КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття ступеня бакалавра
за спеціальністю 121 Інженерія програмного забезпечення
на тему:

РОЗРОБКА WEB-СЕРВІСІВ

Виконав студент 4-го курсу
ТЯПКО Владислав Олександрович

(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
КАТЕРИНИЧ Лариса Олександрівна

(підпис)

Засвідчую, що в цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Роботу розглянуто й допущено до захисту
на засіданні кафедри
інтелектуальних програмних систем
«25» травня 2022р.,
протокол № 10
Завідувач кафедри
Олександр ПРОВОТАР

(підпис)

Київ - 2022

Реферат

Обсяг роботи – 30 сторінки, 22 ілюстрацій, 9 джерел посилань.

Тема: Розробка WEB-сервісу

Перелік ключових слів: WEB-сервіс, Python, Django, Flask, MySQL, PostgreSQL,

SQLite.

Метою роботи є розробка WEB-сервісу, який міг би полегшити життя користувачів. Предметом роботи є браузерний застосунок, який представляє собою WEB-сервіс для замовлення доставки їжі. Для досягнення цієї мети поставлено такі завдання:

- Аналіз існуючих підходів до розробки веб-сервісів.
- Аналіз схожих веб-севісів.
- Визначення особливостей створення програми.
- Вибір технологій для реалізації проекту.
- Проектування архітектури.
- Розробка і тестування веб-сервісу.

Результати роботи: під час виконання курсової роботи було розглянуто та досліджено засоби для розробки WEB-служб, опрацьовано інформацію про можливості використаних бібліотек та фреймворків.

Зміст

Скорочення та умовні позначення

Django - веб-фреймворк на основі Python, безкоштовний і з відкритим вихідним кодом, який відповідає архітектурному шаблону model–template–views (MTV).

Flask — веб-фреймворк, написаний на Python. Його класифікують як мікрофреймворк, оскільки він не вимагає особливих чи бібліотек.

Фреймворк(англ. *Framework*) - інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою, але при цьому вона має ряд обмежень, що задають правила створення структури проєкту та написання коду.

Вступ

Оцінка сучасного стану об’єкта дослідження або розробки. Із розвитком високих технологій WEB-сервіси заполонили людське сьогодення. Якщо на початкових етапах існування Інтернету Web-сервіси обмежувалися забезпеченням завдань пошуку документів і даних, то пізніше з їх допомогою стали вирішуватися комплексні завдання автоматизованого адміністрування розподіленими мережами та сайтами, підтримки ділових процесів, електронної
комерції.

Технології Web-сервісів, базуються на використанні основних стандартів: XML, SOAP, WSDL, UDDI. Існує велика кількість фреймворків та бібліотек для розробки Інтернет застосунків(Express, Django,Laravel, Spring).

Актуальність роботи та підстави для її виконання. Одні із найпоширініших WEB-служб є сервіси електронної комерції. Це зазвичай інтернет-магазини техніки, одягу, квитків, продуктів харчування. Такі ресурси дають змогу, маючи під рукою телефон або інший гаджет, з доступом в Інтернет, не витрачати час на поїздки в магазин та стояння в чергах. Мови програмування, на яких пишуть такі застосунки (Java, JavaScript, Python та інші) мають широкий спектр можливостей і великий арсенал інструментів, якими потрібно вміло володіти. Тому саме, спеціаліст, який добре володіє навичками роботи у даному напрямку, високо цінується інвесторами, роботодавцями, чи бізнес-партнерами.

Мета й завдання роботи. Звертаючи увагу на тенденції і розвиток засобів для розробки WEB-сервісів, метою цієї кваліфікаційної роботи є структуризація знань про WEB-технології, огляд наявних інструментів та вивчення можливостей для їх практичного застосування. Для досягнення мети поставлено такі завдання:

- Ознайомитися із концепцією WEB-прогармування;

- Дослідити засоби для роботи з WEB-службами;
- Виокремити недоліки та переваги використання різних фреймворків;
- Розробити практичну частину для продукту;
- Використати отриманні знання для реалізації власного WEB-сервісу;

Об’єкт, методи й засоби дослідження та розроблення. Об’єктом роботи є процес застосування засобів для розробки WEB-служб .Розробці практичної частини передувало ознайомлення із концепцією розробки WEB-сервісів, проектування, обгрунтування доцільності використання тих чи інших бібліотек та фреймворків.

Під час розробки програмного продукту враховувалися сучасні тенденції та рекомендації, які можна знайти на веб-сайтах для програмістів. Зокрема, опрацьовано статті, в яких зроблена оцінка щодо популярності того чи іншого застосунку і його функціоналу, а також розглянуто переваги та недоліки цього ресурсу під час практичного застосування.

Інструментом для створення Web-сервісу було обрано PyCharm Professional Edition, що підтримує мову програмування Python. Він пропонує широкий вибір бібліотек та фреймворків.

Можливі сфери застосування. На базі реалізованої практичної частини кваліфікаційної роботи можна створювати інтернет-магазини, використовуючи можливості бібліотек та фреймворків. Програмний продукт, можна використовувати у майбутньому, розширюючи та доукомплектовуючи функціонал.

РОЗДІЛ 1. ЗАГАЛЬНИЙ ОГЛЯД WEB-СЕРВІСІВ

1.1 Історія виникнення Інтернет-магазинів та основні задачі, що вони мали вирішувати

Почалася історія Інтернет-магазинів в Америці. До 1990 року Інтернет призначався для військових цілей, а його використання в бізнесі було заборонено регламентом Національного наукового фонду США. У 1990 році великі приватні компанії стали допускатися до мережі. Приблизно в той же час з'явився перший браузер, і поступово все більше людей навчилися користуватися комп'ютерами для доступу до мережі.

Ідея створення першого інтернет-магазину належить Джеффу Безосу, який в 1994 році припустив, що люди, які спілкуються за допомогою Інтернету, також захочуть замовляти товари та послуги онлайн. Розмірковуючи над тим, які товари буде найзручніше купувати в інтернеті, тобто не псуються і не крихкі, він почав продажу книг, аудіо- та відеокасет і дисків. Так в 1995 році відкрився перший інтернет-магазин Amazon, який до цих пір є одним з найбільших і найпопулярніших в світі. До речі серверна частина Amazon написана на мові Java.

Сьогодні спектр онлайн торгівлі настільки великий, що вже складно уявити, чого тільки можна купити через Інтернет. Існують Інтернет-магазини канцтоварів, побутових товарів, одягу, взуття, подарунків - чого тільки немає. Навіть продукти харчування вже можна купувати в мережі. І з кожним днем цей список стає все ширше, а бажаючих купувати товари через Інтернет-магазини з'являється більше і більше. Інтернет дозволив нам економити свій час і витрачати на невеликі покупки в рази менше ресурсів, ніж раніше. Вибір товару

і оплата відбувається настільки швидко, що за цей час ви ледь би приготувалися вийти з дому за покупками.

1.2 Сучасна архітектура WEB-сервісів

Базова платформа веб-сервісів - це XML + HTTP. Всі стандартні веб-сервіси працюють з використанням наступних компонентів:

- SOAP (простий протокол доступу до об'єктів);
- UDDI (універсальне опис, виявлення і інтеграція)**;**
- WSDL (мова опису веб-сервісів)

На Solaris можна створити веб-службу на основі Java, доступну з вашої програми Visual Basic, яка працює в Windows. Ви також можете використовувати C # для створення нових веб-служб в Windows, які можуть бути викликані з вашого веб-додатки, заснованого на JavaServer Pages (JSP) і працює в Linux.

Розглянемо просту систему управління рахунками та обробки замовлень. Співробітники бухгалтерії використовують клієнтське додаток, створений за допомогою Visual Basic або JSP, для створення нових облікових записів і введення нових замовлень клієнтів. Логіка обробки для цієї системи написана на Java і знаходиться на комп'ютері Solaris, який також взаємодіє з базою даних для зберігання інформації.

Кроки для виконання цієї операції наступні:

- Клієнтська програма пов'язує інформацію про реєстрацію облікового запису в повідомлення SOAP;
- Це SOAP-повідомлення відправляється веб-службі як тіло HTTP-запиту POST
- Веб-служба розпаковує запит SOAP і перетворює його в команду, зрозумілу додатком;
- Додаток обробляє інформацію в міру необхідності і відповідає новим унікальним номером облікового запису для цього клієнта;

- Потім веб-служба упаковує відповідь в інше повідомлення SOAP, яке відправляє назад клієнтській програмі у відповідь на свій HTTP-запит;

- Клієнтська програма розпаковує повідомлення SOAP, щоб отримати результати процесу реєстрації облікового запису;

РОЗДІЛ 2. ТЕХНОЛОГІЇ РОЗРОБКИ ТА СТРУКТУРА ПРОЕКТУ

2.1 Основна мета проекту та підбір технологій

Основною метою проекту була освоїти базові навички роботи з технологіями, які використовуються для роботи з WEB-сервісами. Мову програмування, було вирішено використовувати Python. Далі, потрібно було вибрати стек технологій, який буде використовуватися. Вибір став між Django та Flask стеком.

Flask і Django - обидва дорослі, розширювані веб фреймворки, які, в загальному, надають аналогічний функціонал в обробці запитів, підтримці документів, але розрізняються в масштабі відповідальності.

Більшість відмінностей між двома фреймворками випливають із різних підходів, решта — із відмінних основних проектних рішень. Ось невеликий список ключових відмінностей:

Рис.1 Flask vs Django

1.	Flask використовує локальні потоки	Django передає запити там, де потрібно
2.	Flask не підтримує форми за замовчуванням	Django має вбудовані форми, які інтегруються з ORM і панеллю

		адміністратора
3.	Flask надає безпечні cookies файли	Django надає застосунок аутентифікації
4.	Flask не може цим похвалитися, однак є такі інструменти, як SQLAlchemy, які надають аналогічний функціонал	Він заснований на багатоярусній архітектурі, що містить безліч модулів.Django доступний із вбудованою ORM та системою міграції, яка може керувати базами даних
5.	Flask не має вбудованої панелі адміністратора, але є Flask-Admin-розширення, яке допомагає вирішити проблему.	Django включає в собе повністю інтегрований адмін-інтерфейс для керування даними застосунка

Не дивлячись на більш легшу реалізацію проекту, за допомогою Flask, було вирішено використовувати Django стек, в угоду швидкості.

Далі виникло питання, яку ж СУБД (Система Управління Базами Даних) найоптимальніше використати в своєму WEB-сервісі. Звичайно це повинна бути реляційна система управління базами даних, яка реалізовує реляційну модель роботи з даними, яка визначає всю збережену інформацію як набір пов'язаних записів і атрибутів в таблиці. СУБД такого типу використовують структури (таблиці) для зберігання і роботи з даними. Кожен стовпець (атрибут) містить свій тип інформації. Кожен запис в базі даних, що володіє унікальним ключем, передається в рядок таблиці, і її атрибути відображаються в стовпцях таблиці. Зразу на думку спали три СУБД: SQLite, MySQL та PostgreSQL. Звичайно вони всі мають свої переваги та недоліки. Підемо по порядку:

SQLite. Це чудова бібліотека, вбудована в додаток, який її використовує. Будучи файловою БД, вона надає відмінний набір інструментів для більш простої (в порівнянні з серверними БД) обробки будь-яких видів даних. Її переваги, заключаються в тому, що вона

стандартизована і відмінно підходить для розробки та тестування. Немало важливим фактором являється, те що створена база даних зберігається в одному файлі. Недоліками являється, відсутність користувацького керування і додаткових налаштувань.

MySQL. Це найпопулярніша з усіх великих серверних БД. Розібратися в ній дуже просто, та й в мережі про неї можна знайти велику кількість інформації. Хоча MySQL і не намагається повністю реалізувати SQL-стандарти, вона пропонує широкий функціонал. Додатки спілкуються з базою даних через процес-демон. MySQL привертає увагу своєю простотою, безпечністю, швидкістю та масштабованістю, хоча і має проблеми з надійністю, обмеженнями та застоєм в розробці.

PostgreSQL. PostgreSQL - це сама просунута РСУБД, що орієнтується в першу чергу на повну відповідність стандартам і розширюваність. PostgreSQL, або Postgres, намагається повністю відповідати SQL-стандартам ANSI / ISO.

PostgreSQL відрізняється від інших РСУБД тим, що володіє об'єктно-орієнтованим функціоналом, в тому числі повною підтримкою концепту ACID (Atomicity, Consistency, Isolation, Durability). Її переваги це — повна SQL-сумісність, підтримка сторонніми організаціями, розширюваність та об’єктно-орієнтованість. Швидкодія, популярність і проблематичність знайти відповідного провайдера являються проблемами цієї СУБД.

Виходячи з аргументів вище, вирішено було працювати з PostgreSQL, так як її недоліки не були критичними для нашого WEB-сервіса, а переваги являються більш ніж очевидними.

2.2 Огляд створеного WEB-сервісу

З самого початку ми потрапляємо на головну сторінку, де ми можемо бачити категорії наших товарів.

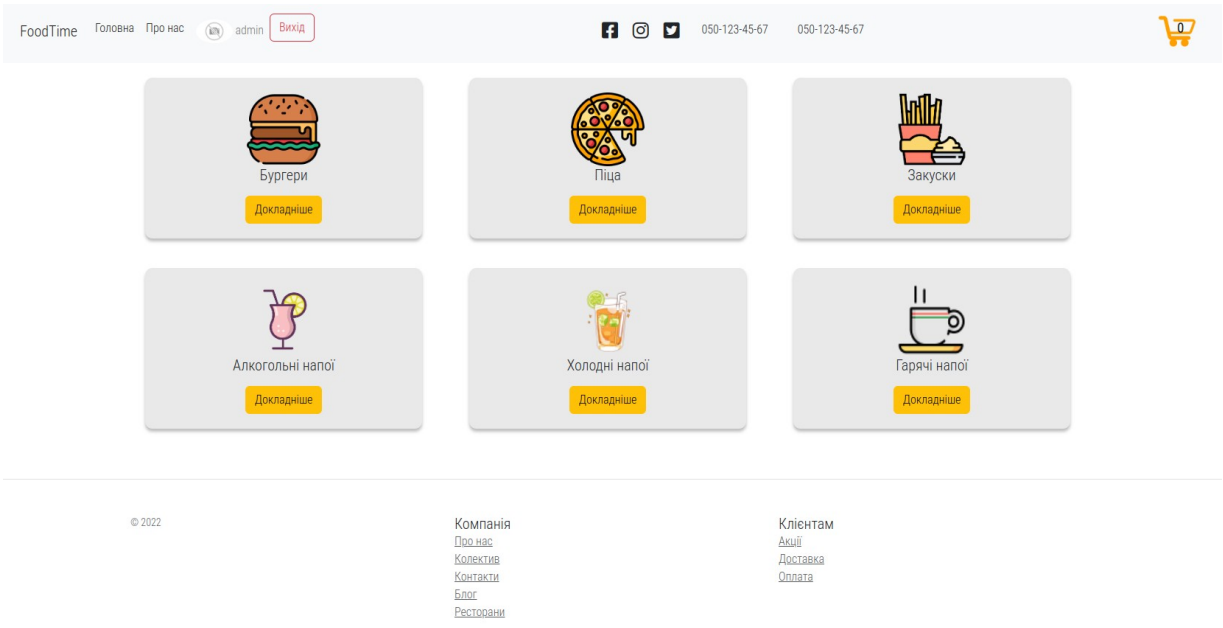

Рис.2 Головна сторінка

Якщо натиснути на кнопку “Докладніше” під будь-якою категорією, то ми зможемо переглянути всі товари цієї категорії. Тут же ми бачимо, що під позиціями знаходиться пагінація. За її допомогою, можна переглянути всі позиції категорії. Також над товарами знаходиться сортування за назвою, ціною та вагою.

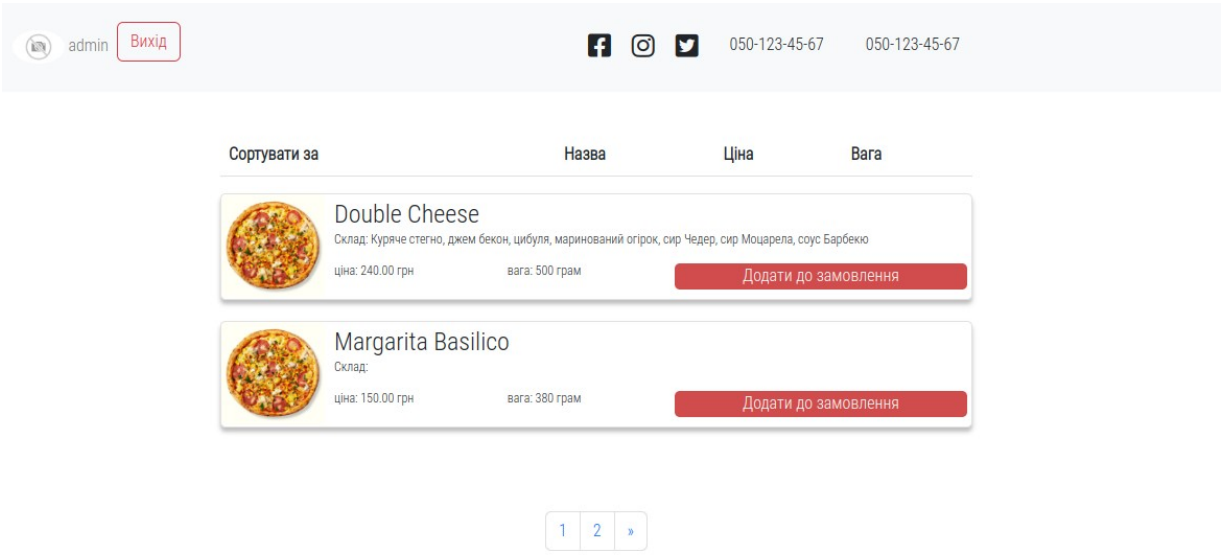

Рис.3 Товари категорії

Натиснувши на кнопку “Додати до замовлення”, ми додаємо товар в корзину, де потім можемо видалити не потрібні нам позиції, якщо ми їх додали випадково.

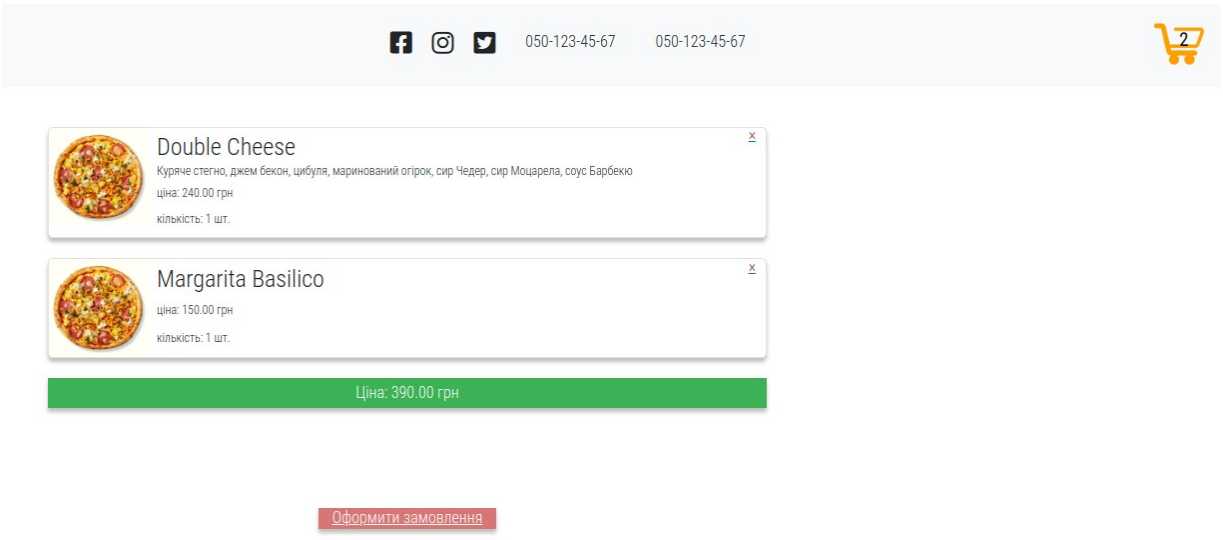

Рис.4 Корзина

Натиснувши на кнопку “Оформити замовлення”, ми переходимо на сторінку з формою, де потрібно вказати свої дані і спосіб оплати.

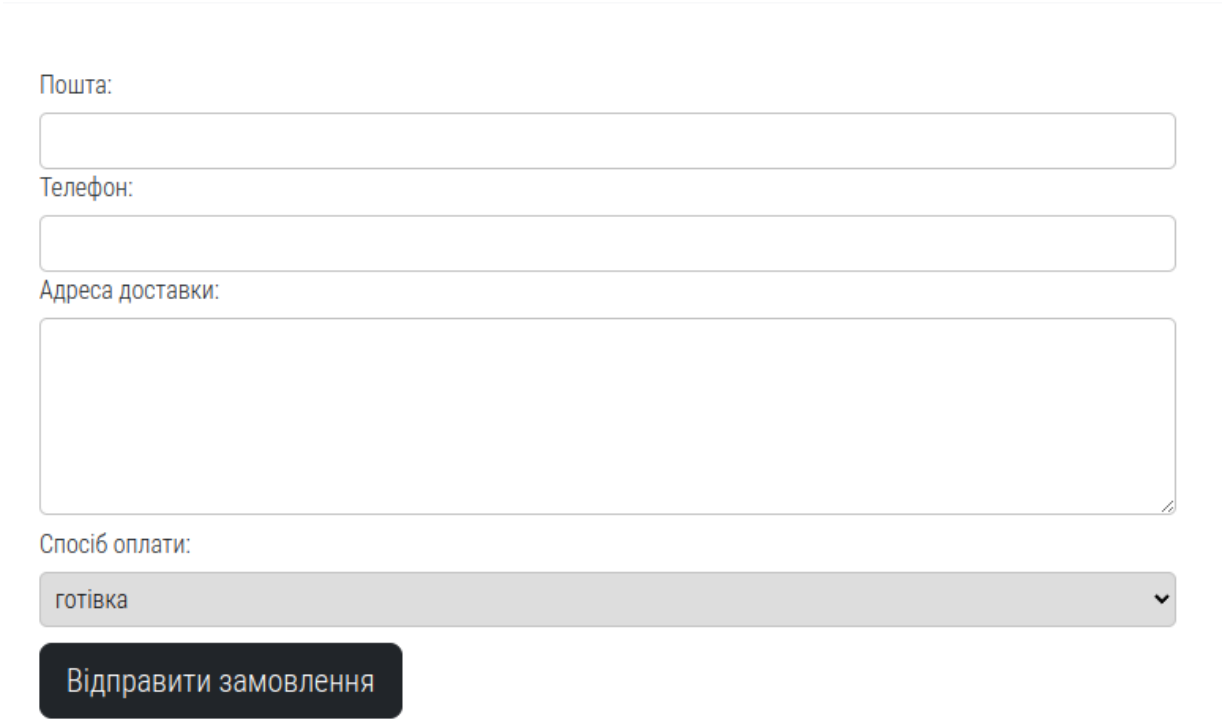

Рис.5 Оформлення замовлення

Якщо ми обираємо спосіб оплати “Онлайн”, тоді нас перенаправляє на сторінку, де нам потрібно оплатити замовлення.

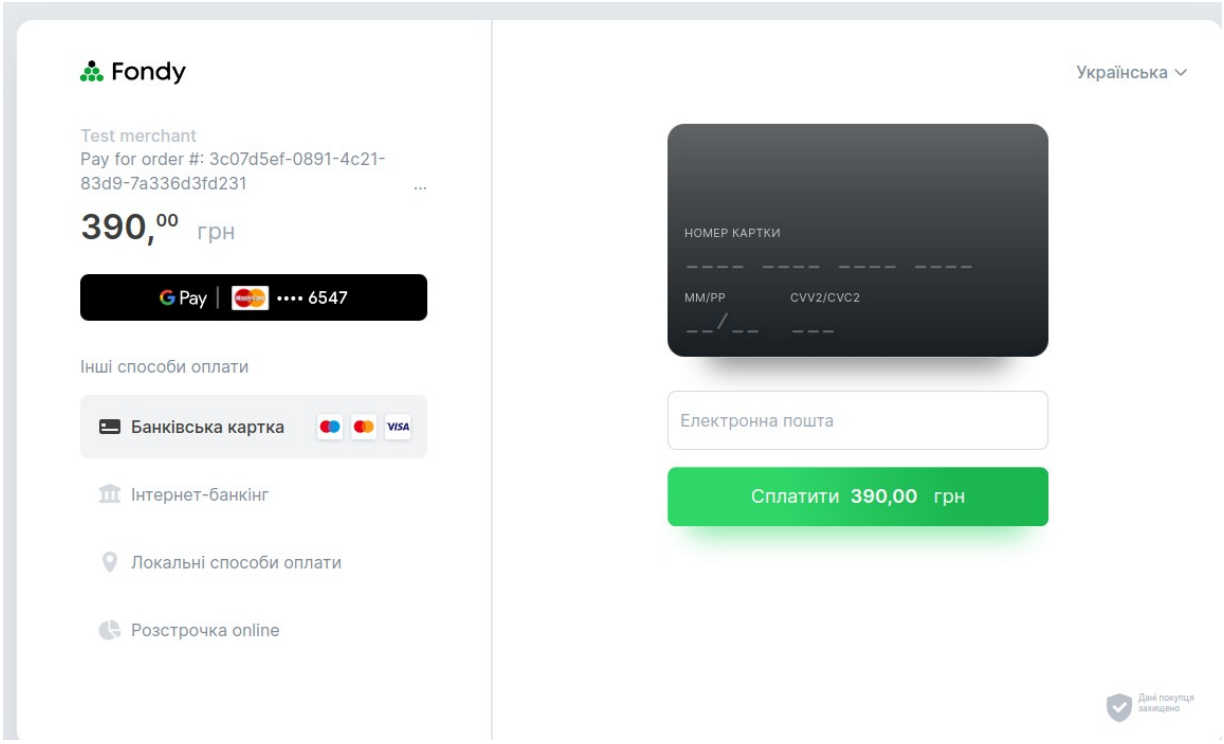

Рис.6 Оплата замовлення

В верхньому колонтитулі ми маємо посилання на соціальні мережі, телефони та можливість зареєструвати нового користувача або авторизувати вже створеного.

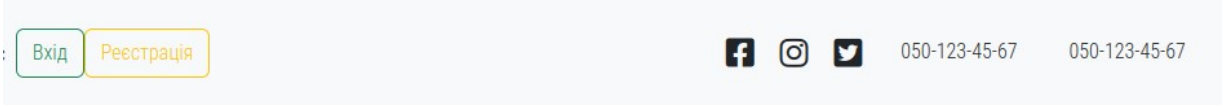

Рис.7 Верхній колонтитул

Для адміністратора є можливість керувати панеллю адміністратора.

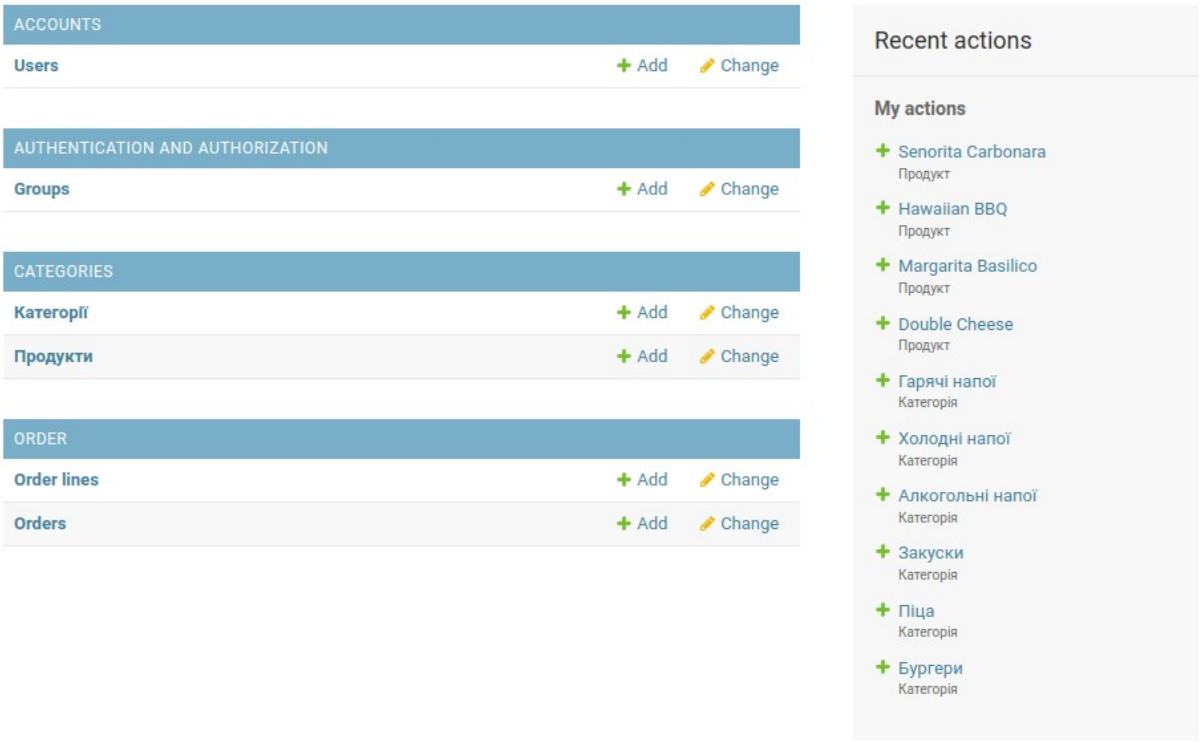

Рис.8 Панель адміністратора

За допомогою цієї панелі він може:

- Додавати нові категорії та товари, їх редагувати чи видаляти.
- Переглядати замовлення та міняти їх статус.
- Отримувати відомості про останні операції в панелі.
- Додавати нові групи користувачів.

Ще наш сервіс надає API наших товарів та замовлень. Це допоможе в майбутньому сильно розширити наш функціонал.

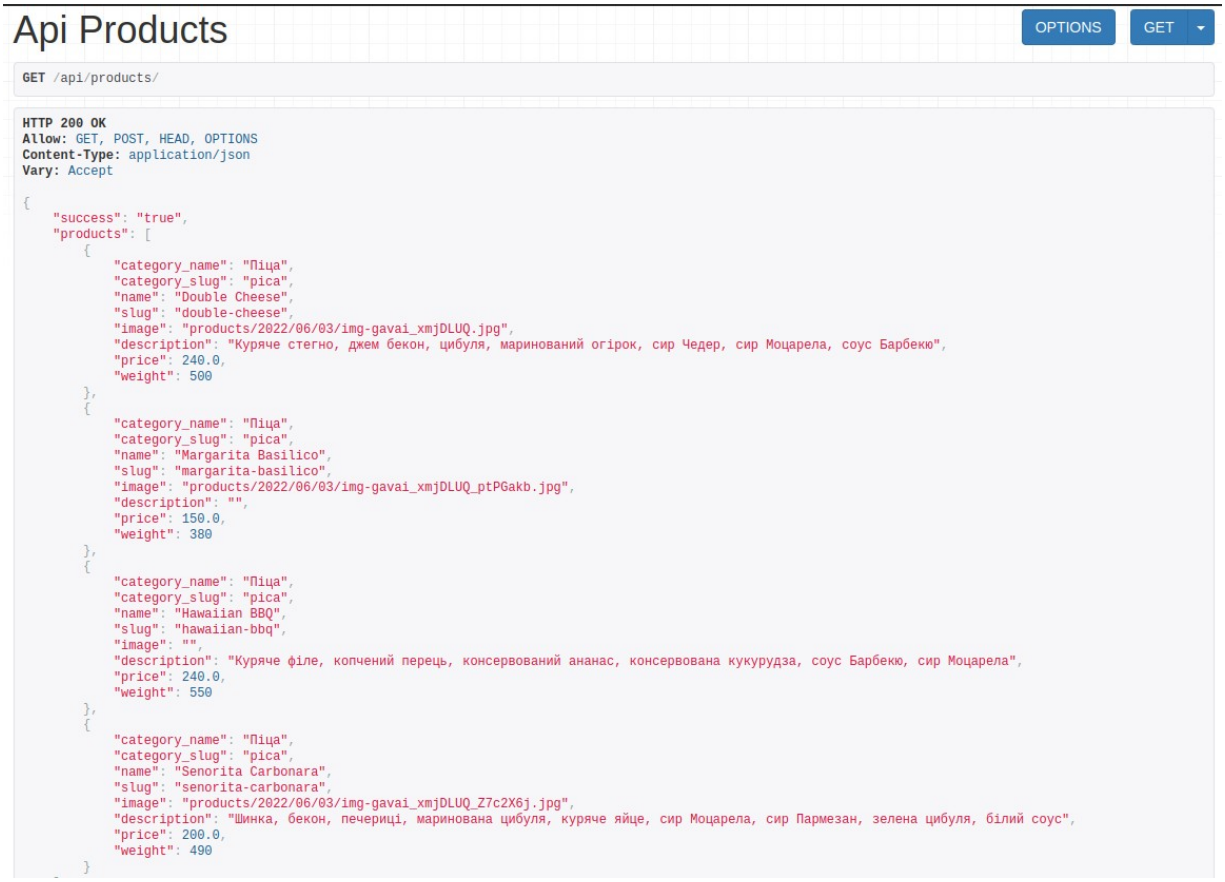

Рис.9 Інтерфейс API

2.3 Структура WEB-сервісу

Фреймворк Django реалізує архітектурний патерн Model-View-Template або скорочено MVT, який є модифікацією розповсюдженого у веб-програмуванні патерну MVC (Model-View-Controller).

Нижче ми можемо спостерігати графічне представлення цього патерну.

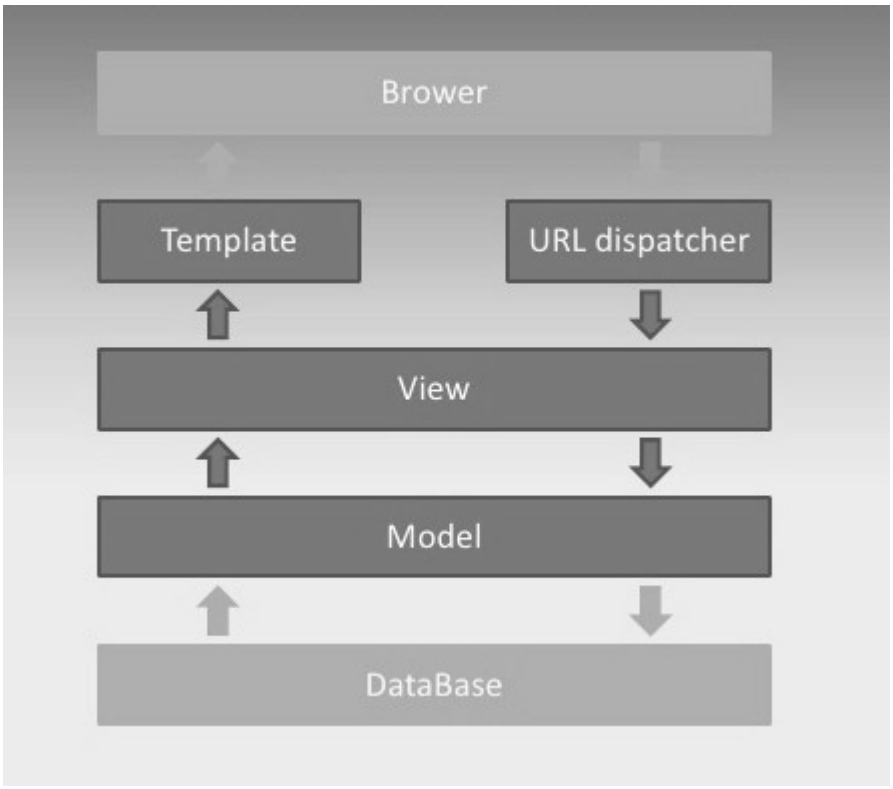

Рис.10 MVT патерн

Основні елементи патерну:

URL dispatcher: при отриманні запиту на підставі запрошеної URL-адреси визначає, який ресурс повинен обробляти даний запит.

View: отримує запит, обробляє його та відправляє у відповідь користувачеві певну відповідь. Якщо для обробки запиту необхідне звернення до моделі та бази даних, то View взаємодіє із нею. Для створення

відповіді може використовуватися Template або шаблони. В архітектурі MVC цьому компоненту відповідають контролери.

Model: описує дані, що використовуються у програмі. Окремі класи, як правило, відповідають таблицям у базі даних.

Template: представляє логіку уявлення у вигляді згенерованої розмітки HTML. У MVC цьому компоненту відповідає View, тобто представлення.

Коли до програми надходить запит, URL dispatcher визначає, з яким ресурсом зіставляється даний запит і передає цей запит обраному ресурсу. Ресурс фактично представляє функцію або View, який отримує запит та певним чином обробляє його. У процесі обробки View може звертатися до моделей та бази даних, отримувати з неї дані, або, навпаки, зберігати у ній дані. Результат обробки запиту відправляється назад, і цей результат бачить користувач у своєму браузері. Як правило, результат обробки запиту є згенерований html-код, для генерації якого застосовуються шаблони (Template).

Почнемо огляд структури нашого web-сервісу з ключової директоріїї **online_store**:

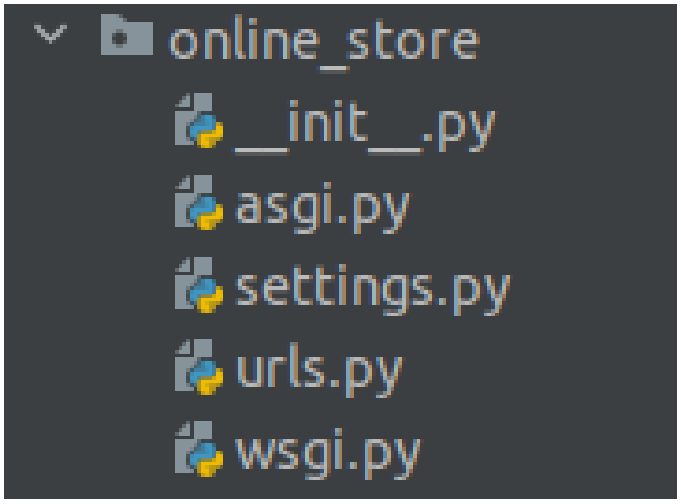

Рис.11 Директорія online_store

settings.py містить усі налаштування проекту. Тут ми реєструємо додатки, задаємо розміщення статичних файлів, налаштування бази даних тощо.

urls.py задає асоціації url адрес з уявленнями. Незважаючи на те, що цей файл може містити всі налаштування URL, зазвичай його ділять на частини, по одній на додаток.

__init__.py — файл для того, щоб Django і Python розпізнавали папку як Python модуль і дозволяє нам використовувати його об'єкти всередині інших частин проекту.

Більш докладно розглянемо основні налаштування в файлі **settings.py**

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'accounts',
    'categories',
    'order',
    'django_filters',
]
```

Рис.12 Налаштування додатків

Після створення додатку, нам потрібно зареєструвати його в проекті, щоб різні утиліти взаємодіяли з ним (наприклад, додавання моделей до бази даних). Додатки реєструються додаванням їх назв до списку **INSTALLED_APPS** у налаштуваннях проекту.

Далі ми переходимо до налаштування бази даних і розберемо параметри:

‘ENGINE’ — система управління базами даних, яку ми використовуємо;

‘NAME’ — назва бази даних;

‘USER’ — користувач, який керує базою;

‘PASSWORD’- пароль для користувача;

‘HOST’ — хост, який використовується під час підключення до бази даних;

‘PORT’ — порт підключення;

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'online_store',
        'USER': 'online_store',
        'PASSWORD': 'online_store',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

Рис.13 Налаштування бази даних

В цьому ж файлі, ми налаштовуємо статичні файли і файли з різними медіа (зображеннями, відео).

```
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
STATIC_URL = '/static/'
STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'static'),
)
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
```

Рис.14 Налаштування файлів

Також, в цьому файлі, є налаштування мови та часового поясу.

```
LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'
```

Рис.15 Налаштування мови та часового поясу

Розглянемо деякі додатки, з яких складається наш сервіс. Перший з них, це додаток, який відповідає за товари та їх категорії.

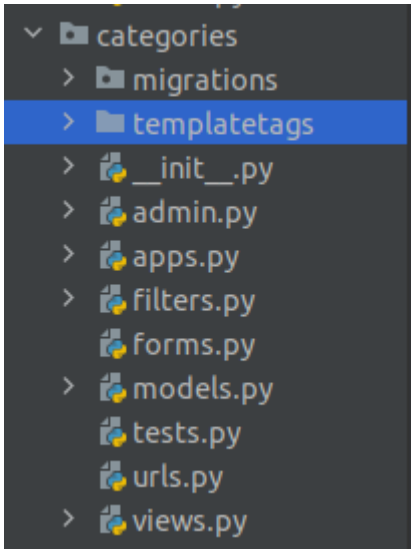

Рис.16 Додаток categories

В файлі **models.py** описуються моделі товарів і категорій.

```python
class Product(models.Model):
    category = models.ForeignKey(Category, related_name='products', on_delete=models.CASCADE)
    name = models.CharField(max_length=200, db_index=True)
    slug = models.SlugField(max_length=200, db_index=True)
    image = models.ImageField(upload_to='products/%Y/%m/%d', blank=True)
    description = models.TextField(blank=True)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    weight = models.PositiveIntegerField()
    available = models.BooleanField(default=True)
```

Рис.17 Опис моделі Product

Ми бачимо клас, що описує модель продуктів, який має поля:

category — вказує, до якої категорії належить товар;

name — назва товару;

slug — URL товару;

image — зображення товару;

description — опис позиції;

price — ціна;

weight — вага страви;

available — наявність товару;

В файлі **views.py** ми розміщаємо логіку нашого додатку, Воно надсилає запит до моделі, яку ми створили заздалегідь, та передає її в шаблон(template).

Нижче, ми бачимо один із прикладів цієї логіки, який виводить всі позиції з однієї категорії і сортує їх за якимось параметром (ціна, вага, назва).

```python
def category_products(request, slug, pk):
    category = get_object_or_404(Category, slug=slug, pk=pk)
    products_category = category.products.all()
    products_filter = ProductFilter(request.GET, queryset=products_category)
    products = get_paginator_items(
        products_filter.qs, settings.PRODUCT_PAGINATE_BY, request.GET.get("page")
    )
    sort_by = request.GET.get("sort_by")

    return render(request, 'products.html', {'category': category,
                                             'products': products, 'sort_by': sort_by})
```

Рис.18 Функція виведення позицій

За цим же принципом, працюють і інші додатки, вони мають схожу структуру. В окрему директорію, винесені templates (шаблони).

Шаблон Django — це текстовий документ або рядок Python, розмічений за допомогою мов шаблонів Django. Деякі конструкції розпізнаються та інтерпретуються механізмом шаблонів. Основними з них є змінні та теги.

Оглянемо деякі елементи директорії **templates**. В файлі **base.html** ми маємо розмітку, за допомогою якої розширюються деякі інші файли, для того, щоб не повторювати код по декілька разів. До цього відноситься нижній та верхній колонтитул.

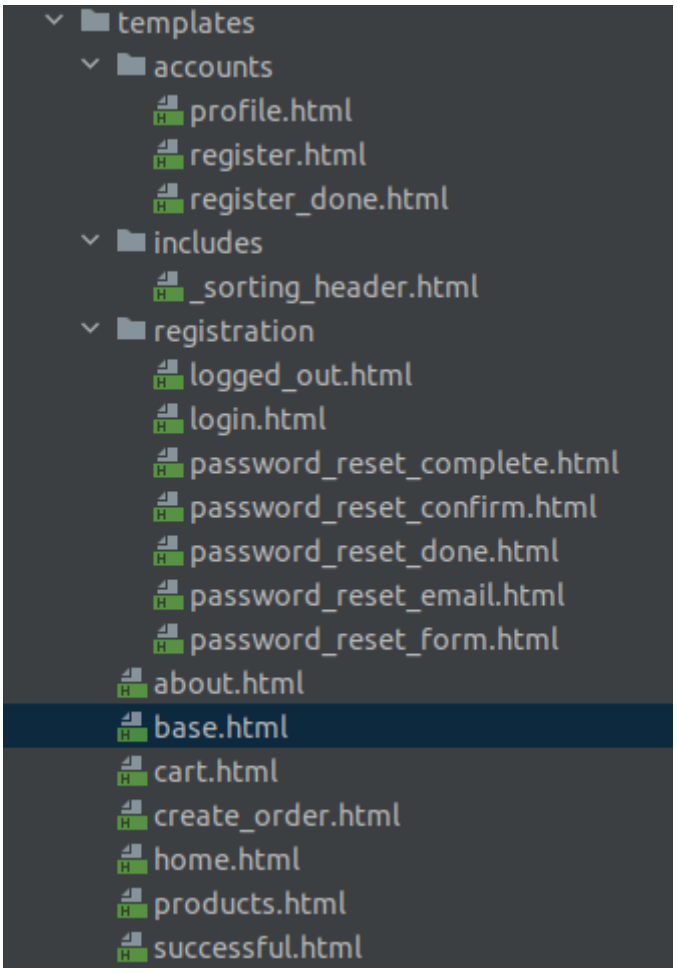

Рис.19 Директорія templates

В файлі **home.html** у нас знаходиться шаблон для головної сторінки. За допомогою цього шаблону, на головну сторінку виводяться всі категорії з посиланнями на всі товари категорії.

```
<div class="categories">
    <div class="container">
        <div class="row justify-content-sm-center justify-content-md-center">
            {% for category in categories %}
                <div class="col-sm-10 sol-md-4 col-lg-4">
                    <div class="category">
                        <img src="{{ category.get_image }}" alt="{{ category.name }}" width="100" height="100">
                        <h5>{{ category.name }}</h5>
                        <div><a href="{{ category.get_absolute_url }}" class="btn btn-warning mt-3">Докладніше</a></div>
                    </div>
                </div>
            {% endfor %}
        </div>
    </div>
</div>
```

Рис.20 Шаблон головної сторінки

Окрему увагу звернемо на директорію **api**, де в нас знаходиться додаток, який реалізовує WEB API за допомогою Django REST framework.

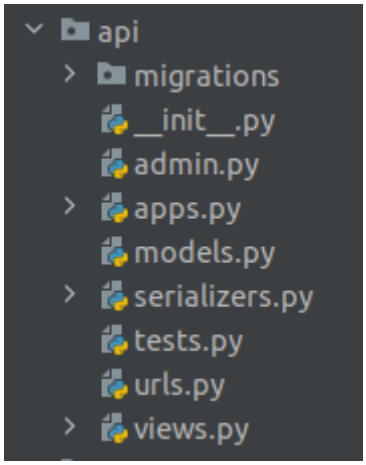

Рис.21 Директорія api

Найцікавіше знаходиться в файлі **serializers.py**, де серіалізуються і десеріалізуються об’єкти. Тут ми бачимо один із класів, який серіалізує наші товари.

```
class ProductSerializerGet(serializers.Serializer):
    category_name = serializers.SerializerMethodField()
    category_slug = serializers.SerializerMethodField()
    name = serializers.CharField()
    slug = serializers.CharField()
    image = serializers.CharField()
    description = serializers.CharField()
    price = serializers.SerializerMethodField()
    weight = serializers.IntegerField()

    def get_category_name(self, obj):
        return (obj.category.name)

    def get_category_slug(self, obj):
        return (obj.category.slug)

    def get_price(self, obj):
        return (obj.price)

    def get_product_weight(self, obj):
        return (obj.weight)
```

Рис.22 Класс ProductSerializerGet

Висновки

Під час написання цієї роботи, було проаналізовано переваги тих чи інших технологій для створення власного WEB-сервісу. Було вирішено розробити сервіс, що допоможе магазину привабити нових клієнтів та дозволить автоматизувати його роботу, шляхом продажу товарів онлайн.

Першим кроком було формулювання функціональних та нефункціональних вимог до системи, що визначило очікувану поведінку системи та проблеми, які вона буде вирішувати. Наступним кроком, було обрано технології за допомогою яких буде написаний веб-сервіс. Python була обрана в якості мови програмування, середовище розробки – PyCharm через свою потужність та багатофункціональність. В якості системи управління базами даних було вирішено використовувати PostgreSQL через свою доступність та надійність. Оскільки веб-сервіс потребує графічний інтерфейс, для динамічної стилізації сторінок було вирішено використовувати HTML та CSS за їх легкість у впровадженні та застосуванні. Наступним кроком були описані сценарії використання. Архітектуру системи було розділено на логічні рівні, задля коректної взаємодії компонентів програми. Основна бізнес логіка була описана мовою програмування Python в класах, що взаємодіють з базою даних, таким же чином описано основні сутності.

Отже, що основні переваги створеного веб-сервісу полягають в простоті використання, зручному й легкому інтерфейсі користувача та швидкому доступу до інформації. Система є відкритою до розширення тому її можна легко модифікувати, наприклад додати логіку ведення інтересів користувачів та підключити інші додатки за допомогою нашого API. Таким чином веб-сервіс має потенціал до розвитку й може стати конкурентно спроможним.

Джерела

1. Історія розвитку інтернет-магазинівРежим доступу:https://oren.aif.ru/dosug/purpose/istoriya_razvitiya_internet-magazinov;
2. Історія розвитку Інтернету. Режим доступу: https://lectureswww.readthedocs.io/1.introduction/history.html;
3. SQLite, MySQL и PostgreSQL: порівнюємо популярні реляційні СУБД. Режим доступу: https://tproger.ru/translations/sqlite-mysql-postgresql-comparison/;
4. Веб-сервіси Режим доступу: https://coderlessons.com/tutorials/veb-razrabotka/izuchite-veb-servisy/veb-servisy-kratkoe-rukovodstvo;
5. Django (webframework) Режим доступу: https://en.wikipedia.org/wiki/Django_(web_framework)
6. Що таке Django? Режим доступу: https://metanit.com/python/django/1.1.php__
7. Tamplates. Режим доступу: https://docs.djangoproject.com/en/4.0/topics/templates/#:~:text=A%20Django%20template%20is%20a,is%20rendered%20with%20a%20context.
8. Прикладний програмний інтерфейс (API). Режим доступу: https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81

9. Flask чи Django? Вибираємо Python вебфреймворк Режим доступу: https://python-scripts.com/flask-or-django