НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАНОЇ ТЕХНІКИ

КУРСОВА РОБОТА

з дисципліни «Інженерія програмного забезпечення»

на тему: «Інтернет – магазин ноутбуків. Функціонал адміністратора та користувача»

Виконали:

студенти ІІ курсу групи ІВ-82

Теряткін Назарій Ігорович

Пінчук Микита Вячеславович

Перевірив:

Болдак Андрій Олександрович

Київ – 2020 рік

# ЗМІСТ

[**ЗМІСТ** 2](#_Toc512258635)

[**ВСТУП** 3](#_Toc512258636)

[**1.** **ОГЛЯД MVC** 4](#_Toc512258637)

[*1.1.* *Загальні положення MVC* 4](#_Toc512258638)

[*1.2.* *Структура MVC* 4](#_Toc512258639)

[*1.3.* *Шаблони програмування, що використовуються в MVC* 4](#_Toc512258640)

[*1.4.* *Завдання MVC* 5](#_Toc512258641)

[*1.5.* *Python* 5](#_Toc512258642)

[*1.6.* *Використання концепції MVC в даній роботі* 5](#_Toc512258643)

[**2.** **ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ** 6](#_Toc512258644)

[*2.1.* *Прецеденти* 6](#_Toc512258645)

[*2.2.* *Діаграма граничних класів* 10](#_Toc512258646)

[2.3. *Таблиця відповідності елементів класів до Django* 10](#_Toc512258647)

[**3.** **РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ** 12](#_Toc512258648)

[*3.1.* *Діаграма класів* 12](#_Toc512258649)

[3.2. *Опис класів* 13](#_Toc512258650)

[**4.** **ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 15](#_Toc512258651)

[**ВИСНОВКИ** 22](#_Toc512258652)

[**ЛІТЕРАТУРА** 23](#_Toc512258653)

[**ДОДАТКИ** 24](#_Toc512258654)

[ДОДАТОК 1: ПРОГРАМНИЙ КОД ПРОЕКТУ 24](#_Toc512258655)

# ВСТУП

Об’єктом розробки даної курсової роботи є програма з функціональністю для управління сайтом магазину ноутбуків, а також моделювання способу замовлення клієнтом певних товарів.

Метою курсової роботи є закріплення теоретичних знань і практичних навичок з проектування, моделювання , розробки та тестування програмного забезпечення з графічним інтерфейсом.

Результат курсової роботи – готовий програмний додаток з графічним інтерфейсом, написаний на мові програмування Python.

Графічний інтерфейс користувача – інтерфейс між комп’ютером і його користувачем, що використовує вікна, меню, вказівний засіб для виконання команд, реалізований за допомогою Django – Python-фреймворку для розробки веб-систем.

1. **ОГЛЯД MVC**
   1. *Загальні положення MVC*

Model-view-controller, (MVC) (Модель–вигляд–контролер) – архітектурний шаблон, що використовується у проектуванні та розробці програмного забезпечення.

Даний шаблон передбачає розділення системи на три частини: модель даних, інтерфейс користувача та модуль керування, так щоб модифікація одного з компонентів мінімально впливала на інші.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

* 1. *Структура MVC*

Програма поділяється на три окремі і взаємопов’язані частини з розподілом функцій між компонентами:

Модель (Model) відповідає за зберігання даних і забезпечення інтерфейсу до них.

* Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.

Вигляд (View) відповідальний за представлення цих даних користувачеві.

* Вигляд може являти собою будь-яке представлення інформації, одержуване на виході. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації.

Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача і передає дані у модель.

* Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель не залежить від процесу вводу чи виводу даних, цей компонент інкапсулює ядро даних і основний функціонал їхньої обробки. Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані. У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас.

* 1. *Шаблони програмування, що використовуються в MVC*

Для реалізації схеми «Model-View-Controller» використовується досить велика кількість шаблонів проектування, основні з яких - «спостерігач», «стратегія», «компонувальник».

Найбільш типова реалізація - в якій уявлення відокремлено від моделі шляхом встановлення між ними протоколу взаємодії, що використовує «апарат подій» (підписка/сповіщення) : при кожній особливій зміні внутрішніх даних в моделі (позначеному як «подія»), вона сповіщає про нього всі залежні представлення і вони оновлюються. Так використовується шаблон «спостерігач».

При обробці реакції користувача представлення вибирає, в залежності від реакції, потрібний контролер, який забезпечує той чи інший зв'язок з моделлю. Для цього використовується шаблон «стратегія», або модифікація з використанням шаблону «команда».

Для можливості однотипного поводження з підоб'єктами складно складеного ієрархічного виду - використовується шаблон «компонувальник».

* 1. *Завдання MVC*

Основною метою застосування концепції MVC є відділення бізнес-логіки (Model) від її візуалізації (View). Через такий поділ виникає можливість повторно використовувати код програми. Найбільш корисним застосуванням концепції є випадок, коли користувач повинен одночасно бачити в різних контекстах ті ж самі дані. Виконуються наступні завдання:

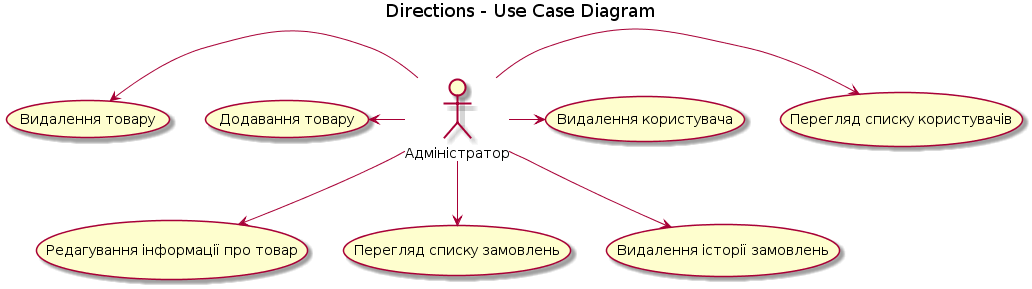
* До однієї моделі можна приєднати кілька видів, при цьому не зачіпаючи реалізацію моделі;
* Не торкаючись реалізації видів, можна змінити реакції на дії користувача, для цього досить використовувати інший контролер;
* Ряд розробників розробляють графічний інтерфейс, або розробляють бізнес-логіку. Можливо досягти того, що програмісти, які займаються розробкою бізнес-логіки, не будуть обізнані про те, який вигляд буде використовуватися.
  1. *Python*

Дану курсову роботу буде виконано, використовуючи мову програмування Python, з використаннямDjango – відкритого Python-фреймворку для розробки веб-систем. Різні компоненти фреймворку між собою пов'язані слабо, тому достатньо будь-яку частину замінити на аналогічну. Архітектура Django подібна на «Модель–вигляд–контролер» (MVC). Однак, те що називається «контролером» в класичній моделі MVC, в Django називається «вигляд» (англ. *view*), а те, що мало б бути «вигляд», називається «шаблон» (англ. *template*). Таким чином, MVC розробники Django називають MTV («Модель-Шаблон-вигляд»).

* 1. *Використання концепції MVC в даній роботі*

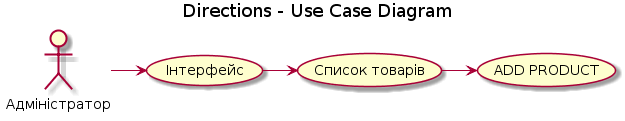
В даній роботі буде використовуватися модифікована версія шаблону MVC, яка об’єднує представлення і контролер в один логічний об’єкт. MVC розділяє систему на дві частини: модель даних (база даних), вигляд даних (графічний інтерфейс користувача) об’єднаний з керуванням (логікою). Це відокремить модель даних від інтерфейсу користувача, щоб зміни в будь-якій з цих частин системи мінімально впливали на іншу частину системи.

1. **ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ**
   1. *Прецеденти*

Адміністратор може додавати та видаляти товари у базі даних; заносити нову або видаляти стару інформацію про товар у базі даних; видаляти користувачів з бази даних; переглядати список користувачів, та їх замовлень; видаляти історію замовлень з бази даних.

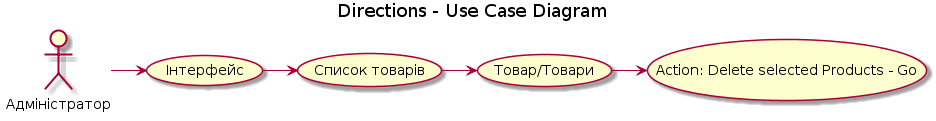
*Рисунок 2.1.* – Можливості адміністратора додатку

Адміністратор може додати товар перейшовши до списку товарів і натиснувши ADD PRODUCT. (Рис.2.2)



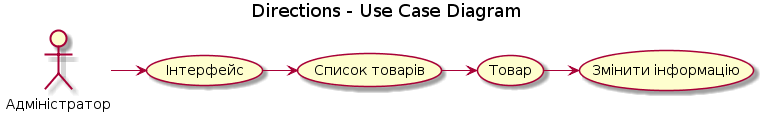
*Рисунок 2.2. –* Операція додавання товару у базу даних

Адміністратор може видаляти товар перейшовши до списку товарів, виділивши потрібні товари і натиснути Action: Delete selected Products - Go. (Рис.2.3)



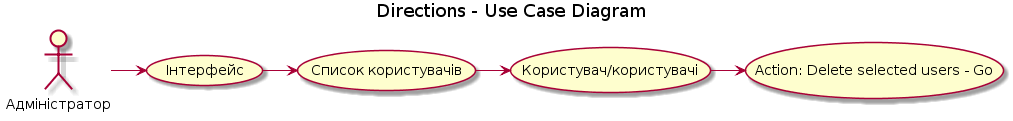
*Рисунок 2.3. –* Операція видалення товару з бази даних

Адміністратор може редагувати інформацію про товар перейшовши до списку товарів, а потім перейшовши до самого товару. (Рис.2.4)

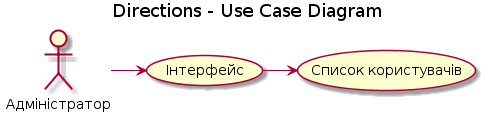


*Рисунок 2.4. –* Операція редагування інформації про товар у базі даних

Адміністратор може видаляти користувача перейшовши до списку користувачів, виділивши потрібних користувачів і натиснути Action: Delete selected users - Go. (Рис.2.5)

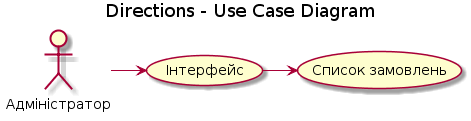


*Рисунок 2.5. –* Операція видалення користувача з бази даних

Адміністратор може переглядати список зареєстрованих користувачів перейшовши до списку користувачів. (Рис.2.6)

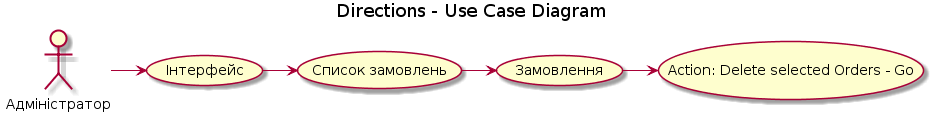
*Рисунок 2.6. –* Операція перегляду списку користувачів у базі даних

Адміністратор може переглядати товари у замовленнях користувачів перейшовши до списку замовлень. (Рис.2.7)



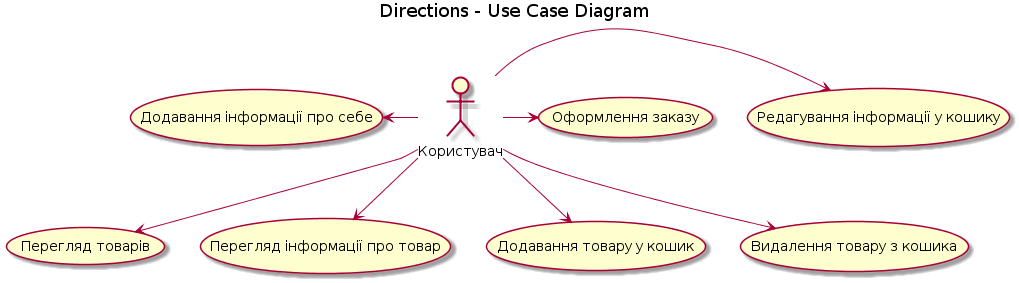
*Рисунок 2.7. –* Операція перегляду списку замовлень у базі даних

Адміністратор може видалити історію замовлення користувача перейшовши до списку замовлень, виділивши потрібні замовлення і натиснути Action: Delete selected Orders - Go. (Рис.2.8)



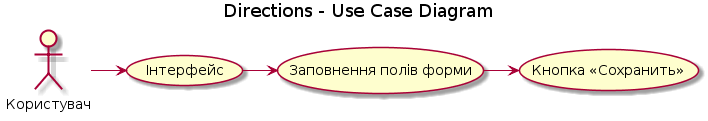
*Рисунок 2.8. –* Операція видалення замовлень з бази даних

Користувач може передавати інформацію про себе; переглядати товари на сайті; переглядати інформацію про певний товар; додавати, видаляти товари з кошика та редагувати інформацію про товари у кошику; оформляти замовлення.



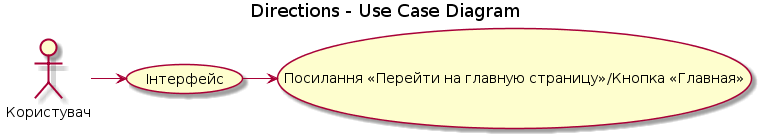
*Рисунок 2.9.* – Можливості користувача додатку

Користувач може додавати інформацію про себе до бази даних, заповнивши форму та натиснувши кнопку «Сохранить». (Рис.2.10)



*Рисунок 2.10.* – Операція додавання інформації про користувача до бази даних

Користувач може переглядати на головній сторінці магазину каталог наявних товарів, натиснувши на посилання «Перейти на главную страницу», або на кнопку «Главная» (Рис.2.11)



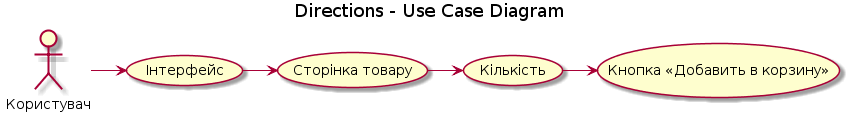
*Рисунок 2.11.* – Операція перегляду товарів

Користувач може переглянути детальну інформацію про товар на сторінці товару натиснувши на зображення відповідного товару на головній сторінці магазину. (Рис.2.12)



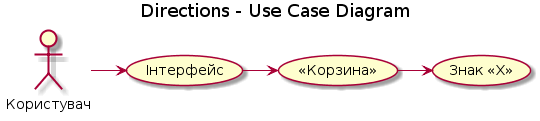
*Рисунок 2.12.* – Операція перегляду інформації про товар

Користувач може додати товар у кошик обравши його кількість та натиснувши кнопку «Добавить в корзину» на сторінці товару. (Рис.2.13)



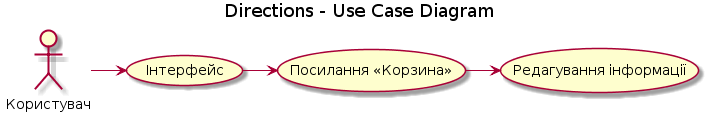
*Рисунок 2.13.* – Операція додавання товару у кошик

Користувач може видалити товар з кошика навівши курсор на посилання «Корзина» і натиснувши на знак «Х» біля товару який хочете видалити. (Рис.2.14)



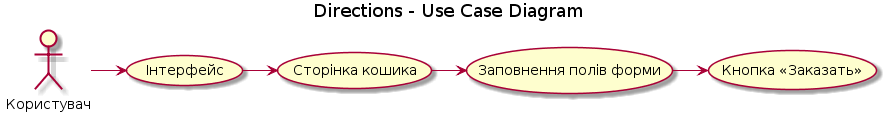
*Рисунок 2.14.* – Операція видалення товару з кошика

Користувач може редагувати інформацію про товари в кошику, перейшовши на сторінку кошика натиском на посилання «Корзина». (Рис.2.15)



*Рисунок 2.15.* – Операція редагування інформації у кошику

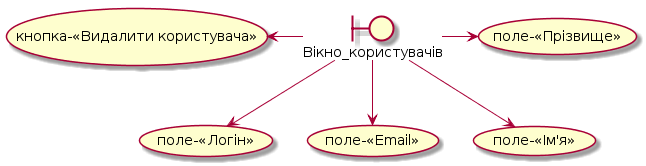
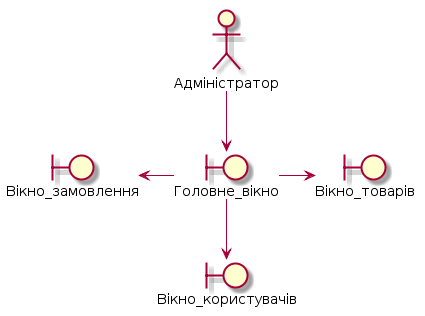
Користувач може оформити замовлення на сторінці кошика, заповнивши форму та натиснувши кнопку «Заказать». (Рис.2.16)

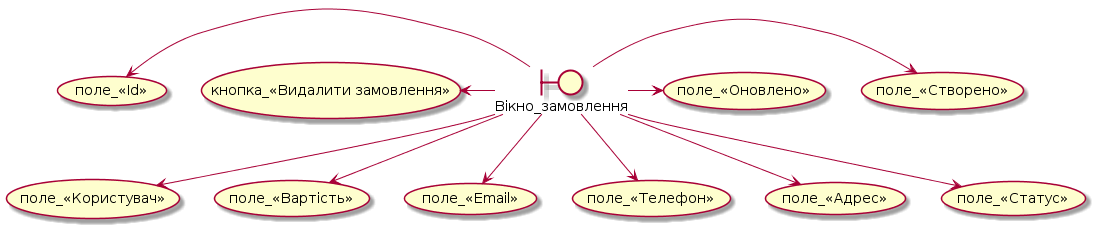


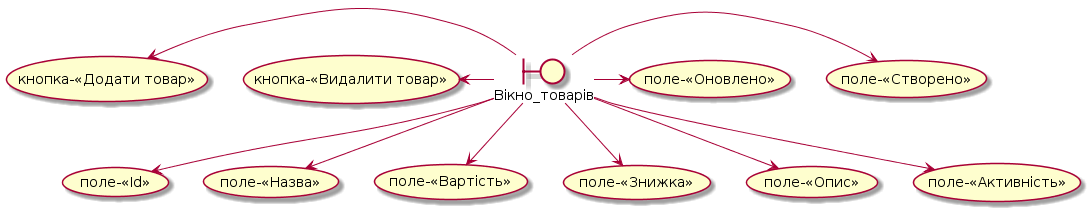
*Рисунок 2.16.* – Операція оформлення замовлення

* 1. *Діаграма граничних класів*

На Рис.2.17 зображено діаграму граничних класів для бази даних, тобто функціоналу адміністратора



**

**

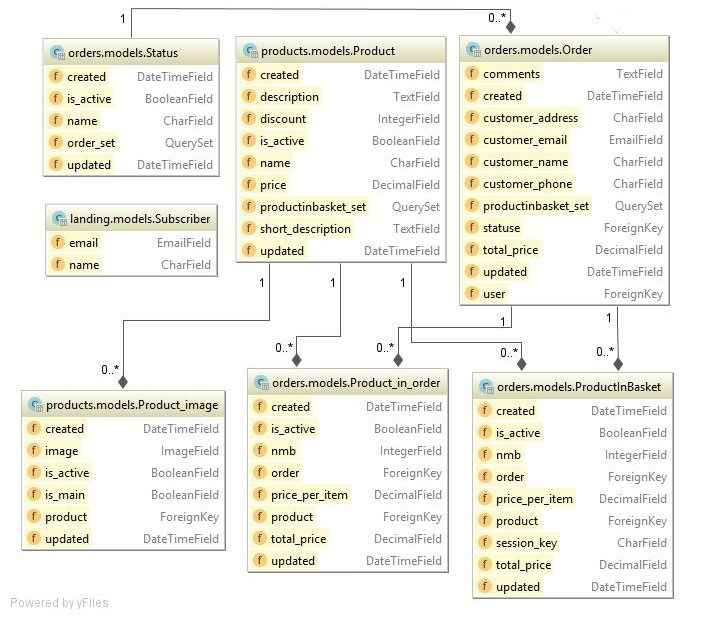
*Рисунок 2.17.* – Діаграма граничних класів

* 1. *Таблиця відповідності елементів класів до Django*

|  |  |
| --- | --- |
| Елемент | Клас, реалізуючий елемент |
| Головне\_вікно | django.models.Model |
| Вікно\_користувачів | django.models.Model |
| Вікно\_замовлення | django.models.Model |
| Вікно\_товарів | django.models.Model |
| кнопка-«Видалити користувача» | django.models.Button |
| поле-«Логін» | django.models.CharField |
| поле-«Email» | django.models.EmailField |
| поле-«Ім'я» | django.models.CharField |
| поле-«Прізвище» | django.models.CharField |
| кнопка\_«Видалити замовлення» | django.models.Button |
| поле\_«Id» | django.models.IdField |
| поле\_«Користувач» | django.models. ForeignKey |
| поле\_«Вартість» | django.models.DecimalField |
| поле\_«Email» | django.models.EmailField |
| поле\_«Телефон» | django.models.CharField |
| поле\_«Адрес» | django.models.CharField |
| поле\_«Статус» | django.models.ForeignKey |
| поле\_«Створено» | django.models.DateTimeField |
| поле\_«Оновлено» | django.models.DateTimeField |
| кнопка-«Видалити товар» | django.models.Button |
| кнопка-«Додати товар» | django.models.Button |
| поле-«Id» | django.models.IdField |
| поле-«Назва» | django.models.CharField |
| поле-«Вартість» | django.models.DecimalField |
| поле-«Знижка» | django.models.IntegerField |
| поле-«Опис» | django.models.TextField |
| поле-«Активність» | django.models.BooleanField |
| поле-«Створено» | django.models.DateTimeField |
| поле-«Оновлено» | django.models. DateTimeField |

1. **РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ**
   1. *Діаграма класів*

На рисунку 3.1 зображено діаграму класів програми. На ній видно класи, що реалізують моделі (models) об’єктів бази даних, та їхні поля, які забезпечують її коректну роботу. Те, що мало б бути «*view*», в Django називається «*template*», і є фактично HTML документами, що в діаграму не входять.



*Рисунок 3.1.* – Діаграма класів

* 1. *Опис класів*

Клас Product – клас, що описує модель товару, тобто його характеристики.

Містить такі поля:

* name – назва товару;
* price – ціна товару;
* discount – (можлива) знижка на товар;
* short\_description – (можливий) короткий опис товару;
* description – опис товару;
* is\_active – чи продається зараз цей товар;
* created – коли товар додали до бази даних;
* updated – коли інформацію про товар останній раз оновлювали.

Клас Order – клас, що описує модель замовлення, тобто його характеристики.

Містить такі поля:

* user –користувач який оформив замовлення;
* total\_price – вартість замовлення;
* customer\_email – email користувача який оформив замовлення;
* customer\_name – ім’я користувача який оформив замовлення;
* customer\_phone – телефон користувача який оформив замовлення;
* customer\_address – адрес користувача який оформив замовлення;
* comments – коментар до замовлення;
* statuse – статус замовлення;
* created – коли замовлення додали до бази даних;
* updated – коли інформацію про замовлення останній раз оновлювали.

Клас Subscriber – клас, що містить список користувачів, що зареєструвалися на сайті.

Містить такі поля:

* email – email користувача;
* name – ім’я користувача.

Клас Product\_image – клас, що містить зображення прив’язує їх до певного товару.

Містить такі поля:

* product – до якого товару відноситься зображення;
* image – зображення;
* is\_main – основне зображення товару;
* is\_active – чи відображається це зображення на сторінці товару;
* created – коли зображення додали до бази даних;
* updated – коли зображення останній раз оновлювали.

Клас Product\_in\_order – клас, що наслідує Product та Order, і являє собою список в якому вказується, які товари входять у яке замовлення.

Містить такі поля:

* order – номер замовлення;
* product – товар у замовлення;
* nmb – кількість одиниць певного товару у замовленні;
* price\_per\_item – ціна за одиницю товару;
* total\_price – ціна всіх одиниць товару;
* is\_active – чи виконується замовлення в даний час;
* created – коли замовлення оформлене;
* updated – коли інформацію про замовлення останній раз оновлювали.

Клас ProducinBasket – клас, що показує, які товари користувач додав у кошик.

Містить такі поля:

* session\_key – ідентифікаційний ключ користувача;
* product – товар у кошику;
* nmb – кількість одиниць певного товару;
* price\_per\_item – ціна за одиницю товару;
* total\_price – ціна всіх одиниць товару;
* is\_active
* created – коли кошик створено (перший захід користувача);
* updated – коли останній раз додавали чи видаляли товари з кошика.

Класи views – класи для створення сторінок на сайті та їхнього заповнення.

Містить такі методи:

для landing\views

* landing(reguest) – функція для створення початкового вікна;
* home(reguest) – функція для створення головного вікна.

для product\views

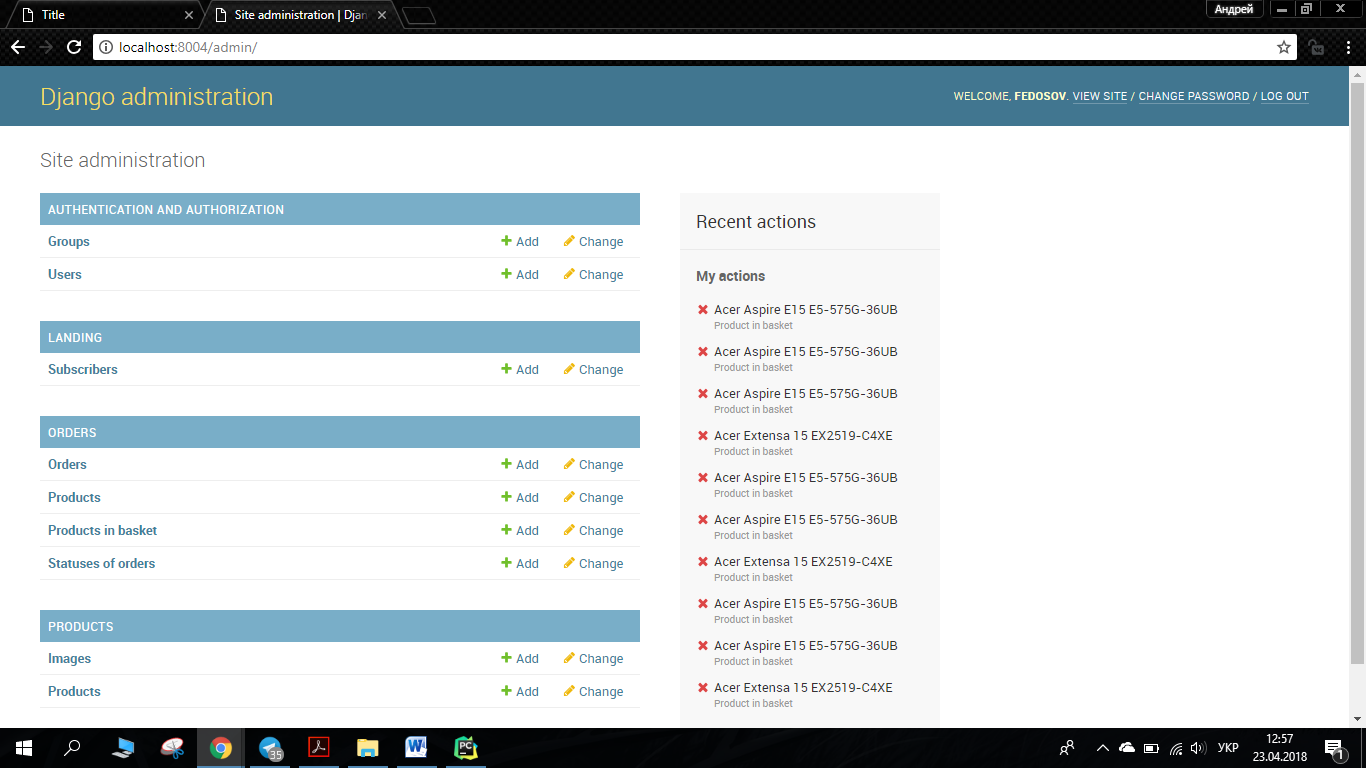
* product(request, product\_id) – функція для створення вікна товару за його id.

для orders\views

* basket\_adding(request) – збереження даних у кошику;
* checkout(request) – функція для створення вікна кошика.

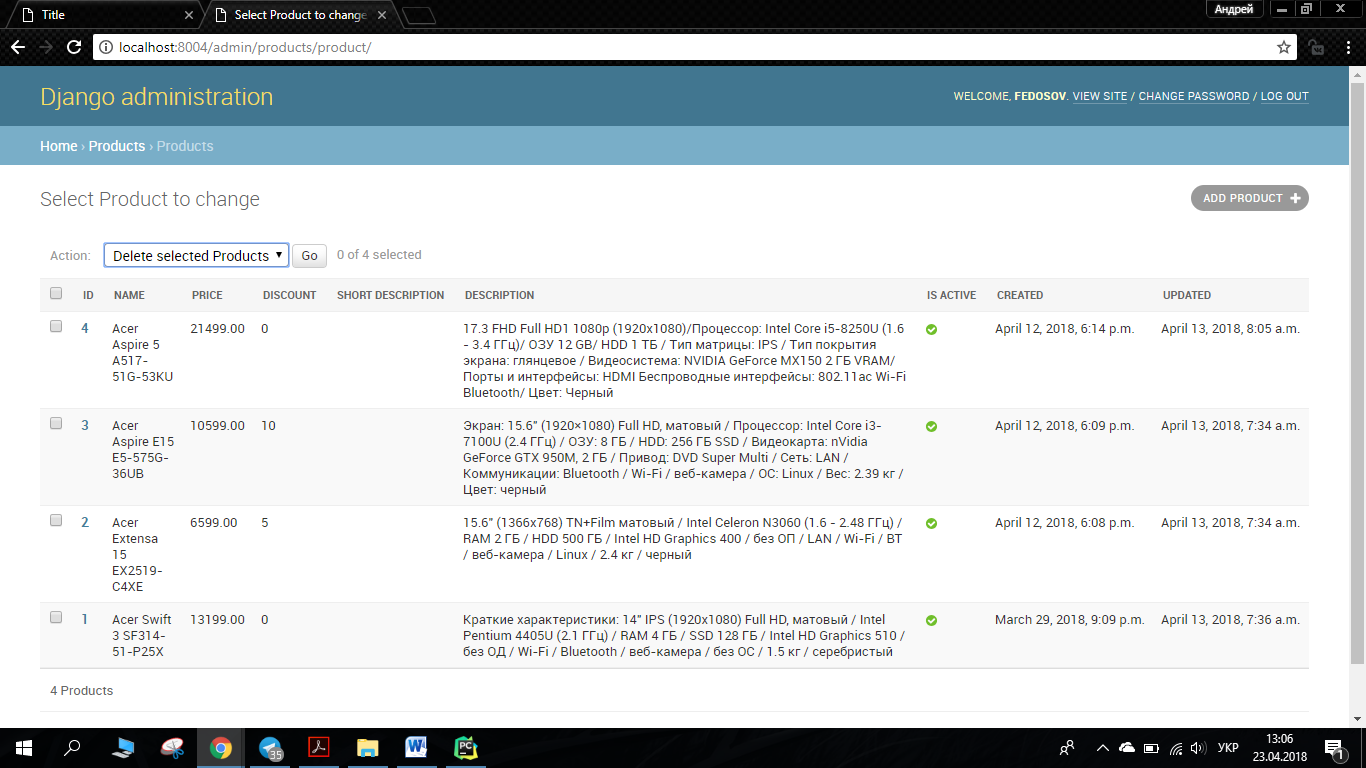
1. **ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

Перейшовши до Головного вікна бази даних, Адміністратор може переглянути існуючі поля бази даних. Вигляд Головного вікна бази даних можна побачити на Рис. 4.1.



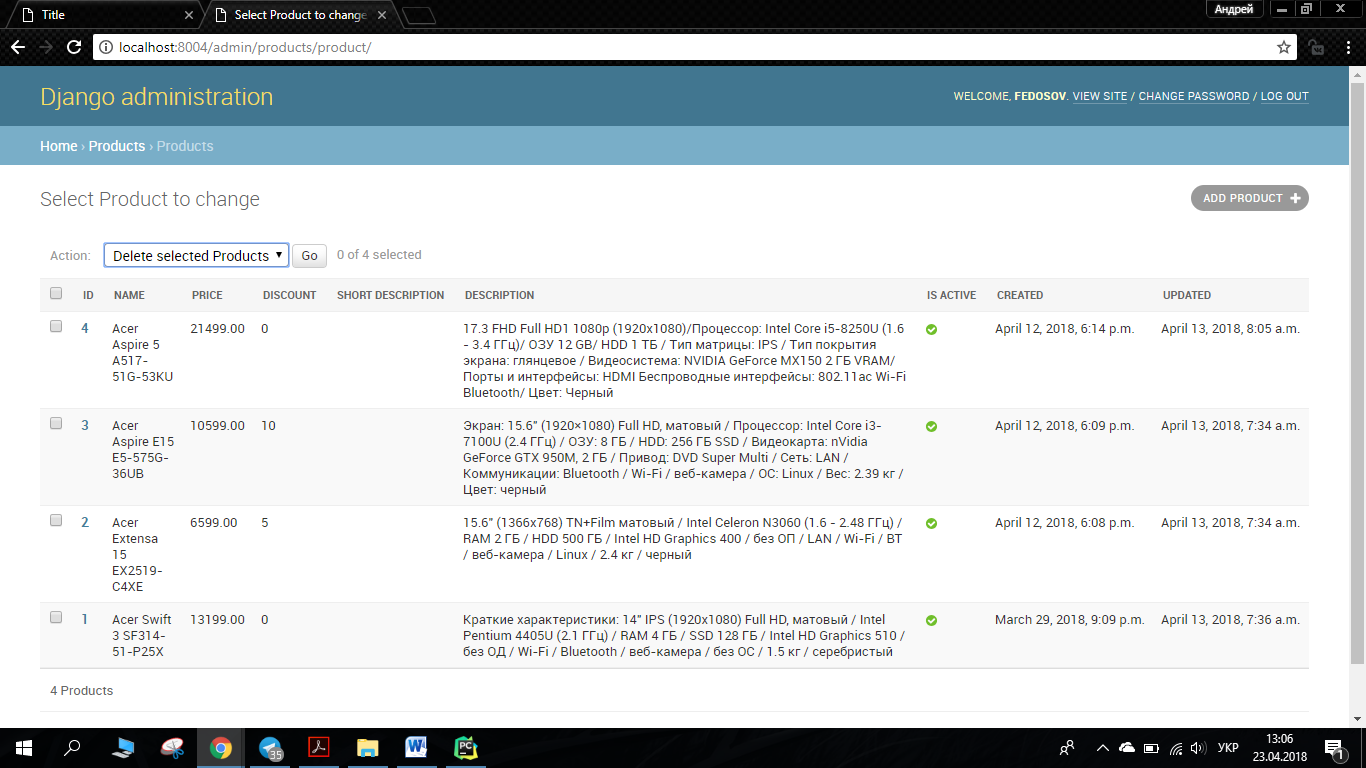
*Рисунок 4.1.* – Головне вікно бази даних

Згідно з прецедентом Рис. 2.2 адміністратор може додати товар у вікні товарів. Вигляд вікна товарів бази даних після додавання товарів, з полями товарів, кнопкою «ADD PRODUCT» можна побачити на Рис. 4.2.



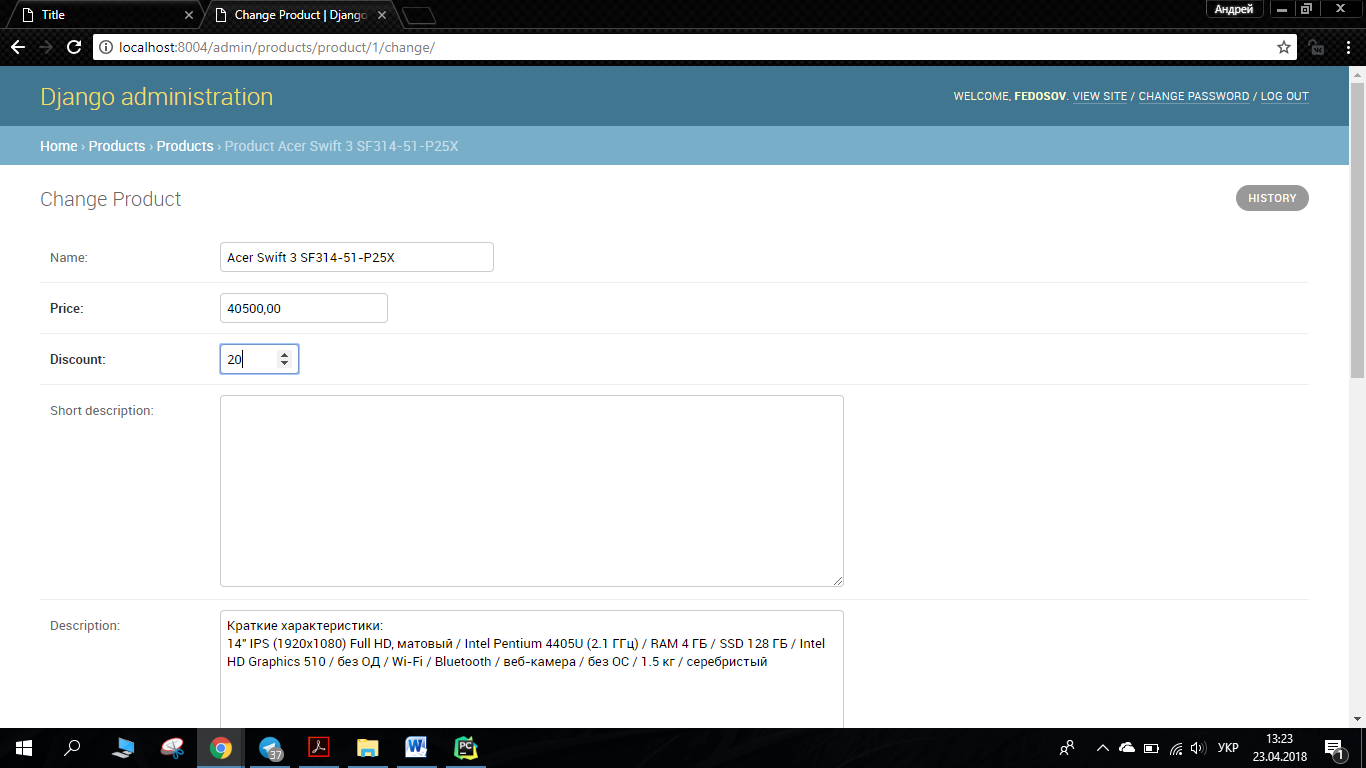
*Рисунок 4.2.* – Вікно товарів бази даних

Згідно з прецедентом Рис. 2.3 адміністратор може видалити товар або товари у вікні товарів. Вигляд вікна товарів бази даних, з полями товарів, кнопкою «Action: Delete selected Products Go » можна побачити на Рис. 4.3.



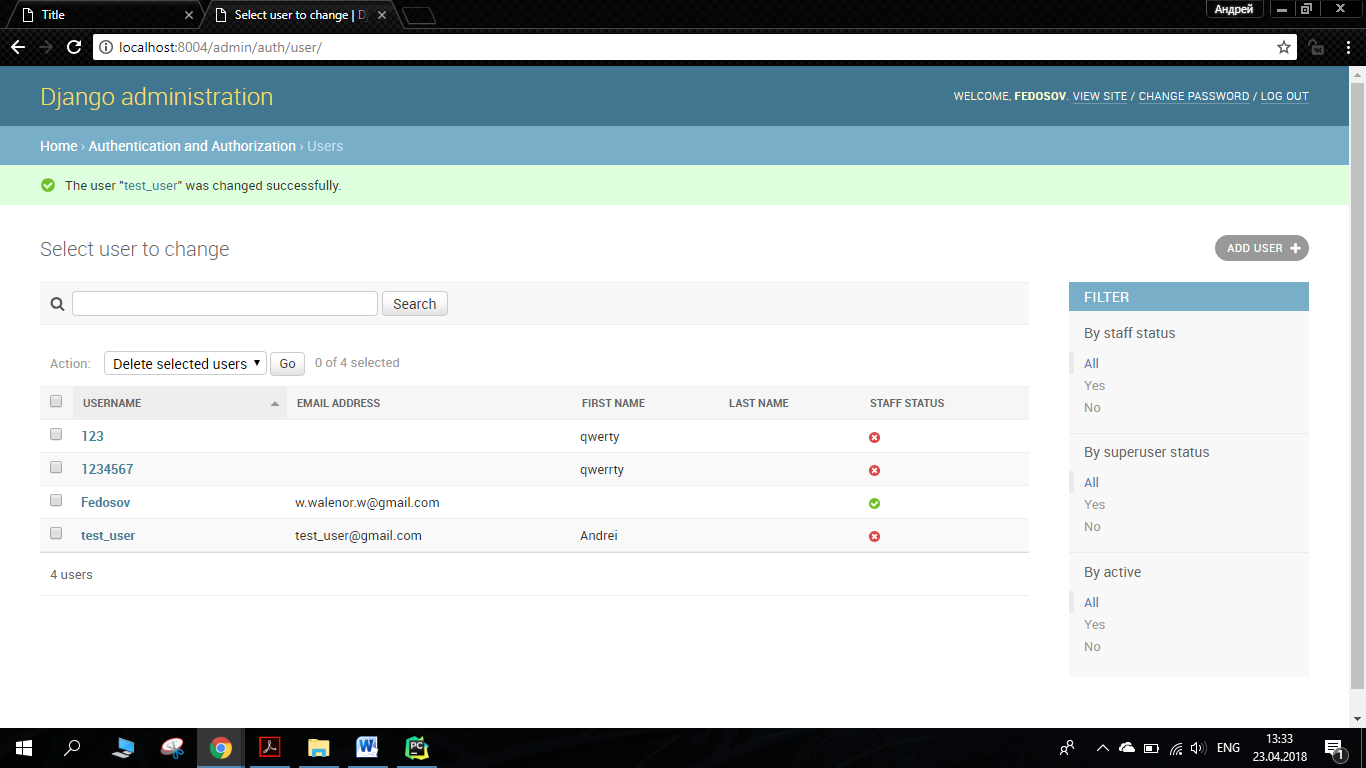
*Рисунок 4.3.* – Вікно товарів бази даних

Адміністратор може змінювати інформацію про товар у вікні товару згідно з прецедентом на Рис. 2.4. Вигляд вікна товару в базі даних, з полями даного товару, та результатом зміни інформації про товар з ID – 1 можна побачити на Рис. 4.4.



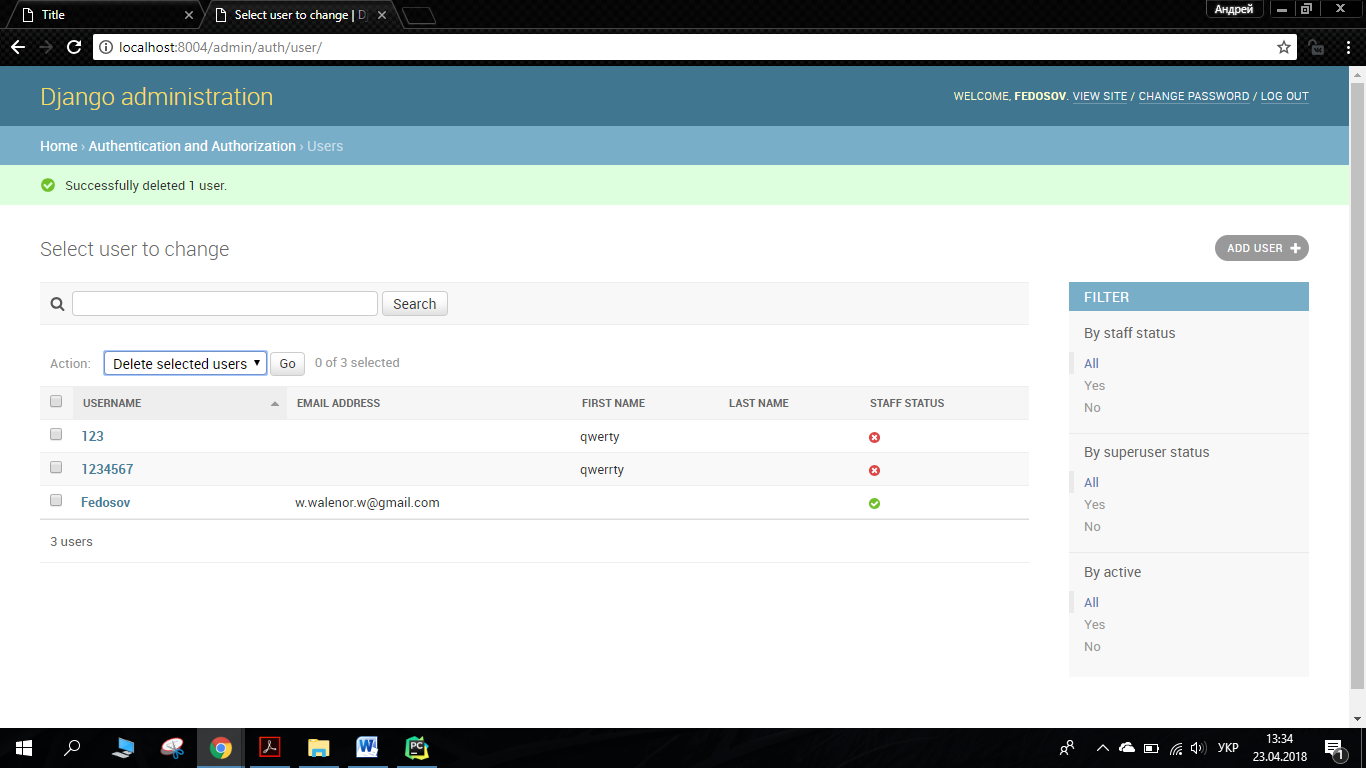
*Рисунок 4.4.* – Вікно товару у базі даних

У вікні користувачів адміністратор може переглянути список користувачів згідно з прецедентом Рис. 2.6, а також згідно з прецедентом Рис. 2.5 може видалити користувача або користувачів. Вигляд вікна користувачів сайту, з полями користувачів, кнопкою «Action: Delete selected users Go» можна побачити на Рис. 4.5.1



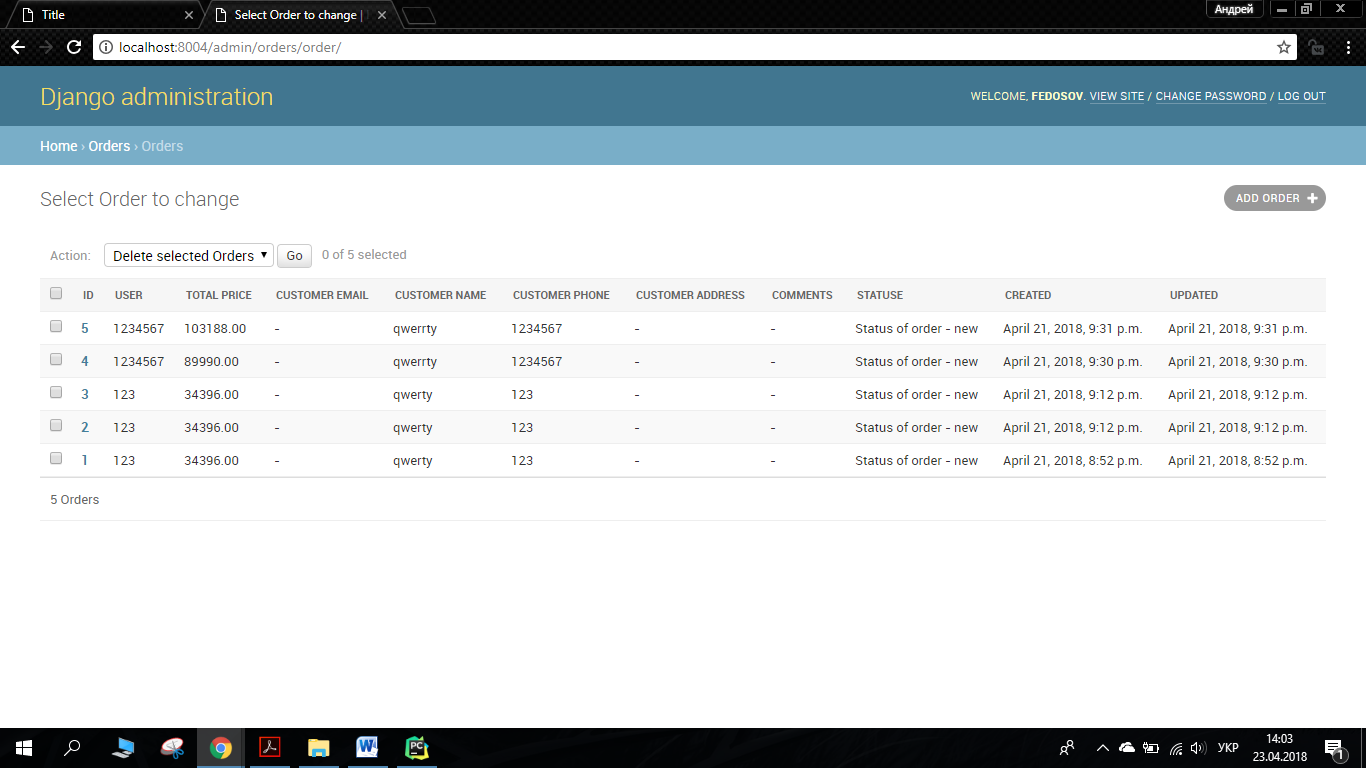
*Рисунок 4.5.1* – Вікно користувачів сайту

Результат видалення користувача test\_user (Рис. 4.5.2)

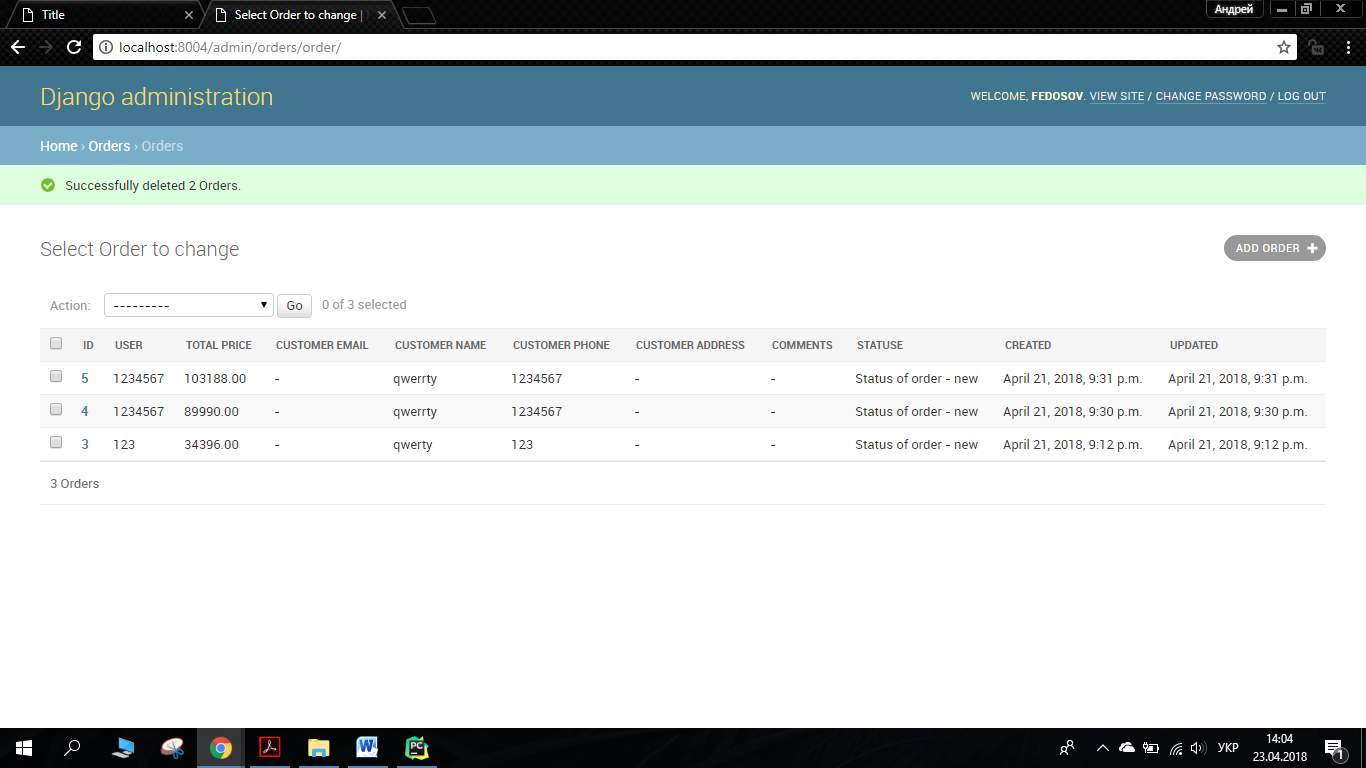


*Рисунок 4.5.2* – Результат видалення користувача

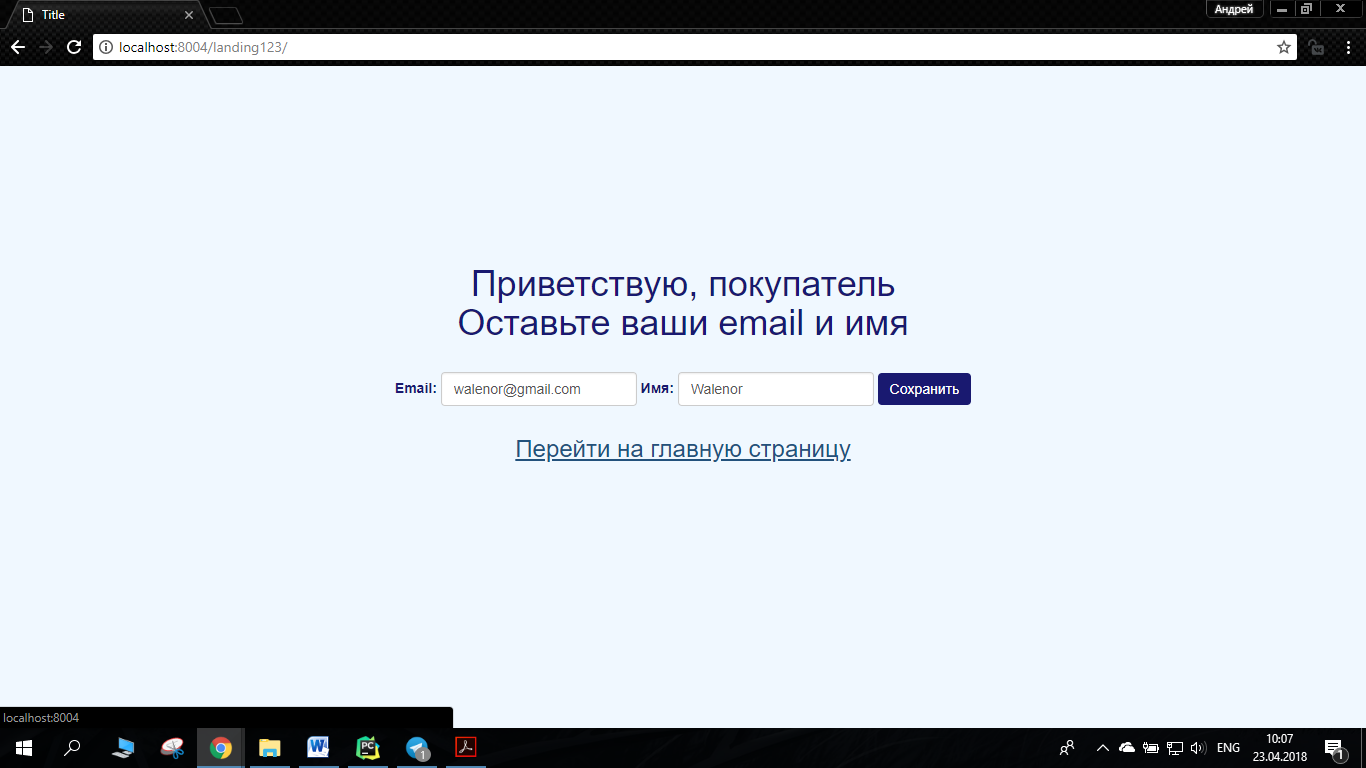
У вікні замовлень адміністратор може переглянути список замовлень згідно з прецедентом Рис. 2.7, а також згідно з прецедентом Рис. 2.8 може видалити історію замовлення або замовлень. Вигляд вікна замовлень, з полями замовлень, кнопкою «Action: Delete selected Orders Go» можна побачити на Рис. 4.6.1



Результат видалення замовлення з ID – 1 і 2 (Рис. 4.6.2)

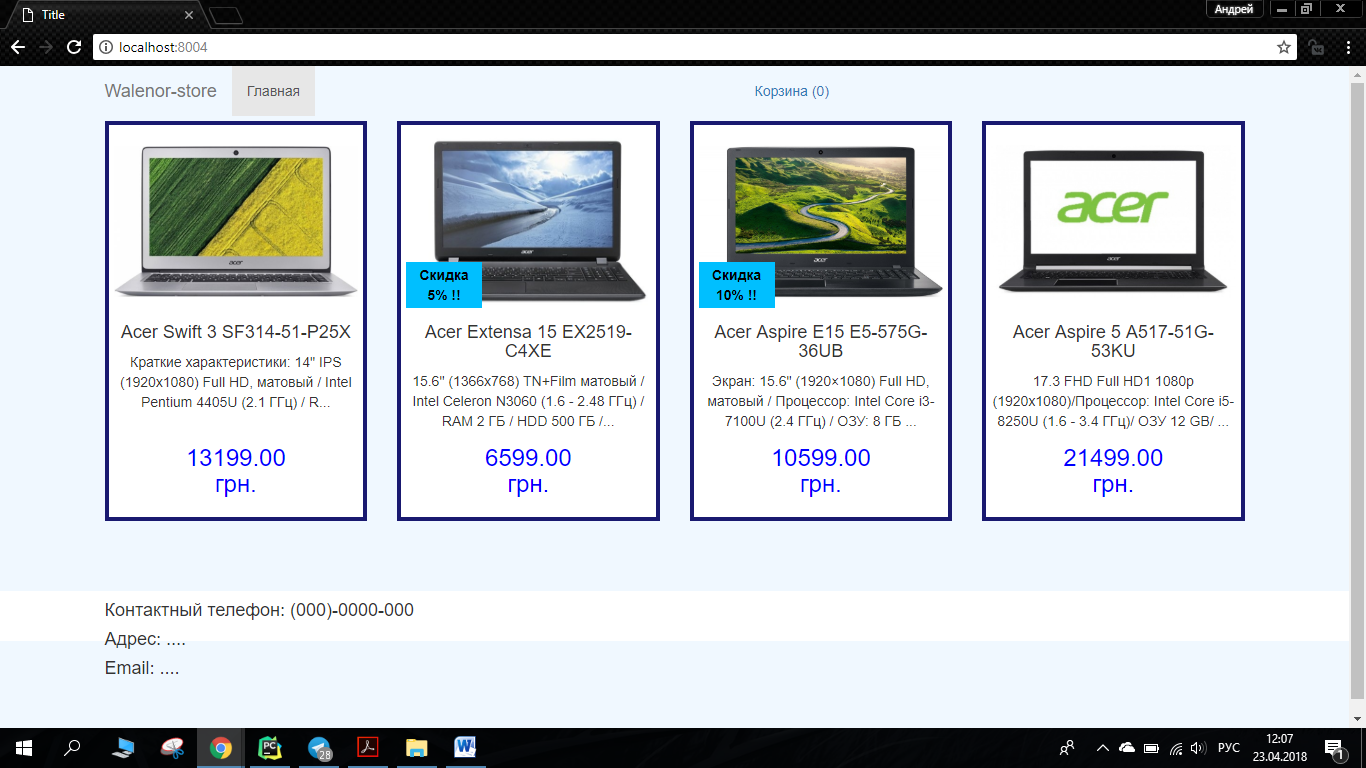


Під час переходу на сайт, користувач бачить вікно вітання і прохання залишити контактні дані про себе згідно з прецедентом на Рис. 2.9, або перейти на головну сторінку магазину, натиснувши на посилання «Перейти на главную страницу» згідно з прецедентом на Рис. 2.10. Вигляд початкової сторінки з кнопкою «Сохранить» та посиланням «Перейти на главную страницу» можна побачити на Рис. 4.1.



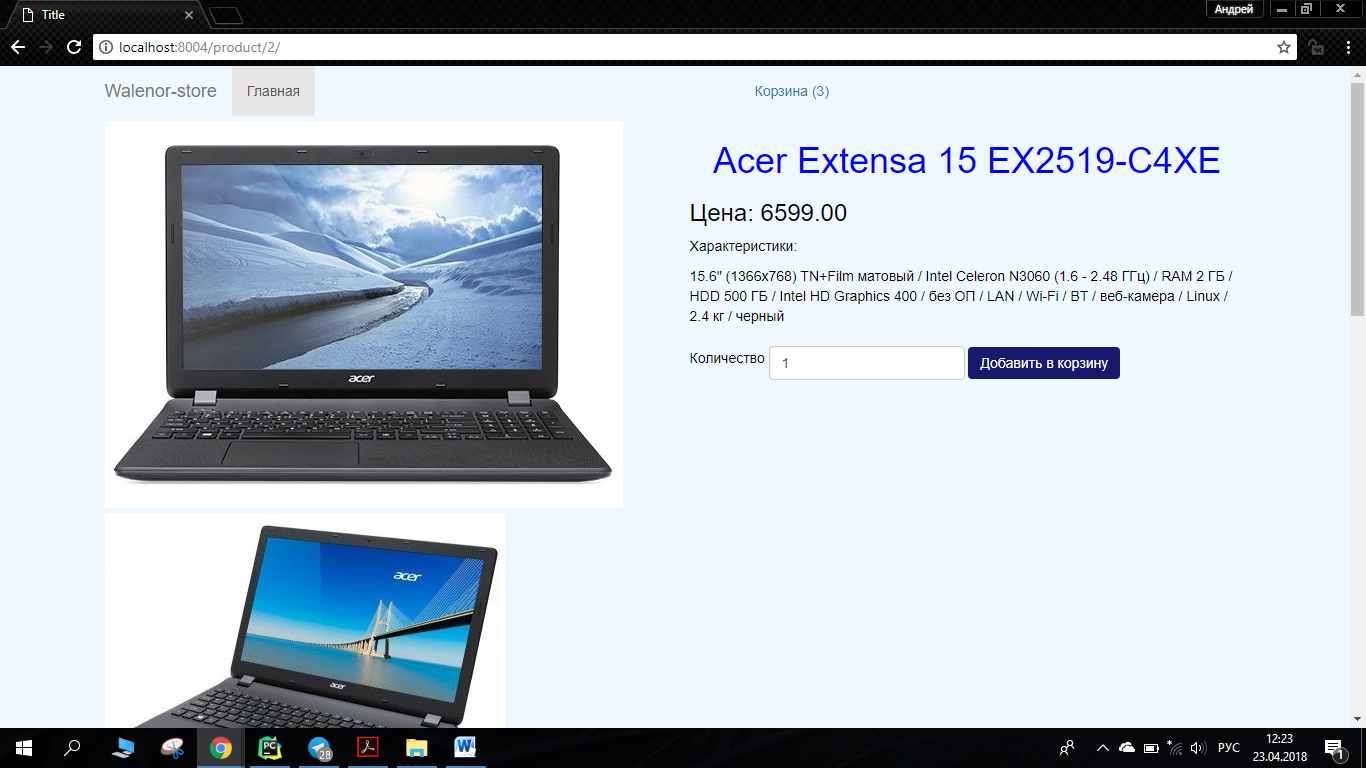
*Рисунок 4.1.* – Початкова сторінка сайту

Натиснувши на посилання «Перейти на главную страницу», користувач переходить на головну сторінку магазину згідно з прецедентом на Рис. 2.10. Вигляд головної сторінки сайту з переліком товарів, короткою інформацією про них, посиланням «Корзина», та контактною інформацією сайту можна побачити на Рис. 4.2.



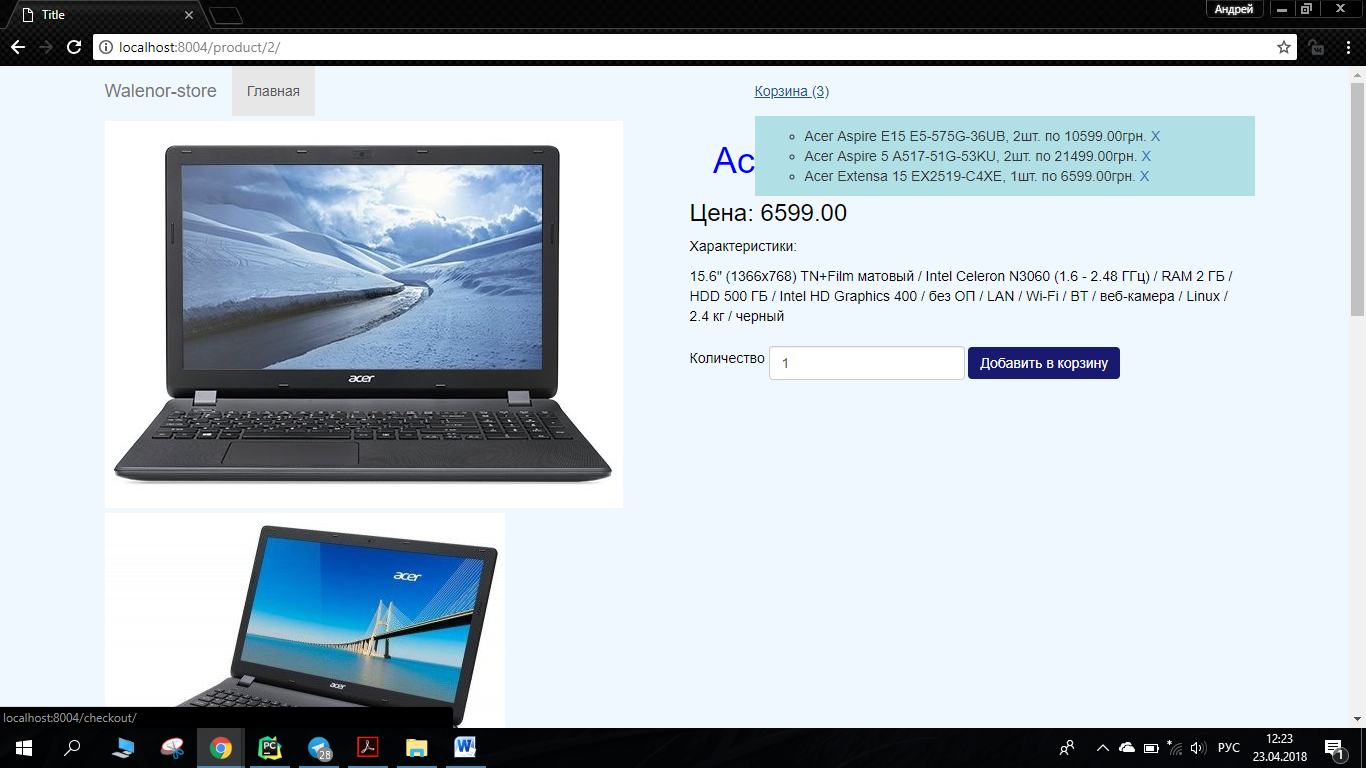
*Рисунок 4.2.* – Головна сторінка сайту

Згідно з прецедентом на Рис. 2.11 користувач може перейти на сторінку певного товару натиснувши на його зображення. Згідно з прецедентом на Рис. 2.12 користувач може додати товар до кошика, і видалити з кошика згідно з прецедентом на Рис. 2.13. Вигляд сторінки товару з інформацією про товар, кількістю товару який хочете додати в кошик, кнопкою «Добавить в Корзину», посиланням «Корзина», кнопкою «Главная», та контактною інформацією сайту можна побачити на Рис. 4.3.



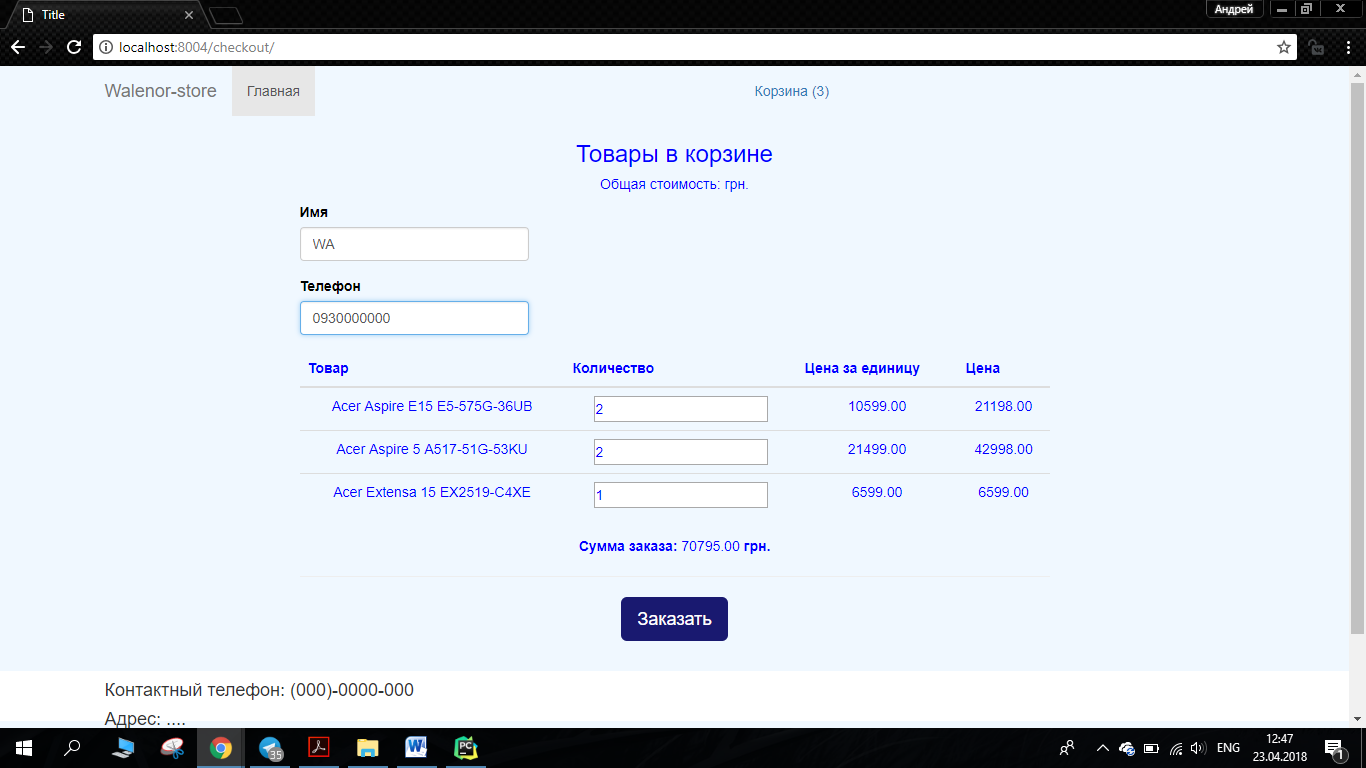
*Рисунок 4.3.* – Сторінка товару

Результат додавання товару в корзину можна побачити на Рис. 4.4.



*Рисунок 4.4.* – Сторінка товару

Користувач може змінювати інформацію щодо свого кошика на сторінці кошика згідно з прецедентом на Рис. 2.14, або оформити замовлення згідно з прецедентом на Рис. 2.15. Вигляд сторінки кошика з інформацією про кількість товару, його вартість, кнопкою «Заказать», посиланням «Корзина», кнопкою «Главная» та контактною інформацією сайту можна побачити на Рис. 4.4.



*Рисунок 4.3.* – Сторінка кошика

В результаті тестування програмного додатку було перевірено на працездатність абсолютно всі прецеденти, які були описані в частині «Проектування програмного додатку». Під час тестування всі прецеденти працюють правильно і без помилок. В даній частині курсової роботи було розроблено візуальне відображення роботи кожного прецеденту.

# ВИСНОВКИ

У ході виконання даної курсової роботи:

Були закріплені теоретичні знання з проектування, моделювання , розробки та тестування програмного забезпечення з графічним інтерфейсом, було вивчено шаблон програмування MVC, покращено практичні навички роботи з мовою програмування Python та Django – Python-фреймворку для розробки веб-систем.

Розроблено механізм створення бази даних, збереження та обробки об’єктів бази даних для сайту інтернет-магазину ноутбуків.

Розроблено графічний інтерфейс, який задовольняє усім вимогам, поставленим під час проектування.

Метою було розробити додаток який дозволяє адміністратору інтернет-магазину вносити зміни до каталогу товарів тобто додавати, видаляти товари та редагувати інформацію про них, а користувачу вільно переміщатися по сайту та інтуїтивно просто робити замовлення.

<https://github.com/Teriatkin/IPZ_project> - посилання на проект.

# ЛІТЕРАТУРА

1. Марк Лутц Программирование на Python. Том 1, 4-е издание – «Символ-Плюс», 2011 – 992 c. — ISBN 978-1-449-35573-9.
2. Бандура, В. В. Архітектура та проектування програмного забезпечення конспект лекцій / В. В. Бандура, Р. І. Храбатин. - Івано-Франківськ: ІФНТУНГ, 2012. - 240 с.
3. Вендров А.М. Проектирование программного обеспечения экономических информационных систем «Финансы и статистика», 2006. - 544 с.
4. Django documentation.
5. http://uk.wikipedia.org/wiki/Шаблони проектування програмного забезпечення.

# ДОДАТКИ

## ДОДАТОК 1: ПРОГРАМНИЙ КОД ПРОЕКТУ

landing\admin.py

**from** django.contrib **import** admin  
**from** .models **import** \*  
  
**class** SubscriberAdmin (admin.ModelAdmin):  
 list\_display = [**"email"**, **"name"**, **"id"**]  
 list\_filter = [**"name"**, ]  
 search\_fields = [**"email"**, **"name"**, **"id"**]  
  
 **class** Meta:  
 model = Subscriber  
  
admin.site.register(Subscriber, SubscriberAdmin)

landing\forms.py

**from** django **import** forms  
**from** .models **import** \*  
  
**class** SubscriberForm(forms.ModelForm):  
  
 **class** Meta:  
 model = Subscriber  
 exclude = [**""**]

landing\models.py

**from** django.db **import** models  
  
**class** Subscriber(models.Model):  
 email = models.EmailField()  
 name = models.CharField(max\_length=100)  
  
 **def** \_\_str\_\_(self):  
 **return "Email: %s; Name: %s"** % (self.email, self.name)

landing\urls.py

**from** django.conf.urls **import** url  
**from** landing **import** views  
  
urlpatterns = [  
 url(**r'^$'**, views.home, name=**'home'**),  
 url(**r'^landing123/$'**, views.landing, name=**'landing'**),  
]

landing\views.py

**from** django.shortcuts **import** render  
**from** .forms **import** SubscriberForm  
**from** products.models **import** \*  
  
**def** landing(reguest):  
 form = SubscriberForm(reguest.POST **or None**)  
 **if** reguest.method == **"POST" and** form.is\_valid():  
 new\_form = form.save()  
 **return** render(reguest, **'landing/landing.html'**, locals())  
  
**def** home(reguest):  
 products\_images = Product\_image.objects.filter(is\_active=**True**, is\_main=**True**)  
 **return** render(reguest, **'landing/home.html'**, locals())

orders\admin.py

**from** django.contrib **import** admin  
**from** .models **import** \*  
  
**class** Product\_in\_orderInline(admin.TabularInline):  
 model = Product\_in\_order  
 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** Product\_in\_orderAdmin(admin.ModelAdmin):  
 list\_display = [field.name **for** field **in** Product\_in\_order.\_meta.fields]  
  
 **class** Meta:  
 model = Product\_in\_order  
  
admin.site.register(Product\_in\_order, Product\_in\_orderAdmin)  
  
  
**class** OrderAdmin(admin.ModelAdmin):  
 list\_display = [field.name **for** field **in** Order.\_meta.fields]  
 inlines = [Product\_in\_orderInline]  
  
 **class** Meta:  
 model = Order  
  
admin.site.register(Order, OrderAdmin)  
  
**class** ProductInBasketAdmin (admin.ModelAdmin):  
 list\_display = [field.name **for** field **in** ProductInBasket.\_meta.fields]  
  
 **class** Meta:  
 model = ProductInBasket  
  
admin.site.register(ProductInBasket, ProductInBasketAdmin)

orders\context\_processors.py

**from** .models **import** ProductInBasket  
  
**def** getting\_basket\_info(request):  
 session\_key = request.session.session\_key  
 **if not** session\_key:  
 request.session[**"session\_key"**] = 123  
 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()

orders\forms.py

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

orders\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=50, 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 "Status of order %s"** % self.name  
  
 **class** Meta:  
 verbose\_name = **"Status of order"** verbose\_name\_plural = **"Statuses of orders"  
  
class** Order(models.Model):  
 user = models.ForeignKey(User, blank=**True**, null=**True**, default=**None**, on\_delete=models.CASCADE)  
 total\_price = models.DecimalField(max\_digits=10, decimal\_places=2, default=0) *# total price for all products in order* customer\_email = models.EmailField(blank=**True**, null=**True**, default=**None**)  
 customer\_name = models.CharField(max\_length=50, blank=**True**, null=**True**, default=**None**)  
 customer\_phone = models.CharField(max\_length=13, 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**)  
 statuse = models.ForeignKey(Status, on\_delete=models.CASCADE)  
  
 created = models.DateTimeField(auto\_now\_add=**True**, auto\_now=**False**)  
 updated = models.DateTimeField(auto\_now\_add=**False**, auto\_now=**True**)  
  
 **def** \_\_str\_\_(self):  
 **return "Order %s %s"** % (self.id, self.statuse.name)  
  
 **class** Meta:  
 verbose\_name = **"Order"** verbose\_name\_plural = **"Orders"  
  
 def** save(self, \*args, \*\*kwargs):  
  
 super(Order, self).save(\*args, \*\*kwargs)  
  
  
**class** Product\_in\_order(models.Model):  
 order = models.ForeignKey(Order, blank=**True**, null=**True**, default=**None**, on\_delete=models.CASCADE)  
 product = models.ForeignKey(Product, blank=**True**, null=**True**, default=**None**, on\_delete=models.CASCADE)  
 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)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 "Product %s"** % self.product.name  
  
 **class** Meta:  
 verbose\_name = **"Product"** verbose\_name\_plural = **"Products"  
  
 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(Product\_in\_order, self).save(\*args, \*\*kwargs)  
  
**def** product\_in\_order\_post\_save(sender, instance, created, \*\*kwargs):  
 order = instance.order  
 all\_products\_in\_order = Product\_in\_order.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=Product\_in\_order)  
  
**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.CASCADE)  
 product = models.ForeignKey(Product, blank=**True**, null=**True**, default=**None**, on\_delete=models.CASCADE)  
 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 = **'Product in basket'** verbose\_name\_plural = **'Products in basket'  
  
  
 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)

orders\urls.py

**from** django.conf.urls **import** url  
**from** orders **import** views  
  
urlpatterns = [  
 url(**r'^basket\_adding/$'**, views.basket\_adding, name=**'basket\_adding'**),  
 url(**r'^checkout/$'**, views.checkout, name=**'checkout'**),  
]

orders\ views.py

**from** django.http **import** JsonResponse  
**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:  
 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**)  
 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**)  
  
 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, statuse\_id=1)  
  
 **for** name, value **in** data.items():  
 **if** name.startswith(**"product\_in\_basket\_"**):  
 products\_in\_basket\_id = name.split(**"product\_in\_basket\_"**)[1]  
 product\_in\_basket = ProductInBasket.objects.get(id=products\_in\_basket\_id)  
 product\_in\_basket.nmb = value  
 product\_in\_basket.order = order  
 product\_in\_basket.save(force\_update=**True**)  
 Product\_in\_order.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)  
  
 **else**:  
 print(**"no"**)  
 **return** render(request, **'orders/checkout.html'**, locals())

products\ admin.py

**from** django.contrib **import** admin  
**from** .models **import** \*  
  
**class** ProductImageInline(admin.TabularInline):  
 model = Product\_image  
 extra = 0  
  
**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)  
  
  
**class** Product\_imageAdmin(admin.ModelAdmin):  
 list\_display = [field.name **for** field **in** Product\_image.\_meta.fields]  
  
 **class** Meta:  
 model = Product\_image  
  
  
admin.site.register(Product\_image, Product\_imageAdmin)

products\ models.py

**from** django.db **import** models  
  
**class** Product(models.Model):  
 name = models.CharField(max\_length=50, blank=**True**, null=**True**, default=**None**)  
 price = models.DecimalField(max\_digits=10, decimal\_places=2, default=0)  
 discount = models.IntegerField(default=0)  
 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 "Product %s"** % self.name  
  
 **class** Meta:  
 verbose\_name = **"Product"** verbose\_name\_plural = **"Products"  
  
  
class** Product\_image(models.Model):  
 product = models.ForeignKey(Product, blank=**True**, null=**True**, default=**None**, on\_delete=models.CASCADE)  
 image = models.ImageField(upload\_to=**"images of products/"**)  
 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 "Image %s"** % self.id  
  
 **class** Meta:  
 verbose\_name = **"Image"** verbose\_name\_plural = **"Images"**

products\ urls.py

**from** django.conf.urls **import** url  
**from** products **import** views  
  
urlpatterns = [  
 url(**r'^product/(?P<product\_id>\w+)/$'**, views.product, name=**'product'**),  
]

products\ views.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())

static\static\_dev\css\ landing.css

**html**, **body**{  
 **height**: 100%;  
 **max-height**: 100%;  
}  
  
.**top-container**{  
 **padding-top**: 13%;  
}  
  
.**main-container**{  
 **background-color**: **aliceblue**;  
 **height**: 100%;  
 **color**: **midnightblue**;  
}  
  
.**main-container form**{  
 **margin-top**: 30**px**;  
}  
  
.**title**{  
 **color**: **midnightblue**;  
 **text-align**: **center**;  
}  
  
.**btn-blue**{  
 **background-color**: **midnightblue**;  
 **color**: **white**;  
 **border**: **none**;  
}  
  
.**btn-blue**:**hover**{  
 **background-color**: **blue**;  
 **color**: **white**;  
 **border**: **none**;  
}  
  
.**text-center**{  
 **margin-top**: 30**px**;  
}

static\static\_dev\css\ landing.css

**html**, **body**{  
 **height**: 100%;  
 **max-height**: 100%;  
 **background-color**: **aliceblue**;  
}  
  
.**navbar** {  
 **background-color**: **white**;  
 **margin-bottom**: 5**px**;  
}  
  
.**wrapper**{  
 **min-height**: 100%;  
}  
  
.**wrapper-content**{  
 **overflow**: **auto**;  
 **padding-bottom**: 180**px**; }  
  
.**footer**{  
 **position**: **relative**;  
 **margin-top**: -150**px**; **height**: 50**px**;  
 **clear**: **both**;  
 **background-color**: **white**;  
}  
  
.**section\_top**{  
 **margin-bottom**: 20**px**;  
}  
.**product-item**{  
 **height**: 400**px**;  
 **background-color**: **white**;  
 **border**: 4**px solid midnightblue**;  
 **position**: **relative**;  
 **padding**: 5**px**;  
 **text-align**: **center**;  
 **margin-bottom**: 40**px**;  
}  
  
.**btn-blue**{  
 **background-color**: **midnightblue**;  
 **color**: **white**;  
 **border**: **none**;  
}  
  
.**btn-blue**:**hover** {  
 **background-color**: **blue**;  
 **color**: **white**;  
 **border**: **none**;  
}  
  
.**text-center-home**{  
 **position**: **absolute**;  
 **color**: **blue**;  
 **left**: 50%;  
 **bottom**: 10**px**;  
 **transform**: **translateX**(-50%);  
}  
  
.**discount-container**{  
 **position**: **absolute**;  
 **top**: 35%;  
 **background**: **deepskyblue**;  
 **width**: 30%;  
 **color**: **black**;  
 **font-weight**: 700;  
 **padding**: 3**px**;  
}  
  
.**text-center**{  
 **color**: **blue**;  
}  
  
.**price\_container**{  
 **color**: **black**;  
}  
  
.**description\_container**{  
 **color**: **black**;  
 **margin-bottom**: 20**px**;  
}  
  
.**product-image-item**{  
 **margin-bottom**: 5**px**;  
}  
  
.**form-group**{  
 **color**: **black**;  
}  
  
.**navbar-top**{  
 **background-color**: **aliceblue**;  
 **border**: **none**;  
 **border-radius**: 0;  
}  
  
.**basket-container**{  
 **position**: **relative**;  
 **width**: 500**px**;  
 **padding**: 15**px** 10**px**;  
}  
  
.**basket-items**{  
 **position**: **absolute**;  
 **top**: 50**px**;  
 **width**: 100%;  
 **background-color**: **powderblue**;  
 **z-index**: 10;  
 **padding**: 10**px**;  
}  
  
.**form-error**{  
 **color**: **red**;  
}

templates\base.html

*<!-- Побудова скелету сторінки сайту -->*

{% **load static** %}  
<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>Title</**title**>  
  
 <**link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.6/css/bootstrap.min.css"**>  
  
 <**link rel="stylesheet" href="**{% **static 'static\_dev/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**>  
  
 <**script src="**{% **static 'static\_dev/js/scripts.js'** %}**"**></**script**>  
</**body**>  
</**html**>

templates\footer.html

*<!-- Побудова footer сторінок сайту з вказаними контактними даними магазину -->*

<**footer class="footer"**>  
 <**div class="container"**>  
 <**div class="text"**>  
 <**h4**>Контактный телефон: (000)-0000-000</**h4**>  
 <**h4**>Адрес: ....</**h4**>  
 <**h4**>Email: ....</**h4**>  
 </**div**>  
 </**div**>  
</**footer**>

templates\navbar.html

*<!-- Побудова navbar сторінок сайту -->* <**nav class="navbar navbar-default navbar-top"**>  
 <**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="#"**>Walenor-store</**a**>  
 </**div**>  
 <**div id="navbar" class="navbar-collapse collapse"**>  
 <**ul class="nav navbar-nav"**>  
 <**li class="active"**><**a href="/"**>Главная</**a**></**li**>  
 </**ul**>  
 <**ul class="nav navbar-nav navbar-right"**>  
 <**li**>  
 <**div class="basket-container"**>  
 <**a href="**{% **url 'checkout'** %}**"**>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** %}  
 </**ul**>  
 </**div**>  
 </**div**>  
 </**li**>  
 </**ul**>  
 </**div**></**div**>  
 </**nav**>

templates\landing\landing.html

*<!-- Побудова Початкової сторінки сайту -->*

{% **load static** %}  
<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>Title</**title**>  
  
 <**link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.6/css/bootstrap.min.css"**>  
  
 <**link rel = "stylesheet" href="**{% **static 'static\_dev/css/landing.css'** %}**"**>  
</**head**>  
<**body**>  
 <**div class="main-container"**>  
 <**div class="container top-container"**>  
 <**div class="row"**>  
 <**div class="col-lg-8 col-lg-offset-2"**>  
 <**h1 class="title"**>Приветствую, покупатель<**br**>  
 Оставьте ваши email и имя</**h1**>  
 <**div**>  
 <**form action="" method="post" class="form-inline text-center"**>{% **csrf\_token** %}  
 <**div class="form-group"**>  
 <**label**>Email: </**label**>  
 <**input name="**{{ **form**.**email**.**name** }}**" class="form-control"**>  
 </**div**>  
 <**div class="form-group"**>  
 <**label**>Имя: </**label**>  
 <**input name="**{{ **form**.**name**.**name** }}**" class="form-control"**>  
 </**div**>  
 <**button type="submit" class="btn btn-blue"**>Сохранить</**button**>  
 </**form**>  
 </**div**>  
 </**div**>  
 </**div**>  
 <**div class="text-center"**>  
 <**h3**><**a href="/"**>Перейти на главную страницу</**a**></**h3**>  
 </**div**>  
 </**div**>  
 </**div**>  
</**body**>  
</**html**>

templates\landing\home.html

*<!-- Побудова Головної сторінки сайту -->*

{% **extends 'base.html'** %}  
{% **load static** %}  
  
{% **block content** %}  
 <**div class="section section1"**>  
 <**div class="container"**>  
 <**div class="row"**>  
 {% **for product\_image in products\_images** %}  
 <**div class="col-lg-3"**>  
 <**div class="product-item"**>  
 <**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**>Скидка {{ **product\_image**.**product**.**discount** }}% !!</**span**>  
 </**div**>  
 {% **endif** %}  
 <**h4**>{{ **product\_image**.**product**.**name** }}</**h4**>  
 <**P**>  
 {{ **product\_image**.**product**.**description**|**truncatechars\_html**:100 }}  
 </**P**>  
 <**div class="text-center-home"**>  
 <**h3**>{{ **product\_image**.**product**.**price** }} грн.</**h3**>  
 </**div**>  
 </**div**>  
 </**div**>  
 {% **endfor** %}  
 </**div**>  
 </**div**>  
 </**div**>  
{% **endblock** %}

templates\orders\checkout.html

*<!-- Побудова сторінки кошика користувача -->*

{% **extends 'base.html'** %}  
{% **load static** %}  
  
{% **block content** %}  
 <**div class="container"**>  
 <**div class="row"**>  
 {% **if products\_in\_basket** %}  
 <**div class="col-lg-8 col-lg-offset-2"**>  
 <**h3 class="text-center"**>Товары в корзине</**h3**>  
 <**h5 class="text-center"**>Общая стоимость:{{ **request**.**session**.**basket**.**total\_amount** }} грн.</**h5**>  
 {{ **request**.**session**.**basket**.**products** }}  
 <**form class="form-horizontal" method="post"**>{% **csrf\_token** %}  
 <**div class="form-group"**>  
 <**div class="col-lg-4"**>  
 <**label**>Имя</**label**>  
 <**span class="form-error"**>{{ **form**.**name**.**errors** }}</**span**>  
 <**input type="text" name="**{{ **form**.**name**.**html\_name** }}**" class="form-control"  
 value="**{{ **request**.**POST**.**name** }}**"** >  
 </**div**>  
 </**div**>  
 <**div class="form-group"**>  
 <**div class="col-lg-4"**>  
 <**label**>Телефон</**label**>  
 <**span class="form-error"**>{{ **form**.**phone**.**errors** }}</**span**>  
 <**input type="text" name="**{{ **form**.**phone**.**html\_name** }}**" class="form-control"  
 value="**{{ **request**.**POST**.**phone** }}**"** >  
 </**div**>  
 </**div**>  
 <**table class="table text-center"**>  
 <**thead**>  
 <**tr**>  
 <**th**>Товар</**th**>  
 <**th**>Количество</**th**>  
 <**th**>Цена за единицу</**th**>  
 <**th**>Цена</**th**>  
 </**tr**>  
 </**thead**>  
 <**tbody**>  
 {% **for product\_in\_basket in products\_in\_basket** %}  
 <**tr**>  
 <**td**>  
 {{ **product\_in\_basket**.**product**.**name** }}  
 </**td**>  
 <**td**>  
 <**input value="**{{ **product\_in\_basket**.**nmb** }}**" type="number" class="product-in-basket-nmb" name="product\_in\_basket\_**{{ **product\_in\_basket**.**id** }}**"**>  
 </**td**>  
 <**td**>  
 <**span class="product-price"**>  
 {{ **product\_in\_basket**.**price\_per\_item** }}  
 </**span**>  
 </**td**>  
 <**td**>  
 <**span class="total-product-in-basket-amount"**>  
 {{ **product\_in\_basket**.**total\_price** }}  
 </**span**>  
 </**td**>  
  
 </**tr**>  
 {% **endfor** %}  
 </**tbody**>  
 </**table**>  
 <**div class="text-center"**>  
 <**b**>Сумма заказа: </**b**> <**span id="total\_order\_amount"**></**span**> <**b**> грн.</**b**>  
 </**div**>  
 <**hr**>  
 <**div class="text-center"**>  
 <**button type="submit" class="btn btn-blue btn-lg"**>Заказать</**button**>  
 </**div**>  
 </**form**>  
 </**div**>  
 {% **else** %}  
 <**h2 class="text-center"**>В корзине нет товаров</**h2**>  
 {% **endif** %}  
 </**div**>  
 </**div**>  
{% **endblock** %}

static\static\_dev\js\scripts.js

**$**(***document***).ready(**function**() {  
 **var** form = **$**(**'#form\_buying\_product'**);  
 ***console***.log(form);  
   
 **function** *basketUpdating*(product\_id, nmb, is\_delete) {  
 **var** data = {};  
 data.**product\_id** = product\_id;  
 data.**nmb** = nmb;  
 **var** csrf\_token = **$**(**'#form\_buying\_product [name="csrfmiddlewaretoken"]'**).val();  
 data[**"csrfmiddlewaretoken"**] = csrf\_token;  
  
 **if** (is\_delete){  
 data[**"is\_delete"**] = **true**;  
 }  
  
 **var** url = form.attr(**"action"**);  
  
 ***console***.log(data)  
  
 **$**.**ajax**({  
 **url**: url,  
 **type**: **'POST'**,  
 **data**: data,  
 **cache**: **true**,  
 success: **function** (data) {  
 ***console***.log(**"OK"**);  
 ***console***.log(data.products\_total\_nmb);  
 **if** (data.products\_total\_nmb || data.products\_total\_nmb == 0){  
 **$**(**'#basket\_total\_nmb'**).text(**"("**+data.products\_total\_nmb+**")"**);  
 ***console***.log(data.products);  
 **$**(**'.basket-items ul'**).**html**(**""**);  
 **$**.each(data.products, **function** (k, v) {  
 **$**(**'.basket-items ul'**).append(**'<li>'**+v.**name**+**', '** + v.**nmb** + **'шт. '** + **'по '** + v.price\_per\_item + **'грн. '** +  
 **'<a class="delete-item" href="" data-product\_id="'**+v.**id**+**'">X</a>'**+  
 **'</li>'**);  
 });  
 }  
 },  
 error: **function** () {  
 ***console***.log(**"error"**)  
 }  
 })  
 }  
   
 form.on(**'submit'**, **function** (e) {  
 e.preventDefault();  
 ***console***.log(**'123'**);  
 **var** nmb = **$**(**'#number'**).val();  
 ***console***.log(nmb);  
 **var** submit\_btn = **$**(**'#submit\_btn'**);  
 **var** product\_id = submit\_btn.data(**"product\_id"**);  
 **var** product\_name = submit\_btn.data(**"product\_name"**);  
 **var** product\_price = submit\_btn.data(**"product\_price"**);  
 ***console***.log(product\_id);  
 ***console***.log(product\_name);  
 ***console***.log(product\_price);  
  
 *basketUpdating*(product\_id, nmb, **is\_delete** = **false**)  
  
 });  
  
 **function** *showingBasket*(){  
 **$**(**'.basket-items'**).removeClass(**'hidden'**);  
 };  
  
 **$**(**'.basket-container'**).mouseover(**function**(){  
 *showingBasket*();  
 });  
  
 **$**(***document***).on(**'click'**, **'.delete-item'**, **function**(e){  
 e.preventDefault();  
 **product\_id** = **$**(**this**).data(**"product\_id"**);  
 **nmb** = 0;  
 *basketUpdating*(**product\_id**, **nmb**, **is\_delete** = **true**)  
 });  
  
 **function** *calculatingBasketAmount*(){  
 **var** total\_order\_amount = 0;  
 **$**(**'.total-product-in-basket-amount'**).each(**function** () {  
 total\_order\_amount += *parseFloat*(**$**(**this**).text());  
 });  
 **$**(**'#total\_order\_amount'**).text(total\_order\_amount.toFixed(2))  
 };  
 **$**(***document***).on(**'change'**, **".product-in-basket-nmb"**, **function** () {  
 **var** current\_nmb = **$**(**this**).val();  
 **var** current\_tr = **$**(**this**).closest(**'tr'**);  
 **var** current\_price = *parseFloat*(current\_tr.find(**'.product-price'**).text()).toFixed(2);  
 **var** total\_amount = *parseFloat*(current\_nmb\*current\_price).toFixed(2);  
 current\_tr.find(**'.total-product-in-basket-amount'**).text(total\_amount);  
  
 *calculatingBasketAmount*();  
 });  
  
 *calculatingBasketAmount*();  
});