best Python Frameworks to learn for Web Development [Ranked] [Electronic resource] // Hackr.io. — Available from : https://hackr.io/blog/python-frameworks.

11. Python: чому вивчати та з чого почати? | GlobalLogic Ukraine [Електронний ресурс] // GlobalLogic Ukraine. — Режим доступу : https://www.globallogic.com/ua/insights/blogs/python-how-to-start/.

12. ТОП 9 фреймворков для Python-разработчиков [Електронний ресурс] // Курсы программирования в Киеве | IT обучение в Web Academy. — Режим доступу : https://web-academy.com.ua/stati/340-9-python.

13. Мартыненко Я. Чому я обираю FastAPI: основні можливості та переваги фреймворку [Електронний ресурс] / Ярослав Мартыненко // ДОУ. — Режим доступу : https://dou.ua/forums/topic/37547/.

14. FastAPI: All You Need to Know About This Python Framework - HackerTrail [Electronic resource] // HackerTrail. — Available from : https://www.hackertrail.com/talent/backend/fastapi-all-you-need-to-know-about-this-trending-python-web-framework/.

15. You Should Start Using FastAPI Now [Electronic resource] // Towards Data Science. — Available from : https://towardsdatascience.com/you-should-start-using-fastapi-now-7efb280fec02.

16. Що таке фреймворки і для чого вони використовуються при веб-розробці - Блог VOLL [Електронний ресурс] // Веб студія VOLL - Інтернет-маркетинг Агентство. — Режим доступу : https://voll.com.ua/uk/blog/frejmvorki-dlya-veb-rozrobki.

17. Що таке Django? · HonKit [Електронний ресурс] // Choose a language · HonKit. — Режим доступу : https://tutorial.djangogirls.org/uk/django/.

18. Django | Введение [Електронний ресурс] // METANIT.COM - Сайт о программировании. — Режим доступу : https://metanit.com/python/django/1.1.php.

19. Django Advantages and Disadvantages - Why You Should Choose Django? - DataFlair [Electronic resource] // DataFlair. — Available from : https://data-flair.training/blogs/django-advantages-and-disadvantages/.

20. Pros and Cons of Django Web Framework for App Development [Electronic resource] // Business 2 Community. — Available from : https://www.business2community.com/tech-gadgets/pros-and-cons-of-django-web-framework-for-app-development-02330165.

21. Pros and Cons of Django Framework- Does It Match Your Next Project's Requirement? [Electronic resource] // Eduonix Blog. — Available from : https://blog.eduonix.com/software-development/pros-and-cons-of-django-framework/.

22. Getting started with Django | Django [Electronic resource] // The web framework for perfectionists with deadlines | Django. — Available from : https://www.djangoproject.com/start/.

23. Ваш перший Django проект! · HonKit [Електронний ресурс] // Choose a language · HonKit. — Режим доступу : https://tutorial.djangogirls.org/uk/django_start_project/.

24. SQL база даних Для чого призначена база даних? [Електронний ресурс] // ukraine.com.ua. — Режим доступу : https://www.ukraine.com.ua/uk/blog/programming/sql-baza-dannih-dlya-chego-prednaznachena-baza-dannih.html.

25. Реляційна база даних — UA5.org [Електронний ресурс] // UA5.org — Методичні матеріали з інформатики. — Режим доступу : https://ua5.org/database/189-reljacjjna-baza-danikh.html.

26. 11 типів сучасних баз даних: короткий опис, схеми і приклади БД [Електронний ресурс] // Только самое интересное. Форум программистов. Ит Новости. Ит юмор. Ит статьи. Новости It компаний, отзывы. senior.ua. — Режим доступу : https://senior.ua/articles/11-tipv-suchasnih-baz-danih-korotkiy-opis-shemi--prikladi-bd.

27. Точна різниця між SQL та NoSQL (знайте, коли використовувати NoSQL та SQL) - Інший [Електронний ресурс] // Огляди, Ігри, Розваги, Може 2022. — Режим доступу : https://uk.myservername.com/sql-vs-nosql-exact-differences.

28. MySQL Advantages and Disadvantages - techstrikers.com [Electronic resource] // Techstrikers learn technology in simple and fast way - techstrikers.com. — Available from : https://www.techstrikers.com/MySQL/advantages-and-disadvantages-of-mysql.php.

29. Pros and Cons of using PostgreSQL for Application Development [Electronic resource] // Aalpha. — Available from : https://www.aalpha.net/blog/pros-and-cons-of-using-postgresql-for-application-development/.

30. SQLite Advantages and Disadvantages - javatpoint [Electronic resource] // www.javatpoint.com. — Available from : https://www.javatpoint.com/sqlite-advantages-and-disadvantages.

31. Чому слід використовувати SQLite [Електронний ресурс] // peterfeatherstone. — Режим доступу : https://uk.peterfeatherstone.com/962-why-you-should-use-sqlite.

32. Advantages and Disadvantages of Microsoft SQL Server [Electronic resource] // Online Technical Courses | ProgramsBuzz. — Available from : https://www.programsbuzz.com/article/advantages-and-disadvantages-microsoft-sql-server.

33. Containerization Explained | IBM [Electronic resource] // IBM - Deutschland | IBM. — Available from: https://www.ibm.com/topics/containerization.

34. IBM - Deutschland | IBM. [Electronic resource] — Режим доступу: https://www.ibm.com/downloads/cas/VG8KRPRM.

35. Powell R. Benefits of containerization [Electronic resource] / Ron Powell // CircleCI. — Available from: https://circleci.com/blog/benefits-of-containerization/.

36. Bigelow S. J. What is Docker and How Does It Work? [Electronic resource] / Stephen J. Bigelow, Meredith Courtemanche // IT Operations. — Available from: https://www.techtarget.com/searchitoperations/definition/Docker.

37. Kubernetes vs Docker: Understanding the Differences and Advantages - Civo.com [Electronic resource] // Civo.com. — Available from: https://www.civo.com/blog/kubernetes-vs-docker-a-comprehensive-comparison.

38. Overview [Electronic resource] // Kubernetes. — Available from: https://kubernetes.io/docs/concepts/overview/.

39. What is Kubernetes? [Electronic resource] // Red Hat - We make open source technologies for the enterprise. — Available from: https://www.redhat.com/en/topics/containers/what-is-kubernetes.
40. Microservices-Architecture // Atlassian — Available from: https://www.atlassian.com/microservices/microservices-architecture/kubernetes-vs-docker.

# ДОДАТОК

Код проекту: https://github.com/ArsenPidhoretskyi/InternetShop