Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Ляхова Євгена Андрійовича*
(ПІБ)

академічної групи *121-19з-1*
(шифр)

спеціальності *121*
(код і назва спеціальності)

освітньої програми Інженерія програмного забезпечення
(назва освітньої програми)

на тему: *Розробка веб сайту інтернет магазину комп'ютерних комплектуючих*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи				
розділів:				
спеціальний	*доц. Кабак Л.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*старш. викл. Мартиненко А.А.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-19з-1* (група) *Ляхова Євгена Андрійовича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб сайту інтернет магазину комп'ютерних комплектуючих*

затверджена наказом ректора НТУ «ДП» від *21.05.2023 р.* № *770-л*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПО й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *доц. Кабак Л.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Ляхов Є.А.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *29.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *10.06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 75 с., 50 рис., 3 дод., 14 джерел.

Об'єкт розробки: Розробка веб сайту інтернет магазину комп’ютерних комплектуючих.

Мета дипломного проекту: забезпечити зручний інструментарій для візуального планування задач, їх простеження та інтерактивної взаємодії між елементами веб сторінки.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної області, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування системи, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження застосунку, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної підсистеми, проведений підрахунок вартості роботи по створенню застосунку та розраховано час на його створення.

Актуальність даного програмного забезпечення визначається необхідністю користувачів планувати свою цілі та задачі, а також простежувати їх за допомогою зручного та зрозумілого інтерфейсу.

Список ключових слів: ВЕБ-ДОДАТОК, REACT, REDUX, JAVASCRIPT, CSS, HTML, JSON, DOM, STORE

РЕФЕРАТ

Explanatory note: 75 pages, 50 pics, 3 apps, 14 sources.
Object of development: a web application for visualization of task planning.

The purpose of the diploma project: to provide convenient tools for visual planning of tasks, their tracking and interactive interaction between the elements of the web page.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the system, determines the input and output data, provides the characteristics of the parameters of hardware, describes the call and application load, describes the program .

In the economic section, the complexity of the developed information subsystem is determined, the cost of work on creating the application is calculated and the time for its creation is calculated.

The relevance of this software is determined by the need for users to plan their goals and objectives, as well as track them through a user-friendly interface.

List of keywords: WEB APP, REACT, REDUX, JAVASCRIPT, CSS, HTML, JSON, DOM, STORE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД - база даних;

СУБД - система управління базами даних;

UI - User Interface, інтерфейс користувача;

ІС - інформаційна система;

ЕОМ - електронно-обчислювальна машина; ПЗ – програмне забезпечення;

HTTP - HyperText Transfer Protocol; DOM – Document Object Model;

HTML - HyperText Markup Language; CSS – Cascading Style Sheets;

JSON - JavaScript Object Notation; UX/UI – User Interface/User Experience.

ВСТУП

У сучасному світі онлайн-торгівля стала однією з найбільш швидкорозвиваючихся галузей економіки. Рік за роком зростає кількість людей, які віддають перевагу здійсненню покупок в Інтернеті замість традиційного способу в магазинах. Це пов'язано зі зручністю та швидкістю покупок, можливістю порівняти ціни та характеристики товарів, а також можливістю зробити покупку з будь-якої точки світу.

Однією з найбільш важливих галузей онлайн-торгівлі є комп'ютерна техніка та комплектуючі. Кожен, хто займається будь-якою сферою діяльності, повинен мати доступ до якісної та надійної комп'ютерної техніки. Однак, пошук необхідних комплектуючих у реальному магазині може бути складним та часозатратним процесом.[7]

Тому метою даної кваліфікаційної роботи є розробка веб-сайту інтернет-магазину комп'ютерних комплектуючих. Головною метою є створення зручної та простої платформи для покупки комп'ютерних комплектуючих онлайн. Процес розробки веб-сайту буде включати в себе аналіз вимог клієнтів, розробку зручного та інтуїтивно зрозумілого інтерфейсу, забезпечення безпеки та швидкості операцій з покупкою товарів.

Результатом даної кваліфікаційної роботи буде готовий веб-сайт інтернет-магазину комп'ютерних комплектуючих, який буде відповідати всім вимогам та стандартам онлайн-торгівлі. Даний веб-сайт забезпечить клієнтам можливість легко та швидко знайти необхідні комплектуючі для своїх комп'ютерів та здійснити покупку в зручний для них спосіб.

Таким чином, дана кваліфікаційна робота має важливе значення для розвитку сучасної онлайн-торгівлі та забезпечення якості та зручності покупок комп'ютерних комплектуючих. Вона має на меті допомогти клієнтам швидко та зручно знайти необхідні комплектуючі та зробити покупку без витрати багато часу та зусиль.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальні відомості з предметної галузі

Інтернет-магазини є невід'ємною частиною сучасної економіки, тому що їх зручність та доступність забезпечують попит на товари та послуги, що пропонуються в мережі Інтернет. Веб-сайти інтернет-магазинів, які пропонують комп'ютерні комплектуючі, стали популярними серед відвідувачів мережі Інтернет завдяки зручності та широкому асортименту товарів.

Предметна галузь, яку буде відображати кваліфікаційна робота, полягає в розробці веб-сайту для інтернет-магазину комп'ютерних комплектуючих. У зв'язку з швидким розвитком технологій та зростанням попиту на комп'ютерні комплектуючі, розробка такого веб-сайту є досить актуальною задачею. Веб-сайт інтернет-магазину з комп'ютерними комплектуючими дозволить користувачам швидко та зручно здійснювати покупки, дізнаватися про новинки та акції магазину.

Основними конкурентними перевагами інтернет-магазину з комп'ютерними комплектуючими є доступні ціни, швидкість та зручність оформлення замовлення, а також широкий асортимент продукції від провідних виробників. Такі інтернет-магазини можуть використовуватися як фізичними особами, так і підприємствами, які займаються продажем комп'ютерних комплектуючих.[1]

Таким чином, загальні відомості з предметної галузі є ключовим елементом аналізу при розробці веб-сайту інтернет-магазину комп'ютерних комплектуючих. Отже, для того, щоб забезпечити ефективне використання технологій у розробці програмного виробу, необхідно глибоко проаналізувати предметну галузь.

У контексті даного дослідження, предметна галузь - це комп'ютерна техніка та її комплектуючі, а також електронна комерція та маркетинг. Комп'ютерна техніка є однією з найшвидше зростаючих технологій, що забезпечує розвиток багатьох галузей, включаючи особисті комп'ютери, мобільні пристрої, хмарні технології та багато іншого.[3]

Інтернет-магазин комп'ютерних комплектуючих - це веб-ресурс, який дозволяє покупцям швидко та зручно здійснювати покупки комп'ютерної техніки та комплектуючих. Інтернет-магазини стали невід'ємною частиною сучасної економіки та торгівлі, що забезпечує зручність та доступність для клієнтів з різних куточків світу. Також, важливо зазначити, що електронна комерція є актуальною та перспективною галуззю, оскільки ринок IT-технологій швидко зростає та розвивається, а це призводить до збільшення кількості онлайн-магазинів та зростання конкуренції в цій сфері.[5]

Також слід зазначити, що ринок комп’ютерної техніки і комплектуючих в Україні активно розвивається, що вказує на потребу у відповідних інструментах для продажу та обслуговування продуктів.

На виробництві комп’ютерних комплектуючих працює значна кількість підприємств, які є потенційними користувачами розроблюваної системи. До них відносяться компанії, що спеціалізуються на виробництві та продажу електронних компонентів, системних блоків, мережевих пристроїв та інших комплектуючих.[3]

Одже важливо відзначити, що в Україні досить широко поширена практика використання інтернет-магазинів для здійснення покупок. Зокрема, велика кількість користувачів використовують інтернет для пошуку необхідних товарів та послуг, що забезпечує попит на веб-сайти інтернет-магазинів.[1]

1.2 Призначення розробки та галузь застосування

Єдиною можливістю для інтернет-магазинів поборотися за перше місце в пошукових системах та залучити увагу цільової аудиторії є пропозиція актуальних, зручних та корисних ресурсів.[9] У цьому контексті, веб-додаток "ComPartsUA" є об'єктом розробки даного проекту, що дозволяє користувачам знайти комплектуючі для свого комп'ютера відомих виробників за різними категоріями. Магазин має бути оснащений сортуванням за категоріями, ціновими фільтрами та підбором виробів за рядом критеріїв. Крім того, користувачі повинні мати можливість перегляду свого замовлення, внесення змін та безпечної передачі своїх даних адміністратору магазину для формування замовлення.

Для адміністратора магазину буде створена спеціальна додаткова панель, у якій він зможе переглядати та редагувати замовлення, переглядати дані клієнтів, створювати нові позначки для категорій та редагувати контент магазину. Збереження даних у базі даних дозволить структурувати всю інформацію про магазин та користувачів.

Backend-частина додатку має забезпечити можливість редагування контенту магазину, безпечну обробку інформації користувачів при створенні замовлень та підвищення продажів через швидкість роботи інтернет-магазину та ефективну обробку замовлень у адміністративній панелі.

1.3 Підстава для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студен виконує кваліфікаційну роботу. Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки «виконання кваліфікаційної роботи» є:

- освітня програма 121 «Інженерія програмного забезпечення»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка»;

– завдання на кваліфікаційну роботу на тему «Розробка веб сайту інтернет магазину комп’ютерних комплектуючих».

1.4 Постановка завдання

Метою кваліфікаційної роботи є створення та управління back-end частиною Інтернет магазину, який продає комп'ютерні комплектуючі.

Програмне забезпечення має мету надання універсального інструменту для відображення контенту та управління цим інтернет-магазином. Для досягнення цієї мети необхідно створити програму, яка забезпечує:

- можливість редагування інформації у адмін-панелі;

- формування web-сторінок на основі шаблону, використовуючи інформацію з бази даних;

- відображення сторінок на різних пристроях;

- оформлення та редагування замовлень.

Для досягнення цих цілей необхідно:

- вивчити предметну галузь розв'язуваної задачі;

- створити алгоритм для реалізації поставленого завдання;

- створити базу даних, клієнтську та серверну програму для роботи з нею.

1.5 Вимоги до програми або програмного виробу

1.5.1 Вимоги до функціональних характеристик

Для досягнення запланованих цілей необхідно, щоб розроблене програмне забезпечення здійснювало наступні дії:

- зберігання даних підсистеми в реляційній базі даних;
- надання віддаленого доступу до програми через веб-браузер на комп'ютері користувача;
- зчитування вхідних даних з пристроїв, хмарних сервісів та за адресними посиланнями;
- обробка отриманої інформації та передача відповіді користувачу.

Для забезпечення виконання цих функцій та оптимальної роботи магазину, у програмі має бути реалізовано:

- можливість доступу до програми через веб-браузер;
- можливість інтеграції з платформою веб-сайту для продажу комплектуючих;
- наявність типової конфігурації, що дозволяє швидко ввести програму в експлуатацію;
- програмно-апаратна переносимість.[13]

1.5.2 Вимоги до інформаційної безпеки

Для уникнення некоректної роботи програми необхідно реалізувати:

- перевірку введених даних;
- обробку виняткових ситуацій;
- виведення повідомлень про помилки;
- можливість повторного введення даних;
- можливість безперервної роботи протягом не менше 120 годин (5 діб);
- підтримку роботи на різних платформах;
- вірогідність виникнення не більше 2 логічних помилок на 1000

операторів за 1 рік експлуатації;

- захист даних в разі відмови.[10]

1.5.3 Вимоги до складу та параметрів технічних засобів

Для забезпечення нормальної та безперебійної роботи даного інтернет-магазину необхідно, щоб комп'ютер, на якому функціонуватиме система інтернет-магазину, відповідав таким вимогам:

- процесор класу Intel Pantium з тактовою частотою не менш 2.4 ГГц та двума ядрами;
- доступ до мережі Internet;
- не менше 16 GB оперативної пам'яті;
- 1 Тб вільного місця на жорсткому диску;
- клавіатура;
- маніпулятор "миша".

Наведені вище технічні характеристики є рекомендованими, тобто якщо використовуються технічні засоби, які відповідають вимогам, що були визначені замовником щодо надійності, швидкості обробки даних та безпеки, програмний застосунок буде функціонувати відповідно до цих вимог.[6]

1.5.4 Вимоги до інформаційної та програмної сумісності

Для нормального функціонування програми необхідно, щоб програмне забезпечення персонального комп'ютера, на якому буде функціонувати веб-орієнтована система, відповідало наступним вимогам:

- операційна система Windows (7+), Linux, MacOs;
- веб-браузер Firefox / Google Chrome / Opera / Microsoft Edge / Safari.

Backend частина додатку має бути реалізована на мові програмування Python з використанням фреймворку Django та СУБД SQLite.

РОЗДІЛ 2

ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Розроблений програмний продукт спрямований на забезпечення використання двома різними типами користувачів - клієнтами і адміністраторами. Кожна з цих категорій має власний набір функціональних можливостей. Клієнт може:

- переглядати увесь список наявних товарів;
- додавати товари до кошика;
- реєструватись на сайті
- авторизовуватись на сайті за допомогою свого унікального логіна та пароля.

Адміністратор сайту на додачу до функцій клієнта може:

- додавати новий товар, категорії та фільтри;
- редагувати або видаляти товар, категорії та фільтри;
- авторизовуватись на сайті за допомогою свого унікального логіна та пароля;
- дивитись інформациію про усіх користувачів;
- редагувати дані користувачів;
- передивлятись статуси замовлень користувачів.

На малюнку, який наведено нижче, зображена діаграма використання, яка відображає основних учасників системи.

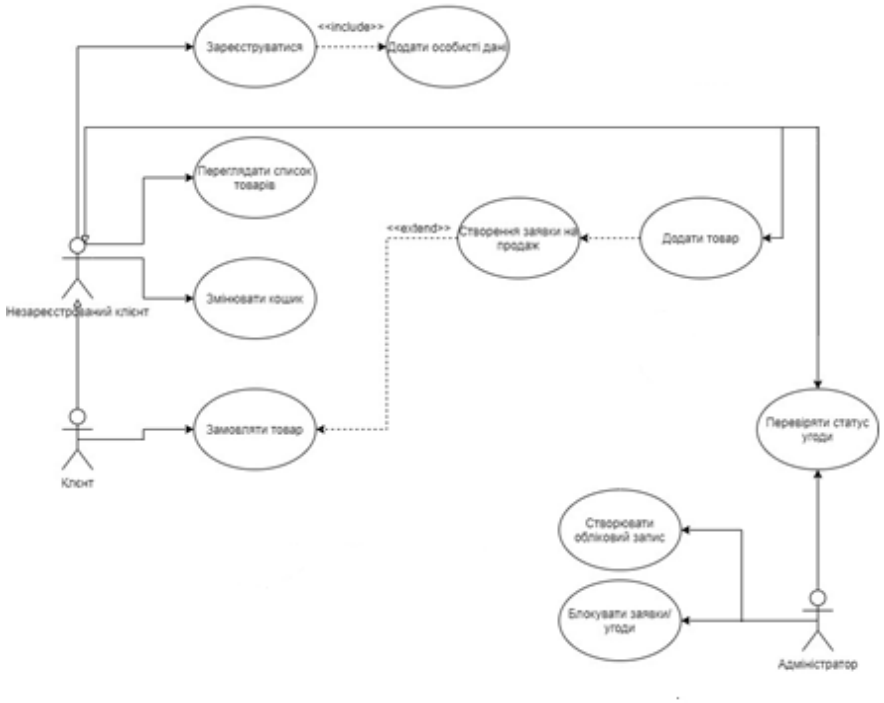


рис 2.1.1. Діаграма use-case

Незареєстрований клієнт може переглядати картки із товаром. Також він може зареєструватися та увійти до свого облікового запису для подальших взаємодій з сайтом. Після цього він може додавати товари в кошик, щоб у майбутньому оформити замовлення.

Адміністратор сайту може робити все те ж, що і звичайний клієнт, але також у нього є доступ до адмі-панелі, де він може переглядати та редагувати всі категорії та фільтри товарів, просамтривати дані користувач та вазимодіяти з ними, а також переглядати їх кошики.

2.2 Опис застосованих математичних методів

У даному програмному продукті нема потреби використовувати математичні методи, оскільки більша частина роботи є зв’язком бази даних,створення записів у ній та пошук необхідних записів.

2.3 Опис використаної архітектури та шаблонів проєктування

Розроблений продукт є серверною складовою веб-додатка, який базується на клієнт-серверній архітектурі. Класична трирівнева клієнт-серверна архітектура ґрунтується на принципах розподілу структури додатків між трьома рівнями логічних обчислень: постачальниками ресурсу (серверами), середовищем зберігання даних (базами даних) та споживачами ресурсу (клієнтами).

MVC

Model-View-Controller – схема розподілення даних програмного продукту, інтерфейсу користувача та керуючої логіки на три окремих компонента:

Модель (Model) надає дані й реагує на команди контролера, змінюючи свій стан;

Уявлення (View) відповідає за представлення даних моделі користувачеві, реагуючи на зміни моделі;

Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін.[3]

2.4 Опис моделей

Для правильного функціонування розробленого продукту необхідна база даних, яка буде зберігати всі необхідні дані. У цій базі даних інформація зберігається у вигляді сутностей, які автоматично створюються під час міграції моделей з файлу. Формат даних у базі даних може бути різним, залежно від типу інформації, яку вони представляють. Наприклад, дані можуть бути збережені у вигляді рядків (строк), чисел, дат з часом або логічних змінних.

Список моделей програмного продукту:

- незареєстрований користувач;
- зареєстрований користувач;
- адміністратор;
- персональні дані;
- продукт.

Модель користувача

Модель користувача є стандартною моделлю фреймворку Django (див. табл. 1.1).[2]

Таблиця 1.1

Модель користувача

Назва колонки	Опис
id	Унікальний ідентифікатор кожного користувача у системі
password	Пароль користувача
last_login	Час останнього входу користувача у систему
is_superuser	Змінна, котра виділяє користувача до групи суперкористувачів, в котрої є більше прав
last_name	Призвіще користувача
email	Електронна пошта користувача
is_staff	Змінна, котра виділяє користувача до групи персоналу, в котрої є більше прав
is_active	Змінна, котра показує, чи активний обліковий запис користувача, чи ні
date_joined	Дата реєстрації користувача
first_name	Ім’я користувача

Модель клієнта

У цій моделі зберігаються ті користувачі, котрі зареєструвалися на сайті. Модель створюється автоматично при реєстрації користувача у системі (див. табл. 1.2).

Таблиця 1.2

Модель клієнта

Назва колонки	Опис
idClient	Унікальний ідентифікатор кожного клієнта
personalDataId	Ключ до моделі персональних даних

Модель персональних даних

У цій моделі зберігаються особисті дані усіх користувачів. Модель створюється автоматично при реєстрації користувача у системі (див. табл. 1.3).

Таблиця 1.3

Модель персональних даних

Назва колонки	Опис
idPersonalData	Унікальний ідентифікатор кожного запису особистих даних користувачів
Login	Логін користувача

Модель покупки

У цій моделі зберігаються заявки на покупку від користувача. Модель створюється автоматично коли клієнт замовляє товар (див. табл. 1.4).

Таблиця 1.4

Модель покупки

Назва колонки	Опис
idPurchase	Унікальний ідентифікатор кожної заявки на покупку
dateOfRequest	Дата створення заявки
dateOfPurchase	Дата запланованої покупки товару
amount	Кількість товару
clientId	Ключ до моделі клієнта, котрий оформив заявку на покупку
isCoupon	Чи є у у заявці купон на знижку

Модель продукту

У цій моделі зберігаються дані товару, котрий додав фермер (див. табл. 1.5).

Таблиця 1.5

Модель продукту

Назва Колонки	Опис
idProduct	Унікальний ідентифікатор кожного товару
Name	Назва продукту
unitOfMeasure	Одиниця виміру товару

2.5. Опис структури програми та алгоритмів її функціонування

Розроблений програмний продукт розподілений по різнім файлам,структура котрих зображена нижче.

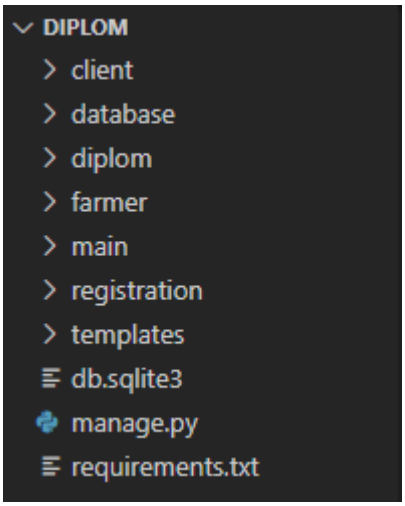


рис 2.5.1 Структура проекту

Проект складається з 5 додатків:

- client;
- database;
- diplom;
- main;
- registration;

А також каталога шаблонів HTML, бази даних, виконуючого файлу проектута текстового документу з вимогами для роботи проетку.

Додаток складається з:

- каталогу кешу байт-коду (__pycache____);

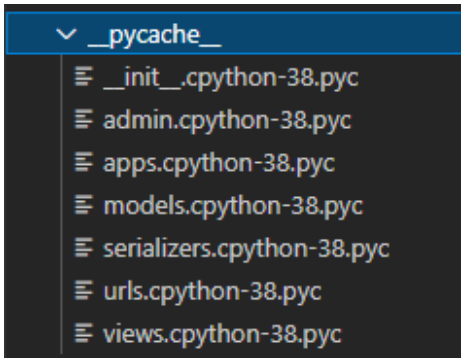


рис 2.5.2 Каталог міграцій

- каталогу з міграціями до бази даних (migrations);

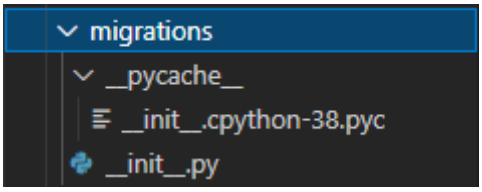


рис 2.5.3 Каталог кешу

- файл позначення пакетів Python (___init.py);
  файл моделей адміністратора (admin.py);
- файл конфігураціх додатку (apps.py); файл з моделями бази даних (models.py);
- файл з серіалайзерами моделей (serializers.py);файл з тестами коду (tests.py);
- файл маршрутизації додатку (urls.py);
- файл з представленнями моделей (views.py);

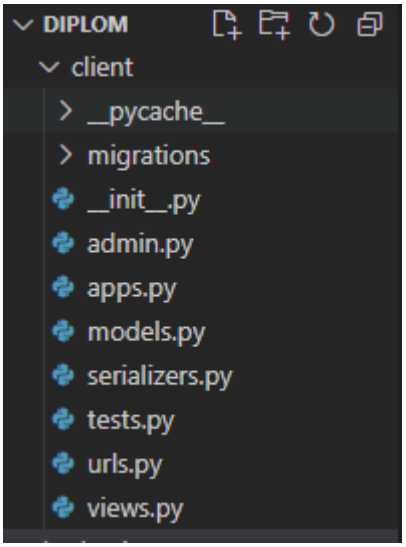


рис 2.5.4 Структура додатка

Всі інші додатки відрізняються тією ж самою структурою, за винятком значень, що містяться в файлах.

В проекті наявні три основні додатки: додаток бази даних (database), додаток реєстрації (main), додаток клієнта (client).

Знизу позначено основні класи та методи відповідно кожному файлудодатка.

Файл Orders/Models

```
class Order(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, related_name='orders')
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    status = models.BooleanField(default=False)
    discount = models.IntegerField(blank=True, null=True, default=None)

    class Meta:
        ordering = ('-created',)

    def __str__(self):
        return f"{self.user} - {self.id}"

    @property
    def get_total_price(self):
        total = sum(item.get_cost() for item in self.items.all())
        if self.discount:
            discount_price = (self.discount / 100) * total
            return int(total - discount_price)
        return total
```

```python
class OrderItem(models.Model):
    order = models.ForeignKey(Order, on_delete=models.CASCADE, related_name='items')
    product = models.ForeignKey(Product, on_delete=models.CASCADE, related_name='order_items')
    price = models.IntegerField()
    quantity = models.SmallIntegerField(default=1)

    def __str__(self):
        return str(self.id)

    def get_cost(self):
        return self.price * self.quantity


class Coupon(models.Model):
    code = models.CharField(max_length=30, unique=True)
    valid_from = models.DateTimeField()
    valid_to = models.DateTimeField()
    discount = models.IntegerField(validators=[MinValueValidator(0), MaxValueValidator(100)])
    status = models.BooleanField(default=False)

    def __str__(self):
        return self.code
```

рис 2.5.5 Файл orders/models

У цьому файлі клас **Order** та **Meta** відпоідають за створення та відновлення статусу замовлень.

Функція **get_total_price** розраховує кінцеву вартість кошику.

Клас **OrderItem** додає певний продукт до кошику, а клас Coupon взаємодіє з цим продуктом через купони, додаючи знижки.

Файл Cart/Views

```python
def detail(request):
    cart = Cart(request)
    return render(request, template_name='cart/detail.html', context={'cart': cart})


@require_POST
def cart_add(request, product_id):
    cart = Cart(request)
    product = get_object_or_404(Product, id=product_id)
    form = Add2CartForm(request.POST)

    if form.is_valid():
        data = form.cleaned_data
        cart.add(product=product, quantity=data['quantity'])
    return redirect('cart:detail')


def cart_remove(request, product_id):
    cart = Cart(request)
    product = get_object_or_404(Product, id=product_id)
    cart.remove(product)
    return redirect('cart:detail')
```

рис 2.5.6 Файл cart/views

У цьому файлі клас **detail** витягує усю інформацію про товар.

Клас **cart_add** викликає функцію додавання товару у кошик через інший скрипт, а клас **cart_remove** прибирає цей товар.

Файл Cart/URLS

```python
from django.urls import path

from cart import views

app_name = 'cart'

urlpatterns = [
    path('', views.detail, name='detail'),
    path('add/<int:product_id>/', views.cart_add, name='cart_add'),
    path('remove/<int:product_id>/', views.cart_remove, name='cart_remove'),
]
```

рис 2.5.7 Файл cart/urls

У цьому скрипті пов’язуються посилання самого сайту із логічними функціями та скриптами кошика.

Файл Cart/Forms

```python
from django import forms


class Add2CartForm(forms.Form):
    quantity = forms.IntegerField(min_value=1, max_value=9, widget=forms.NumberInput(attrs={'class': 'form-control',
                                                                                             'placeholder': 'quantity'}))
```

рис 2.5.8 Файл cart/forms

Даний файл містить форму-кнопку, яка розміщується на певній сторінці сайту зі своїм URL. Вона зв’язується із основними скриптами кошику и при натисканні на неї користувачем, виконує функції прив’язаних скриптів.

Файл Accounts/Views

```python
from django.contrib import messages
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout

# Create your views here.
from accounts.forms import UserLoginForm, UserRegistrationForm
from accounts.models import User


def user_login(request):
    if request.method == 'POST':
        form = UserLoginForm(request.POST)
        if form.is_valid():
            data = form.cleaned_data
            user = authenticate(request, email=data['email'], password=data['password'])
            if user is not None:
                login(request, user)
                messages.success(request, 'login successfully', 'success')
                return redirect('shop:home')
            else:
                messages.error(request, 'username or password is wrong', 'danger')
    else:
        form = UserLoginForm()
    return render(request, 'accounts/login.html', {'form': form})


def user_logout(request):
    logout(request)
    messages.success(request, 'logout successfully', 'success')
    return redirect('shop:home')
```

```python


def register(request):
    if request.method == 'POST':
        form = UserRegistrationForm(request.POST)
        if form.is_valid():
            data = form.cleaned_data
            user = User.objects.create_user(data['email'], data['full_name'], data['phone_number'], data['password'])
            login(request, user)
            messages.success(request, 'you registered successfully', 'success')
            return redirect('shop:home')
    else:
        form = UserRegistrationForm()
    return render(request, 'accounts/register.html', {'form': form})

```

рис 2.5.9 Файл accounts/views

Даний скрипт витягує, редагує та взаімодіє даними користувачів самої бази.

Клас **user_login** витягує з бази даних відповідний логін та пароль користувача, порівняє його, та видає відповідний результат. Якщо зв’язка з логіну та паролю співпадає з певною строкою бази, то результат виконання класу є вдалий і система пропустить користувача далі, а якщо хоч один з параметрів не співпав, то користувач побачить повідомлення про помилку.

Клас **register** бере з певних форм та інпутів користувача дані та створює новий обліковий запис в базі з логіну та паролю користувача. Потім функція перенаправляє користувача на головну сторінку.

Файл Accounts/URLS

```
from django.urls import path

from accounts import views

app_name = "account"

urlpatterns = [
    path('login/', views.user_login, name='login'),
    path('logout/', views.user_logout, name='logout'),
    path('register/', views.register, name='register')
]

```

рис 2.5.10 Файл accounts/urls

У цьому скрипті пов’язуються посилання самого сайту із логічними функціями, базою та скриптами авторизації, логіну та виходу користувача із системи.

Файл Accounts/Models

```python
from django.db import models
from django.contrib.auth.models import AbstractBaseUser

# Create your models here.
from accounts.managers import MyUserManager


class User(AbstractBaseUser):
    email = models.EmailField(max_length=100, unique=True)
    full_name = models.CharField(max_length=100)
    phone_number = models.CharField(max_length=100)
    is_admin = models.BooleanField(default=False)
    is_active = models.BooleanField(default=True)

    objects = MyUserManager()

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['full_name', 'phone_number']

    def __str__(self):
        return self.email

    def has_perm(self, perm, obj=None):
        """Does the user have a specific permission?"""
        # Simplest possible answer: Yes, always
        return True

    def has_module_perms(self, app_label):
        """Does the user have permissions to view the app `app_label`?"""
        # Simplest possible answer: Yes, always
        return True
```

```python

    @property
    def is_staff(self):
        """Is the user a member of staff?"""
        # Simplest possible answer: All admins are staff
        return self.is_admin

```

рис 2.5.10 Файл accounts/models

Даний скрипт перевіряє стани користувача. Чи має він права адміністратора, чи умовного працівника, які властивості повинен зберігати, у даному випадку це пошта, повне ім’я, телефоний номер, стан адміна та активності. Далі ці параметри може бачити адміністратор у адмін-панелі.

Файл Accounts/Managers

```python
from django.contrib.auth.models import BaseUserManager


class MyUserManager(BaseUserManager):
    def create_user(self, email, full_name, phone_number, password):
        if not email:
            raise ValueError('users must have Email')
        if not full_name:
            raise ValueError('users must have Full Name')

        user = self.model(email=self.normalize_email(email), phone_number=phone_number, full_name=full_name)
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, email, full_name, phone_number, password):
        user = self.create_user(email, full_name, phone_number, password)
        user.is_admin = True
        user.save(using=self._db)
        return user

```

рис 2.5.11 Файл accounts/managers

Класс **MyUserManager** дозволяє додавати певних користувачів вручну самому адміністратору за допомогою адмін-панелі.

Файл Accounts/Forms

```python
from django import forms
from accounts.models import User
from django.contrib.auth.forms import ReadOnlyPasswordHashField


class UserCreationForm(forms.ModelForm):
    password1 = forms.CharField(label='password', widget=forms.PasswordInput)
    password2 = forms.CharField(label='confirm password', widget=forms.PasswordInput)

    class Meta:
        model = User
        fields = ('email', 'full_name', 'phone_number')

    def clean_password2(self):
        cd = self.cleaned_data
        if cd['password1'] and cd['password2'] and cd['password1'] != cd['password2']:
            raise forms.ValidationError('passwords must match')
        return cd['password2']

    def save(self, commit=True):
        user = super().save(commit=False)
        user.set_password(self.cleaned_data['password1'])
        if commit:
            user.save()
        return user


class UserChangeForm(forms.ModelForm):
    password = ReadOnlyPasswordHashField()

```

```python
    class Meta:
        model = User
        fields = ('email', 'password', 'full_name', 'phone_number')

    def clean_password(self):
        return self.initial['password']


class UserLoginForm(forms.Form):
    email = forms.EmailField(widget=forms.EmailInput(attrs={'class': 'form-control'}))
    password = forms.CharField(widget=forms.PasswordInput(attrs={'class': 'form-control'}))


class UserRegistrationForm(forms.Form):
    email = forms.EmailField(widget=forms.EmailInput(attrs={'class': 'form-control'}))
    full_name = forms.CharField(widget=forms.TextInput(attrs={'class': 'form-control'}))
    phone_number = forms.CharField(max_length=12,widget=forms.TextInput(attrs={'class': 'form-control'}))
    password = forms.CharField(widget=forms.PasswordInput(attrs={'class': 'form-control'}))
```

рис 2.5.12 Файл accounts/forms

Цей скрипт описує задані кнопки, за допомогою користувач або адміністратор може створювати новий обліковий запис, заходити у власний та виходити з нього.

Клас **UserCreationForm** відповідає якраз за збір информації з отриманих текстових полів інпутів та зберігає ці дані.

Клас **UserChangeForm** допомагає адміністратору продивитись певні дані певних користувачів також через кнопки. Проте у адміністратора не має можливості змінити пароль користувача.

Клас **UserLoginForm** додає логічну кнопку та інпути для заходу користувача на сайт. В ці інпути користувач вводить свої дані, натискає на кнопку «Увійти», та після, система вже йде до функцій та скриптів порівняння його даних з базою, робить свої висновки та вирішує, чи дозволити користувачеві увійти в систему.

Клас **UserRegistrationForm** додає кнопку та інпути для створення нового облікового запису користувача. При заповненні усіх вимагаючих полей, система пропонує користувачеві збергігти цю інформацію та свторити новий акаунт.

Файл Accounts/Admin

```python
# Register your models here.
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import Group

from accounts.forms import UserChangeForm
from accounts.models import User


class UserAdmin(BaseUserAdmin):
    form = UserChangeForm
    add_form = UserCreationForm
    list_display = ('full_name', 'phone_number', 'email', 'is_admin')
    list_filter = ('is_admin',)
    fieldsets = (
        ('Main', {'fields': ('full_name', 'email', 'phone_number', 'password')}),
        ('Personal info', {'fields': ('is_active',)}),
        ('Permissions', {'fields': ('is_admin',)}),
    )
    add_fieldsets = (
        (None, {
            'fields': ('full_name', 'phone_number', 'email', 'password1', 'password2')
        }),
    )
    search_fields = ('email',)
    ordering = ('email',)
    filter_horizontal = ()


admin.site.register(User, UserAdmin)
admin.site.unregister(Group)

```

рис 2.5.13 Файл accounts/admin

Даний скрипт дозволяє керувати інформацією користувачів через адмін панель. Тут є фільтри пошуку, які допомогають адміністратору знайти певний акаунт, якщо їз занадто багато, фільтри виводу інформації для більш зручного її огляду та форматування цих даних.

Файл Shop/URLS

```python
from django.urls import path
from shop import views

app_name = 'shop'

urlpatterns = [
    path('', views.home, name='home'),
    path('category/<slug:slug>/', views.home, name='category_filter'),
    path('<slug:slug>/', views.product_detail, name='product_detail'),
]

```

рис 2.5.14 Файл shop/admin

Даний скрипт зв’язує посилання магазину із категоріями товарів, самими продуктами з іншими виконуючими файлами.

Файл Shop/Models

```python
from django.db import models

# Create your models here.
from django.urls import reverse


class Category(models.Model):
    sub_category = models.ForeignKey('self', on_delete=models.CASCADE, related_name='sub_categories', null=True,
                                     blank=True)
    is_sub = models.BooleanField(default=False)
    name = models.CharField(max_length=200)
    slug = models.SlugField(max_length=200, unique=True)

    class Meta:
        ordering = ('name',)
        verbose_name = 'category'
        verbose_name_plural = 'categories'

    def __str__(self):
        return self.name

    def get_absolut_url(self):
        return reverse('shop:category_filter', args={self.slug})


class Product(models.Model):
    category = models.ManyToManyField(Category, related_name='products')
    name = models.CharField(max_length=200)
    slug = models.SlugField(max_length=200)
    image = models.ImageField(upload_to='products/%Y/%b/%d/')
    description = models.TextField()
    price = models.IntegerField()
    status = models.BooleanField(default=True)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)

    class Meta:
        ordering = ('name',)

    def __str__(self):
        return self.name

    def get_absolut_url(self):
        return reverse('shop:product_detail', args={self.slug, })

```

рис 2.5.15 Файл shop/models

Цей виконуючий файл пов’язує дані, отримані з інших логічним систем, такі як: категорія товарів та самі продукти з загальною базою даних.

Клас **Category** бере з бази категорію товарів. Він містить набір параметрів про сабкатегорію, її ім’я та позицію в каталозі.

Клас **Product** містить інформацію про певний товар. Він містить інформацію про категорію, в який було поміщено товар, його ім’я та зображення, опис, ціну, поточний статус та стан обробки.

Файл Shop/Admin

```python
import var_dump
from django.contrib import admin

# Register your models here.
from shop.models import Category, Product


@admin.register(Category)
class CategoryAdmin(admin.ModelAdmin):
    list_display = ('name', 'slug')
    prepopulated_fields = {'slug': ('name',)}


@admin.register(Product)
class ProductAdmin(admin.ModelAdmin):
    list_display = ('name', 'price', 'status')
    prepopulated_fields = {'slug': ('name',)}
    list_filter = ('status', 'created')
    list_editable = ('price', 'status')
    raw_id_fields = ('category',)
    actions = ('change_status',)

    @admin.action(description='change status of model')
    def change_status(self, request, queryset):
        rows_count = queryset.update(status=True)
        self.message_user(request, f'{rows_count} status has changed')

    # change_status.short_description = 'change status .'

```

рис 2.5.16 Файл shop/admin

Даний скрипт дозволяє адміністратору наповнювати сайт різноманітними товарами. Він містить усі необхідні функції для створення або оновлення даних про певний продукт. За допомогою класу **CategoryAdmin** адміністратор може відображати певні категорії товарів, а за допомогою класу **ProductAdmin** адміністратор може побачити інформацію про певний товар, його назву, статуси та віображенні фільтри.

Файл Settings

```
13  from pathlib import Path
14
15  # Build paths inside the project like this: BASE_DIR / 'subdir'.
16  BASE_DIR = Path(__file__).resolve().parent.parent
17
18  # Quick-start development settings - unsuitable for production
19  # See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/
20
21  # SECURITY WARNING: keep the secret key used in production secret!
22  SECRET_KEY = 'django-insecure-y^xlc@09_p0@_9t3g945%b6tnk0)6afm49rx7=a@4juxy!3u!*'
23
24  # SECURITY WARNING: don't run with debug turned on in production!
25  DEBUG = True
26
27  ALLOWED_HOSTS = []
28
29  # Application definition
30
31  INSTALLED_APPS = [
32      'django.contrib.admin',
33      'django.contrib.auth',
34      'django.contrib.contenttypes',
35      'django.contrib.sessions',
36      'django.contrib.messages',
37      'django.contrib.staticfiles',
38      'accounts.apps.AccountsConfig',
39      'shop.apps.ShopConfig',
40      'cart.apps.CartConfig',
41      'orders.apps.OrdersConfig',
42      'sorl.thumbnail',
43  ]
```

```
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 = 'django_online_shop.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [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',
            ],
        },
    },
]
```

```
WSGI_APPLICATION = 'django_online_shop.wsgi.application'

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

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

# Password validation
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH_USER_MODEL = 'accounts.User'

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.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/3.2/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/3.2/howto/static-files/

STATIC_URL = '/static/'

MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media/'

# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

```

рис 2.5.17 Файл /settings

Це є стандартний скрипт налаштувань проекту, де містяться усі додаткові бібліотеки проекту для нормального функціювання, їх залежності та властивості.

2.6 Обґрунтування та організація вхідних та вихідних даних програми

В програмному продукті, над яким велась розробка, вхідними даними є інформація, яку користувач вводить з клавіатури у спеціальні екранні форми для введення. Всі вхідні дані перевіряються та обробляються стандартними методами Django.

Вхідними даними загального користувача програми є:

- дані форми реєстрації;
- дані форми входу;
- дані профілю;
- дані форми покупки.

Вхідніми даними системи для адміністратора є:

- дані форми реєстрації;

- дані форми входу;
- дані профілю;
- дані форми додавання товару;
- дані форми редагування товару;
- дані категорій;
- дані форми додавання категорій;
- дані форми редагування категорій;
- дані усіх користувачів.

У адміністратора також є усі вхідні дані клієнта.

Усі вищеперераховані дані являються текстовими окрім зображень товарів, що додає до системи адміністратор, вони зберігаютьсяу вигляді файлів.

Вихідними даними системи для клієнта є:

- дані профілю;
- дані товарів;
- дані кошика.

2.7 Опис використаних технологій та програмних продуктів

Python

Це високорівнева мова програмування зі строгою типізацією, яка призначена для спрощення роботи розробника.

Перелік використаних бібліотек:

- Django 3.2;
- django-extensions 3.1.3;
- django-filter 2.4.0;
- djangorestframework 3.12.4;
- dgangorestframework-simplejwt 4.7.0;
- jwt 1.2.0;
- PyJWT 2.1.0;
- PosetgreSQL 1.0.2.

Django

Вільний фреймворк для вебдодатків на мові Python.

Цей фреймворк значно спрощує роботу розробника, оскільки він бере на себе більшу частину роботи, таку як:

- маршрутизація;
- HTTP запити;
- міграція моделей до бази даних.

DjanoAdminExtension

Це дуже зручна модель UI/UX, свторена за допомогою HTML та CSS, яка дозволяє швидко та зручно створювати адміністраторські панелі для керування даними онлайн-магазинів.

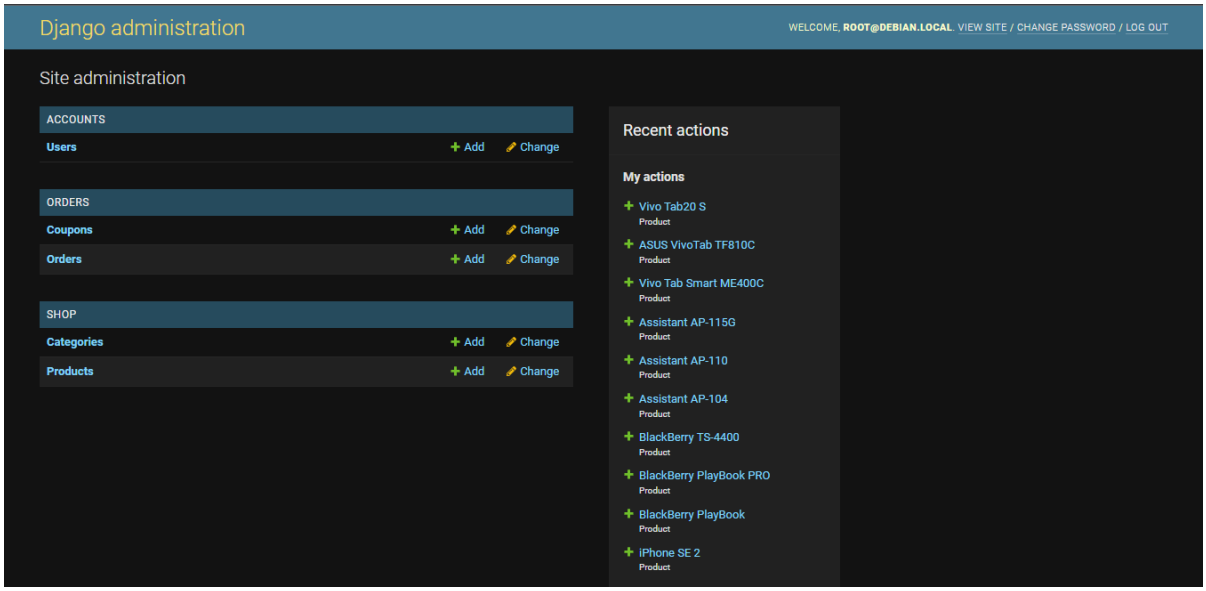


рис 2.7.1 Адміністрована панель DjanoAdminExtension

Navicat for PostgreSQL

Додаток для зручного відображеня створених СУБД на основі PostgreSQL

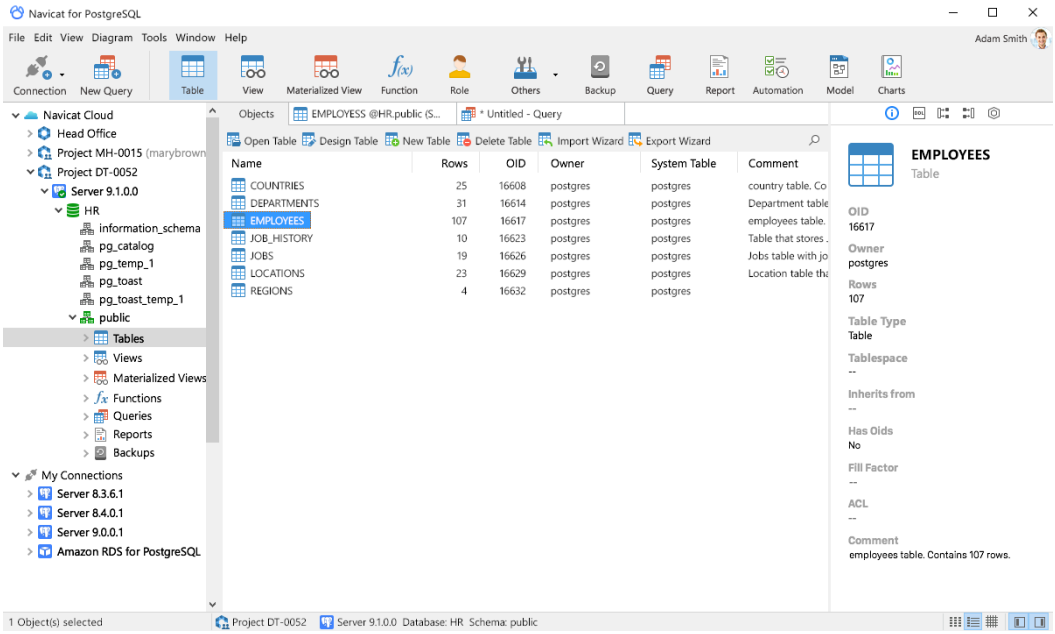


рис 2.7.2 Navicat for PostgreSQL

2.7.1. Використані технічні засоби

Для запуску розробленої програми можна використовувати будь-який пристрій, на якому є доступ до мережі інтернет та на який можна встановити браузер. Це може бути як мобільний телефон так і персональний комп’ютер. Для запуску сервера потрібна система Windows чи Linux.

2.7.2. Використані програмні засоби

Для функціонування розробленого програмного продукту можна використовувати будь-яку операційну систему, яка має стабільне підключення до мережі Інтернет та яка підтримує мову програмування Python та на неї можна встановити Django. У даному випадку було використовано віртуальне середовище за допомогою програми Oracle VirtualBox та системи Debian.

2.7.3. Виклик та завантаження програми

Для запуску сервера потрібно:

1. Встановити віртуальне середовище Oracle VirtualBox.
2. Імпортувати середовище .ova у програму Oracle VirtualBox.
3. Запустити імпортовану віртуальну машину.
4. Дочекатися, поки завантажиться система Debian.
5. Увійти у користувача системи за логіном user та паролем user.
6. Увести команду ip addr.
7. Знайти та переписати адресу віртуальної машини eth0 або inet.
8. Увести команду cd ~/django-online-shop для того, щоб відкрити папку з проектом.
9. Увести команду source venv/bin/activate.
10. Увести команду python manage.py runserver 0.0.0.0:8000 для запуску

сервера.

11. Відкрити браузер на будь-якому пристрої та увести ip адресу віртуальної машини, додаючи порт :8000.

12. Щоб зайти на адміністратовану панель керування сайтом, потрібно до порту додати /admin, щоб вийшло :8000/admin.

13. Для керування адміністративною панелью потрібно увести пошту root@debian.local та пароль root.

2.7.4 Опис інтерфейсу користувача

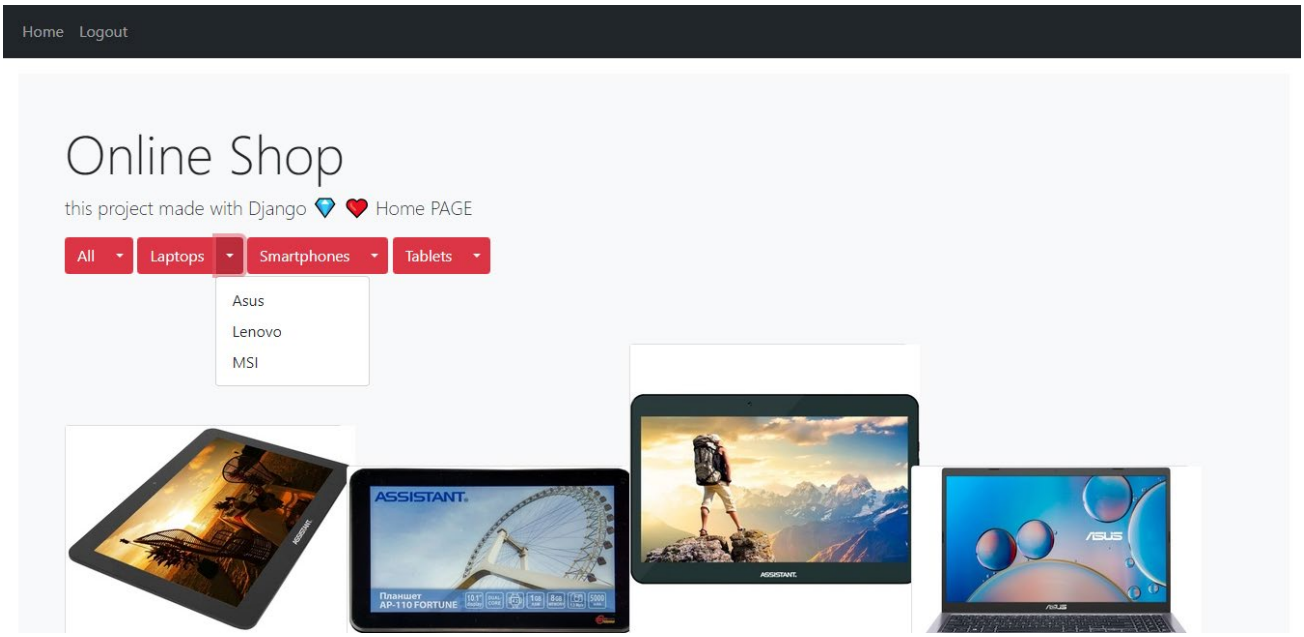


рис 2.7.3 Головна сторінка сайту

Це головна сторінка сайту, де користувач може продивитися увесь каталог товарів, отфільтрувати їх по брендам, або певним категоріям.

Також користувач може увійти у свій обліковий запис, зареєструвати новий, або вийти з поточного.

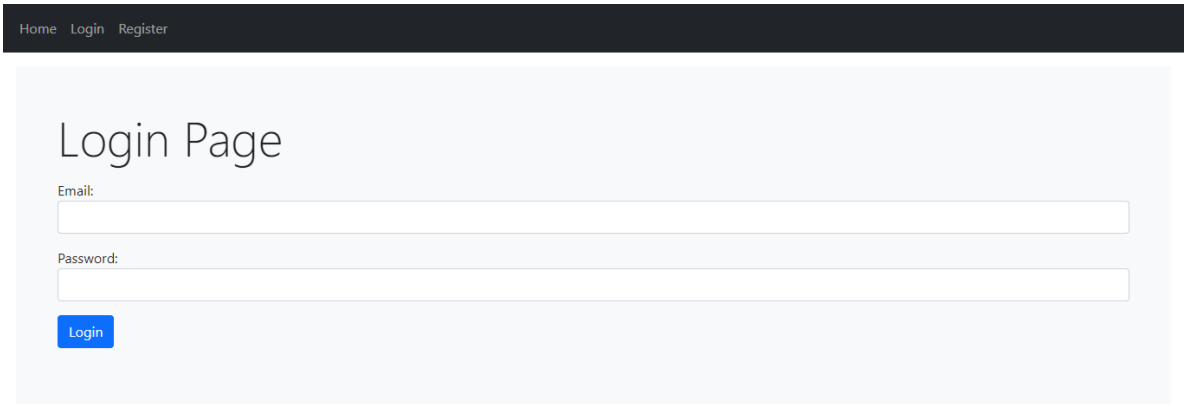


рис 2.7.4 Сторінка із логіном

На цієй сторінці користувач може зайти в свій обліковий запис.

рис 2.7.5 Сторінка із реєстрацією

На цієй сторінці користувач може зайти створити новий обліковий запис.

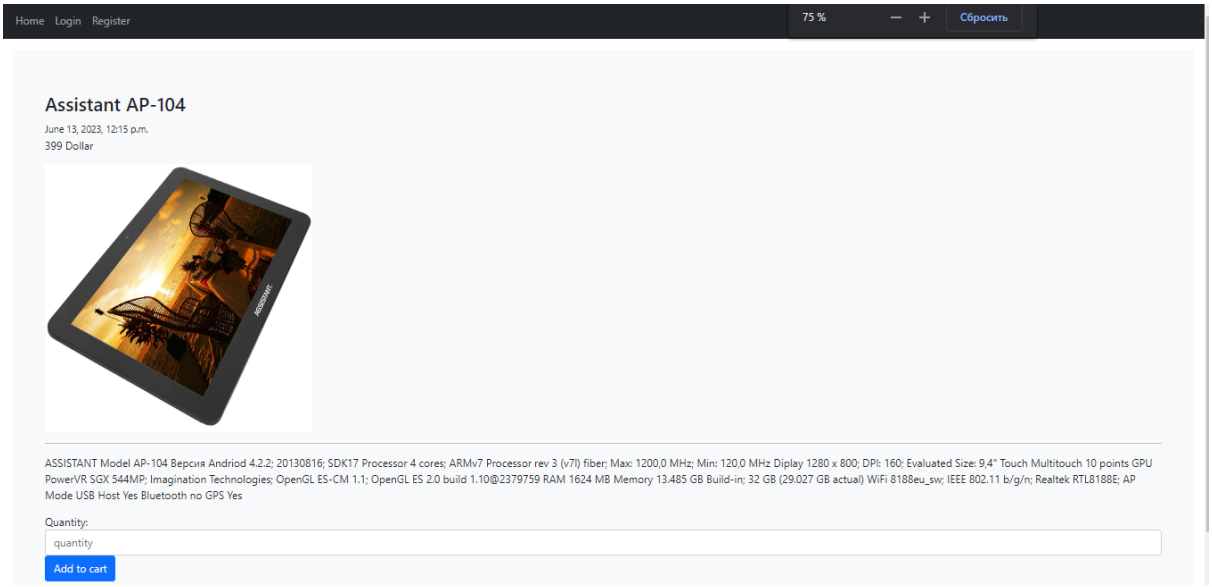


рис 2.7.6 Сторінка із реєстрацією

Ця сторінка відоброжає детальний опис певного товару, на який перейшов користувач. Тут він може обрату кількість товару та додати його до кошика.

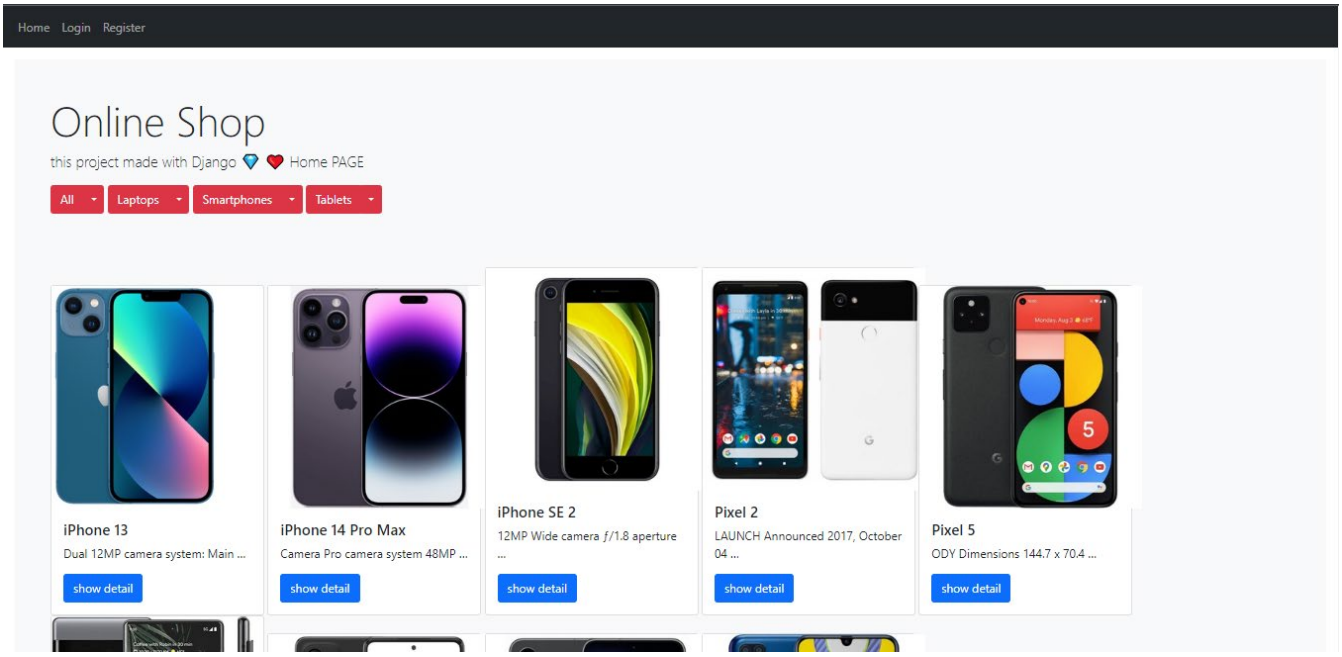
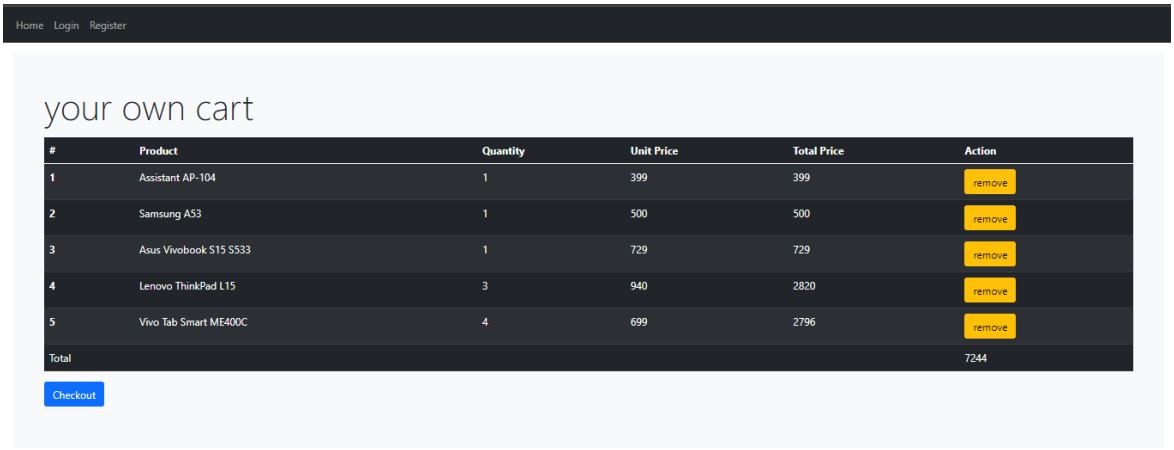


рис 2.7.7 Сторінка із обраною категорією товарів

Дане зображення відображає вибір фільтру категорій товару для подальшого додавання продуктів до кошика.



рис

2.7.8 Сторінка з кошиком клієнта

Дана сторінка відображає повний набір обраних товарів клієнта та повну їх вартість. Також тут відображена можливість прибрати певні позиції.

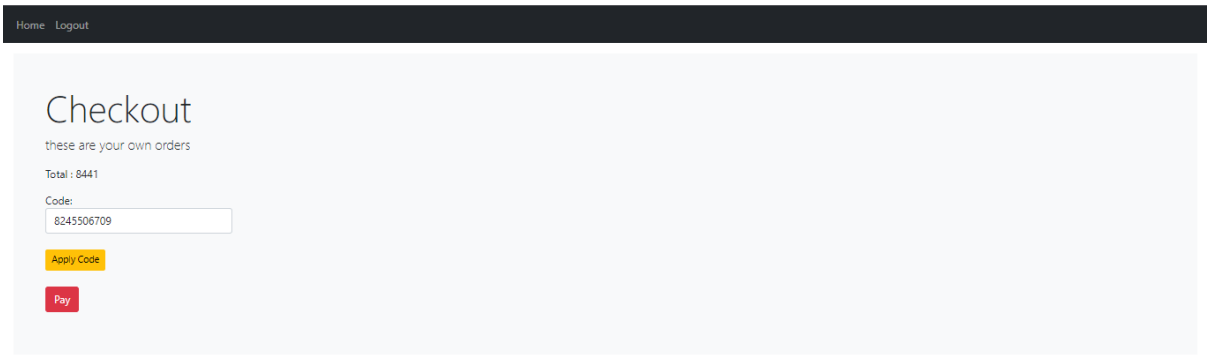


рис 2.7.9 Сторінка придбання товарів

Це зображення показує фінальний етап взаємодії користувача із сайтом. Тут він може ввести певний купон на знижку, який видає адміністратор або продавець та переходить до оплати усього кошика.

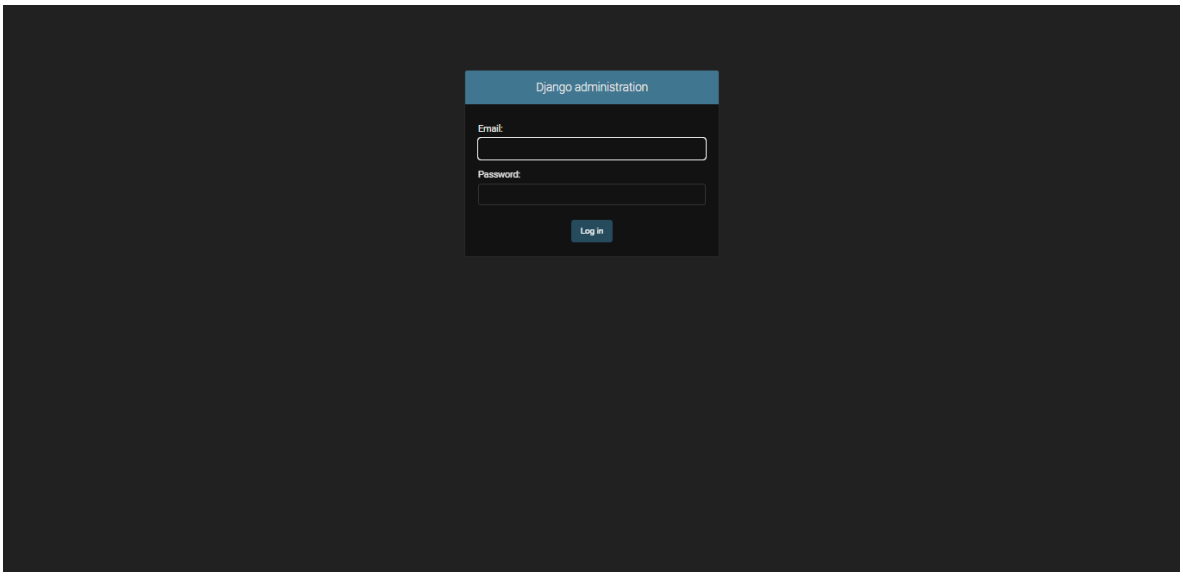


рис 2.7.10 Адмін-панель з логіном

На цьому зображенні користувач може увійти в адміністративну панель.

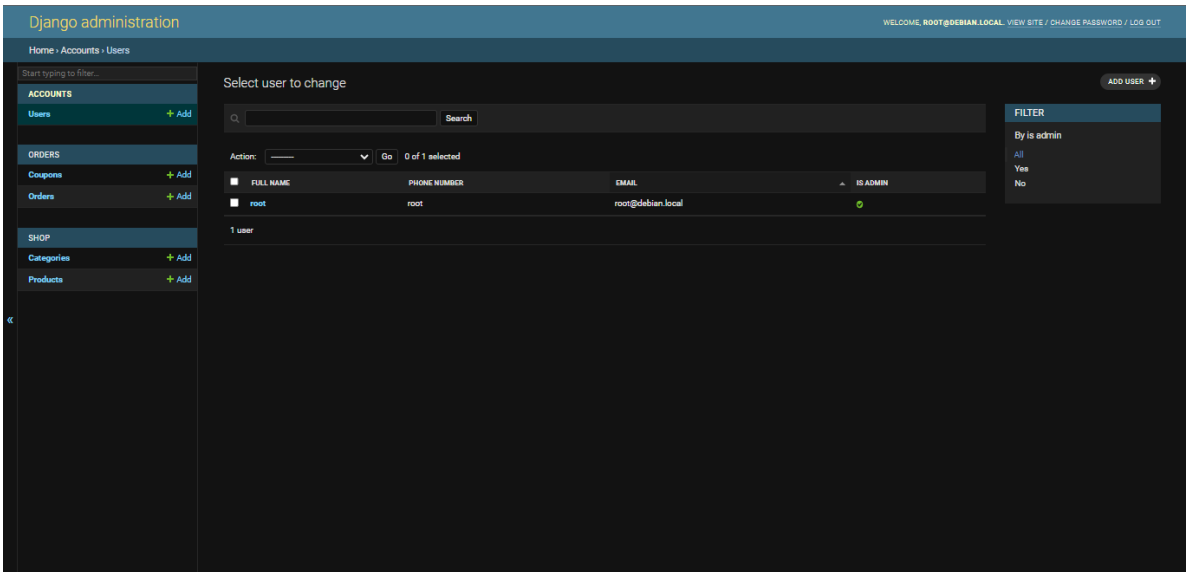


рис 2.7.11 Адмін-панель з переглядом усіх користувачів

На даному знімку можно побачити адміністровану панель зі списком усіх

користувачів. Адміністратор може обрати певних клієнтів та тип взаємодії з ними. Наприклад, перейти до картки клієнту або видалити його. Також тут присутні фільтри вибору декільох користувачів.

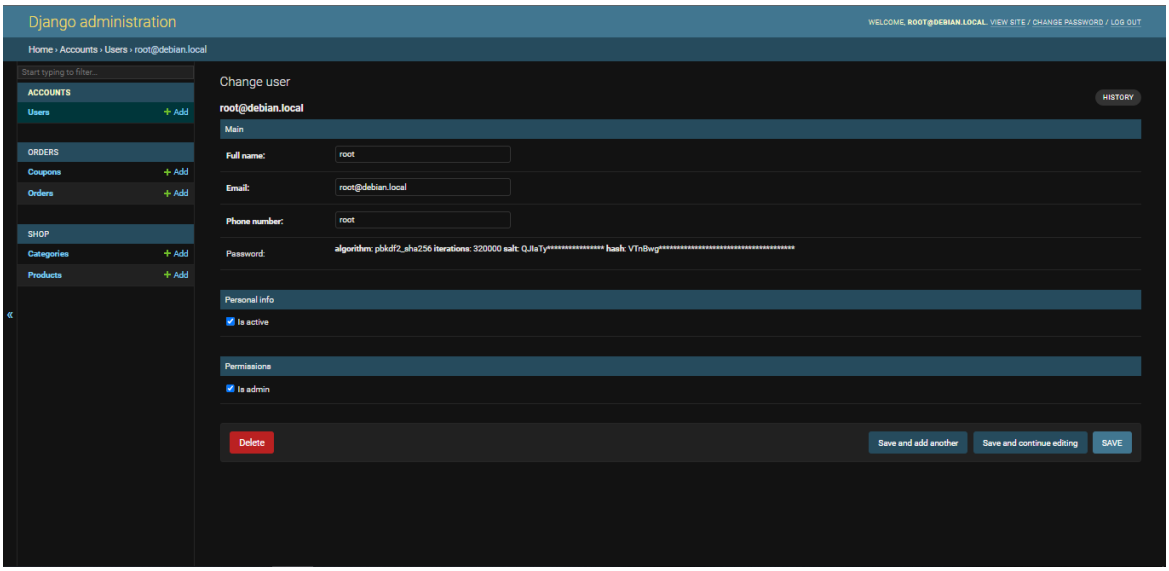


рис 2.7.12 Адмін-панель з переглядом певного користувача

Тут зображена певна картка користувача. З її допомогою адміністратор може змінити дані клієнта, його логін, пошту, телефоний номер, заблокувати клієнта або надати йому права адміністратора. Також адміністратор може видалити аккаунт, та збергігти зміни.

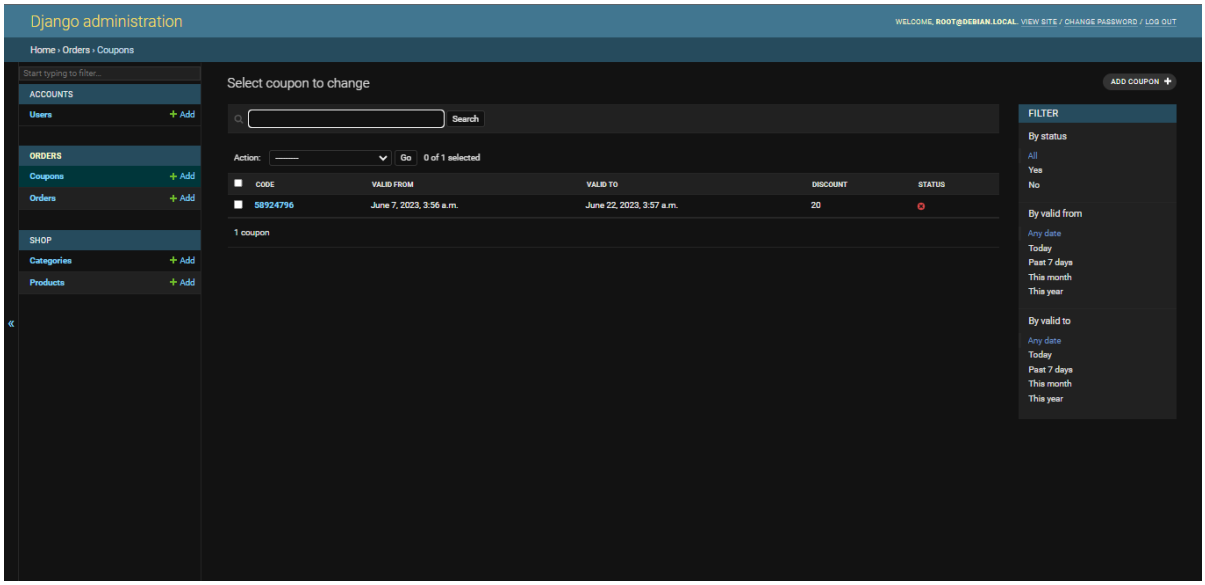


рис 2.7.13 Адмін-панель з переглядом усіх користувача

На цьому зображенні відображаються усі наявні купони магазину. Адміністратор може додату новий, побачити повну інформацію про купони, обрати декілька та змінити.

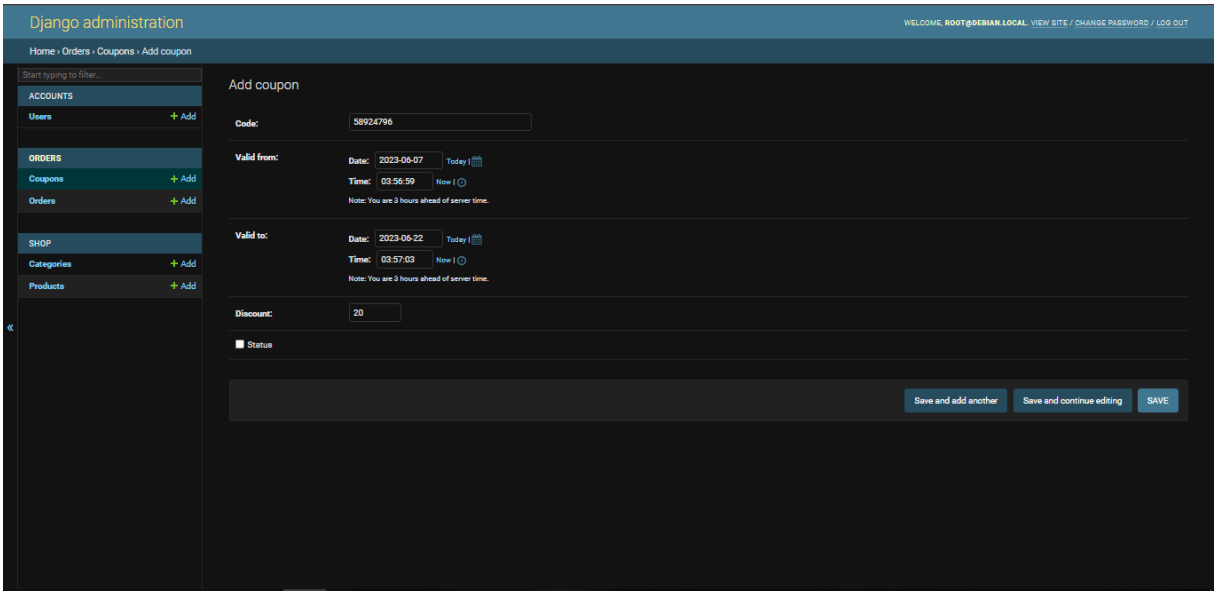


рис 2.7.14 Адмін-панель з переглядом певного купона

Тут адміністратор має можливість створити новий купон. Він може додати його ідентифікатор, дату дії та знижку, який цей купон видає. Також у адміністратора є можливість переглядати історію змін по обраному купону.

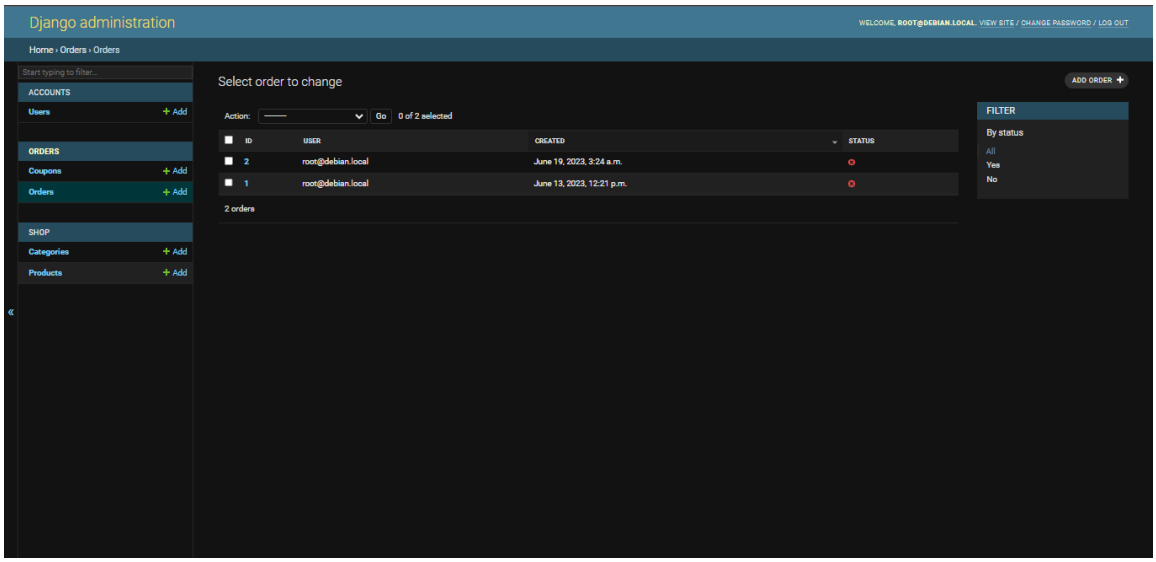


рис 2.7.15 Адмін-панель з переглядом усіх замовлень

На цьому зображенні адміністратор може передивлятись усі наявні замовлення, які поступили в магазин. Він також має можливість передивлятись ці замовлення, редагувати їх та створювати нові вручну.

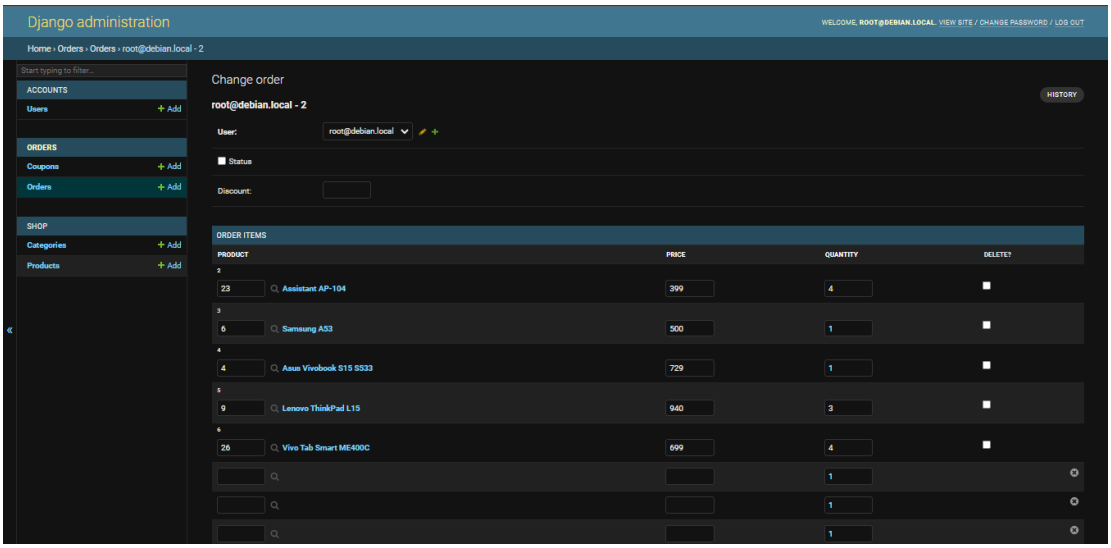


рис 2.7.14 Адмін-панель з переглядом певного замовлення

Тут зображена можливість огляду певного замовлення адміністратором. Можна побачити позиції товарів, які обрав клієнт, їх кількість, чи була застосована знижка, який саме користувач зробив замовлення, загальну вартість замовлення та його історію. Також адміністратор має право внести зміни до цього замовлення або зовсім видалити його.

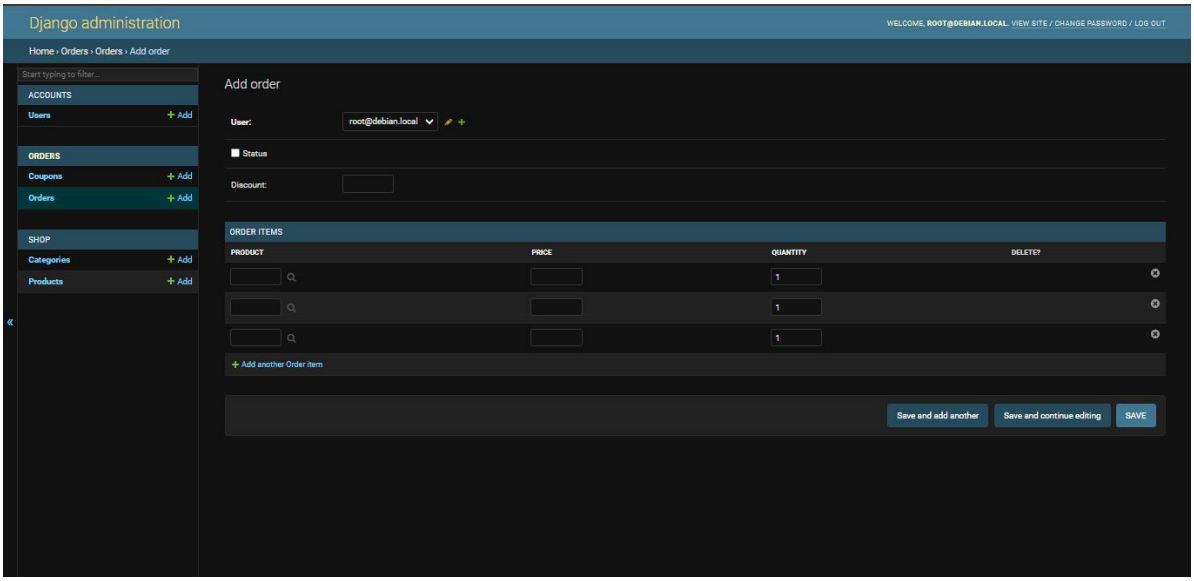


рис 2.7.15 Адмін-панель з додаванням нового замовлення

На цьому зображенні адміністратор може створити нове замовлення, додаючи певні товари до кошику, редагуючі їх ціни та кількість. Також адміністратор має змогу додати знижку на все замовлення та обрати певного користувача. Так як адміністратор має змогу додати певний товар до кошику, то в нього повинна бути реалізована зручна панель для пошуку.

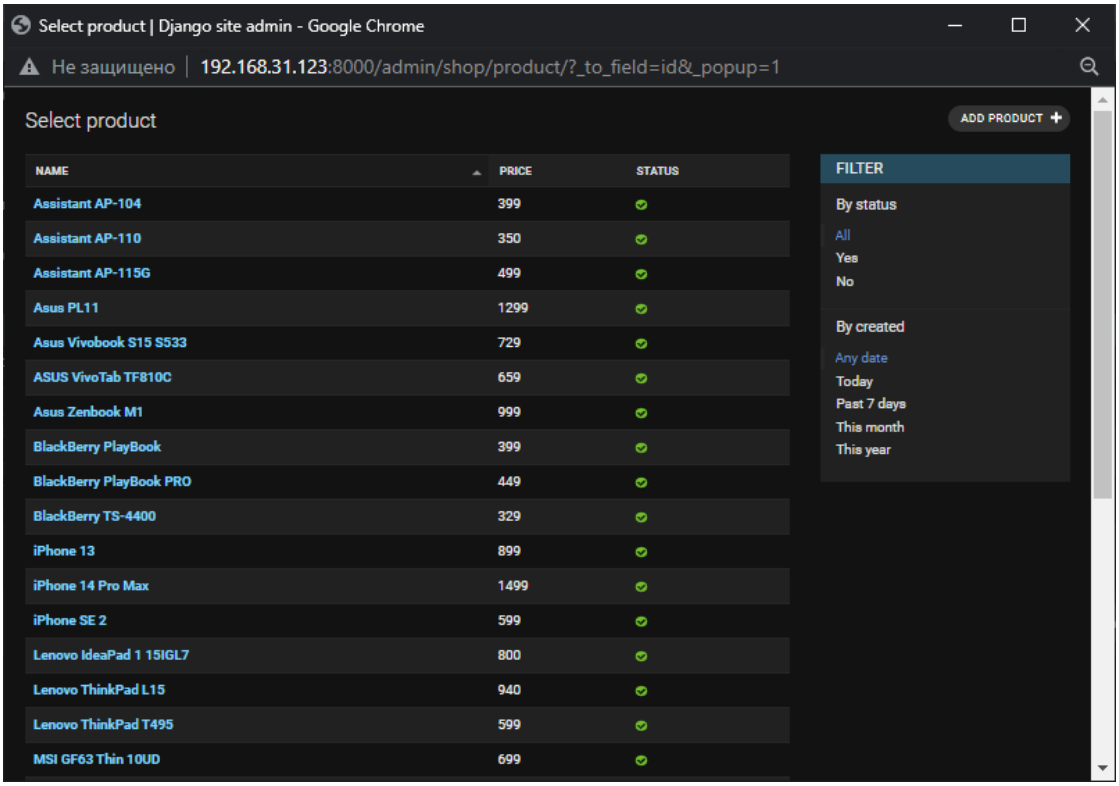


рис 2.7.16 Адмін-панель з додаванням нового товару

На цьому зображенні можна побачити увесь набор товарів, представлений у магазині. Тут є зручний пошук по фільтрам статусу та датою створення картки з цим товаром. Також можно побачити назву певного товару та його поточну ціну. Щоб додати новий товар, або категорію, редагувати вже існуючи, потрібно перейти до відповидної вкладки.

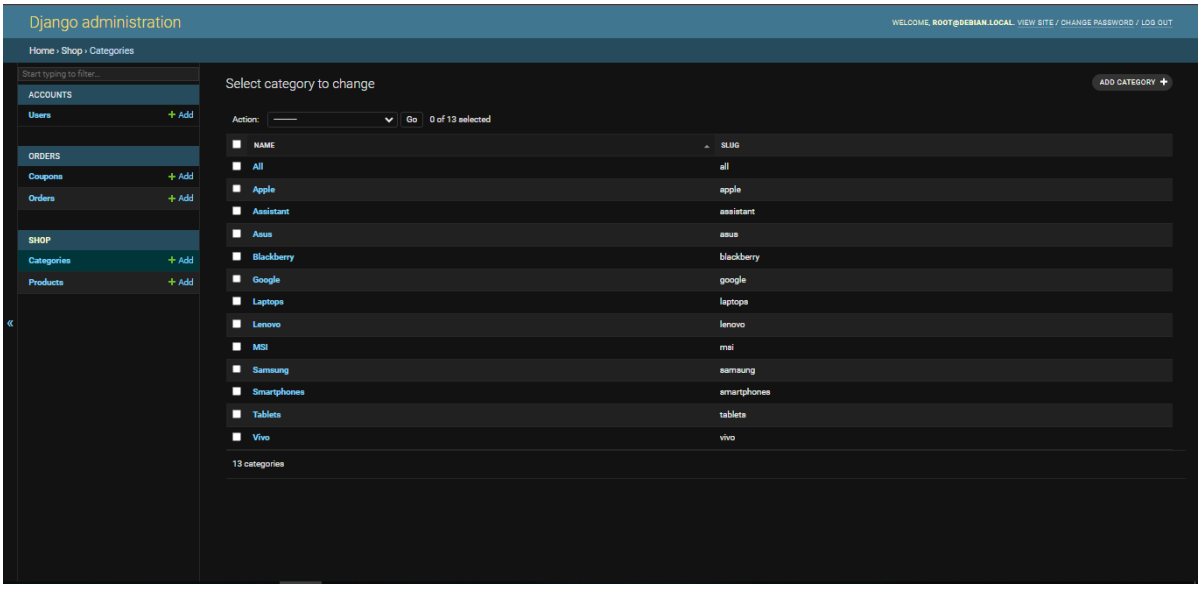


рис 2.7.17 Адмін-панель з наявними категоріями магазину

Тут зображені усі представлені категорії магазину. Адміністратор може додати нову, або переглянути чи видалити вже існуючу.

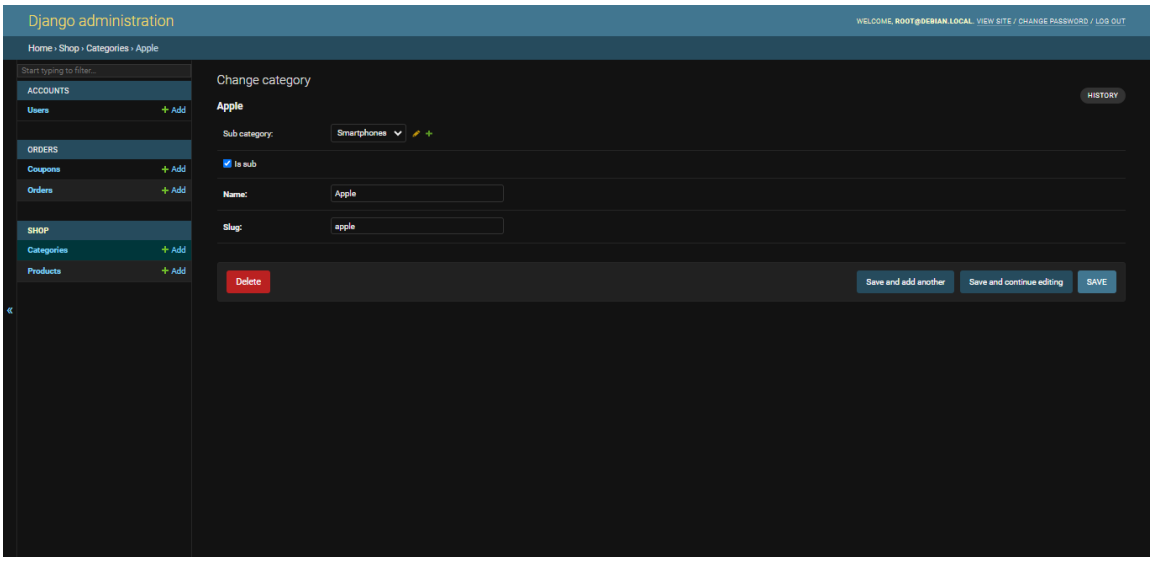


рис 2.7.18 Адмін-панель з поточною категорією магазину

При натисканні на певну категорію магазину відкривається картка із докладною інформацією про неї. Тут адміністратор може змінити її назву та субкатегорію, якщо це потрібно.

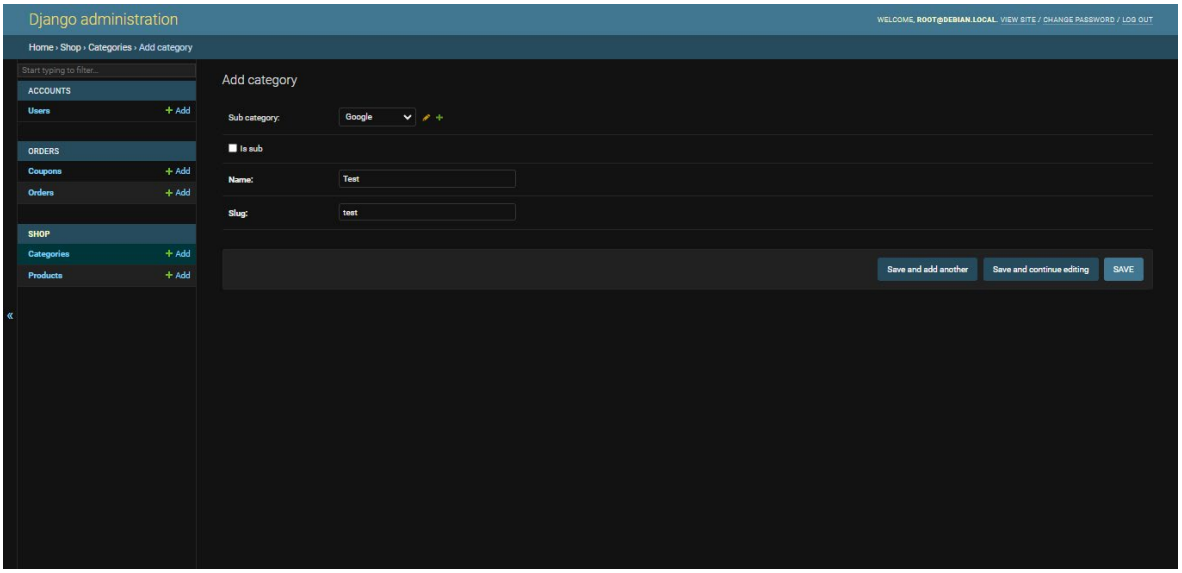


рис 2.7.19 Адмін-панель з додаванням категорії

Якщо адміністратору потрібно додати нову категорію товарів, він натискає на кнопку “Add category”. Відкривається панель, де він може обрати назву категорії, зробити її субкатегорією та зберігти зміни.

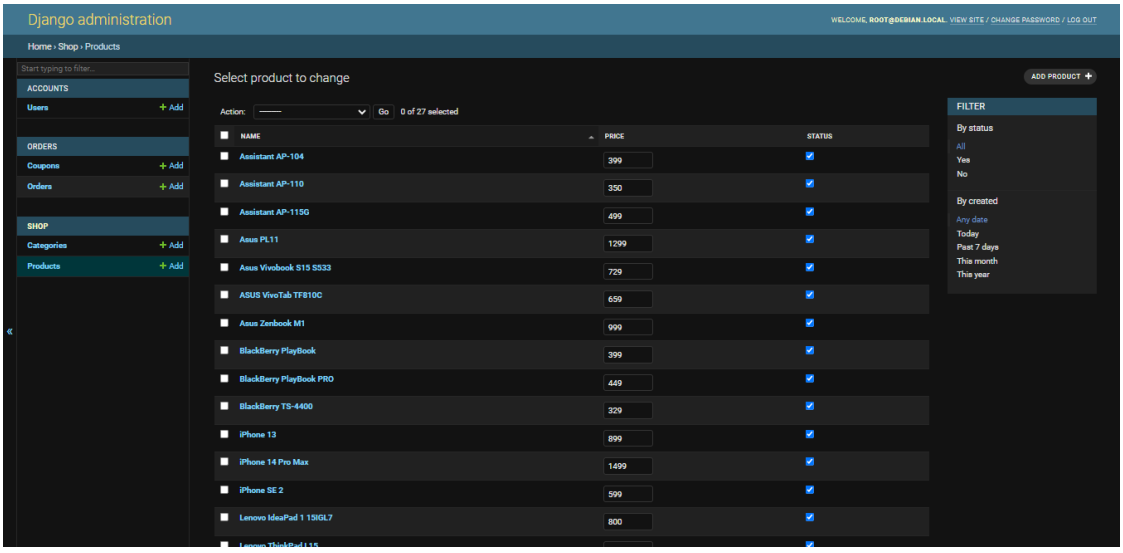


рис 2.7.20 Адмін-панель з переглядом усіх наявних товарів

Це одна з найголовніших панелій керування сайтом – панель перегляду, додавання та редагування товарів.

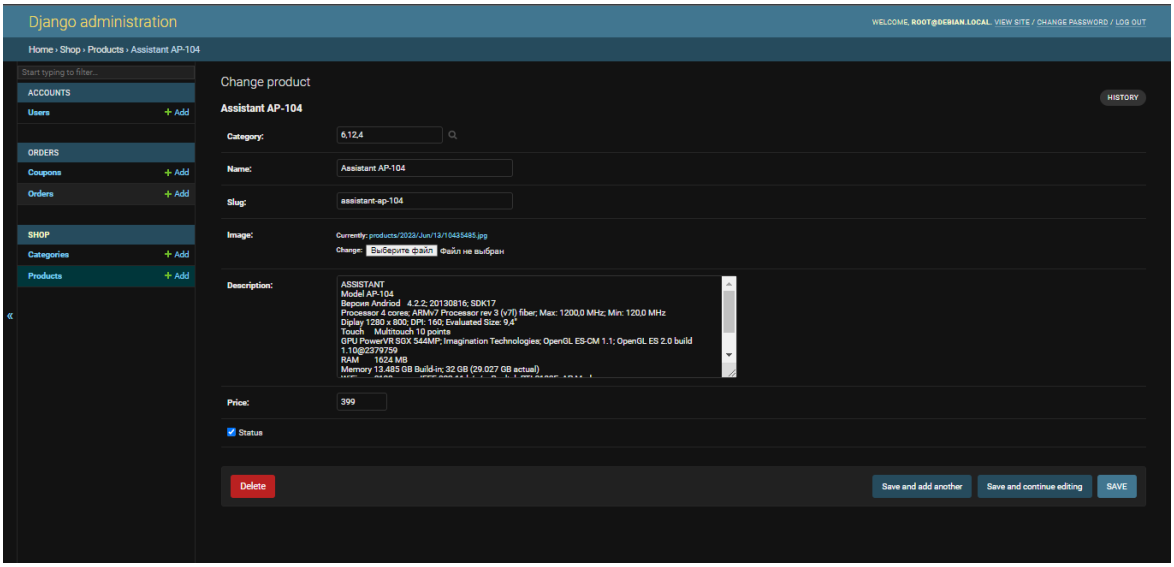


рис 2.7.21 Адмін-панель з переглядом певного товару

На цьому зображенні адміністратор має змогу додати повну інформацію по товару, включаючи його категорію, зображення, детальный опис, ціну статус активності.

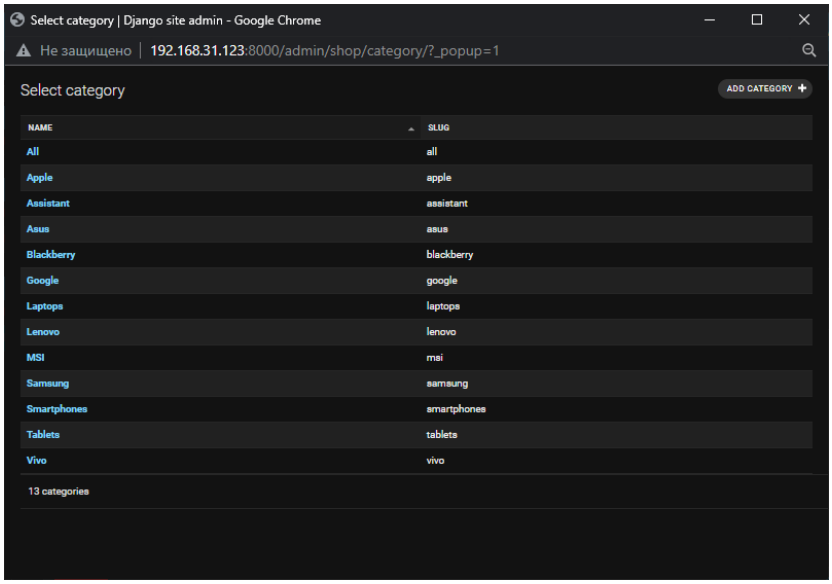


рис 2.7.22 Адмін-панель з переглядом категорій

При натискані кнопки додачі категорії до певного товару відкривається ця панель із списком усіх можливих категорій. Також тут є кнопка додати категорію для зручності.

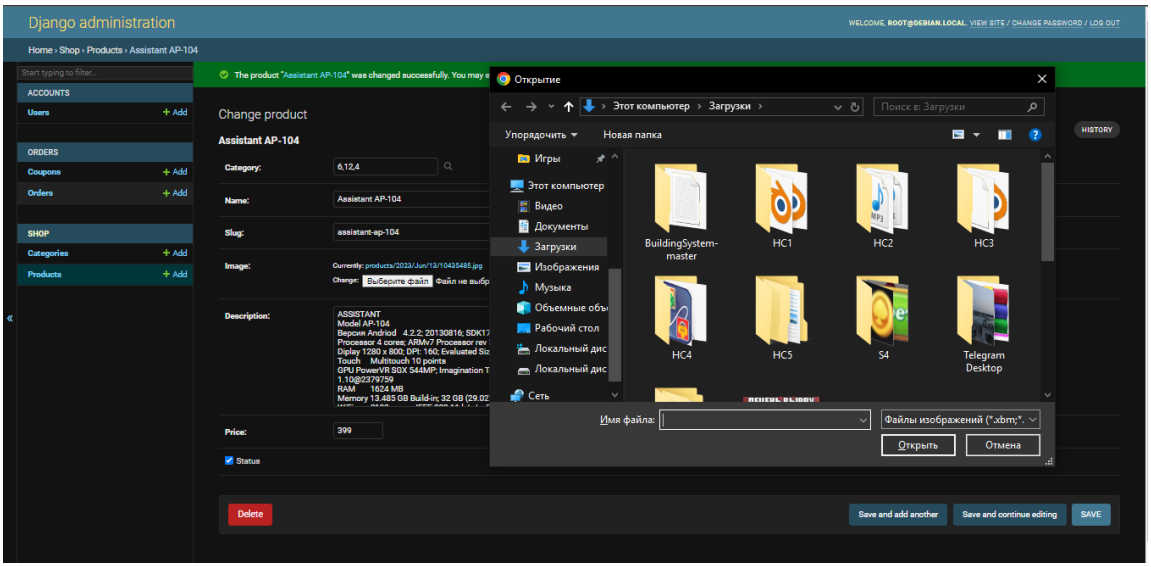


рис 2.7.23 Додавання зображення на товар

При вибору зображення для товара відкривається стандартний провідник операційної системи, в якому працює адміністратор.

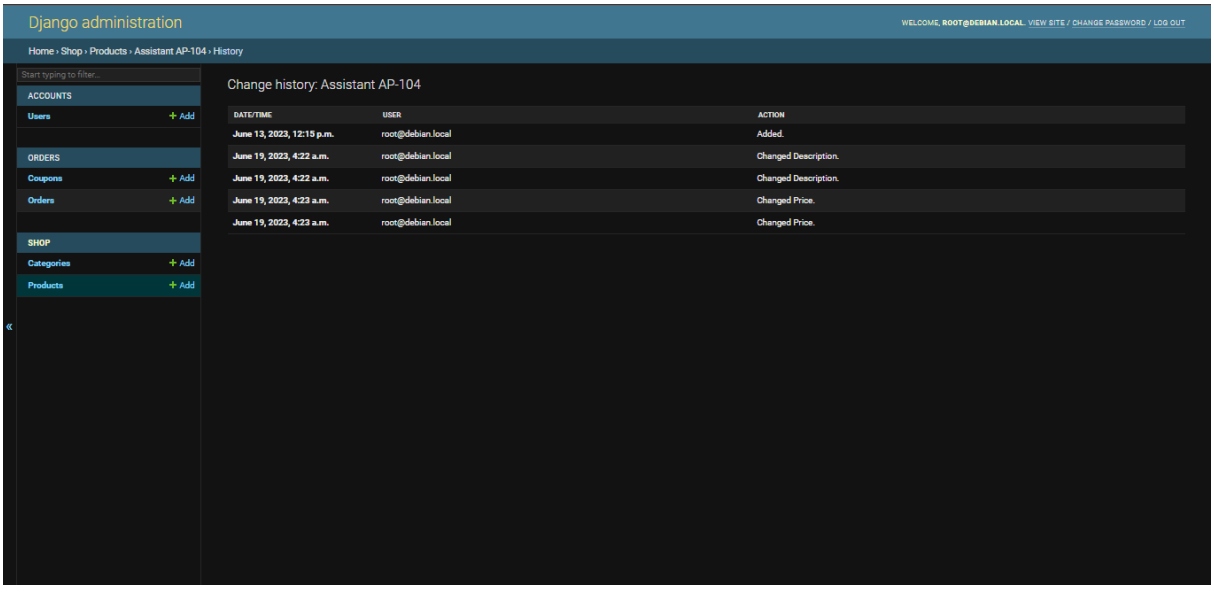


рис 2.7.24 Адмін-панель з історією товара

Так виглядає адмін-панель, якщо натиснути на кнопку “History” будь-якого елементу.

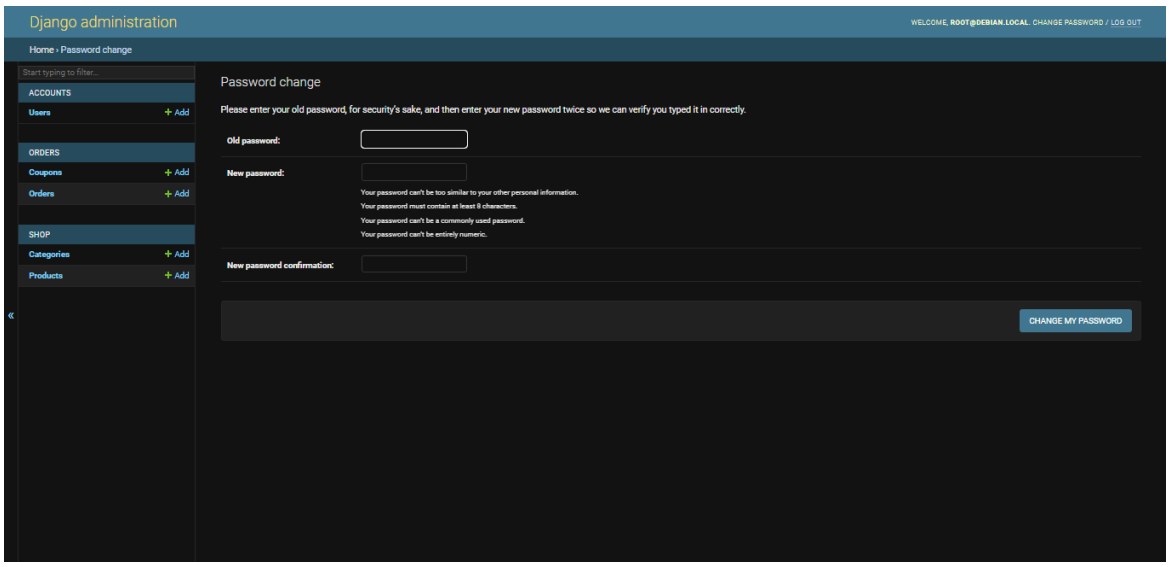


рис 2.7.25 Адмін-панель зі зміною паролю поточного облікового запису

Якщо у адміністратора виникне ситуація, при якої йому потрібно буде змінити поточний пароль, то він зможе натиснути кнопку «Change password» зправа зверху, щоб відкрилась дана панель.

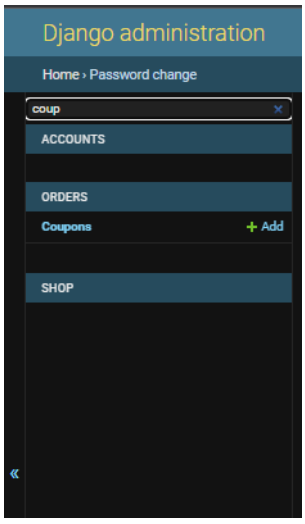


рис 2.7.26 Пошук у адмін-панелі

Так виглядає пошук по вкладкам адмін-панелі.

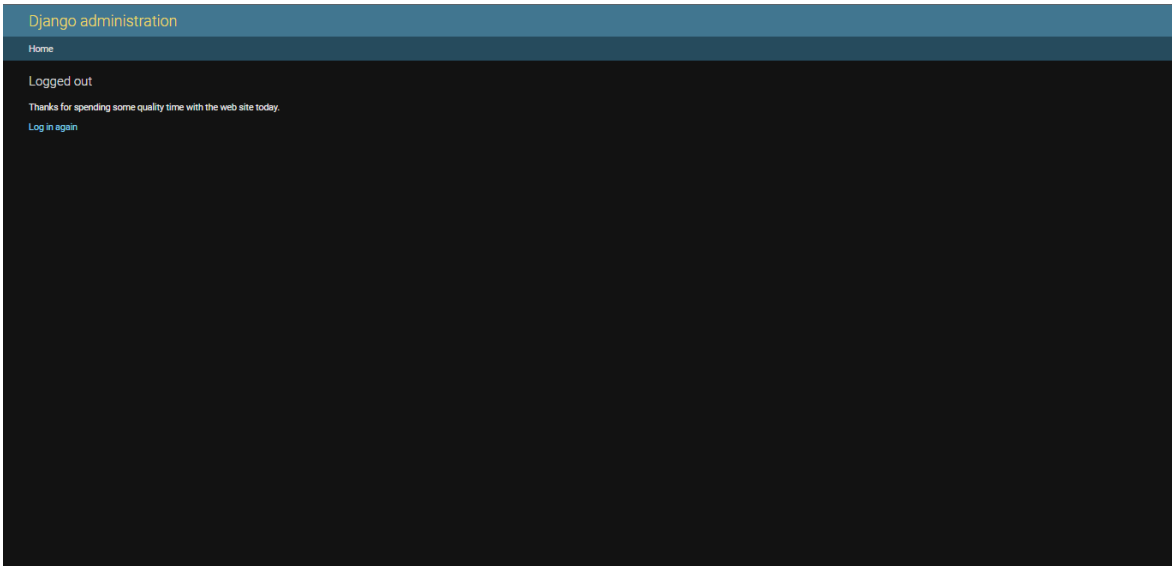


рис 2.7.27 Делогін адміністратора

Ще одна з можливостей адміністратора – вихід із власного облікового запиу для його зміни. Також можна побачити зправа кнопку «View Site», яка дає змогу одразу ж перейти на головний сайт магазину та побачити недавні зміни.

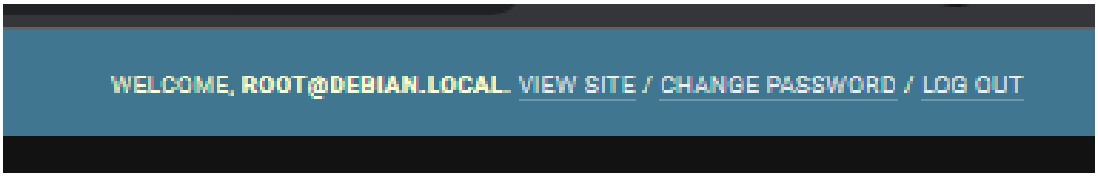


рис 2.7.28 Кнопки адмін-панелі

РОЗДІЛ 3
ЕКОНОМІЧНА ЧАСТИНА

Під час розробки програмного забезпечення важливими етапами є визначення трудомісткості розробки і розрахунок витрат на створення програмного продукту.

3.1. Визначення трудомісткості розробки програмного забезпечення

Задані дані:

1. передбачуване число операторів (підпрограм) – 1200;
2. коефіцієнт складності програми – 1,5;
3. коефіцієнт корекції програми в ході її розробки – 0,2;
4. годинна заробітна плата програміста [1], грн/год – 150;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,3;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,0;
7. вартість машино-години ЕОМ, грн/год – 21 (розрахунок нижче).

Собівартість машино-години ЕОМ, грн/год:

$$M = \frac{S_1 + S_2 + S_3}{H}, \text{грн/год}, \qquad (3.1)$$

де S_1 – річна сума амортизації (оновлення, обслуговування, ремонтування та ін.), грн;

S_2 – річні витрати на електроенергію, грн;

S_3 – річні витрати на ПЗ, грн;

H – дійсний годовий фонд часу роботи, годин.

$$M = \frac{40000 + 5000 + 12000}{2710} = 14{,}8 + 1{,}8 + 4{,}4 = 21, \text{ грн/год},$$

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \qquad (3.2)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q * C * (1 + p), \qquad (3.3)$$

де q - передбачуване число операторів;

C - коефіцієнт складності програми;

p - коефіцієнт корекції програми в ході її розробки.

$$Q = 1200 * 1,5 * (1 + 0,2) = 2160.$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q * B}{(75..85) * k}, \text{ людино-годин.} \qquad (3.4)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

$$t_u = \frac{2160 * 1,3}{80 * 1} = \frac{2808}{80} = 35,1, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25) * k}, \text{ людино-годин,} \quad (3.5)$$

$$t_a = \frac{2160}{20 * 1} = 108, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25) * k}, \text{ людино-годин,} \quad (3.6)$$

$$t_n = \frac{2160}{25 * 1} = 86,4, \text{ людино-годин,}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5) * k}, \text{ людино-годин,} \quad (3.7)$$

$$t_{отл} = \frac{2160}{4 * 1} = 540, \text{ людино-годин;}$$

- за умови комплексного налагодження завдання:

$$t^k_{отл} = 1,5 * t_{отл}, \text{ людино-годин,} \quad (3.8)$$

$$t^k_{отл} = 1,5 * 540 = 810, \text{ людино-годин.}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \quad (3.9)$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{15..20 * k}, \text{ людино-годин,} \quad (3.10)$$

$$t_{др} = \frac{2160}{20 * 1} = 108, \text{ людино-годин.}$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0,75 * t_{др}, \text{ людино-годин,} \quad (3.11)$$

$$t_{до} = 0,75 * 108 = 81, \text{ людино-година,}$$

$$t_{д} = 108 + 81 = 189 \text{ людино-годин.}$$

Тепер розрахуємо трудомісткість ПЗ:

$$t = 81 + 35,1 + 108 + 86,4 + 540 + 189 = 1039,5, \text{ людино-годин.}$$

3.2. Витрати на створення програмного забезпечення

Витрати на створення ПЗ *Кпо* включають витрати на заробітну плату виконавця програми *Зз/п* і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{по} = З_{зп} + З_{мв}, \text{ грн.} \quad (3.12)$$

Заробітна плата виконавців визначається за формулою:

$$З_{зп} = t * С_{пр}, \text{ грн,} \quad (3.13)$$

де t - загальна трудомісткість, людино-годин;
$С_{пр}$ - середня годинна заробітна плата програміста, грн/година.

$$З_{зп} = 1039,5 * 150 = 155925, \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми:

$$З_{мв} = t_{отл} * С_{мч}, \text{ грн,} \quad (3.14)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год,
$С_{мч}$ - вартість машино-години ЕОМ, грн/год,
$С_{мч}$ = 21, грн/год.

$$З_{мв} = 540 * 21 = 11340, \text{ грн.}$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення ПЗ:

$$К_{по} = 155925 + 11340 = 167265, \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k * F_p}, \text{ міс.,} \qquad (3.15)$$

де B_k - число виконавців (приймається 1),
F_p - місячний фонд робочого часу (40 годин на тиждень F_p = 176 годин).

$$T = \frac{1039,5}{1 * 176} = 5,9, \text{ міс.}$$

Висновок:

Розробка сайту з продажу компь’ютерних комплектуючих розроблено як продукт для користувачів. Вартість даного продукту становить 167 тис. грн. і не вимагає додаткових витрат. Очікуваний час розробки становить 5,9 місяців. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Середня заробітна плата за видами економічної діяльності по місяцях (Державна служба статистики України) / URL: https://ukrstat.gov.ua/operativ/operativ2005/gdn/Zarp_ek_m/Zp_ek_m_u/arh_zpm_u.html (дата звернення 25.05.2023).

2. Python's documentation. URL: https://www.python.org/doc/

3. Django documentation. URL: https://docs.djangoproject.com/en/3.2/

4. Library documentation. URL: https://stackoverflow.com/

5. Description of structures. URL: https://habr.com/

6. Simple JWT documentation. URL: https://django-rest-framework-simplejwt.readthedocs.io/en/latest/

7. Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – Австрия, 2011. – 1280 с.

8. Лутц М. Программирование на Python, том I, 4-е издание. – Пер. с англ. – Австрия, 2011. – 992 с.

9. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. – Австрия, 2011. – 992 с.

10. Гэддис Т. Начинаем программировать на Python. – 4-е изд.: Пер. с англ. –Германия, 2019. – 768 с.

11. Лучано Рамальо Python. К вершинам мастерства. – М.: ДМК Пресс, 2016. – 768с.

12. Свейгарт, Эл. Автоматизация рутиных задач с помощью Python: практическое руководство для начинающих. Пер. с англ. — М.: Вильямс, 2016. – 592 с.

13. Рейтц К., Шлюссер Т. Автостопом по Python. – Германия, 2017. – 336 с.

14. Лергейц Билл Простой Python. Современный стиль программирования. – Гемания, 2016. – 480 с.: – (Серия «Бестсепперы O’Reilly»).

ДОДАТОК А

КОД ПРОГРАМИ

```
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-y^xlc@09_p0@_9t3g945%b6tnk0)6afm49rx7=a@4juxy!3u!*'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'accounts.apps.AccountsConfig',
    'shop.apps.ShopConfig',
    'cart.apps.CartConfig',
    'orders.apps.OrdersConfig',
    'sorl.thumbnail',
]

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 = 'django_online_shop.urls'


TEMPLATES = [
    {
```

```
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates']
        ,
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_p rocessors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'django_online_shop.wsgi.application'

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

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

```
# Password validation
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH_USER_MODEL = 'accounts.User'

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/3.2/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/3.2/howto/static-files/

STATIC_URL = '/static/'

MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media/'

# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
from django.db import models

# Create your models here.
from django.urls import reverse



class Category(models.Model):
    sub_category = models.ForeignKey('self', on_delete=models.CASCADE,
related_name='sub_categories', null=True,
                                     blank=True)
    is_sub = models.BooleanField(default=False)
    name = models.CharField(max_length=200)
    slug = models.SlugField(max_length=200, unique=True)
```

```
    class Meta:
        ordering = ('name',)
        verbose_name = 'category'
        verbose_name_plural = 'categories'

    def __str__(self):
        return self.name

    def get_absolut_url(self):
        return reverse('shop:category_filter', args={self.slug})


class Product(models.Model):
    category = models.ManyToManyField(Category, related_name='products')
    name = models.CharField(max_length=200)
    slug = models.SlugField(max_length=200)
    image = models.ImageField(upload_to='products/%Y/%b/%d/')
    description = models.TextField()
    price = models.IntegerField()
    status = models.BooleanField(default=True)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)

    class Meta:
        ordering = ('name',)

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

```
    def get_absolut_url(self):
        return reverse('shop:product_detail', args={self.slug, })
from django.urls import path
from shop import views

app_name = 'shop'

urlpatterns = [
    path('', views.home, name='home'),
    path('category/<slug:slug>/', views.home, name='category_filter'),
    path('<slug:slug>/', views.product_detail, name='product_detail'),
]
from django.shortcuts import render, get_object_or_404

# Create your views here.
from cart.forms import Add2CartForm
from shop.models import Product, Category


def home(request, slug=None):
    products = Product.objects.filter(status=True)
    categories = Category.objects.filter(is_sub=False)
    if slug:
        category = get_object_or_404(Category, slug=slug)
        products = products.filter(category=category)
    return render(request, 'shop/home.html', {'products': products, 'categories': categories})


def product_detail(request, slug):
```

```
    product = get_object_or_404(Product, slug=slug)
    form = Add2CartForm()
    return render(request, 'shop/product_detail.html', {'product': product, 'form': form})
from django import forms


class Add2CartForm(forms.Form):
    quantity = forms.IntegerField(min_value=1, max_value=9,
widget=forms.NumberInput(attrs={'class': 'form-control',
                                                    'placeholder': 'quantity'}))
from django.shortcuts import render, get_object_or_404, redirect

# Create your views here.
from cart.forms import Add2CartForm
from cart.utils.cart import Cart
from shop.models import Product
from django.views.decorators.http import require_POST


def detail(request):
    cart = Cart(request)
    return render(request, template_name='cart/detail.html', context={'cart': cart})


@require_POST
def cart_add(request, product_id):
    cart = Cart(request)
    product = get_object_or_404(Product, id=product_id)
    form = Add2CartForm(request.POST)
```

```
    if form.is_valid():
        data = form.cleaned_data
        cart.add(product=product, quantity=data['quantity'])
    return redirect('cart:detail')


def cart_remove(request, product_id):
    cart = Cart(request)
    product = get_object_or_404(Product, id=product_id)
    cart.remove(product)
    return redirect('cart:detail')
```

ДОДАТОК Б

ВІДГУК

керівника економічного

розділу на кваліфікаційну

роботу бакалавра

на тему:

«Розробка веб сайту інтернет магазину комп'ютерних комплектуючих»

студента групи 121-19з-1 Ляхова Євгена Андрійовича

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Ляхов Євгеній.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Ляхов Євгеній.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Ляхов Євгеній.rar	Архів. Містить коди програми
Презентація	
Презентація Ляхов Євгеній.ppt	Презентація кваліфікаційної роботи