МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Інтернет магазин на базі фреймворку Django»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Шелехов І.В.**

Студент гр. ІН–63 **Коропець М.О.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедри Довбиш А.С.

"_____" _____2020 р.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи ІН-63 спеціальності "Комп'ютерні науки" денної форми навчання Коропець Максима Олександровича.

Тема: " Інтернет-магазин на базі фреймворку Django"

Затверджена наказом по СумДУ

№_____ от _____2020 р.

Зміст пояснювальної записки: 1) огляд всіх доступних методів розробки; 2) постановка задачі й налаштування робочої середи; 3) огляд методів та технологій створення магазину на фрейморку Django 4) розробка програмного продукту та налаштування бази даних 5) перевірка кінцевого результату.

Дата видачі завдання "_____" _____2020р.

Керівник випускної роботи _____ Шелехов І.В.

Завдання прийняв до виконання _____Коропець М.О.

РЕФЕРАТ

Записка:. 52 стор., 24 рис., 1 додаток, 15 джерел.

Об’єкт дослідження — процес проектування інтернет магазину.

Мета роботи — розробка інтернет магазину використовуючи мову Python на фреймворку Django з використанням бази даних SqLite.

Методи дослідження — методи проеткування веб-орієнтованих інформаційних систем.

Результати — створено інтернет магазин, в якому користувач може придбати або поставити на продажу будь-який товар. Магазин було реалізовано на мові Python на популярному фреймворку Django. При цьому сайт не містить зайвої анімації і тому він простий для різних браузерів та доволі швидко завантажується на різних швидкостях інтернету .

ІНТЕРНЕТ МАГАЗИН, PYTHON, ФРЕЙМОРК, DJANGO

ЗМІСТ

ВСТУП

Робота присвячена темі розробки Інтернет магазину на фрейморку Django з використанням бібліотек розроблених під нього.

Обрана тема є доволі актуальною, тому що на сьогодні дуже багато людей надають перевагу покупкам в інтернеті. Це досить зручно, адже купуючи різні товари в інтернет магазинах не потрібно навіть виходити з дому. Для бізнесменів створення таких інтернет магазинів вигідне, тому що це не лише зручно, а й створює додатковий трафік клієнтів. Крім того це може бути ще й інформаційною сторінкою для користувачів, які хочуть дізнатися додаткову інформацію про магазин, або подивитися наявність того чи іншого товару та його ціну. Також перевагою є те, що клієнти мають можливість одразу порівнювати ціни товарів в різних інтернет магазинах. Для цього створено вже досить багато безкоштовних сервісів. Також існують кешбек сервіси, які допоможуть зекономити гроші при купівлі товару. У світі та Україні кожного дня стає все більше користувачів Internet і тому виростає кількість нових клієнтів у інтернет магазинах. Ось чому вони завжди будуть актуальними.

Для того щоб почати розробку магазину потрібно зрозуміти основні можливості відповідних фреймворків та виділити їх переваги та недоліки.

1 ОГЛЯД ВІДОМИХ РІШЕНЬ

1.1 Відомі Python-фреймворки

Фреймворк - інструмент, що полегшує процес написання та використання веб-додатків.

На мові Python є доволі багато відомих та якісних фреймворків. До вибору фреймворка потрібно підходити відповідально адже це буде впливати на якість нашого майбутнього магазину. Маже кожний фреймворк можна використовувати для створення магазину.

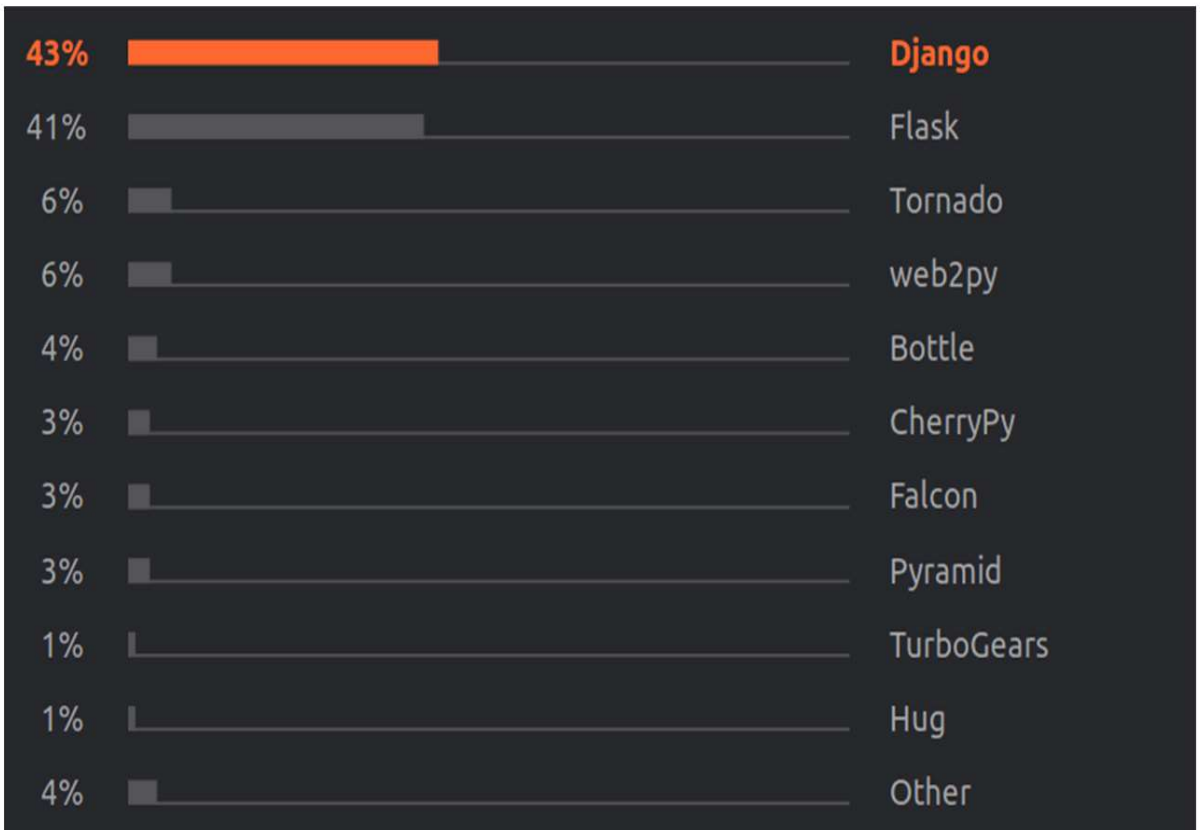


Рисунок 1.1 – Графік популярності Фреймворків Phyton

Як бачимо на графіку на данний момент можна виділити чотири найпопулярніші фреймворки це Django , Flask , Tornado, web2py.

Потрібно вибрати один фреймворк який найбільше нам буде підходити. Переваги та недоліки фреймворків:

Flask:

Рисунок 1.2 – Фреймворк Flask

- гарна продуктивність
- мінімалістичний і не має обмежень
- можно реалізувати саме те, що ти хочеш
- достатня кількість пакетів гарний захист
- запобігання витоку інформації та запобігання веб-атак
- вбудований сервер розробки

Tornado:

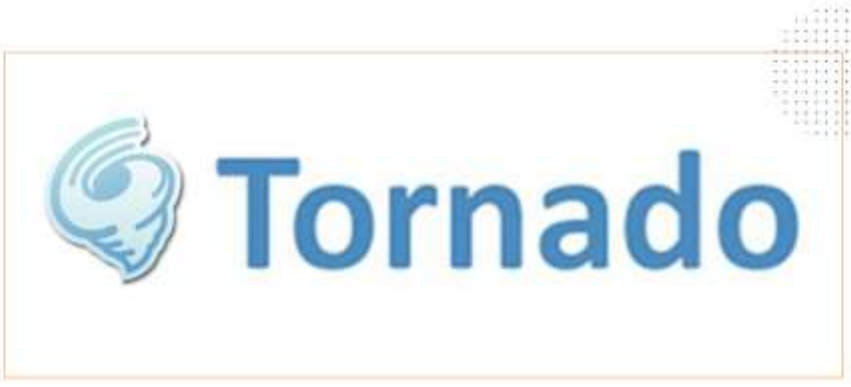


Рисунок 1.3 – Фреймворк Tornado

- вбудована підтримка аутентифікації користувача
- підтримка перекладу та локалізації
- гарна продуктивність
- якісна структура

- неблокуючий HTTP-клієнт
- реалізація сторонніх схем аутинтефікації

Web2py:

Рисунок 1.4 – Фреймворк Web2py

- система з відкритим кодом з повноцінним стеком Python
- навчальний інструмент з акцентом на простоті використання
- має власний веб-IDE
- зручний у використанні
- не має вимог до встановлення налаштувань

1.2 Постановка завдання

Метою роботи є розробка та програмна реалізація Web-орієнтованої системи для Інтернет-магазину. Для досягенення поставленої мети необхідно було виконати такі завдання:

1) дослідження популярних фреймворків для розробки магазину і вибір потрібного для розробки ;
2) огляд популярних інтернет магазинів та створення прототипу проекту який відрізняється тим, що не матиме зайвої інформації про товар і буде мати високу швидкість звавантаження;
3) розробка frontend частини та верстка компонентів.
4) розробка backend частини
5) тестування Інтернет-магазину

2 ВИБІР МЕТОДУ РІШЕННЯ

2.1 Переваги та недоліки розробки на Django

Рисунок 2.1 – Фреймворк Django

Коли виникає певна ідея, трансформувати її на мові програмування і задати їй реальну форму за допомогою Django займе всього кілька хвилин. Те, що Django знаходиться у вільному доступі, дає можливість помітно спростити процес веб розробки, так як розробник може сфокусуватися на процесі дизайну і розробки функціоналу програми. Таким чином, Django - це ідеальний інструмент для створення веб-проектів.[3]

Django з'явився в 2005 році, і поступово став одним з кращих фреймворків, який допомагав і допомагає тисячам розробників виконувати ту чи іншу роботу протягом декількох хвилин. Спочатку Django був фреймворком для мови Python, з відмінним функціоналом, Django помітно спростив ряд складнощів в розробці веб додатків, надавши цій роботі більш спрощений підхід. Добре відомо, що Python є одним з найбільш використовуваних мов програмування завдяки простоті у вивченні, дизайну і гнучкості, що робить його практично досконалим мовою програмування. [7]

Переваги Django:

- Швидкість: Django був розроблений, щоб допомогти розробникам створити додаток настільки швидко, на скільки це можливо. Це включає в себе формування ідеї, розробку і випуск проекту, де Django

економить час і ресурси на кожному з цих етапів. Таким чином, його можна назвати ідеальним рішенням для розробників, для яких питання дедлайну є в пріоритеті.

- Повна комплектація: Django працює з десятками додаткових функцій, які помітно допомагають з аутентифікацією користувача, картами сайту, адмініструванням вмісту, RSS і багато чим іншим. Дані аспекти допомагають здійснити кожен етап веб розробки.
- Безпека: Працюючи в Django, ви отримуєте захист від помилок, пов'язаних з безпекою які ставлять під загрозу проект. При цьому ефективно використовується механізм логінів і паролів, система користувальницької аутентифікації.
- Масштабованість: фреймворк Django найкращим чином підходить для роботи з найвищими трафіками. Отже, логічно, що велика кількість завантажених сайтів використовують Django для задоволення вимог, пов'язаних з трафіком.
- Різнобічність: менеджмент контенту, наукові обчислювальні платформи, навіть великі організації - з усім цим можна ефективно справлятися за допомогою Django.

Недоліки Django:

- Використання шаблону маршрутизації із зазначенням URL
- Django занадто монолітний
- Все базується на ORM Django
- Компоненти розгортаються спільно
- Необхідно вміння володіти всією системою для якісної роботи

Коли розглядається проект з дедлайном, використання Django для вирішення поставленого завдання - грамотне рішення. Більш того, для прискорення робочого процесу, є можливість встановити для користувача свою конфігурацію. Раніше, коли розробка коду вимагала великої кількості

часу, така можливість дозволила помітно спростити цей процес. Вартість розробки знизилася практично на 80%.[9]

2.2 Огляд розробки на вибраному фреймворку

Щоб почати нашу розробку сайту потрібно встановити Django. Для цього потрібно прописати в командній строці :

```
pip install Django==1.9.1.
```

Якщо правильно встановлено Django, то після запуску django-admin –version буде відображена поточна версія фреймворку.

Тепер створимо проект. Це можна зробити наступним чином:

```
django-admin startproject django_example.
```

Як тільки створення проекту буде завершено, наша директорія проекту буде виглядати так:

- `django_example / __ init__.py` - порожній файл, який говорить Python, що дана директорія повинна сприйматися як пакет.
- `django_example / settings.py` - містить конфігурацію нашого проекту.
- `urls.py` - тут оголошуються URL.
- `django_example / wsgi.py` - за допомогою нього додаток може працювати з веб-сервером по протоколу WSGI.
- `manage.py` дозволяє взаємодіяти з проектом.

Тепер прийшов час запустити наш додаток. Для цього в командному рядку потрібно написати

```
python manage.py runserver
```

Після цього в адресному рядку браузера потрібно написати:

```
http://127.0.0.1:8000/
```

Якщо все правильно зроблено повинно вивести надпис, що все правильно.

Тепер створимо найпростіший додаток. Потрібно знову ж таки прописати:

```
python manage.py startapp riddles
```

Як тільки додаток створений, давайте напишемо простий вигляд, за правилами Django всі види повинні зберігатися в файлі `views.py`

```python
from django.http import HttpResponse


def index(request):
    return HttpResponse("Hello, World!")
```

Рисунок 2.2 – Файл views.py

Тепер, щоб прив'язати наш вид до URL, створимо файл `urls.py`

```python
from django.conf.urls import url

from . import views

app_name = 'riddles'

urlpatterns = [
    url(r'^$', views.index, name='index'),
]
```

Рисунок 2.3 – Файл urls.py

У `urls.py` ми повинні написати наступне:

```
from django.conf.urls import include, url
from django.contrib import admin

urlpatterns = [
    url(r'^riddles/', include('riddles.urls')),
    url(r'^admin/', admin.site.urls),
]
```

Рисунок 2.3 – Файл urls.py

Тепер, якщо ми запустимо наш додаток `http://127.0.0.1:8000/riddles/` ми побачимо «Hello, World!».

2.3 Відомі моделі для Django

- django-hvad

Цікава реалізація зберігання багатомовних полів в моделях. Для кожної перекладної моделі буде створена додаткова таблиця.

```
class DjangoModel(TranslatableModel):
    name = models.CharField(max_length=255, unique=True)
    author = models.CharField(max_length=255)

    translations = TranslatedFields(
        description = models.TextField(),
        description_author = models.CharField(max_length=255),
    )

    def __unicode__(self):
        return self.name
```

Рисунок 2.4 – django-hvad

- django-whatever (django-any)

Модуль дозволяє відмовитися від створення громіздких фікстур для тестів, замінивши їх простим створенням об'єктів моделей на льоту,

заповнюючи поля випадковими даними. Завдяки цьому тести стають більш читабельними і підтримуваними.

```
from django_any import any_model

class TestMyShop(TestCase):
    def test_order_updates_user_account(self):
        account = any_model(Account, amount=25, user__is_active=True)
        order = any_model(Order, user=account.user, amount=10)
        order.proceed()

        account = Account.objects.get(pk=account.pk)
        self.assertEquals(15, account.amount)
```

Рисунок 2.5 – django-whatever

- django-guardian

Додаток реалізує відсутність прав на об'єкт, а не на всю модель (object-level permissions). Починаючи з Django 1.2 бекенд аутентифікації підтримує перевірку прав на об'єкт, але це не реалізовано в самому Django. django-guardian успішно заповнює цю прогалину. Єдиним виявленим недоліком є, мабуть, те, що наявність глобальних прав на модель не дає прав на конкретний об'єкт і вимагає окремої перевірки.

- django-sphinx

Sphinx, популярний пошуковий движок, - наше все, завдяки його швидкості, гнучкості. Незважаючи на популярність, існує одне єдине рішення-інтегратор для Django - django-sphinx.

Пошукові запити через менеджер моделей, можна уточнювати такі параметри як вага полів або назви індексів прямо в описі класу моделі на основі зазначених параметрів вміє автоматично генерувати sphinx-конфиг.

2.4 Вибір середовища розробки

PyCharm - IDE для професійної розробки на Python, розроблений JetBrains для Windows, Linux і macOS. PyCharm надає великий набір інструментів з коробки: вбудований відладчик і інструмент запуску тестів, профілювальник Python, повнофункціональний вбудований термінал, інструменти для роботи з базами даних. IDE інтегрована з популярними системами контролю версій, містить вбудований SSH-термінал, підтримує можливості віддаленої розробки та віддалені інтерпретатори, а також інтеграцію з Docker і Vagrant.

Якщо користувач використовує систему стеження за вадами, потрібно підключити PyCharm до баг-трекера, щоб працювати з завданнями прямо з IDE. Для цього потрібно просто вказати сервер баг-трекера.[6]

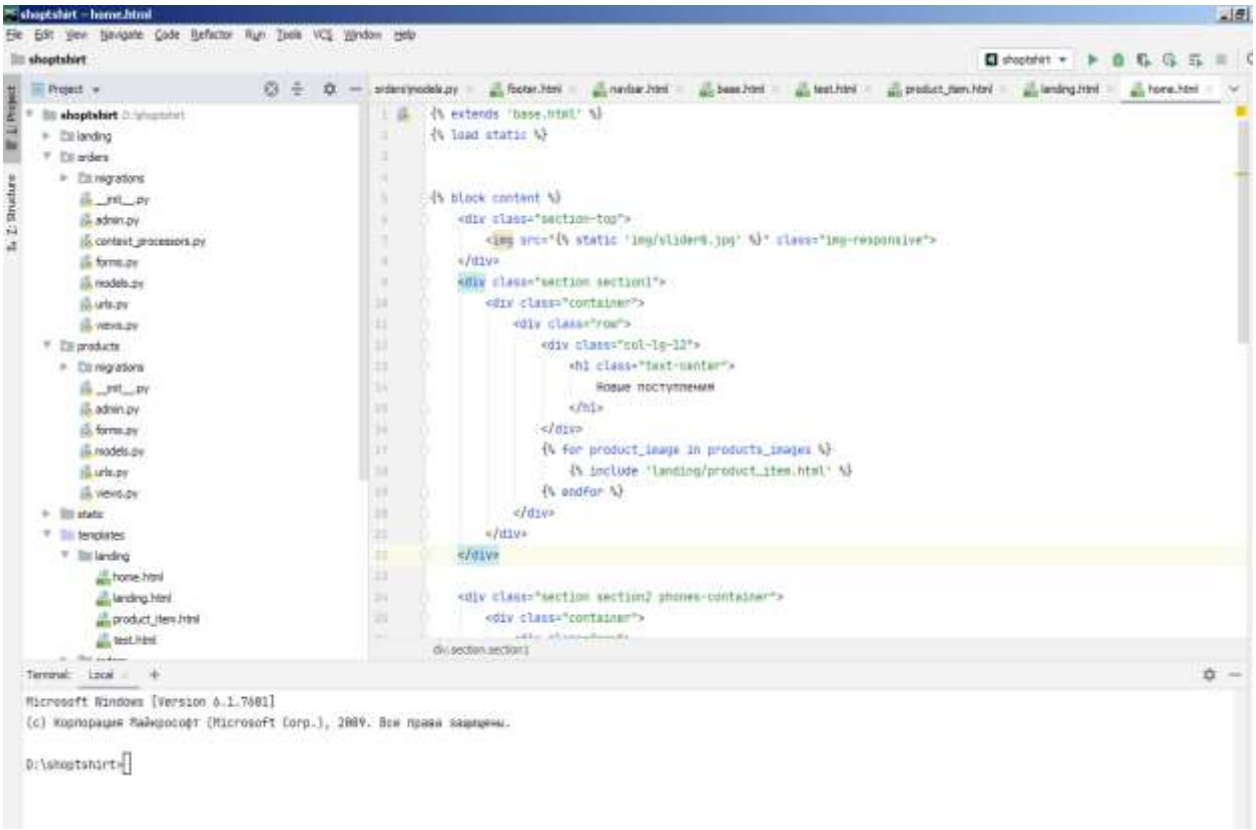


Рисунок 2.6 – Інтерфейс PyCharm

Переваги PyCharm:

- Максимальна продуктивність

PyCharm подбає про рутинні завдання, а ви зможете зосередитися на більш важливі речі. Працює в PyCharm, ви економите час - для більшості завдань не потрібно відривати руки від клавіатури.

- Допомога при написанні коду

PyCharm знає все про вашому коді. Розумний механізм аналізу коду забезпечує точне автодоповнення, пошук помилок і швидкі виправлення, зручну навігацію по коду та інші корисні функції.

- Підвищення якості коду

PyCharm допомагає писати красивий код, який легко підтримувати. IDE контролює якість коду за допомогою перевірок відповідності вимогам PEP8, розумних рефакторингов і безлічі інспекцій, а також надає допомогу при тестуванні.

- Все що потрібно

PyCharm створюється програмістами для програмістів, тому в ньому є все необхідне для продуктивної розробки на Python.

2.5 Вибір бази даних

Рисунок 2.7 – SQLite

На відміну від більшості інших баз даних SQL SQLite не має окремого серверного процесу. SQLite читає і пише безпосередньо в звичайний файл на диску. Повна база даних SQL з численними таблицями, індексами, тригерами

міститься в єдиному файлі на жорсткому диску. Формат файлу бази даних є міжплатформним. Можна вільно копіювати базу даних між 32- і 64-розрядних системами. Подібні особливості роблять SQLite популярним вибором в якості формату файлів додатків.

Ще один аргумент на користь SQLite - її компактність. З усіма включеними налаштуваннями, в залежності від параметрів оптимізації компілятора, розмір бібліотеки може становити менше 250 Кб. Якщо додаткові настройки не використовуються, розмір бібліотеки SQLite може бути скорочений до 180 Кб і трохи менше. Як стверджують розробники, SQLite може працювати при мінімальному стеку близько 16 Кб і дуже невеликий динамічної пам'яті - 100 Кб. Це дозволяє використовувати SQLite як популярний механізм бази даних в таких пристроях з обмеженнями пам'яті, як мобільні телефони, кишенькові персональні комп'ютери, MP3-програвачі і так далі. В даному випадку має місце співвідношення між використанням пам'яті і швидкістю. Чим більше відводиться пам'яті для SQLite, тим швидше вона працює. Проте робочі характеристики зазвичай і так досить високі навіть в середовищах з низьким обсягом пам'яті.

Як стверджується на офіційному сайті, SQLite має репутацію надійного механізму[8]. Близько двох третин вихідного коду відводиться на тестування і перевірку. Автоматизований тестує набір програм виконує сотні тисяч перевірок, що обробляють мільйони індивідуальних SQL-запитів, досягаючи при цьому більш ніж 99% охоплення. SQLite миттєво реагує на збої в розподілі пам'яті і дискові помилки введення-виведення. Все це перевірено автоматичними тестуваннями при використанні спеціального обладнання, яке імітує збої в роботі системи. Зрозуміло, навіть при таких випробуваннях все ще є певні помилки. Але на відміну від деяких інших подібних проектів SQLite відкрито і чесно повідомляє про свої дефектах в наданому списку помилок, що включає список критичних помилок з похвилинною хронологією повідомлень про помилки та зміни коду.

Базовий код SQLite підтримується міжнародною групою розробників, які постійно працюють з її застосуванням. Розробники продовжують розширювати можливості SQLite, покращувати її надійність і роботу, поки підтримується зворотна сумісність з виданими специфікаціями інтерфейсу, синтаксисом SQL і форматом файлу бази даних. Вихідний код абсолютно вільний для використання. При цьому також доступні і професійні послуги служби підтримки. Розробники сподіваються, що користувачі знайдуть SQLite корисним, і закликають використовувати його добре: створювати якісні і надійні програми, які будуть зручні і прості в застосуванні.

Недоліки SQLite:

- RIGHT і FULL OUTER JOIN. Реалізовано тільки LEFT OUTER JOIN
- Частково реалізований ALTER TABLE. Доступні тільки RENAME TABLE і ADD COLUMN
- Часткова підтримка Тригер. Доступний тільки FOR EACH ROW Тригер
- Запис у VIEWS. У SQLite VIEWS доступні тільки на читання. Частково обходиться через тригер
- В силу реалізації бази даних, як єдиного файлу і відходу від концепції «клієнт-сервер», не використовуються можливості GRANT і REVOKE
- За замовчуванням відключені зовнішні ключі. Це зроблено для забезпечення сумісності.

Переваги SQLite:

- Типізація

SQLite використовує динамічну типизацію даних. Можливі типи полів: NULL, INTEGER, REAL, TEXT, BLOB

- Надійність

Ситуація з покриттями тестами вихідного коду SQLite в деякому роді є легендою. Це пов'язано з тим, що тестами описано практично всі можливі (і

неможливі) ситуації. Коду, що описує тести, набагато більше, ніж коду SQLite. Природно, що покриття коду тестами 100%.

Сам же підхід до тестування використовується розробниками SQLite гідний наслідування. І може бути взятий як ідеал, до якого йдуть прагнути. [8]

Для того щоб запустити SQL-код необхідно виконати наступну команду: `sqlite3 ~ / example.sqlite`

Далі відкриється інтерактивний SQLite-shell, куди можна вводити команди.

Приклад як створити просту струкртуру каталогу товарів:

```sql
CREATE TABLE category (name TEXT NOT NULL);

INSERT INTO category (name) VALUES
    ('Тапки'),
    ('Самолёты'),
    ('Ноутбуки');


CREATE TABLE product (
    name TEXT NOT NULL,
    price NUMERIC NOT NULL,
    category REFERENCES category(rowid)
);

INSERT INTO product (name, price, category) VALUES
    ('Босоножки', 1.17, 1),
    ('Вьетнамки', 2.36, 1),
    ('Макасины', 4.99, 1),
    ('ИЛ-2', 556000, 2),
    ('Суперджет 100', 1500000, 2),
    ('Ту-160', 25000000, 2),
    ('Dell', 590, 3),
    ('Lenovo', 200, 3),
    ('Sony', 437, 3);
```

Рисунок 2.8 – Створення БД

Розглянемо як розширити існуючу структуру під нові вимоги Вимоги будуть наступними:

- Для категорії «Капці» необхідно додати поле «Розмір»
- Для категорії «Літаки» необхідно додати поле «Місткість».
- Для категорії «Ноутбуки» необхідно додати поле «процессорів».

```
1  ALTER TABLE product ADD size INTEGER;
2
3  ALTER TABLE product ADD capacity INTEGER;
4
5  ALTER TABLE product ADD processor TEXT;
```

Рисунок 2.9 – Розширення структури БД

Суттєвим недоліком даного підходу є той факт, що поля одних категорій товарів є обов'язковими для інших категорій. Тобто, наприклад, поле «розмір», яке додано для категорії «Капці», стає обов'язковим і для товарів з категорії «Літаки» і «Ноутбуки».

Класична реляційна модель не дозволяє зручно застосувати окремі поля для різних категорій. Можна, звичайно, використовувати поле типу BLOB і складати туди все, що душі заманеться, але тоді загубляться такі принади реляційних СУБД, як індекси, тригер (або вони дуже ускладняться) і фільтрація в запитах.

Можна ще, як варіант, залишити базові поля в одній таблиці, а решта винести в додаткові, зв'язавши їх зовнішніми ключами. Але це, знову ж таки, непотрібне ускладнення.

Тому пропоную залишити чудову реляційну базу SQLite в спокої. Вона чудово справляється з покладеними на неї обов'язками. Завдяки чому і є, мабуть, найпопулярнішою вбудовується базою даних в світі. [8]

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Проектування

На даному етапі будемо використовувати діаграми потоків даних (Data Flow Diagrams - DFD), що дозволяють подати ієрархію функціональних процесів, пов'язаних потоками даних. Мета такого подання - продемонструвати, яким чином кожен процес перетворює свої вхідні дані у вихідні, а також виявити взаємозв’язки між цими процесами.

Для побудови DFD традиційно використовуються дві різні нотації, які відповідають методам Йордона-ДеМарко і Гейне-Серсона. Ці нотації незначно відрізняються один від одного графічним зображенням символів.

При цьому модель системи визначається як ієрархія діаграм потоків даних, що описують асинхронний процес перетворення інформації від її введення в систему до видачі споживачеві. Джерела інформації (зовнішні сутності) породжують інформаційні потоки (потоки даних), які переносять інформацію до підсистем або процесів. Ті, в свою чергу, перетворюють інформацію і породжують нові потоки, які переносять інформацію до інших процесів або підсистем, накопичувачів даних або зовнішнім сутностей - споживачам інформації.

Діаграми верхніх рівнів ієрархії (контекстні діаграми) визначають основні процеси або підсистеми з зовнішніми входами і виходами. Вони деталізуються за допомогою діаграм нижнього рівня. Така декомпозиція триває, створюючи багаторівневу ієрархію діаграм, до тих пір, поки не буде досягнутий рівень декомпозиції, на якому деталізувати процеси далі втрачає сенс.

Склад діаграм потоків даних

Основними компонентами діаграм потоків даних є:

- зовнішні сутності;
- системи і підсистеми;

- процеси;
- накопичувачі даних;
- потоки даних.

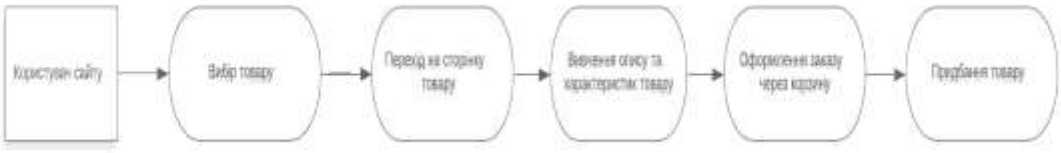


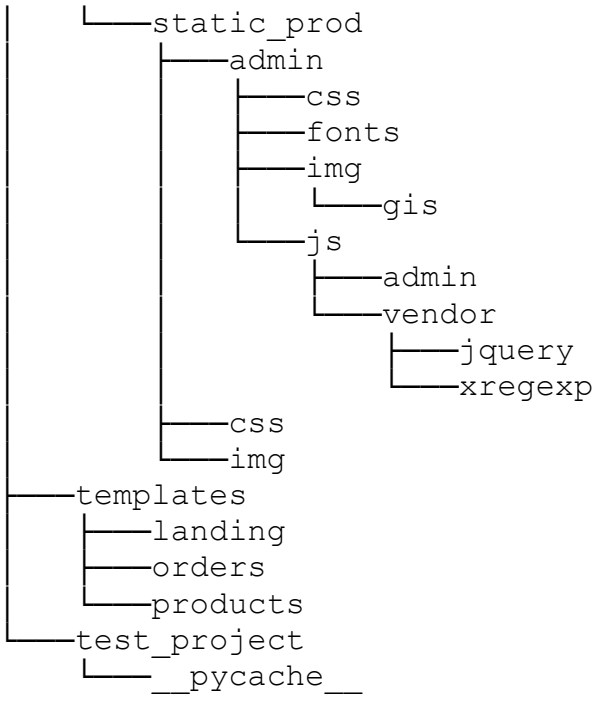
Рисунок 3.1 –DFD діаграмма

На DFD діаграммі, рисунок 3.1, можна бачити процес замовлення товару

3.2 Програмна реалізація

Створюємо структуру файлів інтернет магазина

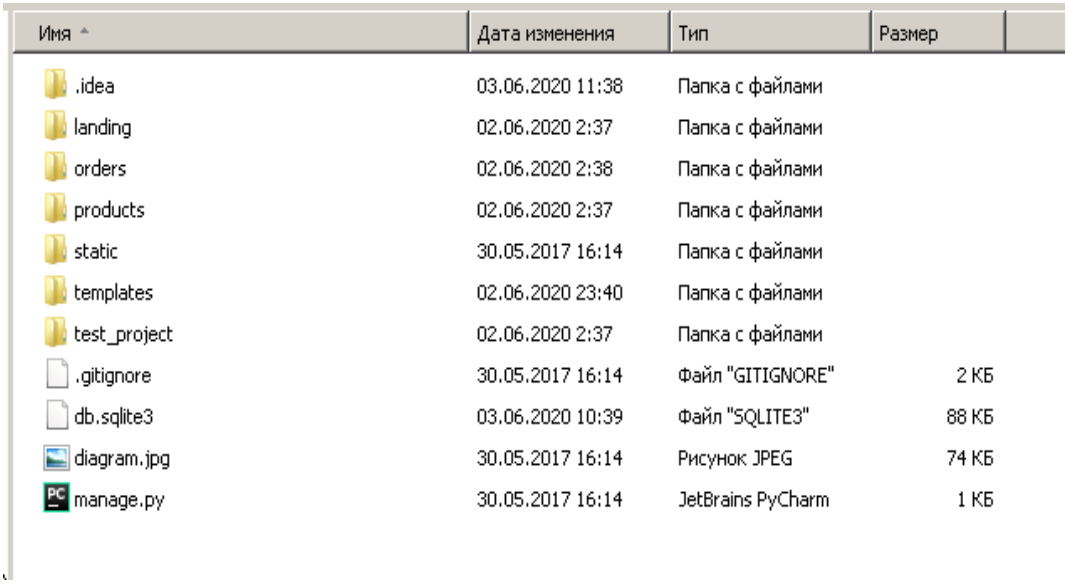
```
├───.idea
│   ├───dataSources
│   └───inspectionProfiles
├───landing
│   ├───migrations
│   │   └───__pycache__
│   └───__pycache__
├───orders
│   ├───migrations
│   │   └───__pycache__
│   └───__pycache__
├───products
│   ├───migrations
│   │   └───__pycache__
│   └───__pycache__
├───static
│   ├───media
│   │   └───products_images
│   ├───static_dev
│   │   ├───css
│   │   ├───img
│   │   └───js
```



Проект складається з:

- папки landing
- папки orders,
- папки products ,
- папки static ,
- папки templares ,
- папки test_projct
- файл базы данных db.sqlite3
- файлу manage.py

Вся файлова структура проекту зображена на рис.3.2



Имя	Дата изменения	Тип	Размер
.idea	03.06.2020 11:38	Папка с файлами	
landing	02.06.2020 2:37	Папка с файлами	
orders	02.06.2020 2:38	Папка с файлами	
products	02.06.2020 2:37	Папка с файлами	
static	30.05.2017 16:14	Папка с файлами	
templates	02.06.2020 23:40	Папка с файлами	
test_project	02.06.2020 2:37	Папка с файлами	
.gitignore	30.05.2017 16:14	Файл "GITIGNORE"	2 КБ
db.sqlite3	03.06.2020 10:39	Файл "SQLITE3"	88 КБ
diagram.jpg	30.05.2017 16:14	Рисунок JPEG	74 КБ
manage.py	30.05.2017 16:14	JetBrains PyCharm	1 КБ

Рисунок 3.2 – Структура файлів проекту

Опис структури деяких файлів:

- settings.py містить в собі всі настройки проекту. Тут ми реєструємо додатки, задаємо розміщення статичних файлів, налаштування бази даних і так далі.
- urls.py задає асоціації url адрес з уявленнями.
- wsgi.py використовується для налагодження зв'язку між вашим Django додатком і веб-сервером.
- Файл manage.py використовується для створення програм, роботи з базами даних.
- В папці orders знаходяться файли для конфігурації вікна заказів та доставки. В папці products знаходяться файли для конфігурації вікна товарів. А в products-images зображення всіх товарів. В static розташовуються файли стилів сайту, скрипти, зображення тошо. В templates розташовуються макети сторінок сайту

База Данних

Рисунок 3.3 – Таблиці бази данних sqlite3

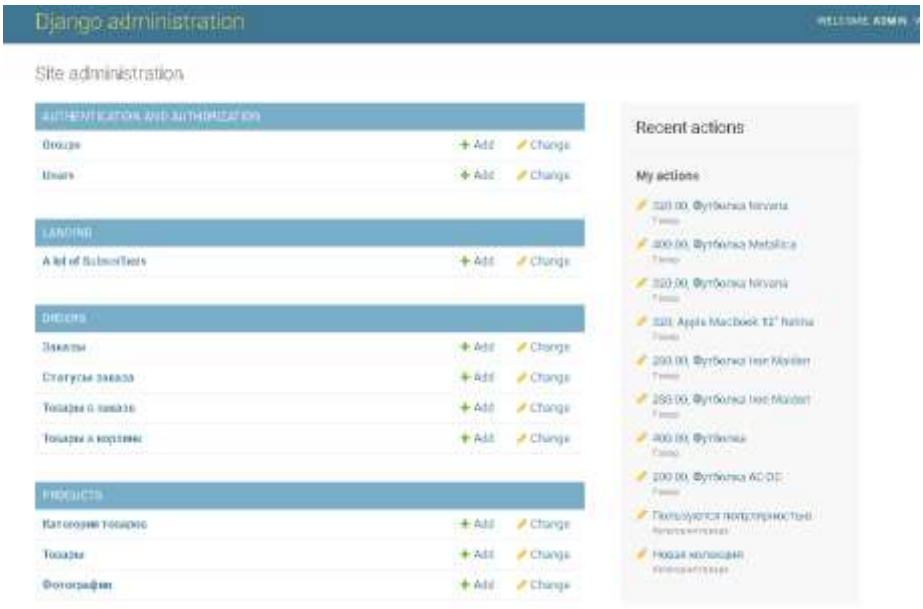


Рисунок 3.4 – Зовнішній вигляд адмін-панелі Django

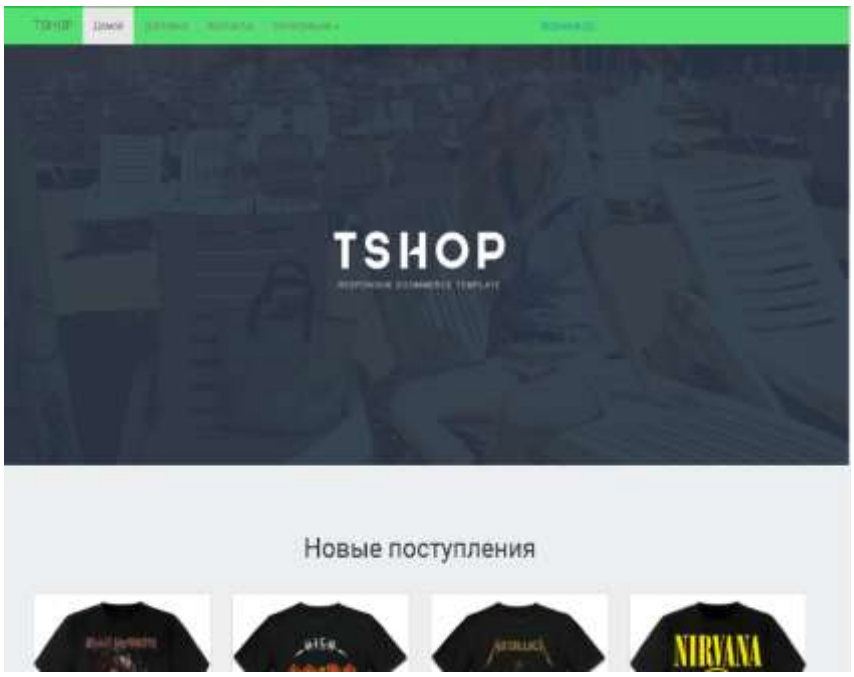


Рисунок 3.5 – Зовнішній вигляд верхньої частини головної сторінки

Контентом головної сторінки є баннер, який зображений на рисунку 3.5, з усіма товарами. Під яким знаходяться різні категорії товарів. На товарі є можливість або відкрити повну сторінку товару та кнопка добавити в корзину, також під фото є короткий опис товару. Зовнішній вигляд картки відображений на рисунку 3.6.

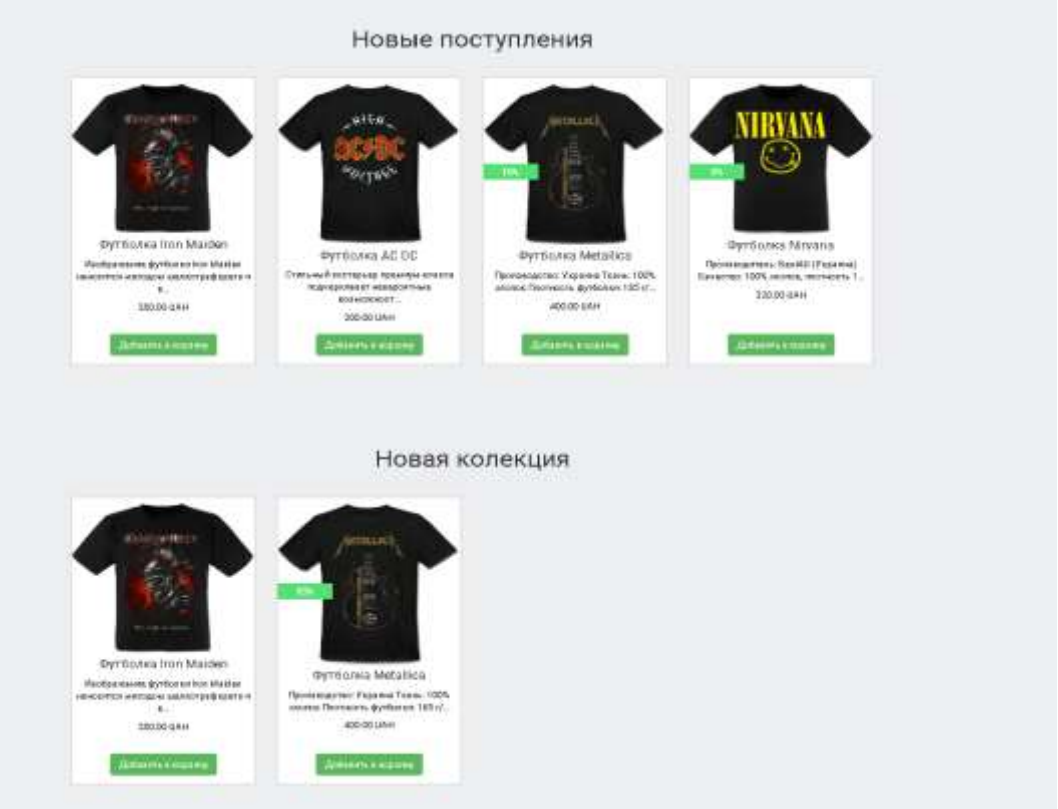


Рисунок 3.6 – Зовнішній вигляд нижньої частини головної сторінки

Рисунок 3.7 – Зовнішній вигляд картки товару

Також на картці товару використовується кнопка для того щоб було можна відразу добавити товар в корзину з головної сторінки.

Рисунок 3.8 – Зовнішній вигляд сторінки товару

На сторінці товару знаходится поле в яке потрібно написати кількість необхідного товару для купівлі. При натисканні кнопки “Купить” вся кількість товарів добавится в корзину

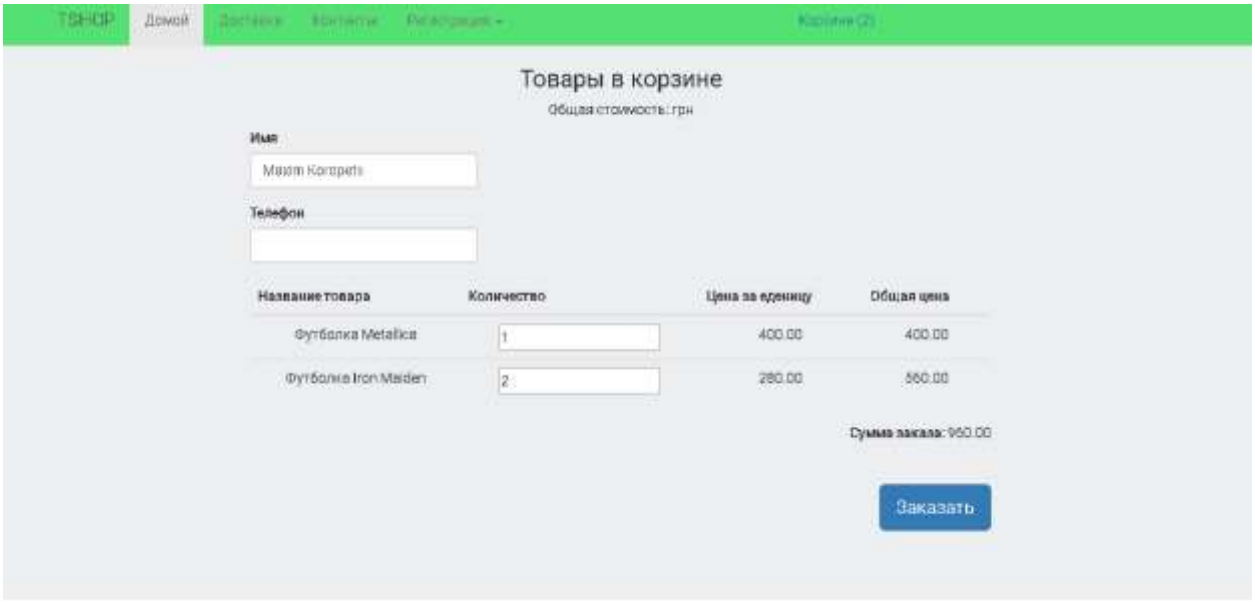


Рисунок 3.10 – Зовнішній вигляд сторінки корзини

Тут відображаются товари та їх кількість які були раніше добавлені сюди через картку товару з головної сторінки або через сторінку товару

ВИСНОВКИ

Робота виконана на фреймоврку Django. Це прогресивний фреймворк, який використовується для створення користувацьких інтерфейсів.

Після перегляду усіх популярних популярних фреймворків на мові Python було вирішено використати для реалізації завдання саме Django, а як базу даних використовувати SqLite.

Даний проект являє собою інтернет магазин на якому користувач може переглянути та купити будь який товар.

Кінцевий готовий проект є якісним продуктом який не містить зайвої анімації і тому він простий для різних браузерів та доволі швидко завантажується на різних швидкостях інтернету завдяки використанню сучасної мови Python, простого дизайну та нових технологій у сфері веб розробки.

СПИСОК ЛІТЕРАТУРИ

1. Керівництво по Python. – https://python-scripts.com/
2. Python-фреймворки: тенденції 2020 року. –waksoft.susu.ru/2020/01/14/top-5-frejmvorkov-python-dlya-web-razrabotki-v-2020-godu
3. Огляд Django. – https://djbook.ru/
4. Документація Django. – https://tproger.ru/translations/create-your-first-django-app
5. Документація Django. – https://habr.com/ru/company/SECL_GROUP/blog/218921/
6. ТОП 10 кращих фреймворків для Python.– https://django.fun/
7. Документація Django .– https://vuetifyjs.com/ru/components/api-explorer/
8. Документація SqLite.– https://proglib.io/p/sqlite-tutorial
9. Інструментарій для розробки на Dgango .– https://djbook.ru/
10. Лутц, М. Програмування на Python. Т. 1 / М. Лутц. - М.: Символ, 2016. - 992 с
11. Лутц М. Програмування на Python. Т. 2 / М. Лутц. - М.: Символ, 2016. - 876 с.
12. Метиз Е. Вивчаємо PYTHON. Програмування програм, візуалізація данных, веб-додатки / Е. Метиз. - СПб.: Москва, 2017. - 496 с.
13. Дронов В. Django 2.1. Практика створення веб-сайтів на Python/ В. Дронов. – БХВ-Петербург,2016. – 672 с.
14. Pinkham A. Django Unleashed / Pinkham A. - Sams, 2012.– 700 с.
15. Форс Д. Django: Розробка веб-додатків на Python / Д. Форс, П. Біссекс, У. Чан. – Символ-плюс, 2010.– 267 с.

ДОДАТОК

Програмний код проекту

Папка `landing`:

Файл `admin.py`

```
from django.contrib import admin
from .models import *


class SubscriberAdmin (admin.ModelAdmin):
    # list_display = ["name", "email"]
    list_display = [field.name for field in Subscriber._meta.fields]
    list_filter = ['name',]
    search_fields = ['name', 'email']

    fields = ["email"]

    # exclude = ["email"]
   # inlines = [FieldMappingInline]
   # fields = []
    # #exclude = ["type"]
   # #list_filter = ('report_data',)
   # search_fields = ['category', 'subCategory', 'suggestKeyword']

    class Meta:
        model = Subscriber

admin.site.register(Subscriber, SubscriberAdmin)
```

Файл `forms.py`

```
from django import forms
from .models import *


class SubscriberForm(forms.ModelForm):

    class Meta:
        model = Subscriber
        exclude = [""]
```

Файл `Models.py`

```
from django.db import models


class Subscriber(models.Model):
    email = models.EmailField()
    name = models.CharField(max_length=128)
```

```
    def __str__(self):
        return "Пользователь %s %s" % (self.name, self.email,)

    class Meta:
        verbose_name = 'MySubscriber'
        verbose_name_plural = 'A lot of Subscribers'
```

Файл `urls.py`

```
from django.conf.urls import url, include
from django.contrib import admin
from landing import views

urlpatterns = [
    url(r'^$', views.home, name='home'),
    url(r'^landing123/$', views.landing, name='landing'),
]
```

Файл `vievs.py`

```
from django.shortcuts import render
from .forms import SubscriberForm
from products.models import *


def landing(request):
    name = "CodingMedved"
    current_day = "03.01.2020"
    form = SubscriberForm(request.POST or None)

    if request.method == "POST" and form.is_valid():
        print (request.POST)
        print (form.cleaned_data)
        data  = form.cleaned_data
        print (data["name"])

        new_form = form.save()

    return render(request, 'landing/landing.html', locals
    ())


def home(request):
    products_images = ProductImage.objects.filter(
    is_active=True, is_main=True, product__is_active=True)
    products_images_phones = products_images.filter(
    product__category__id=1)
    products_images_laptops = products_images.filter(
    product__category__id=2)
    return render(request, 'landing/home.html', locals())
```

Папка `Orders`

Файл `admin.py`

```
from django.contrib import admin
from .models import *


class ProductInOrderInline(admin.TabularInline):
    model = ProductInOrder
    extra = 0


class StatusAdmin (admin.ModelAdmin):
    list_display = [field.name for field in Status._meta
    .fields]

    class Meta:
        model = Status

admin.site.register(Status, StatusAdmin)
class OrderAdmin (admin.ModelAdmin):
    list_display = [field.name for field in Order._meta.
    fields]
    inlines = [ProductInOrderInline]

    class Meta:
        model = Order
```

Файл `context-processor.py`

```
from .models import ProductInBasket


def getting_basket_info(request):

    session_key = request.session.session_key
    if not session_key:
        #workaround for newer Django versions
        request.session["session_key"] = 123
        #re-apply value
        request.session.cycle_key()

    products_in_basket = ProductInBasket.objects.filter(
    session_key=session_key, is_active=True,
    order__isnull=True)
    products_total_nmb = products_in_basket.count()

    return locals()
```

Файл `forms.py`

```
from django import forms
from .models import *
class CheckoutContactForm(forms.Form):
    name = forms.CharField(required=True)
    phone = forms.CharField(required=True)
```

Файл `models.py`

```
from django.db import models
from products.models import Product
from django.db.models.signals import post_save
from django.contrib.auth.models import User


class Status(models.Model):
    name = models.CharField(max_length=24, blank=True, null=True, default=None)
    is_active = models.BooleanField(default=True)
    created = models.DateTimeField(auto_now_add=True, auto_now=False)
    updated = models.DateTimeField(auto_now_add=False, auto_now=True)

    def __str__(self):
        return "Статус %s" % self.name

    class Meta:
        verbose_name = 'Статус заказа'
        verbose_name_plural = 'Статусы заказа'


class Order(models.Model):
    user = models.ForeignKey(User, blank=True, null=True, default=None, on_delete=
    models.DO_NOTHING)
    total_price = models.DecimalField(max_digits=10, decimal_places=2, default=0)
    #total price for all products in order
    customer_name = models.CharField(max_length=64, blank=True, null=True, default
    =None)
    customer_email = models.EmailField(blank=True, null=True, default=None)
    customer_phone = models.CharField(max_length=48, blank=True, null=True,
    default=None)
    customer_address = models.CharField(max_length=128, blank=True, null=True,
    default=None)
    comments = models.TextField(blank=True, null=True, default=None)
    status = models.ForeignKey(Status, on_delete=models.DO_NOTHING)
    created = models.DateTimeField(auto_now_add=True, auto_now=False)
    updated = models.DateTimeField(auto_now_add=False, auto_now=True)

    def __str__(self):
        return "Заказ %s %s" % (self.id, self.status.name)

    class Meta:
        verbose_name = 'Заказ'
        verbose_name_plural = 'Заказы'

    def save(self, *args, **kwargs):

        super(Order, self).save(*args, **kwargs)


class ProductInOrder(models.Model):
    order = models.ForeignKey(Order, blank=True, null=True, default=None,
    on_delete=models.DO_NOTHING)
    product = models.ForeignKey(Product, blank=True, null=True, default=None,
    on_delete=models.DO_NOTHING)
    nmb = models.IntegerField(default=1)
    price_per_item = models.DecimalField(max_digits=10, decimal_places=2, default=
    0)
    total_price = models.DecimalField(max_digits=10, decimal_places=2, default=0)
```

```
    #price*nmb
    is_active = models.BooleanField(default=True)
    created = models.DateTimeField(auto_now_add=True, auto_now=False)
    updated = models.DateTimeField(auto_now_add=False, auto_now=True)

    def __str__(self):
        return "%s" % self.product.name

    class Meta:
        verbose_name = 'Товар в заказе'
        verbose_name_plural = 'Товары в заказе'


    def save(self, *args, **kwargs):
        price_per_item = self.product.price
        self.price_per_item = price_per_item
        print (self.nmb)

        self.total_price = int(self.nmb) * price_per_item

        super(ProductInOrder, self).save(*args, **kwargs)



def product_in_order_post_save(sender, instance, created, **kwargs):
    order = instance.order
    all_products_in_order = ProductInOrder.objects.filter(order=order, is_active=
    True)

    order_total_price = 0
    for item in all_products_in_order:
        order_total_price += item.total_price

    instance.order.total_price = order_total_price
    instance.order.save(force_update=True)


post_save.connect(product_in_order_post_save, sender=ProductInOrder)


class ProductInBasket(models.Model):
    session_key = models.CharField(max_length=128, blank=True, null=True, default=
    None)
    order = models.ForeignKey(Order, blank=True, null=True, default=None,
    on_delete=models.DO_NOTHING)
    product = models.ForeignKey(Product, blank=True, null=True, default=None,
    on_delete=models.DO_NOTHING)
    nmb = models.IntegerField(default=1)
    price_per_item = models.DecimalField(max_digits=10, decimal_places=2, default=
    0)
    total_price = models.DecimalField(max_digits=10, decimal_places=2, default=0)
    #price*nmb
    is_active = models.BooleanField(default=True)
    created = models.DateTimeField(auto_now_add=True, auto_now=False)
    updated = models.DateTimeField(auto_now_add=False, auto_now=True)

    def __str__(self):
        return "%s" % self.product.name
```

```
    def save(self, *args, **kwargs):
        price_per_item = self.product.price
        self.price_per_item = price_per_item
        self.total_price = int(self.nmb) * price_per_item

        super(ProductInBasket, self).save(*args, **kwargs)
```

Файл `urls.py`

```
from django.conf.urls import url, include
from django.contrib import admin
from . import views

urlpatterns = [

    url(r'^basket_adding/$', views.basket_adding, name='basket_adding'),
    url(r'^checkout/$', views.checkout, name='checkout'),

]
```

Файл `views.py`

```
from django.http import JsonResponse, HttpResponse, HttpResponseRedirect
from .models import *
from django.shortcuts import render
from .forms import CheckoutContactForm
from django.contrib.auth.models import User


def basket_adding(request):
    return_dict = dict()
    session_key = request.session.session_key
    print (request.POST)
    data = request.POST
    product_id = data.get("product_id")
    nmb = data.get("nmb")
    is_delete = data.get("is_delete")

    if is_delete == 'true':
        ProductInBasket.objects.filter(id=product_id).update(is_active=False)
    else:
        new_product, created = ProductInBasket.objects.get_or_create(session_key
        session_key, product_id=product_id,
                                                                     is_active=
True, defaults={"nmb": nmb})
        if not created:
            print ("not created")
            new_product.nmb += int(nmb)
            new_product.save(force_update=True)

    #common code for 2 cases
    products_in_basket = ProductInBasket.objects.filter(session_key=session_key,
    is_active=True, order__isnull=True)
    products_total_nmb = products_in_basket.count()
    return_dict["products_total_nmb"] = products_total_nmb

    return_dict["products"] = list()

    for item in  products_in_basket:
        product_dict = dict()
        product_dict["id"] = item.id
        product_dict["name"] = item.product.name
        product_dict["price_per_item"] = item.price_per_item
        product_dict["nmb"] = item.nmb
```

```
        return_dict["products"].append(product_dict)

    return JsonResponse(return_dict)


def checkout(request):
    session_key = request.session.session_key
    products_in_basket = ProductInBasket.objects.filter(session_key=session_key,
    is_active=True, order__isnull=True)
    print (products_in_basket)
    for item in products_in_basket:
        print(item.order)
    form = CheckoutContactForm(request.POST or None)
    if request.POST:
        print(request.POST)
        if form.is_valid():
            print("yes")
            data = request.POST
            name = data.get("name", "3423453")
            phone = data["phone"]
            user, created = User.objects.get_or_create(username=phone, defaults={
            "first_name": name})

            order = Order.objects.create(user=user, customer_name=name,
            customer_phone=phone, status_id=1)

            for name, value in data.items():
                if name.startswith("product_in_basket_"):
                    product_in_basket_id = name.split("product_in_basket_")[1]
                    product_in_basket = ProductInBasket.objects.get(id=
                    product_in_basket_id)
                    print(type(value))

                    product_in_basket.nmb = value
                    product_in_basket.order = order
                    product_in_basket.save(force_update=True)

                    ProductInOrder.objects.create(product=product_in_basket.
                    product, nmb = product_in_basket.nmb,
                                                  price_per_item=product_in_basket
                                                  .price_per_item,
                                                  total_price = product_in_basket.
                                                  total_price,
                                                  order=order)

            return HttpResponseRedirect(request.META['HTTP_REFERER'])
        else:
            print("no")
    return render(request, 'orders/checkout.html', locals())
```

Папка `Products`

Файл `Admin.py`

```
from django.contrib import admin
from .models import *


class ProductImageInline(admin.TabularInline):
    model = ProductImage
    extra = 0
```

```
class ProductCategoryAdmin(admin.ModelAdmin):
    list_display = [field.name for field in ProductCategory._meta.fields]

    class Meta:
        model = ProductCategory

admin.site.register(ProductCategory, ProductCategoryAdmin)


class ProductAdmin (admin.ModelAdmin):
    list_display = [field.name for field in Product._meta.fields]
    inlines = [ProductImageInline]

    class Meta:
        model = Product

admin.site.register(Product, ProductAdmin)
```

Файл `models.py`

```
from django.db import models


class ProductCategory(models.Model):
    name = models.CharField(max_length=64, blank=True, null=True, default=None)
    is_active = models.BooleanField(default=True)

    def __str__(self):
        return "%s" % self.name

    class Meta:
        verbose_name = 'Категория товара'
        verbose_name_plural = 'Категория товаров'


class Product(models.Model):
    name = models.CharField(max_length=64, blank=True, null=True, default=None)
    price = models.DecimalField(max_digits=10, decimal_places=2, default=0)
    discount = models.IntegerField(default=0)
    category = models.ForeignKey(ProductCategory, blank=True, null=True, default=
    None, on_delete=models.DO_NOTHING)
    short_description = models.TextField(blank=True, null=True, default=None)
    description = models.TextField(blank=True, null=True, default=None)
    is_active = models.BooleanField(default=True)
    created = models.DateTimeField(auto_now_add=True, auto_now=False)
    updated = models.DateTimeField(auto_now_add=False, auto_now=True)

    def __str__(self):
        return "%s, %s" % (self.price, self.name)

    class Meta:
        verbose_name = 'Товар'
        verbose_name_plural = 'Товары'


class ProductImage(models.Model):
    product = models.ForeignKey(Product, blank=True, null=True, default=None,
    on_delete=models.DO_NOTHING)
    image = models.ImageField(upload_to='products_images/')
    is_main = models.BooleanField(default=False)
    is_active = models.BooleanField(default=True)
    created = models.DateTimeField(auto_now_add=True, auto_now=False)
```

```
    updated = models.DateTimeField(auto_now_add=False, auto_now=True)

    def __str__(self):
        return "%s" % self.id

    class Meta:
        verbose_name = 'Фотография'
        verbose_name_plural = 'Фотографии'
```

Файл `vievs.py`

```
from django.shortcuts import render
from products.models import *


def product(request, product_id):
    product = Product.objects.get(id=product_id)


    session_key = request.session.session_key
    if not session_key:
        request.session.cycle_key()

    print(request.session.session_key)
 return render(request, 'products/product.html', locals())
```

Папка `test_project`

Файл `settings.py`

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

    'landing',
    'products',
    'orders',
]


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',
]

ROOT_URLCONF = 'test_project.urls'
```

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')]
        ,
        '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',

                'orders.context_processors.getting_basket_info',
            ],
        },
    },
]

WSGI_APPLICATION = 'test_project.wsgi.application'


# Database
# https://docs.djangoproject.com/en/1.10/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
# Password validation
# https://docs.djangoproject.com/en/1.10/ref/settings/#auth-password-validators

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',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/1.10/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True
```

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.10/howto/static-files/

STATIC_URL = '/static/'

STATICFILES_DIRS = (
  os.path.join(BASE_DIR, "static", "static_dev"),
)


STATIC_ROOT = os.path.join(BASE_DIR, "static", "static_prod")

MEDIA_URL = '/media/'

MEDIA ROOT = os.path.join(BASE DIR, "static", "media")
```

Файл `urls.py`

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


urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^', include('landing.urls')),
    url(r'^', include('products.urls')),
    url(r'^', include('orders.urls')),
] \
            + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT) \
            + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Файл `manage.py`

```
import os
import sys

if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "test_project.settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError:
        # The above import may fail for some other reason. Ensure that the
        # issue is really that Django is missing to avoid masking other
        # exceptions on Python 2.
        try:
            import django
        except ImportError:
            raise ImportError(
                "Couldn't import Django. Are you sure it's installed and "
                "available on your PYTHONPATH environment variable? Did you "
                "forget to activate a virtual environment?"
            )
        raise
    execute_from_command_line(sys.argv)
```

Папка `Templars`

Файл `home.html`

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


{% block content %}
    <div class="section-top">
        <img src="{% static 'img/slider0.jpg' %}" class="img-responsive">
    </div>
    <div class="section section1">
        <div class="container">
            <div class="row">
                <div class="col-lg-12">
                    <h1 class="text-center">
                        Новые поступления
                    </h1>
                </div>
                {% for product_image in products_images %}
                    {% include 'landing/product_item.html' %}
                {% endfor %}
            </div>
        </div>
    </div>

    <div class="section section2 phones-container">
        <div class="container">
            <div class="row">
                <div class="col-lg-12">
                    <h1 class="text-center">
                        Новая колекция
                    </h1>
                </div>
                {% for product_image in products_images_phones %}
                    {% include 'landing/product_item.html' %}
                {% endfor %}

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

    <div class="section section3 laptops-container">
        <div class="container">
            <div class="row">
                <div class="col-lg-12">
                    <h1 class="text-center">
                        Пользуются популярностью
                    </h1>
                </div>
                {% for product_image in products_images_laptops %}
                    {% include 'landing/product_item.html' %}
                {% endfor %}

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

Файл `product-item.html`

```
        <div>
            <a href="{% url 'product' product_image.product.id %}">
                <img src="{{ product_image.image.url }}" class="img-responsive">
            </a>
        </div>

        {% if product_image.product.discount %}
            <div class="discount-container">
                <span>{{ product_image.product.discount }}%</span>
            </div>
        {% endif %}

        <h4>{{ product_image.product.name }}</h4>
        <p>
            {{ product_image.product.description|truncatechars_html:70 }}
        </p>
        <div>
            {{ product_image.product.price }} UAH
        </div>
        <div class="add-to-card-btn">
            <button class="btn btn-success">
                Добавить в корзину
            </button>
        </div>
    </div>
</div>

        <![endif]-->

        <!-- Favicon and touch icons -->
        <link rel="shortcut icon" href="assets/ico/favicon.png">
        <link rel="apple-touch-icon-precomposed" sizes="144x144" href=
        "assets/ico/apple-touch-icon-144-precomposed.png">
        <link rel="apple-touch-icon-precomposed" sizes="114x114" href=
        "assets/ico/apple-touch-icon-114-precomposed.png">
        <link rel="apple-touch-icon-precomposed" sizes="72x72" href=
        "assets/ico/apple-touch-icon-72-precomposed.png">
        <link rel="apple-touch-icon-precomposed" href=
        "assets/ico/apple-touch-icon-57-precomposed.png">

    </head>
    <body>

        <!-- Loader -->
       <div class="loader">
          <div class="loader-img"></div>
       </div>

        <!-- Top content -->
        <div class="top-content">

          <!-- Top menu -->
         <nav class="navbar navbar-inverse navbar-no-bg" role="navigation">
            <div class="container">
               <div class="navbar-header">
                  <button type="button" class="navbar-toggle collapsed" data-
                  toggle="collapse" data-target="#top-navbar-1">
                     <span class="sr-only">Toggle navigation</span>
                     <span class="icon-bar"></span>
                     <span class="icon-bar"></span>
                     <span class="icon-bar"></span>
```

```
            </button>
            <a class="navbar-brand" href="index.html">Marco - Bootstrap
            Landing Page</a>
         </div>
         <!-- Collect the nav links, forms, and other content for toggling
         -->
         <div class="collapse navbar-collapse" id="top-navbar-1">
            <ul class="nav navbar-nav navbar-right">
               <li><a class="scroll-link" href="#features">Features</a></li>
               <li><a class="scroll-link" href="#testimonials">Testimonials
               </a></li>
               <li><a class="scroll-link" href="#pricing">Price</a></li>
               <li><a class="scroll-link" href="#how-it-works">How it works
               </a></li>
               <li><a class="scroll-link" href="#about-us">About</a></li>
               <li><a class="scroll-link" href="#contact">Contact</a></li>
            </ul>
         </div>
      </div>
   </nav>

   <div class="inner-bg">
       <div class="container">

          <div class="row">
              <div class="col-sm-8 col-sm-offset-2 text">
                  <h1 class="wow fadeInLeftBig">Begin Using Our Product
                  Today</h1>
                  <div class="description wow fadeInLeftBig">
                     <p>
                        Subscribe by entering your email address in the
                        form below and we'll send you a link to start
                        right away.
                        Working MailChimp form!
                     </p>
                  </div>
                  <div class="subscribe wow fadeInUp">
                  <form class="form-inline" role="form" action=
                  "assets/subscribe.php" method="post">
                      <div class="form-group">
                        <label class="sr-only" for="subscribe-email">
                        Email address</label>
                         <input type="text" name="email" placeholder=
                         "Enter your email..." class="subscribe-email
                         form-control" id="subscribe-email">
                      </div>
                      <button type="submit" class="btn">Subscribe</
                      button>
                  </form>
                  <div class="success-message"></div>
                  </div>
              </div>
          </div>

          <div class="row">
            <div class="col-sm-8 col-sm-offset-2 top-video-link
            medium-paragraph wow fadeInUp">
               <a href="#" class="launch-modal" data-modal-id=
               "modal-how-it-works">
                  <span class="top-video-link-icon"><i class="fa
                  fa-play"></i></span>
                 <span class="top-video-link-text">See how it works</
                 span>
```

```
                    </a>
                </div>
            </div>

            <div class="row">
                <div class="col-sm-12 top-social wow fadeInUp">
                    <a href="#"><i class="fa fa-facebook"></i></a> <span
                    class="divider-2"></span>
                <a href="#"><i class="fa fa-twitter"></i></a> <span class=
                "divider-2"></span>
                <a href="#"><i class="fa fa-google-plus"></i></a> <span class
                ="divider-2"></span>
                <a href="#"><i class="fa fa-instagram"></i></a> <span class=
                "divider-2"></span>
                <a href="#"><i class="fa fa-pinterest"></i></a>
                </div>
            </div>

        </div>
    </div>

</div>

<!-- Features -->
<div class="features-container section-container">
    <div class="container">
        <div class="row">
            <div class="col-sm-12 features section-description wow fadeIn">
                <h2>Features</h2>
                <div class="divider-1 wow fadeInUp"><span></span></div>
                <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit
                , sed do eiusmod tempor incididunt ut labore et.
                    Ut wisi enim ad minim veniam, quis nostrud tempor
                    incididunt ut labore et.</p>
            </div>
        </div>
        <div class="row">
            <div class="col-sm-4 features-box wow fadeInUp">
                <div class="features-box-icon"><i class="fa fa-eye"></i></
                div>
                <h3>Easy To Use</h3>
                <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit
                , sed do eiusmod tempor incididunt ut labore et.</p>
            </div>
            <div class="col-sm-4 features-box wow fadeInDown">
                <div class="features-box-icon"><i class="fa fa-thumbs-up"></
                i></div>
                <h3>Responsive Design</h3>
                <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit
                , sed do eiusmod tempor incididunt ut labore et.</p>
            </div>
            <div class="col-sm-4 features-box wow fadeInUp">
                <div class="features-box-icon"><i class="fa fa-cog"></i></
                div>
                <h3>Bootstrap Engine</h3>
                <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit
                , sed do eiusmod tempor incididunt ut labore et.</p>
            </div>
        </div>
        <div class="row">
            <div class="col-sm-4 features-box wow fadeInUp">
                <div class="features-box-icon"><i class="fa fa-play"></i></
                div>
```

```
                    <h3>Lots Of Videos</h3>
                    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit
                    , sed do eiusmod tempor incididunt ut labore et.</p>
                </div>
                <div class="col-sm-4 features-box wow fadeInDown">
                    <div class="features-box-icon"><i class="fa fa-phone"></i></
                    div>
                    <h3>Mobile App</h3>
                    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit
                    , sed do eiusmod tempor incididunt ut labore et.</p>
                </div>
                <div class="col-sm-4 features-box wow fadeInUp">
                    <div class="features-box-icon"><i class="fa fa-commenting"
                    ></i></div>
                    <h3>Big Community</h3>
                    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit
                    , sed do eiusmod tempor incididunt ut labore et.</p>
                </div>
            </div>
            <div class="row">
                <div class="col-sm-12 section-bottom-button wow fadeInUp">
                    <a class="btn btn-link-1 scroll-link" href="#pricing">
                    Plans &amp; Pricing</a>
                </div>
            </div>
        </div>
    </div>

<!-- More features 2 -->
    <div class="more-features-2-container section-container
    section-container-gray-bg">
        <div class="container">
            <div class="row">
                <div class="col-sm-7 more-features-2-box wow fadeInLeft">
                    <img src="assets/img/devices/ipad.png" alt="">
                </div>
                <div class="col-sm-5 more-features-2-box
                more-features-2-box-right wow fadeInUp">
                    <h3>We have more features for you:</h3>
                    <ul>
                        <li><i class="fa fa-check-circle"></i> 15+ Layouts</li>
                        <li><i class="fa fa-check-circle"></i> 30+ Forms (
                        registration, login, contact &amp; subscription forms)</
                        li>
                        <li><i class="fa fa-check-circle"></i> Responsive Layout
                        </li>
                        <li><i class="fa fa-check-circle"></i> Bootstrap
                        Framework</li>
                        <li><i class="fa fa-check-circle"></i> MailChimp
                        Subscription Form</li>
                    </ul>
                    <a class="btn btn-link-1 scroll-link" href="#pricing">See
                    Pricing</a>
                </div>
            </div>
        </div>
    </div>
```

```
<!-- Many options -->
<div class="many-options-container section-container">
   <div class="container">
      <div class="row">
         <div class="col-sm-5 many-options-box wow fadeInLeft">
            <h3>Many options to choose from:</h3>
            <p>
               Lorem ipsum dolor sit amet, consectetur adipisicing elit
               , sed do eiusmod tempor incididunt ut labore et.
               Ut wisi enim ad minim veniam, quis nostrud.
            </p>
            <p>
               Exerci tation ullamcorper suscipit lobortis nisl ut
               aliquip ex ea commodo consequat.
               Ut wisi enim ad minim veniam, quis nostrud exerci tation
                ullamcorper suscipit lobortis nisl.
            </p>
            <p>
               <a class="learn-more scroll-link" href="#features">Learn
                More</a>
            </p>
         </div>
         <div class="col-sm-7 many-options-box many-options-box-right
         wow fadeInUp">
            <img src="assets/img/devices/ipad2.png" alt="">
         </div>
      </div>
   </div>
</div>

<!-- Testimonials -->
<div class="testimonials-container section-container
section-container-image-bg">
   <div class="container">
      <div class="row">
         <div class="col-sm-12 testimonials section-description wow
         fadeIn">
            <h2>Testimonials</h2>
            <div class="divider-1 wow fadeInUp"><span></span></div>
             <p>Take a look below to read what our clients have said
             about us:</p>
         </div>
      </div>
      <div class="row">
         <div class="col-sm-10 col-sm-offset-1 testimonial-list wow
         fadeInUp">
            <div role="tabpanel">
               <!-- Tab panes -->
               <div class="tab-content">
                  <div role="tabpanel" class="tab-pane fade in active"
                  id="tab1">
                     <div class="testimonial-image">
                        <img src="assets/img/testimonials/1.jpg" alt=""
                        data-at2x="assets/img/testimonials/1.jpg">
                     </div>
                     <div class="testimonial-text">
                        <h3>5 / 5 Stars</h3>
                           <p>
                              "Lorem ipsum dolor sit amet, consectetur
                              adipisicing elit, sed do eiusmod tempor
                              incididunt ut labore et.
                              Lorem ipsum dolor sit amet, consectetur
                              adipisicing elit, sed do eiusmod tempor
```

```
                                    incididunt ut labore et.
                                    Lorem ipsum dolor sit amet, consectetur
                                    ..."<br>
                                    <a href="#">Lorem Ipsum, dolor.co.uk</a>
                                </p>
                            </div>
                        </div>
                        <div role="tabpanel" class="tab-pane fade" id="tab2">
                            <div class="testimonial-image">
                                <img src="assets/img/testimonials/2.jpg" alt=""
                                data-at2x="assets/img/testimonials/2.jpg">
                            </div>
                            <div class="testimonial-text">
                                <h3>4 / 5 Stars</h3>
                                    <p>
                                    "Ut wisi enim ad minim veniam, quis
                                    nostrud exerci tation ullamcorper
                                    suscipit lobortis nisl ut aliquip
                                    ex ea commodo consequat. Ut wisi enim ad
                                    minim veniam, quis nostrud exerci tation
                                    ullamcorper suscipit
                                    lobortis nisl ut aliquip ex ea commodo
                                    consequat..."<br>
                                    <a href="#">Minim Veniam, nostrud.com</a>
                                </p>
                            </div>
                        </div>
                        <div role="tabpanel" class="tab-pane fade" id="tab3">
                            <div class="testimonial-image">
                                <img src="assets/img/testimonials/3.jpg" alt=""
                                data-at2x="assets/img/testimonials/3.jpg">
                            </div>
                            <div class="testimonial-text">
                                <h3>5 / 5 Stars</h3>

    </body>

</html>
```

Папка `static`

Файл `landing.css`

```
html, body{
    height: 100%;
    max-height: 100%;
    font-family: 'Roboto', sans-serif;
}

.top-container{
    padding-top: 10%;
}

.general-container{
    background: url('../img/bg_image.jpg');
    height: 100%;
}

.general-container form{
    margin-top: 30px;
}
```

```
.general-container form label{
    font-weight: 100;
    color: white;
}

.btn-orange{
    background-color: #e8643e;
    border: none;
}

.btn-orange:hover{
    background-color: #dc5129;
    border: none;
}

.title{
    text-align: center;
    color: white;
    font-weight: 100;
    line-height: 50px;
}
```

Файл `style.css`

```
html, body{
    height: 100%;
    max-height: 100%;
    font-family: 'Roboto', sans-serif;
    background-color: #edeef0;
}

.navbar{
    margin-bottom: 0;
}

.wrapper{
    min-height: 100%;
}

.wrapper-content{
    overflow: auto;
    padding-bottom: 180px; /* must be same height as the footer */
}

.footer{
    position: relative;
    margin-top: -180px; /* negative value of footer height */
    height: 180px;
    clear: both;
    background-color: white;
}

.section-top{
    margin-bottom: 20px
}

.product-description-tabs{
    padding: 10px;
}

.section{
    padding: 50px 0;
}
```

```
.section h1{
    margin-bottom: 40px;
}

.product-item{
    height: 460px;
    background-color: white;
    border: 1px solid lightgrey;
    position: relative;
    padding: 10px 0 10px 0;
    text-align: center;
    margin-bottom: 10px;
}

.add-to-card-btn{
    position: absolute;
    bottom: 15px;
    left: 50%;
    transform: translateX(-50%);
}

.discount-container{
    position: absolute;
    top: 30%;
    background: #55e073;
    width: 30%;
    color: white;
    font-weight: 700;
    padding: 3px;
}

.section-delivery{
    height: 300px;
    background-color: darkseagreen;
    text-align: center;
    color: white;
}

.product-image-item{
    padding: 5px;
    margin-bottom: 5px;
}
.navbar-top{
    min-height: 10px;
    height: 20px;
    background-color: green;
    border: none;
    border-radius: 0;
}

.navbar-main{
    background-color: #55e073;
    border: none;
    border-radius: 0;
}

.basket-container{
    position: relative;
    width: 400px;
    padding: 15px 10px;
}
```

```
.basket-items{
    position: absolute;
    top: 50px;
    width: 100%;
    background-color: #55e093;
    z-index: 10;
    padding: 10px;
}

.form-error{
    color: red;
}
```

Папка `product`

Файл `navbar.html`

```
<nav class="navbar navbar-default navbar-top">

    </nav>
    <nav class="navbar navbar-default navbar-main">
      <div class="container">
        <div class="navbar-header">
          <button type="button" class="navbar-toggle collapsed" data-toggle=
          "collapse" data-target="#navbar" aria-expanded="false" aria-controls=
          "navbar">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="#">ROCK SHOP</a>
        </div>
        <div id="navbar" class="navbar-collapse collapse">
          <ul class="nav navbar-nav">
            <li class="active"><a href="/">Домой</a></li>
            <li><a href="#about">Доставка</a></li>
            <li><a href="#contact">Контакты</a></li>
            <li class="dropdown">
              <a href="#" class="dropdown-toggle" data-toggle="dropdown" role=
              "button" aria-haspopup="true" aria-expanded="false">Регистрация <
              span class="caret"></span></a>
              <ul class="dropdown-menu">
                <li><a href="#">Войти</a></li>
                <li><a href="#">Регистрация</a></li>
                <li><a href="#">Забыл пароль</a></li>
                <li role="separator" class="divider"></li>
                <li class="dropdown-header">Дополнительно</li>
                <li><a href="#">Востановить номер</a></li>
                <li><a href="#">Востановить Email</a></li>
              </ul>
            </li>
          </ul>
          <ul class="nav navbar-nav navbar-right">
            <li>
                <div class="basket-container">
                    <a href="">Корзина <span id="basket_total_amount"></span><
                    span id="basket_total_nmb">
                        {% if products_total_nmb %}
                            ({{ products_total_nmb }})
                        {% else %}
                            (0)
                        {% endif %}
```

```
                    </span></a>
                    <div class="basket-items hidden">
                        <ul class="">
                            {% for product_in_basket in products_in_basket %}
                                <li>
                                    {{ product_in_basket.product.name }} {{
                                    product_in_basket.nmb }} шт.
                                    по {{ product_in_basket.price_per_item }} грн.
                                    <a class="delete-item" href="" data-
                                    product_id="{{ product_in_basket.id }}">x</a>
                                </li>
                            {% endfor %}
                            <li>
                                <a href="{% url 'checkout'  %}">
                                    Оформить заказ
                                </a>
                            </li>
                        </ul>
                    </div>
                </div>
            </li>


        </ul>
      </div><!--/.nav-collapse -->
    </div>
  </nav>
```

Файл `base.html`

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <!-- Bootstrap-->
    <link rel="stylesheet" href="
    https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.6/css/bootstrap.m
    in.css">
    <link href="
    https://fonts.googleapis.com/css?family=Roboto:100,100i,300,300i,400" rel=
    "stylesheet">
    <!-- Custom styles -->
    <link rel="stylesheet" href="{% static 'css/style.css' %}">
</head>
<body>
    <div class="wrapper">
        <div class="wrapper-content">
            {% include 'navbar.html' %}

            {% block content %}
            {% endblock content %}
        </div>
    </div>
     {% include 'footer.html' %}
    <!-- jQuery -->
    <script src="https://code.jquery.com/jquery-2.1.4.min.js"></script>
    <!-- Latest compiled and minified JavaScript -->
    <script src="
    https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"
    integrity=
    "sha384-0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ7xS"
    crossorigin="anonymous"></script>
    <script src="{% static 'js/scripts.js' %}"></script>
</body>
</html>
```