Вінницький національний технічний університет
Факультет комп'ютерних систем і автоматики
Кафедра системного аналізу та інформаційних технологій

# МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

# «Інформаційна технологія аналізу та прогнозування цін інтернет-магазинів для електронних товарів»

Виконав: студент 2 курсу, групи 2ІСТ-20м
спеціальності 126 – «Інформаційні системи та технології»
_____ Ющук І. О.

Керівник: к.т.н., доц. каф. САІТ
_____Козачко О. М.
«___» _____ 2021 р.

Опонент: к.т.н., доц. каф. АІІТ
_____ Кабачій В. В.
«___» _____ 2021 р.

**Допущено до захисту**

Завідувач кафедри САІТ
_____ д.т.н., проф. Мокін В. Б.
«___» _____ 2021 р.

Вінниця ВНТУ – 2021 рік

Вінницький національний технічний університет
Факультет комп'ютерних систем і автоматики
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – ІІ-й (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 126 Інформаційні системи та технології
Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ

_____ д.т.н., проф. Мокін В. Б.

«___» _____ 2021 р.

**ЗАВДАННЯ**
**НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ**
Ющуку Ігору Олеговичу

1. Тема роботи: «Інформаційна технологія аналізу та прогнозування цін інтернет магазинів для електронних товарів»,
керівник роботи: Козачко О. М., к.т.н., доц. каф. САІТ,
затверджені наказом закладу вищої освіти від «___»__ _____2021 року №__
2. Строк подання студентом роботи «___»_____2021 року
3. Вихідні дані до роботи:

Датасети з даними про товари зібрані з популярних українських інтернет магазинів.

4. Зміст текстової частини:
   - аналіз предметної області;
   - розробка моделі прогнозування рекомендованої ціни електронних товарів;
   - розробка аналітичної системи моніторингу цін електронних товарів в інтернет-магазинах;
   - економічна частина.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
   - UML діаграма прецедентів;
   - UML діаграма активності додавання даних;
   - UML діаграма активності роботи скрапера;
   - UML діаграма активності редагування даних;
   - головна сторінка веб-додатку;
   - каталог товарів;
   - сторінка детальної інформації товару.

6. Консультанти розділів МКР

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 5 | Ратушняк О.Г.,<br>к.е.н., доц. каф. ЕПВМ | | |
| | | | |

7. Дата видачі завдання «___»_____2021 року

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів МКР | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз предметної області | 09.2021 | |
| 2 | Підготовка даних для аналізу | 09.2021 | |
| 3 | Розробка моделі прогнозування | 09.2021 | |
| 4 | Реалізація системи та розробка інструкції користувача | 10.2021 | |
| 5 | Економічна частина | 10.2021 | |
| 6 | Оформлення матеріалів до захисту МКР | 11.2021 | |

Студент _____ Ющук І. О.

Керівник роботи _____ Козачко О. М.

# АНОТАЦІЯ

УДК 004.08

Ющук І. О. Інформаційна технологія аналізу та прогнозування цін інтернет-магазинів для електронних товарів. Магістерська кваліфікаційна робота зі спеціальності 126 – інформаційні системи та технології, освітньо-професійна програма – інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2021. 128 с.

На укр. мові. Бібліогр.: 20 назв; рис.: 63; табл.: 9.

Магістрська кваліфікаційна робота присвячена аналізу та прогнозуванню цін електронних товарів в інтернет-магазинах. Проведено розвідувальний аналіз даних та відібрано оптимальний набір ознак, за якими слід будувати модель. Розроблена математична модель, яка дозволяє спрогнозувати рекомендовану ціну необхідного електронного товару та визначити найкращий інтернет-магазин для його купівлі. Прогнозування рекомендованої ціни здійснювалася на основі даних зібраних за допомогою розробленого веб-скрапера, який дозволяє збирати дані про товари з популярних українських інтернет-магазинів, а також розроблено веб-додаток для представлення проведених досліджень. Система веб-скрапера написана на мові Python з використанням, інструментів Selenium та Requests в комплексі з Beautifulsoup4, а для розробки веб-додатку був використаний фреймворк Django.

Ілюстративна частина складається з 7 плакатів із результатами моделювання.

У розділі економічної частини розглянуто питання про доцільність розробки та впровадження інформаційної технологія аналізу та прогнозування цін інтернет-магазинів для електронних товарів.

Ключові слова: аналіз даних, веб-скрапінг, побудова моделей, електронна комерція, розробка програмного модуля

# ABSTRACT

Yushchuk IO Information technology of analysis and forecasting of prices of online stores for electronic goods. Master's thesis in specialty 126 – information systems and technologies, educational and professional program – information technology data and image analysis. Vinnytsia: VNTU, 2021. 128 p.

In Ukrainian language. Bibliographer: 20 titles; fig.: 63; table: 9.

The master's qualification work is devoted to the analysis and forecasting of prices of electronic goods in online stores. An exploratory analysis of the data was conducted and the optimal set of features on which to build the model was selected. Developed a mathematical model that allows you to predict the recommended price of the required electronic product and determine the best online store to buy it. The forecast of the recommended price was carried out on the basis of data collected with the help of a developed web-scraper, which allows collecting data on goods from popular Ukrainian online stores, as well as a web application for presenting research. The web scrapper system was written in Python using Selenium and Requests tools in combination with Beautifulsoup4, and the Django framework was used to develop the web application.

The illustrative part consists of 7 posters with simulation results.

In the section of the economic part the question of expediency of development and introduction of information technology of the analysis and forecasting of the prices of online stores for electronic goods is considered.

Keywords: data analysis, web scraping, model building, e-commerce, software module development

# ЗМІСТ

# ВСТУП

На сучасному етапі розвитку інформаційних технологій використання комп'ютера для збереження незліченної кількості видів інформації стає єдиним засобом, який надає широкі можливості управління інформацією. Напевно найважливішу роль отримання інформації відіграє мережа Інтернет. За дослідженнями проведеними в 2018 році у всьому світі послугами інтернет користуються понад 4 млрд. осіб, а в Україні 25 млн. що становить більше половини всього населення країни [1].

Інтернет вважається найбільш розвиненою інформаційною системою, за допомогою якої здійснюється комунікація між незліченною кількістю інтернет користувачів. За допомогою всесвітньої мережі забезпечується доступ до більш як мільярду Web-сайтів. Якщо порівняти це з показниками України, де налічується близько 200-300 тисяч сайтів, а кількість Web-серверів забезпечуючих роботу цих сайтів налічує 4,5 тисячі [2].

**Актуальність теми** обумовлена тим, що все частіше інтернет застосовується для замовлення товарів, тому досить доцільним є сервіс, який допоможе аналізувати ціни товарів в різних інтернет магазинах та обрати такий, в якому ціна та якість буде оптимальною.

**Мета і задачі дослідження**. Метою цієї роботи є визначення оптимальної купівельної ціни необхідного електронного товару за рахунок аналізу цін на товари в різних магазинах та прихильності клієнтів до того чи іншого магазину.

Для досягнення поставленої мети було поставлено та вирішено такі задачі:

– провести аналіз предметної області;

– розробити веб-скрапер, який дозволяє збирати дані про електронні товари з різних інтернет-магазинах;

– розробити модель прогнозування рекомендованої ціни товарів на основі статистичних даних отриманих за допомогою розробленого веб-скрапера, щоб визначити оптимальний інтернет-магазин для здійснення купівлі товару;

– розробити аналітичну систему моніторингу цін електронних товарів, в якій користувач зможе визначити інтернет-магазин з оптимальною ціною та якістю товару;

**Об’єкт дослідження** – процес аналізу цін електронних товарів з різних інтернет магазинів.

**Предмет дослідження** – методи та засоби для аналізу даних, задля визначення найкращого інтернет-магазину, в якому ціна та якість електронного товару є оптимальною.

**Новизна одержаних результатів.** Подальшого розвитку набув метод прогнозування ціни електронних товарів інтернет-магазинів, який на відміну від інших, враховує відгуки покупців та забезпечує їх пошук на основі рекомендованої середньозваженої ціни серед усіх інтернет-магазинів.

**Практичне значення** роботи полягає у можливості використання розробленої інформаційної технології, для визначення товарів, що найбільше відповідають їх потребам.

**Апробація результатів магістерської кваліфікаційної роботи.** Результати кваліфікаційної роботи доповідались на Всеукраїнській науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, 2021-2022 рр.).

**Публікації результатів магістерської кваліфікаційної роботи**. Опубліковано тези в збірнику матеріалів Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, 2021-2022 рр.) [1].

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Огляд існуючих аналогів

Здебільшого більшість аналогічних систем можна винести в категорію компаратори.

Що стосується бізнесу електронної комерції, більшість секторів працюють в умовах жорсткої конкуренції, коли маленькі цифрові магазини борються за те, щоб привернути увагу користувачів і зрештою спонукати їх купувати на їхніх веб-сайтах. Потужний інструмент, який допоможе вам отримати більше трафіку та запропонувати більш конкурентоспроможні ціни, — це порівняльник цін.

Таким чином, онлайн-компаратори цін — це інструменти, які постійно сканують та відстежують зміни цін на товари вашого конкурента в режимі реального часу.

Залежно від того, в якій галузі та вертикалі ви працюєте (хоча зазвичай у більшості галузей це саме так), існує значна кількість конкурентів, які пропонують однакові продукти за однаковою або дуже схожою ціною. Це означає:

– Для користувачів ціна є вирішальним фактором, коли мова йде про вибір, де купити продукт. У цьому немає жодних сумнівів.

– Обсяг цін, які потрібно сканувати, дуже великий через значну кількість конкурентів. Тому вам доведеться вкладати занадто багато часу і грошей.

Ось тут вступає в гру порівняння цін в Інтернеті. Оскільки він порівнює ціни та пропозиції в Інтернеті в режимі реального часу, він забезпечує високу конкурентоспроможність у порівнянні з вашими цифровими конкурентами.

Порівняння цін — це інструменти, які допомагають Інтернету. Користувачі, які визначають ціни на продукти, отримані в різних магазинах електронної комерції. Ці компаратори допомагають здійснювати покупки, знайти найкращі пропозиції та пропозиції.

Таким чином, ми маємо намір впровадити інтелектуальний компаратор цін за допомогою технології Data Mining (DM), яка шукає продукти в різних магазинах електронної комерції та знаходить подібні результати продуктів.

Цей проект спрямований на оцінку існуючих технологій можливості, інтегруючи вилучення інформації з Інтернету та аналіз даних за допомогою методів DM в порівняльниках цін, створення механізмів, які дозволяють нам виконувати різні підходи. Оцінка буде проводитися на основі а порівняння, де виділяємо відмінності, переваги і недоліки інтелектуального порівняння цін і порівняння поточних ринкових цін з існуючими методологіями які підтверджують правдивість результатів методик застосовано.

E-Katalog – сервіс порівняння цін та характеристик товарів, сайт належить торговій системі Nadavi. Розробка проекту була запущена в 2000 році, а вже в 2001 в мережі була запущена повноцінна версія сайту. Також в 2007 році був запущений сервіс Megazilla – більш проста версія E-Katalog. Даний ресурс також дозволяв порівняти ціна, вибрати потрібний товар, але при цьому має полегшений інтерфейс та позбавлений додаткових функцій, присутніх в E-Katalog. Інтерфейс сервісу представлений на рисунку 1.1.

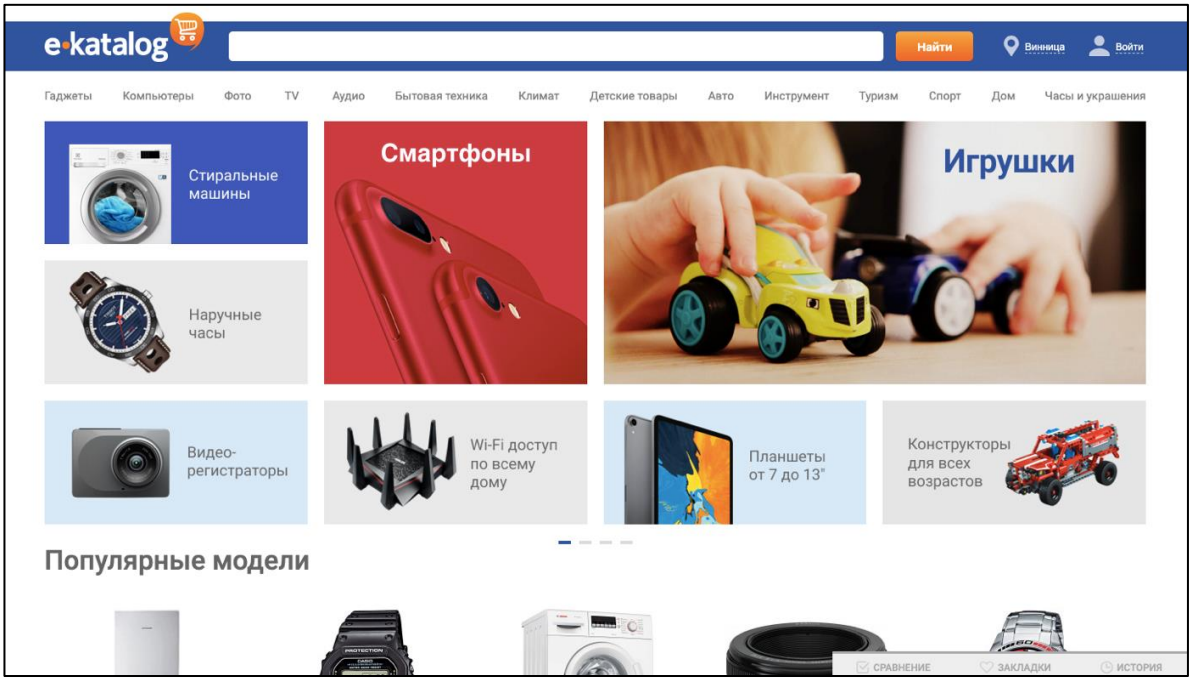



Рисунок 1.1 – Домашня сторінка інтернет-магазину E-Katalog

До плюсів можна віднести:

– існування двох версій, завдяки чому кожен користувач може обрати те, що саме для нього буде зручніше;

– великий вибір та можливість порівняння товарів, так як сервіс надає можливість перегляду товарів з багатьох популярних магазинів України;

– можливість залишати відгуки та перегляд відгуків інших клієнтів;

– на сайті існує розділ з статтями які містять корисні поради, короткі огляди товарів та іншу корисну інформацію з світу техніки;

До мінусів системи відносяться:

– сайт існує лише на російській мові;

– сайт надає інформацію лише про ціни представлених магазинів.

Розглянемо також сервіс порівняння цін Price.ua (рис. 1.2).

Price.ua — сайт для порівняння цін в різних магазинах. Входить до складу міжнародного інтернет-холдингу Universal Commerce Group. Ресурс має механізм пошуку товарів і послуг в інтернет-магазинах, порівнюючи їх ціни і характеристики. Ресурс отримує оплату за кожен перехід покупця на сайт, де продається той чи інший товар.

Окрім каталогу товарів, сайт містить відгуки покупців, а також рейтинг фірм, сформований користувачами. Це допомагає оцінити надійність продавця.

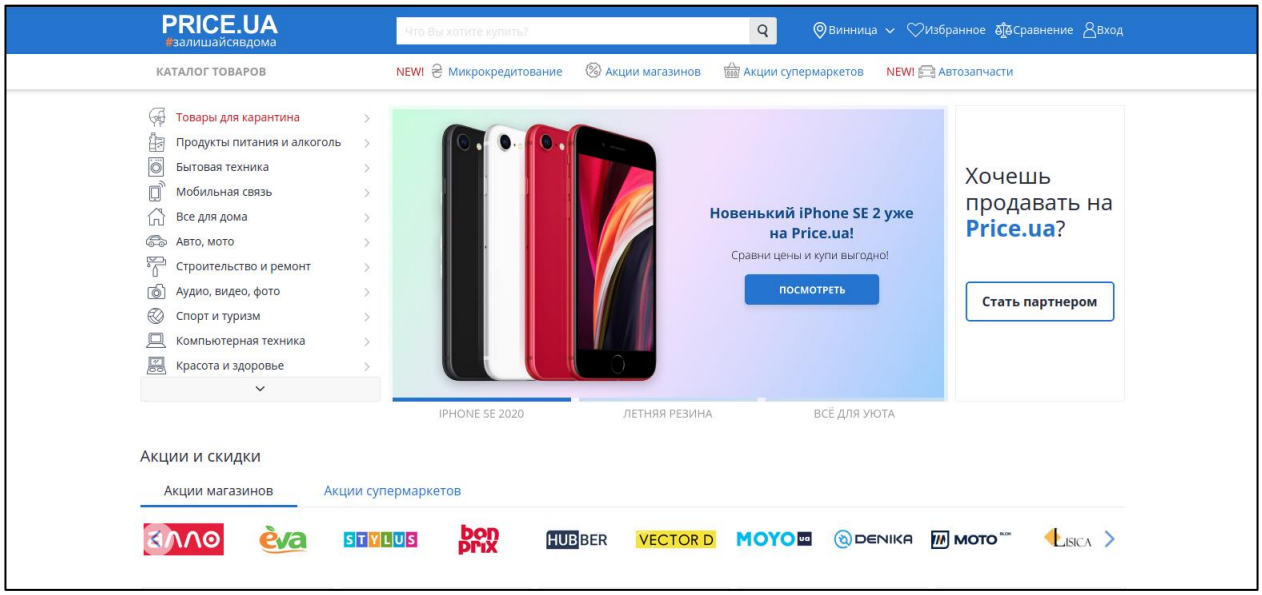



Рисунок 1.2 – Домашня сторінка інтернет-магазину Price.ua

Плюси ресурсу:

– зручний інтерфейс пошуку товарів;

– великий вибір та можливість порівняння товарів, так як сервіс надає можливість перегляду товарів з багатьох популярних магазинів України;

– можливість залишати відгуки та перегляд відгуків інших клієнтів як тесктових так і відео оглядів обраного товару;

– на сайті існує розділ з статтями які містять корисні поради, короткі огляди товарів та іншу корисну інформацію з світу техніки;

– ресурс має свої соціальні сторінки в чотирьох соціальних мережах де можна дізнаватись про новини та оновлення ресурсу.

Мінуси ресурсу:

– відсутність українського інтерфесу;

– велика кількість рекламних банерів на сторінках;

– сайт надає інформацію лише про ціни представлених магазинів.

Останнім досліджуваним аналогом буде hotline.ua (рис. 1.3). Ресурс був створений в жовтні 1992 року як каталог прайс-листів на комп'ютерну техніку і спочатку поширювався за допомогою модемного доступу. У 1993 р. з'явилася його паперова версія, яка щотижня виходила до кінця 2005 року. З 1 січня 2006 почав свою роботу інтернет-проект – hotline.ua. У березня 2014 року з'явилася українська версія сайту. Щоденна аудиторія сайту становить 250 000. Відвідувачів, щомісячна аудиторія – 7.5млн користувачів.

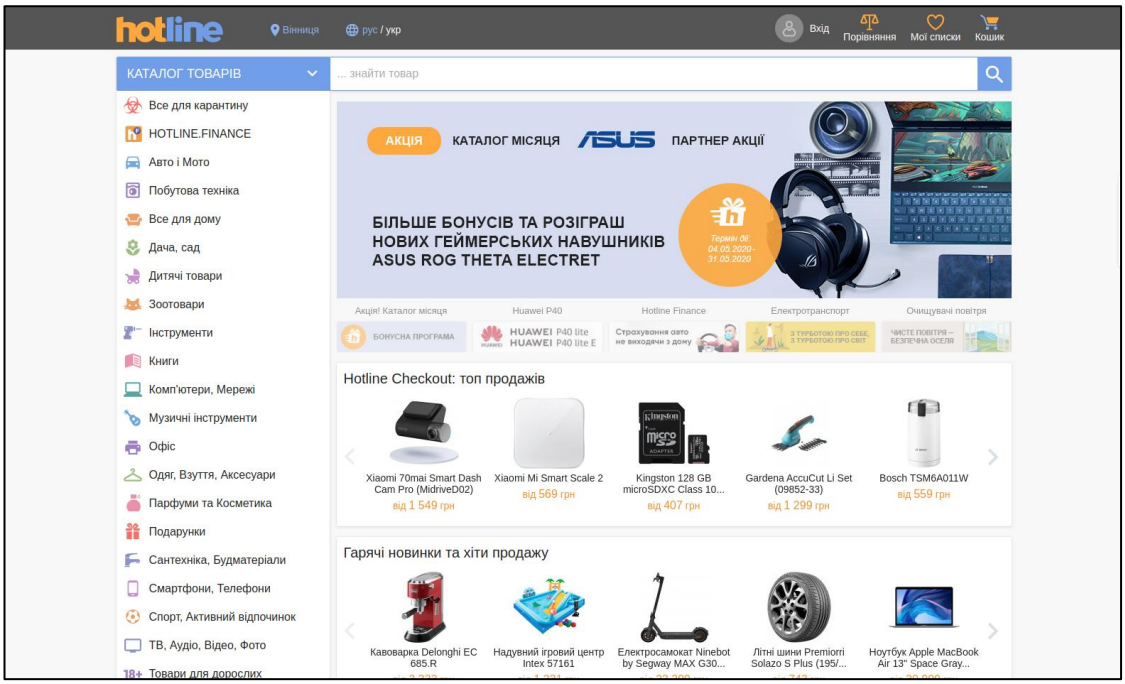



Рисунок 1.3 – Домашня сторінка інтернет-магазину hotline.ua

Основні можливості ресурсу:

– великий вибір та можливість порівняння товарів, так як сервіс надає можливість перегляду товарів з багатьох популярних магазинів України;

– можливість залишати відгуки та перегляд відгуків інших клієнтів як про сам товар та і про окремі магазини представлені на сайті;

– рейтинги користувачів який формується виходячи з кількості і корисності його відгуків про товари, про магазини, відеоогляди і відповідей на питання інших користувачів. Відеоогляди, зняті самими користувачами, цінуються вище, ніж додані ними сторонні ролики;

– гід покупця де можна знайти статті на тему «Як обрати», розділ оновлюється щотижнево, що забезпечує завжди корисну та актуальну інформацію про товари.

– Hotline Checkout – це сервіс, що дозволяє користувачам hotline.ua розрахуватися платіжними картами за товари, представлені на hotline.ua інтернет-магазинами. В цьому випадку hotline виступає як "посередник" при оформленні замовлення і проведенні платежу.

Проаналізувавши аналоги, можна зробити висновок, що в кожного ресурсу

є свої переваги та недоліки, особливо помітна проблема з українською мовою в найпопулярніших українських інтернет-магазинів. На рисунку 1.4 представлена порівняльна таблиця цих трьох сервісів.

| Основні функції | e-k | PRICE.UA | h |
|---|---|---|---|
| Україномовний інтерфейс | ✗ | ✗ | ✓ |
| Можливість оплати беспосередньо на | ✗ | ✗ | ✓ |
| Соціальні сторінки | ✗ | ✓ | ✗ |
| Спрощена версія сайту | ✓ | ✗ | ✗ |
| Розділ з статтями | ✓ | ✓ | ✗ |

Рисунок 1.4 – Порівняльна таблиця

## 1.2 Аналіз методів прогнозування рекомендованої ціни на товар

Найбільш відомими методами прогнозування є регресіннйий аналіз, штучні нейронні мережі та нечітка логіка.

Першим розглянутим методом буде регресійний аналіз.

У статистичному моделюванні регресійний аналіз являє собою набір статистичних процесів для оцінки відносин між змінними. Вона включає в себе безліч методів моделювання та аналізу декількох змінних, коли основна увага приділяється взаємозв'язку між залежною змінною і однією або кількома незалежними змінними (або "предикторами"). Більш конкретно, регресійний аналіз допомагає зрозуміти, як типове значення залежної змінної (або "змінної критерію") змінюється, коли будь-яка з незалежних змінних змінюється, а інші незалежні змінні фіксуються.

Найчастіше, регресійний аналіз оцінює умовне очікування залежної змінної з урахуванням незалежних змінних – тобто середнє значення залежної

змінної, коли незалежні змінні фіксовані. Рідше фокус знаходиться на квантилі або іншому параметрі розташування умовного розподілу залежної змінної з урахуванням незалежних змінних. У всіх випадках слід оцінювати функцію незалежних змінних, що називається функцією регресії. В регресійному аналізі також представляє інтерес характеризувати зміну залежної змінної навколо прогнозування функції регресії з використанням розподілу ймовірностей. Пов'язаний, але чіткий підхід – це аналіз необхідних умов, який оцінює максимальне (а не середнє) значення залежної змінної для даного значення незалежної змінної (лінійка стелі, а не центральна лінія) для ідентифікації яке значення незалежної змінної є необхідним, але недостатнім для заданого значення залежної змінної [4].

Регресійний аналіз – це добре відома методика статистичного навчання, яка корисна для висновку про зв’язок між залежною змінною Y та p незалежними змінними X=[X1|…|Xp]. Залежна змінна Y також відома як змінна відповіді або результат, а змінні Xk (k=1,…,p) як предиктори, пояснювальні змінні або коваріати. Точніше, регресійний аналіз має на меті оцінити 00математичне відношення f() для пояснення Y у термінах X як, Y=f(X), використовуючи спостереження (xi,Yi),i=1,…,n, зібрані на n спостережувані статистичні одиниці. Якщо Y описує одновимірну випадкову величину, регресія називається одновимірною регресією, інакше її називають багатовимірною регресією. Якщо Y залежить тільки від однієї змінної x (тобто p=1), регресія називається простою, інакше (тобто p>1) регресія називається множинною, див. Faraway, 2004; Fahrmeir та ін., 2013; Jobson, 1991; Rao, 2002; Sen and Srivastava, 1990).

Для стислості в цій главі ми обмежуємо нашу увагу одновимірною регресією (простою та множинною), так що Y=(Y1,Y2…Yn)T представляє вектор спостережуваних результатів і представляє планову матрицю спостережуваних коваріатів, де xi=(xi1,…,xip)T (або xi=(1,xi1,…,xip)T ). У цьому параметрі X=[X1|…|Xp] є p-вимірною змінною (p≥1).

Регресійний аналіз є найстарішою і, мабуть, найбільш широко використовуваною багатофакторною технікою в соціальних науках. На відміну

від попередніх методів, регресія є прикладом аналізу залежності, в якому змінні не розглядаються симетрично. У регресійному аналізі мета полягає в тому, щоб отримати прогноз однієї змінної з урахуванням значень інших. Щоб задовольнити цю зміну точки зору, використовується інша термінологія та позначення. Прогнозована змінна зазвичай позначається y, а змінні-провісники — x з доданими індексами, щоб відрізняти одну від іншої. У лінійній множинній регресії ми шукаємо лінійну комбінацію предикторів (часто називають регресорними змінними). Наприклад, у освітніх дослідженнях нас може цікавити, наскільки успішність у школі може бути передбачена домашніми обставинами, віком або успішністю в минулому. На практиці регресійні моделі оцінюються методом найменших квадратів за допомогою відповідного програмного забезпечення. Важливі практичні питання стосуються найкращого вибору найкращих змінних регресора, перевірки значущості їхніх коефіцієнтів та встановлення меж довіри до прогнозів.

Наступний метод аналізу – це штучні нейронні мережі.

Штучні нейронні мережі – це обчислювальні моделі, які надихають людський мозок. Багато з останніх досягнень були зроблені в області штучного інтелекту, включаючи розпізнавання голосу, розпізнавання зображень, робототехніку з використанням штучних нейронних мереж. Штучні нейронні мережі – це біологічне моделювання, яке виконується на комп’ютері для виконання певних конкретних завдань, як-от:

– Кластеризація
– Класифікація
– Розпізнавання образів

Штучна нейронна мережа — це взаємозв'язана мережа вузлів, уподібнена до безкрайої мережі нейронів у головному мозку. Тут кожним круговим вузлом представлено штучний нейрон, а стрілкою — з'єднання виходу одного штучного нейрону зі входом іншого (рис 1.5).

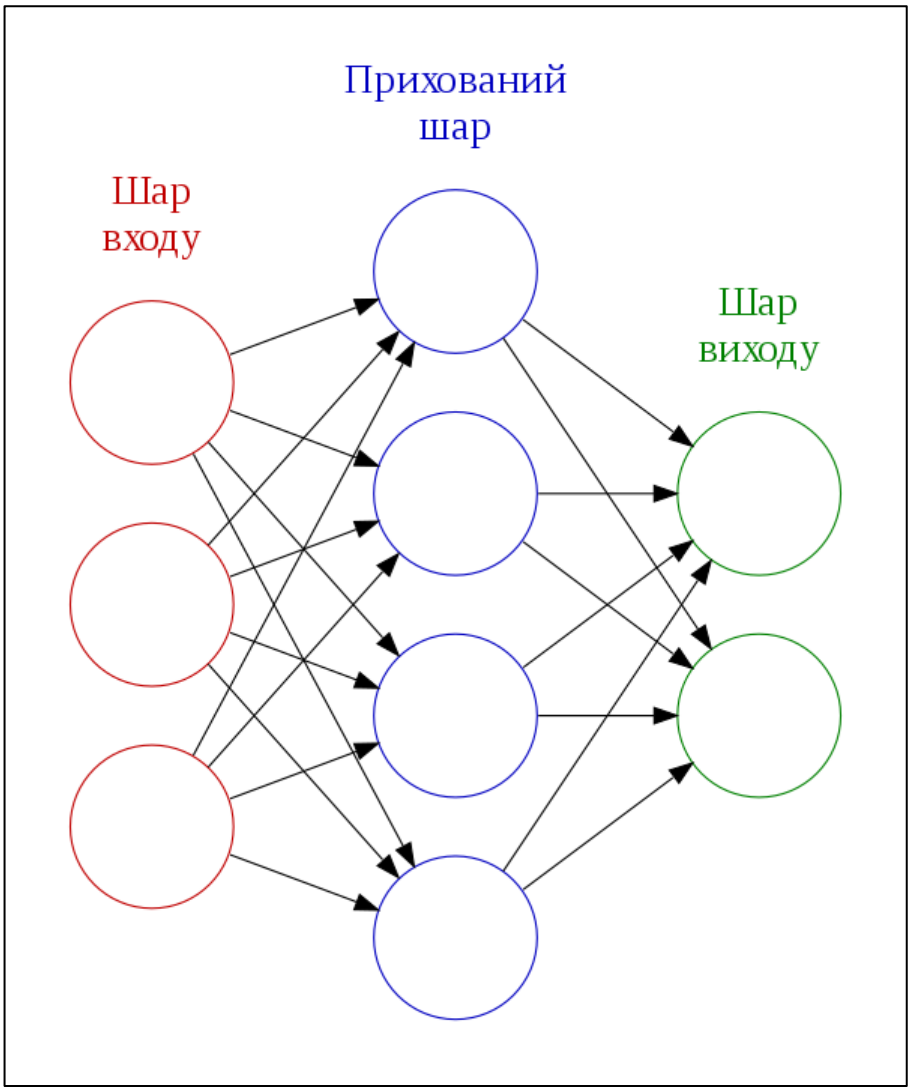



Рисунок 1.5 – Схема роботи штучної нейронної мережі

Штучні нейронні мережі, загалом, – це біологічно натхненна мережа штучних нейронів, налаштованих на виконання певних завдань. Ці біологічні методи обчислень відомі як наступний великий прогрес у обчислювальній індустрії [5]. Сама нейронна мережа не є алгоритмом, а скоріше рамкою для багатьох різних алгоритмів машинного навчання для спільної роботи та обробки складних вхідних даних [6]. Такі системи "навчаються" виконувати завдання, розглядаючи приклади, як правило, не запрограмовані з якимись конкретними правилами. Наприклад, при розпізнаванні зображень вони можуть навчитися ідентифікувати зображення, які містять кішок, аналізуючи приклади зображень, які були позначені вручну як "кіт" або "без кішки" і використовуючи результати для ідентифікації кішок в інших зображеннях. Вони роблять це без будь-яких

попередніх знань про кішок, наприклад, що вони мають хутро, хвости, вуса і котячі обличчя. Замість цього вони автоматично генерують ідентифікаційні характеристики з навчального матеріалу, який вони обробляють.

ШНМ базується на колекції з'єднаних одиниць або вузлів, які називаються штучними нейронами, які вільно моделюють нейрони в біологічному мозку. Кожне з'єднання, подібно синапсам в біологічному мозку, може передавати сигнал від одного штучного нейрона до іншого. Штучний нейрон, який отримує сигнал, може обробляти його, а потім сигналізувати про додаткові штучні нейрони, пов'язані з ним.

У звичайних реалізаціях ШНМ сигнал при з'єднанні між штучними нейронами є дійсним числом, і вихід кожної штучної нейронної системи обчислюється деякою нелінійною функцією суми її входів. Зв'язки між штучними нейронами називаються «ребрами». Штучні нейрони і ребра, як правило, мають вагу, що регулюється, як навчання. Вага збільшує або зменшує силу сигналу при з'єднанні. Штучні нейрони можуть мати такий поріг, що сигнал надсилається тільки, якщо агрегатний сигнал перетинає цей поріг. Як правило, штучні нейрони агрегуються в шари. Різні шари можуть виконувати різні види перетворень на своїх входах. Сигнали проходять від першого шару (вхідного шару) до останнього шару (вихідного шару), можливо, після проходження шарів кілька разів.

І останнім розглянутим методом буде нечітка логіка.

Нечітка логіка – це форма багатозначної логіки, в якій значення істинності змінних можуть бути будь-якими реальними числами від 0 до 1, обидва включно. Він використовується для управління концепцією часткової істини, де значення істини може коливатися між абсолютно правдивою та абсолютно хибною. [6] Навпаки, в булевій логіці значення істинності змінних можуть бути лише цілими значеннями 0 або 1.

Термін нечітка логіка був введений 1965 р. Пропозицією теорії нечітких множин Лотфі Заде. Однак нечітка логіка вивчалася з 1920-х років як нескінченно цінна логіка, зокрема Лукасевич та Тарскі.

Нечітка логіка заснована на спостереженні, що люди приймають рішення на основі неточної та нечислової інформації. Нечіткі моделі або множини – це математичний засіб подання нечіткості та неточності інформації (звідси термін нечіткий). Ці моделі мають можливість розпізнавати, представляти, маніпулювати, інтерпретувати та використовувати дані та інформацію, які є невиразними та не мають певної визначеності.

Для кращого розуміння нечіткої логіки на рисунку 1.6 представлено графік оцінки кімнатної температури.

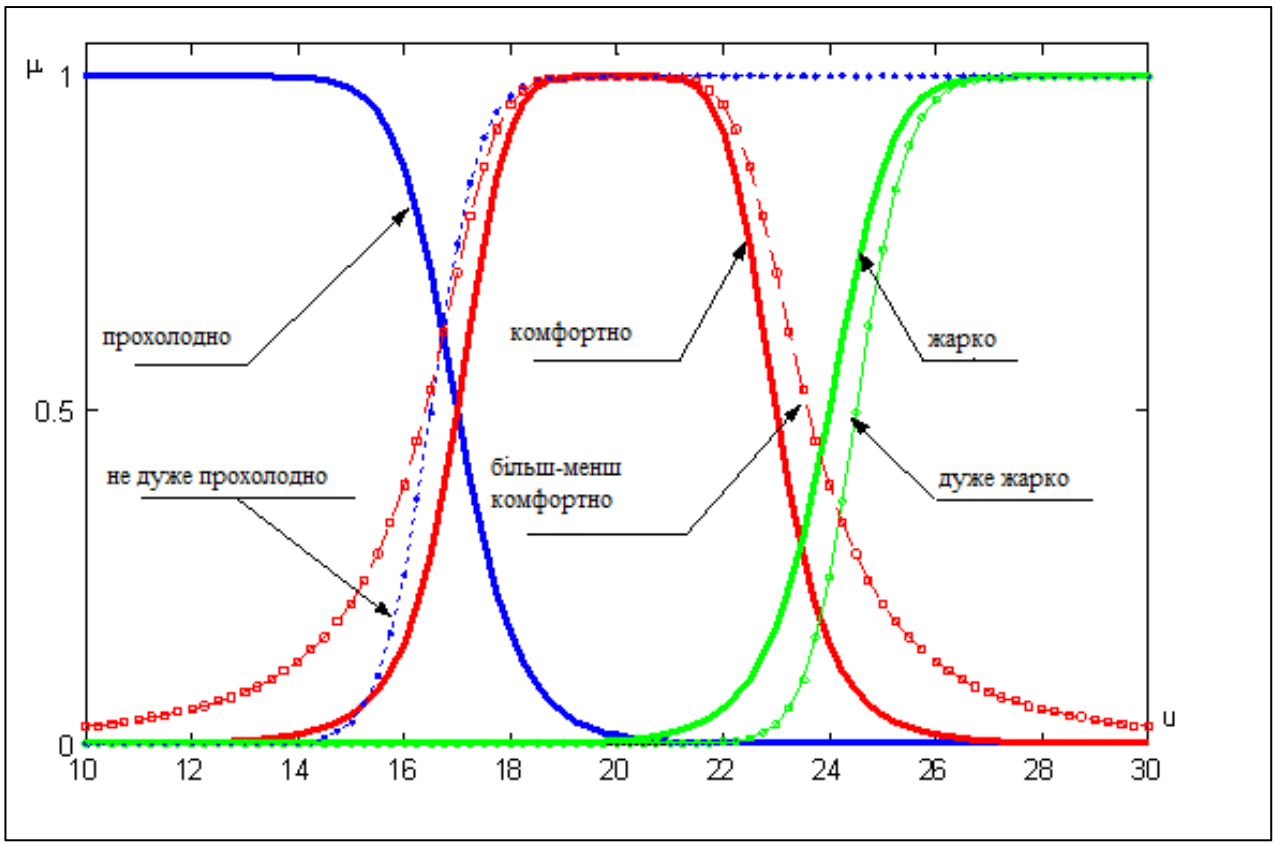



Рисунок 1.6 – Графік оцінки кімнатної температури

Нечітка логіка призначена для моделювання логічних міркувань з нечіткими або неточними твердженнями на кшталт «Петро молодий (багатий, високий, голодний тощо)». Це відноситься до сімейства багатозначних логік, де істинні значення інтерпретуються як ступені істини. Істинність логічно складеної пропозиції, як-от «Карлес високий, а Кріс багатий», визначається

істинністю її компонентів. Іншими словами, як і в класичній логіці, нав’язується істинність-функціональність.

Нечітка логіка виникла в контексті теорії нечітких множин, введеної Лотфі Заде (1965). Нечітка множина присвоює елементам всесвіту ступінь належності, як правило, дійсне число з інтервалу [0,1]. Нечітка логіка виникає шляхом присвоєння ступеня істинності судженням. Стандартний набір значень істинності (ступенів) — це реальний одиничний інтервал [0,1], де 0 позначає «повністю невірно», 1 являє собою «повністю істину», а інші значення відносяться до часткової істинності, тобто проміжні ступені правда.

«Нечітка логіка» часто розуміється в дуже широкому сенсі, який включає всі види формалізмів і прийомів, що стосуються систематичної обробки певних ступенів (див., наприклад, Nguyen & Walker 2000). Зокрема, в інженерних контекстах (нечіткое керування, нечітка класифікація, м’які обчислення) воно спрямоване на ефективні обчислювальні методи, толерантні до неоптимальності та неточності (див., наприклад, Ross 2010). Цей запис зосереджується на нечіткій логіці в обмеженому сенсі, створеній як дисципліна математичної логіки після фундаментальної монографії Петра Гаєка (1998) і в наш час зазвичай називається «математична нечітка логіка». Докладніше про історію різних варіантів нечіткої логіки можна знайти в Bělohlávek, Dauben, & Klir 2017.

Математична нечітка логіка зосереджується на логіках, заснованих на істинно-функціональному обліку часткової істини і вивчає їх у дусі класичної математичної логіки, досліджуючи синтаксис, теоретичну семантику моделей, системи доказів, повноту тощо; як на рівні пропозиції, так і на рівні предиката

## 1.3 Обґрунтування і вибір методів та інструментів

Для реалізації представленої системи доцільним буде розробка скраперу для збирання інформації з інтернет магазинів а також веб-додаток на якому буде представлена вся інформація, що дозволить будь яким користувачам з можливістю виходу до мережі інтернет користуватись системою. Для збереження

та обробки будь яких даних юуде використано реляційну базу дланих, а для відображення та використання іншого функціоналу системи буде розроблений веб-додаток зі зручним для користувача інтерфейсом.

### 1.3.1 Вибір засобів реалізації інтерфейсу користувача

Для реалізації інтерфейсу системи було використано мову розмітки HTML, мову стилю сторінок CSS та мову програмування JavaScript. Мова розмітки гіпертексту (HTML) – це стандартна мова розмітки для документів, призначених для відображення у веб-браузері. Цьому можуть допомогти такі технології, як каскадні таблиці стилів (CSS) та мови сценаріїв, такі як JavaScript.

Каскадні таблиці стилів (CSS) – це таблиця стилів, яка використовується для опису подання документа, написаного мовою розмітки, як HTML [7]. CSS – це основна технологія всесвітньої павутини, поряд із HTML та JavaScript.

CSS розроблений, щоб забезпечити розділення презентації та контенту, включаючи макет, кольори та шрифти. Цей поділ може покращити доступність контенту, забезпечити більшу гнучкість та контроль у конкретизації характеристик презентації, дасть змогу декільком веб-сторінкам обмінюватися форматуванням, вказавши відповідний CSS в окремому файлі .css, а також зменшити складність та повторність структурного вмісту [8].

JavaScript, що часто скорочується як JS, – мова програмування, що відповідає специфікації ECMAScript. JavaScript – це високорівневий, часто своєчасно складений та багатопарадигмальний. Він має синтаксис фігурної дужки, динамічне введення тексту, орієнтовану на прототип об'єктну орієнтацію та функції першого класу.

Поряд із HTML та CSS, JavaScript є однією з основних технологій всесвітньої павутини [9]. JavaScript включає інтерактивні веб-сторінки і є невід'ємною частиною веб-додатків. Переважна більшість веб-сайтів використовують його для поведінки на стороні клієнта, а всі основні веб-браузери мають спеціальний механізм JavaScript для його виконання.

Як мова для багатьох парадигм, JavaScript підтримує керовані подіями, функціональні та імперативні стилі програмування. Він має інтерфейси прикладного програмування (API) для роботи з текстом, датами, регулярними виразами, стандартними структурами даних та Модель об'єкта документа (DOM). Однак сама мова не включає жодного вводу / виводу (вводу / виводу), такого як мережеві засоби, сховища чи графічні засоби, оскільки хост-середовище (як правило, веб-браузер) надає ці API.

Для спрощення процесу верстки сайту буде використовуватись фреймворк Bootstrap. Bootstrap (також відомий як Twitter Bootstrap) – це вільний набір інструментів для створення сайтів і веб-додатків. Включає в себе HTML і CSS-шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript-розширення.

Bootstrap використовує сучасні напрацювання в області CSS і HTML, тому необхідно бути уважним при підтримці старих браузерів.

Вибір зумовлений тим, що в сукупності представлені мови є чи не найкращим рішенням для розробки інтерфейсу веб-додатку в наш час, що також є за статисктикою найпопулярнішим рішенням існуючим проектів в мережі.

### 1.3.2 Вибір засобів для обробки даних та реалізації серверної частини веб-додатку

Для реалізація веб-скрапера та серверної частини системи було обрано мову програмування Python.

У сучасному конкурентному світі кожен шукає шляхи для інновацій та використання нових технологій. Веб-скрейпинг (також званий вилученням веб-даних або скрейпінгом даних) надає рішення для тих, хто хоче отримати доступ до структурованих веб-даних в автоматизованому режимі. Веб-скрейпинг корисний, якщо загальнодоступний веб-сайт, з якого ви хочете отримати дані, не має API, або він має, але надає лише обмежений доступ до даних.

У сучасному конкурентному світі кожен шукає шляхи для інновацій та використання нових технологій. Веб-скрепінг (також званий вилученням веб-даних або скрейпінгом даних) надає рішення для тих, хто хоче отримати доступ до структурованих веб-даних в автоматизованому режимі. Веб-скрейпинг корисний, якщо загальнодоступний веб-сайт, з якого ви хочете отримати дані, не має API, або він має, але надає лише обмежений доступ до даних.

Веб-скрепінг – це процес автоматизованого збору структурованих веб-даних. Це також називається вилученням веб-даних. Деякі з основних випадків використання веб-скрейпінгу включають моніторинг цін, аналіз цін, моніторинг новин, генерацію потенційних клієнтів і дослідження ринку серед багатьох інших.

Загалом, вилучення веб-даних використовується людьми та підприємствами, які хочуть використовувати величезну кількість загальнодоступних веб-даних для прийняття більш розумних рішень.

Якщо ви коли-небудь копіювали та вставляли інформацію з веб-сайту, ви виконували ту ж функцію, що й будь-який веб-скребок, тільки в мікроскопічному, ручному масштабі. На відміну від повсякденного, приголомшливого процесу вилучення даних вручну, веб-скрейпінґ використовує інтелектуальну автоматизацію для отримання сотень, мільйонів або навіть мільярдів точок даних із, здавалося б, нескінченного кордону Інтернету.

І це не повинно дивувати, оскільки веб-скрейпінґ надає щось дійсно цінне, чого не може ніщо інше: він надає вам структуровані веб-дані з будь-якого загальнодоступного веб-сайту.

Більше, ніж сучасна зручність, справжня сила веб-скрейпінгу даних полягає в його здатності створювати та впроваджувати деякі з найбільш революційних бізнес-додатків у світі. Термін «трансформаційний» навіть не означає, як деякі компанії використовують дані, отримані в Інтернеті, щоб покращити свою діяльність, інформуючи керівників про рішення, аж до індивідуального досвіду обслуговування клієнтів.

Вилучення веб-даних – також широко відоме як копіювання даних – має величезний спектр застосувань. Інструмент зняття даних може допомогти вам швидко й точно автоматизувати процес вилучення інформації з інших веб-сайтів. Він також може переконатися, що отримані вами дані акуратно організовані, що спрощує аналіз і використання для інших проектів.

У світі електронної комерції скрегінг даних широко використовується для моніторингу цін конкурентів. Це єдиний практичний спосіб для брендів перевірити ціни на продукти та послуги своїх конкурентів, що дозволяє їм точно налаштовувати власні цінові стратегії та залишатися на випередження. Він також використовується як інструмент для виробників, щоб переконатися, що роздрібні продавці дотримуються інструкцій щодо цін на свою продукцію. Організації та аналітики, що вивчають ринок, залежать від вилучення даних з Інтернету, щоб оцінити настрої споживачів, відстежуючи онлайнові огляди продуктів, статті новин та відгуки.

У фінансовому світі існує величезна кількість програм для вилучення даних. Інструменти скрейпінгу даних використовуються для отримання інформації з новин, використовуючи цю інформацію для визначення інвестиційних стратегій. Так само дослідники та аналітики залежать від вилучення даних для оцінки фінансового стану компаній. Компанії, що надають страхові та фінансові послуги, можуть видобути багатий масив альтернативних даних, витягнутих з Інтернету, щоб розробити нові продукти та політики для своїх клієнтів.

На цьому програми для вилучення веб-даних не закінчуються. Інструменти скрейпінгу даних широко використовуються в моніторингу новин і репутації, журналістиці, моніторингу SEO, аналізі конкурентів, маркетингу на основі даних і генерації потенційних клієнтів, управлінні ризиками, нерухомості, академічних дослідженнях та багато іншого.

Python – інтерпретована мова програмування високого рівня, загального призначення. Створена Гідо ван Россумом і вперше випущена в 1991 році, філософія дизайну Python підкреслює читабельність коду завдяки помітному

використанню значного пробілу. Його мовні конструкції та об'єктно-орієнтований підхід мають на меті допомогти програмістам написати чіткий логічний код для малих та масштабних проектів [10].

Python – мова програмування з багато парадигми. Об'єктно-орієнтоване програмування та структуроване програмування повністю підтримуються, і багато його функцій підтримують функціональне програмування та орієнтоване на аспекти програмування (у тому числі метапрограмування та метаоб'єктів (так звані магічні методи)). Багато інших парадигм підтримуються через розширення, включаючи дизайн за контрактом та логічне програмування [11].

Для збору та оновлення даних про товари будуть використані фреймворк Selenium та бібліотека requests в комплексі з beautifullsoup4.

Для автоматизованого збору товарів буде використано фреймворк автоматизованої роботи з браузером Selenium. Selenium WebDriver, або просто Selenium – це драйвер браузера, тобто програмна бібліотека, що не має інтерфейсу користувача, яка дозволяє різним іншим програмам взаємодіяти з браузером, керувати його поведінкою, отримувати від браузера якісь дані і змушувати браузер виконувати якісь команди.. Selenium пропонує інструмент відтворення для складання функціональних взаємодій з браузером без необхідності вивчати мову тестового сценарію (Selenium IDE). Він також надає тестовий домен-мову (Selenese) для написання тестів на ряді популярних мов програмування, включаючи C #, Groovy, Java, Perl, PHP, Python, Ruby та Scala. Потім скрипти можуть працювати на більшості сучасних веб-браузерів. Selenium працює на Windows, Linux та macOS. Це програмне забезпечення з відкритим кодом, що випускається під ліцензією Apache License 2.0.

Для оновлення інформації для актуальних товарів в базі даних за допомогою асинхронної черги задач будуть використані бібліотеки requests та beautifullsoup4. Requests — це бібліотека HTTP для мови програмування Python. Requests — одна з найпопулярніших бібліотек Python, яка не входить до складу Python. Було запропоновано, щоб запити розповсюджувалися з Python за замовчуванням. Дана бібліотека використовується для отримання вихідного

коду сторінки, далі для вилучення потрібної для нас інформації використано бібліотеку beautifullsoup4. Beautiful Soup — це бібліотека Python для вилучення даних із файлів HTML та XML. Він являє собою аналізатор, який забезпечує ідіоматичні способи навігації, пошуку та модифікації дерева аналізу. Зазвичай це економить програмістам години або дні роботи.

В свою чергу для реалізації серверної частини системи буде використанно фреймворк Django.

Django – це високорівневий Pyrhon веб-фреймворк з відкритим кодом, який дозволяє швидко створювати безпечні та підтримувані веб-сайти, та відповідає архітектурній схемі модельного шаблону (MTV). Його підтримує Django Software Foundation (DSF), американська незалежна організація, створена як неприбуткова [12].

Django – це фреймворк для створення веб-застосунків за допомогою мови програмування Python.

Django був створений у 2005 році, коли веб-розробники з газети Lawrence Journal-World стали використовувати Python як мову для створення веб-сайтів. А у 2008 році вийшов публічний перший реліз фреймворку. На сьогоднішній день він продовжує розвиватись. Так, поточною версією фреймворку на момент написання цієї статті є версія 2.0, яка вийшла 3 грудня 2017 року. Та й постійно виходять підверсії.

Django є досить популярним. Він використовується на багатьох сайтах, у тому числі таких, як Pinterest, PBS, Instagram, BitBucket, Washington Times, Mozilla та багатьох інших.

Фреймворк є безкоштовним. Він розвивається як Open Source, його вихідний код відкритий, його можна знайти репозиторії на github.

Фреймворк Django реалізує архітектурний патерн Model-View-Template або скорочено MVT, який є модифікацією розповсюдженого у веб-програмуванні патерну MVC (Model=-View-Controller) [13].

Схематично ми можемо представити архітектуру MVT Django представлено на рисунку 1.7.

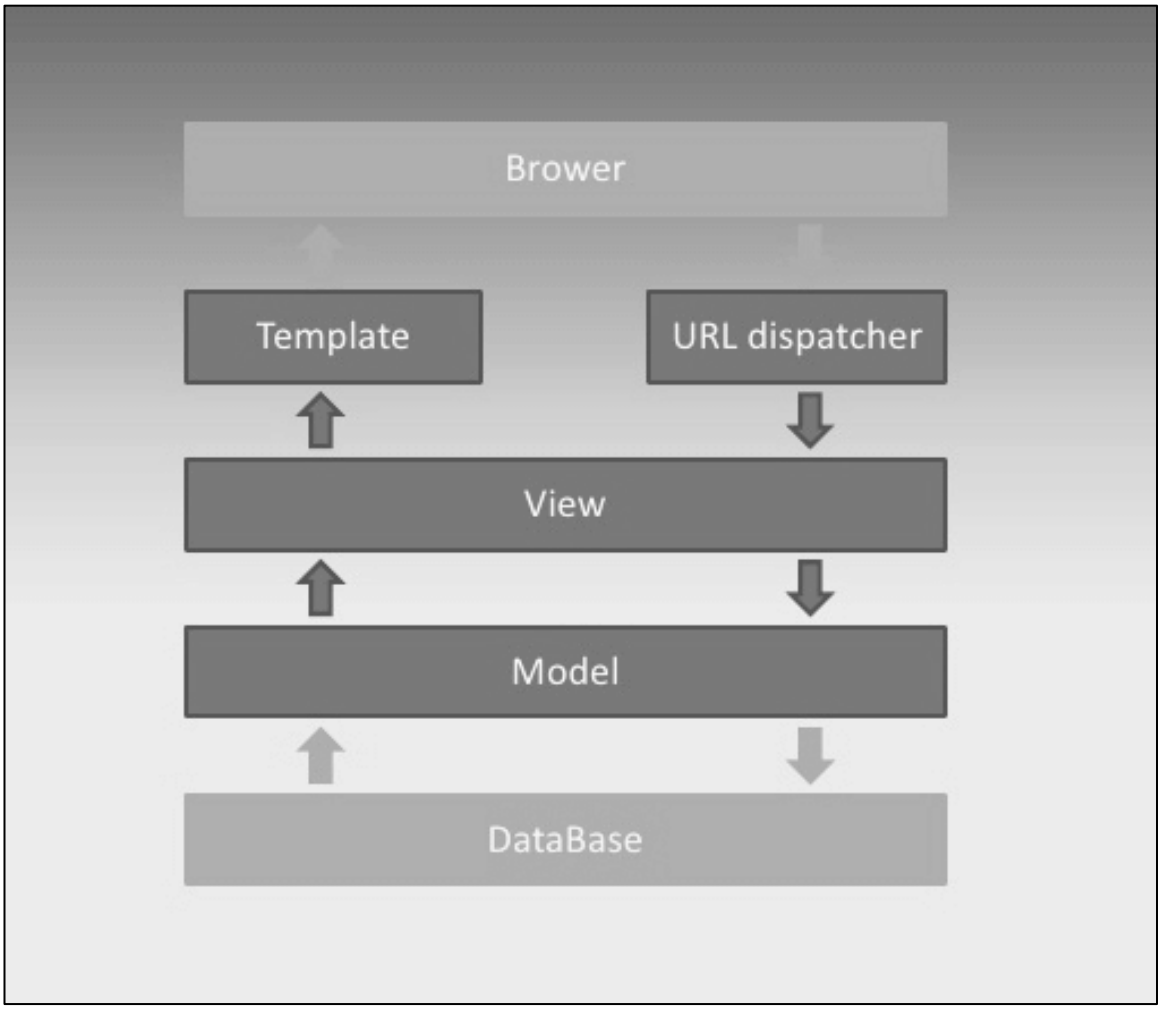



Рисунок 1.7 – Схема роботи MVT

Основні елементи патерну:

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

– View: отримує запит, обробляє його та відправляє у відповідь користувачеві певну відповідь. Якщо обробки запиту необхідно звернення до моделі та базі даних, то View взаємодіє із нею. Для створення відповіді може використовуватися Template або шаблони. В архітектурі MVC цьому компоненту відповідають контролери (але не уявлення).

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

– Template: представляє логіку уявлення у вигляді згенерованої розмітки

HTML. У MVC цьому компоненту відповідає View, тобто уявлення [14].

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

Переваги саме цього фреймворку:

– Django слідує філософії «все включено» і надає майже все, що розробники можуть захотіти зробити «з коробки». Оскільки все, що вам потрібно, є частиною єдиного продукту, все це бездоганно працює разом, відповідає послідовним принципам проектування і має велику і актуальну документацію;

– Django може бути використаний для створення практично будь-якого типу веб-сайтів – від систем керування контентом та wiki до соціальних мереж та новинних сайтів. Він може працювати з будь-яким середовищем клієнта і може доставляти контент практично в будь-якому форматі (включаючи HTML, RSS-канали, JSON, XML і т. д.);

– Django допомагає розробникам уникнути багатьох поширених помилок безпеки, надаючи фреймворк, розроблений, щоб «робити правильні речі» для автоматичного захисту сайту. Наприклад, Django надає безпечний спосіб керування обліковими записами користувачів та паролями, уникаючи поширених помилок, таких як розміщення інформації про сеанс у файли cookie, де вона вразлива (замість цього файли cookie містять лише ключ, а фактичні дані зберігаються в базі даних) або безпосереднє зберігання паролів замість хеш пароля;

– Django використовує компонентну "shared-nothing" архітектуру (кожна її частина є незалежною від інших і, отже, може бути замінена або змінена, якщо

це необхідно). Чіткий розділ розділу означає, що Django може масштабуватися при збільшенні трафіку, шляхом додавання обладнання на будь-якому рівні: сервери кешування, сервери баз даних або сервери додатків. Одні з найбільш завантажених сайтів успішно масштабували Django (наприклад, Instagram та Disqus, якщо назвати лише два з них);

– код Django написаний з використанням принципів та шаблонів проектування, які заохочують створення підтримуваного та повторно використовуваного коду. Зокрема, у ньому використовується принцип "Don't Repeat Yourself" (DRY, "не повторюйся"), тому немає непотрібного дублювання, що скорочує обсяг коду. Django також сприяє групуванню пов'язаних функціональних можливостей у повторно використовувані "додатки" і, на нижчому рівні, групує пов'язаний код у модулі (відповідно до шаблону Model View Controller (MVC)).

В якості системи управління баз даних, яка напряму буде взаємодіяти з веб-фреймворком Django було обрано реляційно орієнтовано СУБД PostgreSQL.

PostgreSQL (вимовляється як post-gress-Q-L) — це система управління реляційною базою даних (СУБД) з відкритим вихідним кодом, розроблена всесвітньою командою волонтерів. PostgreSQL не контролюється жодною корпорацією чи іншою приватною особою, а вихідний код доступний безкоштовно.

PostgreSQL, який спочатку називався Postgres, був створений в UCB професором інформатики на ім’я Майкл Стоунбрейкер. Stonebraker заснував Postgres у 1986 році як продовження свого попередника Ingres, який зараз належить Computer Associates [15].

– 1977-1985 рр. − Розроблено проект під назвою INGRES:
    1. Підтвердження концепції для реляційних баз даних;
    2. Засновано компанію Ingres в 1980 році;
    3. Куплений Computer Associates в 1994 році.
– 1986-1994 рр. – ПОСТГРЕС:
    1. Розробка концепцій в INGRES з акцентом на об'єктну орієнтацію та

мову запитів – Quel;

2. Кодова база INGRES не була використана як основа для POSTGRES;
3. Комерціалізується як Illustra (куплена Informix, куплена IBM).

– 1994-1995 рр. − Postgres95:

1. Підтримка SQL була додана в 1994 році;
2. Випущений як Postgres95 у 1995 році;
3. Перевипущена як PostgreSQL 6.0 у 1996 році.
4. Створення глобальної команди розробників PostgreSQL

PostgreSQL працює на всіх основних операційних системах, включаючи Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) і Windows. Він підтримує текст, зображення, звуки та відео, а також включає інтерфейси програмування для C/C++, Java, Perl, Python, Ruby, Tcl та Open Database Connectivity (ODBC) [16].

PostgreSQL підтримує велику частину стандарту SQL і пропонує багато сучасних функцій, включаючи наступні:

– складні SQL-запити,
– підвибір SQL,
– сторонні ключі,
– тригер,
– перегляди,
– транзакції,
– багатоверсійний контроль паралельності (MVCC),
– потокова реплікація (станом на 9.0),
– гарячий режим очікування (станом на 9.0).

Також для реалізації асинхронної черги задач, яка допоможе з актуалізацією потрібної інформації про представлені на сторінках товари, в розробці використано такі програмні рішення, як Celery та Redis. Черги завдань використовуються як механізм для розподілу роботи між потоками або машинами. Вхідні дані черги завдань — це одиниця роботи, яка називається завданням, виділені робочі процеси потім постійно відстежують чергу для

виконання нової роботи. Celery спілкується за допомогою повідомлень, зазвичай використовуючи брокера для посередництва між клієнтами та працівниками. Щоб ініціювати завдання, клієнт ставить повідомлення в чергу, а потім брокер доставляє повідомлення працівнику. Система Celery може складатися з кількох працівників і брокерів, поступаючись місцем високої доступності та горизонтального масштабування. Celery написано на Python, але протокол може бути реалізований будь-якою мовою. На додаток до Python є node-celery для Node.js, клієнт PHP, gocelery для golang і rusty-celery для Rust. В якості брокера повідомлень для налаштування черги задач всередині системи обрано NoSQL СУБД Redis. Redis – резидентна система управління базами даних класу NoSQL з відкритим вихідним кодом, що працює зі структурами даних типу "ключ – значення". Використовується як баз даних, так реалізації кешів, брокерів повідомлень. Орієнтована досягнення максимальної продуктивності на атомарних операціях.

Для нормальної роботи всієї системи суспільно, а також для зручного розгортання на будь якій обчислювальній системі використано Docker, який дозволить помістити всі складові проекту в «контейнер». Docker — це набір продуктів «платформа як послуга», які використовують віртуалізацію на рівні ОС для доставки програмного забезпечення в пакетах, які називаються контейнерами. Контейнери ізольовані один від одного і об'єднують власне програмне забезпечення, бібліотеки та файли конфігурації; вони можуть спілкуватися один з одним через чітко визначені канали. Оскільки всі контейнери використовують послуги одного ядра операційної системи, вони використовують менше ресурсів, ніж віртуальні машини.

Docker — це платформа для контейнерування з відкритим кодом. Це дозволяє розробникам упаковувати програми в контейнери — стандартизовані виконувані компоненти, що поєднують вихідний код програми з бібліотеками операційної системи (ОС) і залежностями, необхідними для виконання цього коду в будь-якому середовищі. Контейнери спрощують доставку розподілених

додатків і стають дедалі популярнішими, оскільки організації переходять на власну хмарну розробку та гібридні мультихмарні середовища.

Розробники можуть створювати контейнери без Docker, але платформа спрощує, спрощує та безпечніше створювати, розгортати та керувати контейнерами. Docker — це, по суті, набір інструментів, який дозволяє розробникам створювати, розгортати, запускати, оновлювати та зупиняти контейнери за допомогою простих команд та автоматизації, що економить роботу, через єдиний API.

Docker також відноситься до Docker, Inc. (посилання знаходиться за межами IBM), компанії, яка продає комерційну версію Docker, і до проекту з відкритим кодом Docker (посилання знаходиться за межами IBM), якому Docker, Inc. та багато інших організацій і особи сприяють.

Контейнери стають можливими завдяки ізоляції процесів і можливостям віртуалізації, вбудованим у ядро Linux [17]. Ці можливості – такі як групи керування (Cgroups) для розподілу ресурсів між процесами та простори імен для обмеження доступу процесів або видимості в інших ресурсах або областях системи – дозволяють кільком компонентам програми спільно використовувати ресурси одного екземпляра хоста, що працює. система майже так само, як гіпервізор дозволяє кільком віртуальним машинам (ВМ) спільно використовувати центральний процесор, пам’ять та інші ресурси одного апаратного сервера.

Як результат, контейнерна технологія пропонує всю функціональність і переваги віртуальних машин, включаючи ізоляцію додатків, економічно ефективну масштабованість і одноразову можливість, а також важливі додаткові переваги:

– Менша вага: на відміну від віртуальних машин, контейнери не несуть корисне навантаження цілого екземпляра ОС і гіпервізора; вони включають лише процеси та залежності ОС, необхідні для виконання коду. Розміри контейнера вимірюються в мегабайтах (порівняно з гігабайтами для деяких віртуальних машин), вони краще використовують апаратну ємність і мають

швидший час запуску.

– Підвищена ефективність використання ресурсів: за допомогою контейнерів ви можете запускати в кілька разів більше копій програми на тому самому обладнанні, ніж за допомогою віртуальних машин. Це може зменшити витрати на хмару.

– Покращена продуктивність розробників: у порівнянні з віртуальними машинами, контейнери швидше і легше розгортають, надають і перезапускають. Це робить їх ідеальними для використання в конвеєрах безперервної інтеграції та безперервної доставки (CI/CD), а також краще підходять для команд розробників, які використовують методи Agile і DevOps.

### 1.3.3 Вибір середовища реалізації

Для написання коду буде використано два середовища – Visual Studio Code та Pycharm. Для реалізації користувацького інтерфесу буде використано Visual Studio Code.

Особливості включають підтримку налагодження, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти, рефакторинг коду та вбудований Git. Користувачі можуть змінювати тему, ярлики клавіатури, налаштування та встановлювати розширення, що додають додаткових функцій. Вихідний код – вільний та відкритий, випущений згідно з дозвільною ліцензією MIT. Скомпільовані двійкові файли безкоштовно для будь-якого використання. У опитуванні розробників Stack Overflow 2019 Visual Studio Code потрапив у найпопулярніший інструмент середовища для розробників, 50,7% із 87317 респондентів заявили, що використовують його.

В свою чергу для написання скрапера та серверної частини веб-додатку буде використано Pycharm.

PyCharm – це інтегроване середовище розробки (IDE), яке використовується в комп'ютерному програмуванні, спеціально для мови Python. Він розроблений чеською компанією JetBrains. [6] Він забезпечує аналіз коду,

графічний налагоджувач, інтегрований тестер одиниць, інтеграцію з системами управління версіями (VCSes) та підтримує веб-розробки з Django, а також Data Science з Anaconda.

PyCharm – кросплатформенний та працює на операційних системах Windows, macOS та Linux. Community Edition випускається під ліцензією Apache, також є Professional Edition з додатковими функціями – випущений під власною ліцензією.

Отже, в процесі розгляду технічних та програмних засобів для реалізації аналітичної системи моніторингу цін інтернет-магазинів було проведено аналіз існуючих на ринку аналогічних рішень, обґрунтувано обрані методи вирішення поставлених задач, інструменти та середовища реалізації, обрано засоби організації та реалізації бази даних та обрано засоби реалізації інтерфейсу користувача.

## 1.4 Висновки

В цьому розділі проведено огляд предметної області, який показав про необхідність розробки сервісу для визначення інтернет-магазину, в якому товар буде опимальною за ціною та якістю, а також обгрунтовано вибір методів та інструментів для реалізації такого сервісу. Для визначення найкращого магазину проведено аналіз відомих методів прогнозування рекомендованої ціни на електронні товари, який показав про необхідніть врахування користувацьких відгуків.

# 2 РОЗРОБКА МОДЕЛІ ПРОГНОЗУВАННЯ РЕКОМЕНДОВАНОЇ ЦІНИ ЕЛЕКТРОННИХ ТОВАРІВ

## 2.1 Підготовка даних

Так як представлений веб-додаток повинен представляти користувачу можливість аналізувати ринок товарів, то інформацію про товари потрібно звідкись збирати. Для рішення цієї задачі було розроблено веб-скрапер.

Веб-скрапінг – це процес збору даних, які відображаються в інтерфесі веб-сайтів. Програмне забезпечення для скрапінгу веб-сторінок може отримати доступ до всесвітньої павутини безпосередньо за допомогою протоколу передачі гіпертексту (HTTP) або через веб-браузер [18]. Хоча веб-скрапінг може бути здійснено вручну користувачем програмного забезпечення, термін зазвичай стосується автоматизованих процесів, реалізованих за допомогою бота або веб-сканера. Це форма копіювання, в якій конкретні дані збираються та копіюються з Інтернету, як правило, в центральну локальну базу даних або електронну таблицю для подальшого пошуку або аналізу.

Розшифровка веб-сторінки включає в себе завантаження сторінки та вилучення з неї. Тому сканування веб-сторінки є головним компонентом скрапінгу веб-сторінок для отримання інформації та подальшої обробки. Вміст сторінки може бути проаналізований, шуканий, переформатований, його дані скопійовані в електронну таблицю тощо. Веб-скрепери зазвичай виймають щось із сторінки, щоб використовувати його з іншою метою десь в іншому місці. Прикладом може бути пошук та копіювання імен та номерів телефонів або компаній та їх URL-адресу до списку.

Веб-скрапінг використовується для скрапінгу контактів, як компонент програм, що використовуються для індексації веб, веб-видобутку та видобутку даних, онлайн-моніторингу зміни цін та порівняння цін, скрапінг інформації про товари (для спостереження за конкуренцією), збору списків нерухомості, моніторинг погодних даних, виявлення змін на веб-сайтах, дослідження,

відстеження присутності та репутації в Інтернеті, веб-розміщення та інтеграція веб-даних [19].

Веб-сторінки створюються за допомогою текстових мов розмітки (HTML та XHTML) і часто містять безліч корисних даних у текстовій формі. Однак більшість веб-сторінок розроблені для кінцевих користувачів людини, а не для зручності автоматизованого використання. В результаті були розроблені спеціалізовані інструменти та програмне забезпечення для полегшення скрапінгу веб-сторінок.

В представленому випадку буде використовуватись фреймворк Selenium.

Принцип роботи скрипта:

– Підключення до браузера;

– з’єднання з базою даних;

– відносно того до якої категорії відноситься товар скрипт переходить до виконання потрібної функції;

– збір списку всіх активних товарів представлених в потрібній категорії товару;

– відкриття потрібних сторінок товару та збір інформації;

– внесення зібраної інформації до бази даних;

– вибір наступного товару та перехід до пункту 4.

Код реалізації скрапера наведений на рис. 2.1 – 2.3.

```
# Підключення до браузера
executable_path = "/home/grow/Documents/Diplom/Pars_eShop/chromedriver"
chromeOptions = webdriver.ChromeOptions()
driver = webdriver.Chrome(executable_path=executable_path)

# з'єднання з базою даних
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="",
    database="eShop"
)
cursor = mydb.cursor()
```

Рисунок 2.1 – Частина коду яка відповідає підключення до браузера та БД

```
def cycle(id):
    print(id)
    query = "select count(id) from FoxApp_contributor;"
    cursor.execute(query)
    counter = cursor.fetchone()
    while id <= counter[0]:
        query = "select category_id from FoxApp_contributor where id = '" + str(id) + "';"
        cursor.execute(query)
        type_ = cursor.fetchone()
        try:
            if type_[0] == 1:
                pc(id)
            elif type_[0] == 2:
                laptop(id)
            elif type_[0] == 3:
                smartphone(id)
        except:
            id = id + 1
            cycle(id)
        id = id + 1
```

Рисунок 2.2 – Частина коду яка визначає категорію товару

```
citrus = link[2]
allo = link[3]
stylus = link[4]
driver.get(itbox)
driver.find_element_by_class_name("characteristic-toggle").click()
elements1 = []
info = driver.find_elements_by_tag_name("td")
for elem in info:
    elements1.append(elem.text)
i = 0
for elem in elements1:
    i = i+1
    if elem == "Серия процессора":
        cpu = elements1[i]
    if elem == "Частота процессора, ГГц":
        speed = elements1[i]
    if elem == "Модель видеокарты":
        videocard = elements1[i]
    if elem == "Тип памяти":
        ram_type = elements1[i]
    if elem == "Объем установленной памяти":
        ram = elements1[i].split("ГБ")[0]
    if elem == "Типы внутренних накопителей":
        hd_type = elements1[i]
    if elem == "Объем HDD" or elem == "Объем SSD":
        hd = elements1[i].split(" ")[0]
try:
    itbox_price = driver.find_element_by_class_name("stuff-price").text.split("грн")[0].replace(' ', '')
    itbox_price = int(itbox_price)
except:
    itbox_price = 0
try:
    driver.get(stylus)
    driver.implicitly_wait(5)
    stylus_price = driver.find_element_by_class_name("regular-price").text.split("грн")[0].replace(' ', '')
    stylus_price = int(stylus_price)
except:
    stylus_price = 0
if int(hd) < 10:
    hd = int(hd)*1000
cpu_serial = cpu.split(' ')[0] + ' ' + cpu.split(' ')[1] + ' ' + cpu.split(' ')[2]
category_id = 1
print(id.split("'")[1], cpu, speed, videocard, ram_type, ram, hd_type, hd, itbox_price, rozetka_price, citrus_price, allo_price,
      stylus_price, category_id, cpu_serial)
sql = "INSERT INTO FoxApp_products (id, cpu, speed, videocard, ram_type, ram, hd_type, hd, " \
      "diagonal, main_cam, back_cam, front_cam, color, os, price_itbox, price_rozetka, " \
      "price_citrus, price_allo, price_stylus, category_id, model_id, cpu_serial) " \
      "VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
val = (id.split("'")[1], cpu, float(speed), videocard, ram_type, int(ram), hd_type, hd, DEFAULT, DEFAULT,
       DEFAULT, DEFAULT, DEFAULT, stylus_price: int ce, rozetka_price,
       citrus_price, allo_price, stylus_price, category_id, id.split("'")[1], cpu_serial)
print(val)
cursor.execute(sql, val)
mydb.commit()
```

Рисунок 2.3 – Частина коду яка збирає дані про товар та додає їх до БД

Зібрані дані зберігаються в базі даних та в подальшому будуть використані для потрібного аналізу, а також дана база даних буде підключена до веб-додатку в якому потрібні дані будуть представлені для користувачів.

Фрагмент вибірки підготовлених даних наведено в таблиці 2.1.

Таблиця 2.1 – Фрагмент вибірки

| CPU | Speed | Videocard | Ram_t | ram | hd_type | hdd | P_itbox | P_rozeta |
|---|---|---|---|---|---|---|---|---|
| Intel | 2.3 | Intel Iris | DDR4 | | SSD | 256 | 37999 | |
| Intel | 1.8 | Intel HD | DDR4 | | SSD | 128 | 23999 | 23999 |
| AMD | 2.1 | Nvidia GF | DDR4 | | HDD | 500 | 20999 | 22999 |
| AMD | 2.3 | Radeon Vega | DDR5 | | SSD | 512 | 21999 | 20999 |

2.2 Розвідувальний аналіз

В таблиці 2.2 наведено статистичні характеристики цін на товари, що продаються в різних інтернет-магазинах.

Таблиця 2.2 – Статистичні характеристики електронних товарів

| ID | itbox | rozetka | citrus | allo | stylus | AVG | DISP | STDEV |
|---|---|---|---|---|---|---|---|---|
| 1 | 37999 | | 364999 | | 35239 | 36579 | 1909200 | 1381 |
| 2 | 23999 | 23999 | 23999 | 23999 | 23999 | 23999 | 0 | 0 |
| 3 | 20999 | 22999 | 24999 | | 20691 | 23533 | 3810496 | 1952 |
| 4 | 21999 | 20999 | 21999 | 25299 | 11357 | 22197 | 3351012 | 1830 |

З таблиці 2.2 видно, що розкид ціни від його математичного очікування є достатньо великим. На рисунку 2.4 представлено зміни ціни на ноутбук “ASUS M5098DA-EJ068” протягом 2020 року.

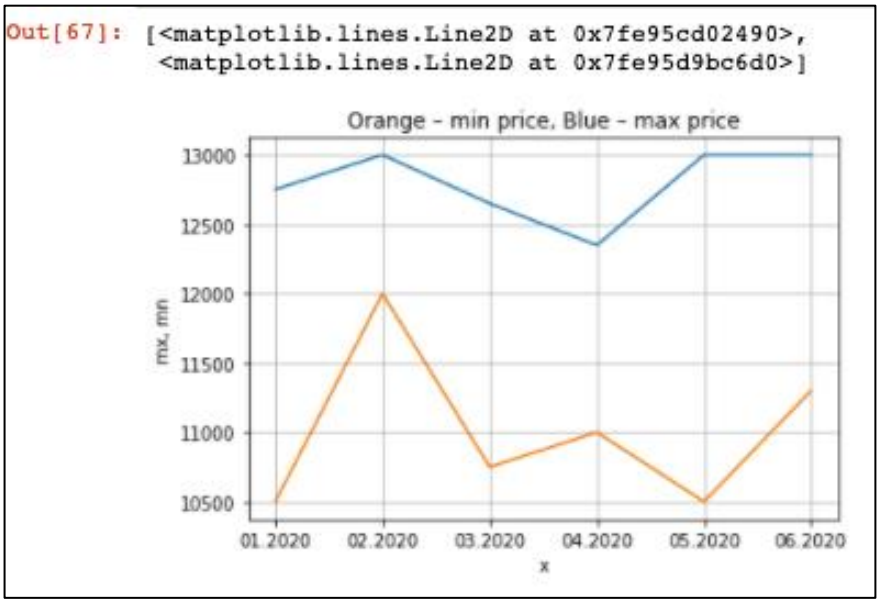



Рисунок 2.4 – Зміни ціни протягом 2020 року

На рисунку 2.5 наведено розподіл ціни відносно кількості відгуків для товарів з кожного інтернет-магазину. Візуально видно що користувачі найчастіше обирають відносно середні показники ціни та відгуків, так як здебільшого в інтернет магазинах з малою ціною, мало відгуків, а тому і нема надії на якість придбаного товару в представленому інтернет магазині.

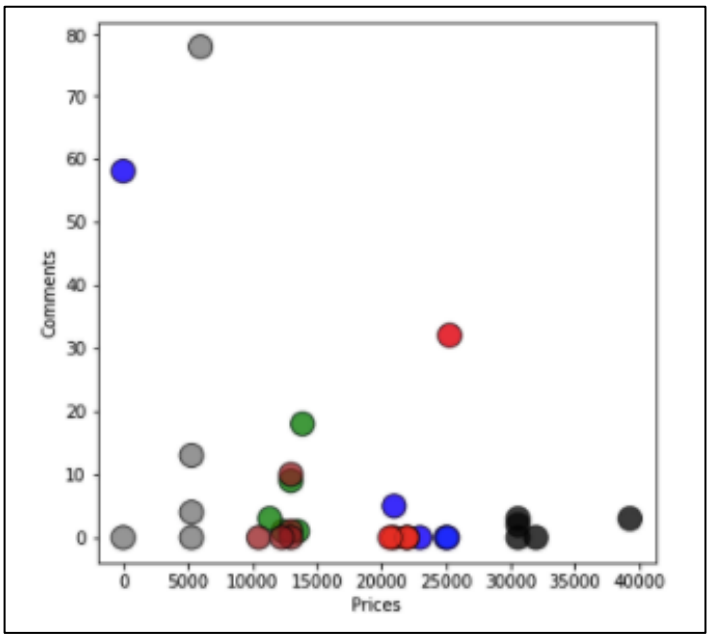



Рисунок 2.5 – Графік розподілу цін

## 2.3 Розробка модуля прогнозування ціни

Представимо модель прогнозування рекомендованої ціни на товар за такою залежністю:

$$y = f(V, P)\ , \tag{2.1}$$

де y – рекомендована ціна

V = { v1, v2, …, $v_m$} – вектор відгуків, в якому елемент $v_i$ означає кількість відгуків на товар для і-го магазину, i=$1,\vec{m}$.

p = {$p_1$, $p_2$, …,$p_m$} – вектор цін, в якому елемент $p_i$ означає ціну на товар для і-го магазину, i = $1,\vec{m}$, а m – кількість магазинів.

Розрахуємо зважену вагу відгуків на товар для і-го магазину таким чином:

$$w_i = \frac{vi}{v_1+v_2+\cdots+v_m}, i = 1,\vec{m} \tag{2.2}$$

причому $\sum_{i=1}^{m} wi = 1$

Далі вирішимо рекомендовану ціну за таким співвідношенням:

$$p = \frac{1}{m}*(p_1*w_1+p_2*w_2+\ldots.+p_m*w_n) \tag{2.3}$$

У формулі (2.3) рекомендована ціна вираховується як середня зважена ціна товару серед усіх інтернет магазинів. Зрозуміло, що чим більше вага, тим більше залишених відгуків користувачами, і тому ціна з інтернет магазину в якому найбільше відгуків буде більше впливати на рекомендовану ціну. Реалізація алгоритму прогнозування представлена на рисунку 2.6.

```
def best_price(self):
    com = [self.itbox_com_count, self.rozetka_com_count, self.citrus_com_count, self.allo_com_count,
           self.stylus_com_count]
    prices = [self.price_itbox * com[0], self.price_rozetka * com[1], self.price_citrus * com[2],
              self.price_allo * com[3], self.price_stylus * com[4]]
    count = 0
    for i in range(len(prices)):
        if prices[i] == 0:
            del com[i - count]
            count = count + 1
    price = (sum(prices) / sum(com))
    round(price, 0)
    print(price)
    return price
```

Рисунок 2.6 – Алгоритм прогнозування найкращої ціни на ринку

На рисунку 2.7 наведено результат роботи моделі для товару “ASUS M5098DA-EJ068” за 2020 рік.

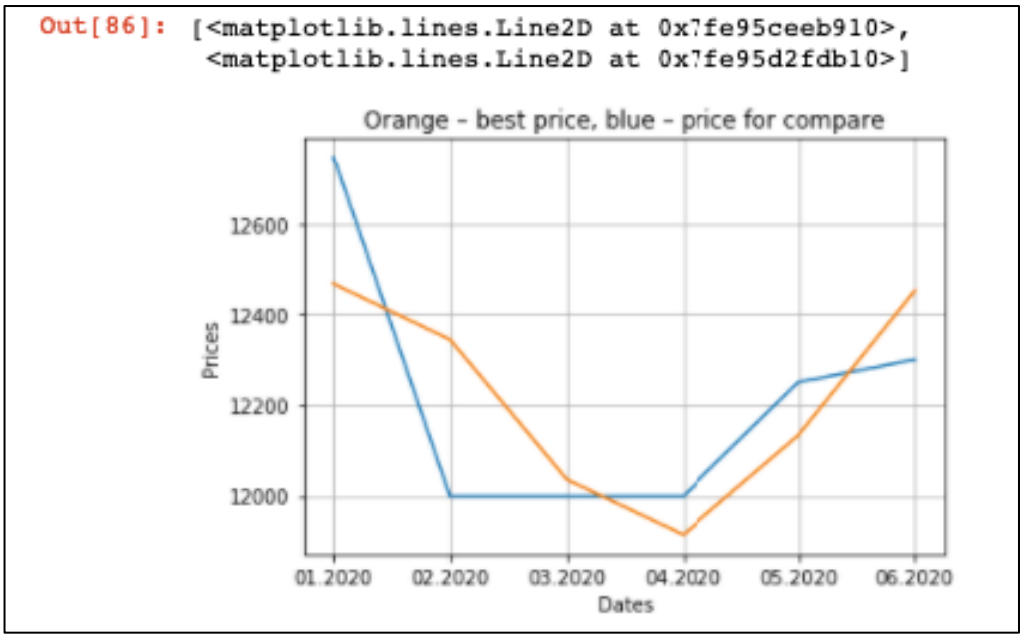



Рисунок 2.7 – Результат роботи моделі

Також даному рисунку наведено найпопулярнішу ціну за діапазон часу, що розглядається. Популярна ціна визначена в магазині, в якому кількість відгуків є найбільшою для представленого товару. З графіка видно, що рекомендована ціна достатньо точно описує найпопулярнішу ціну. Похибка складає близько 5%.

## 2.4. Висновки

В цьому розділі проведено розвідувальний аналіз цін на електронні товари в різних інтернет-магазинах, який показав досить широкий діапазон варіювання, що ускладнює процес купівлі товару. Крім того розроблено математичну модель прогнозування рекомендованої ціни на основі кількості відгуків про товар в різних інтернет-магазинів. Запропонована модель дозволяє визначити інтернет-магазин, в якому товар є оптимальним за ціною та якістю.

# 3 РОЗРОБКА АНАЛІТИЧНОЇ СИСТЕМИ МОНІТОРИНГУ ЦІН ЕЛЕКТРОННИХ ТОВАРІВ В ІНТЕРНЕТ-МАГАЗИНАХ

## 3.1 Формування процесів та методів роботи системи

Основним завданням аналітичної системи моніторингу цін інтернет-магазинів для електронних товарів є моніторинг популярних інтернет-магазинів та предоставлення користувачу аналізу цін. Процес моніторингу інтернет магазинів та обробки даних полягає в срапінгу інформації з популярних інтернет-магазинів україни та в подальшій обробці їх перед внесенням до бази даних. Функціонал системи моніторингу інтернет-магазинів для ролі адміністратор:

– додавання товарів для збору інформації в базу даних;

– додати, видалити або відредагувати інформацію про внесені товари в базі даних;

– запуск скрипта, який в свою чергу збере інформацію з сайтів та додасть її до бази даних.

UML діаграма прецедентів системи моніторингу інтернет-магазинів представлена на рисунку 3.1.

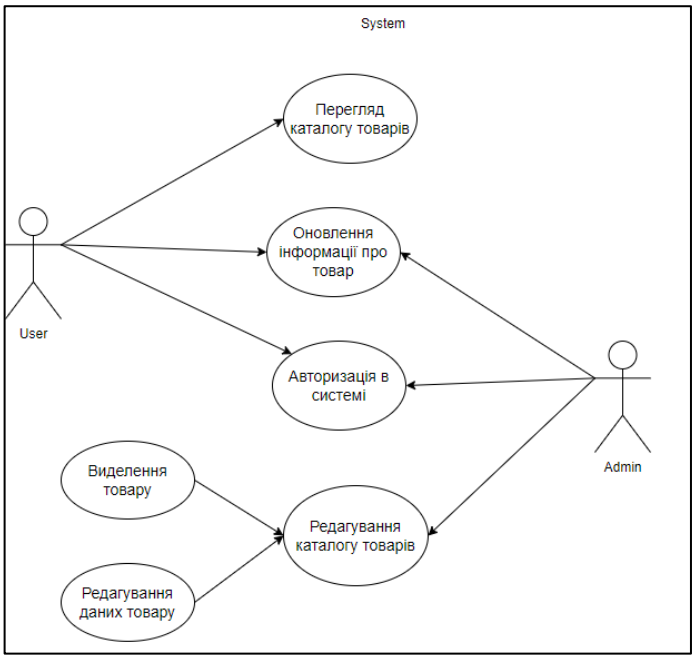



Рисунок 3.1 – UML діаграма прецедентів

Для забезпечення зручності створення та редагування даних, або ж використання веб-додатку в цілому побудовано UML-діаграми активності (Activity Diagram) для кожного процесу відповідно.

На рисунку 3.2 зображеня UML діаграма авторизації та реєстрації нової користувачів в системі.

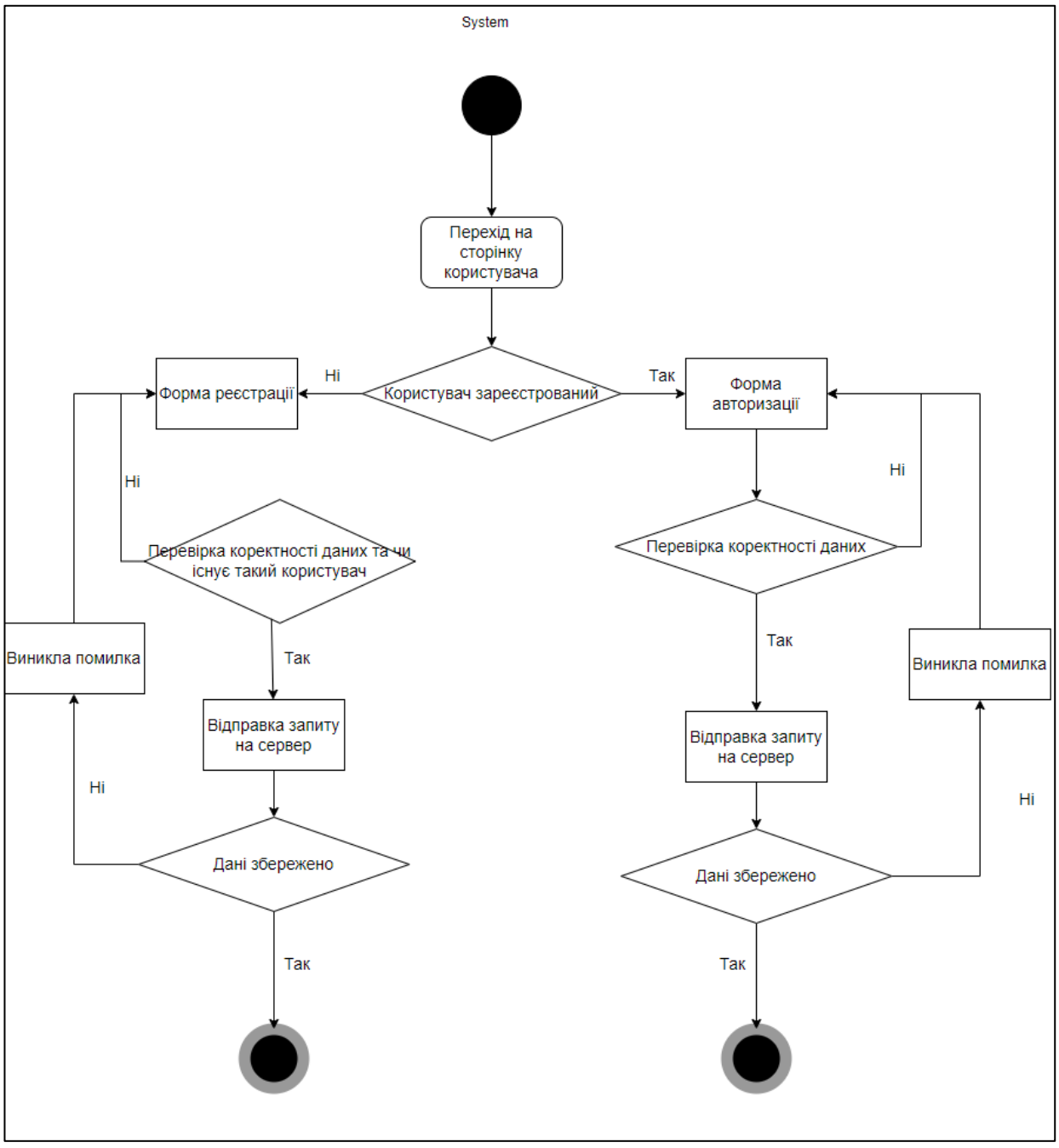



Рисунок 3.2 – UML діаграма активності додавання даних

На рисунках 3.3, 3.4 зображено діаграми активності системи керування іншими процесами, що пов’зані з обробкою даних системи.

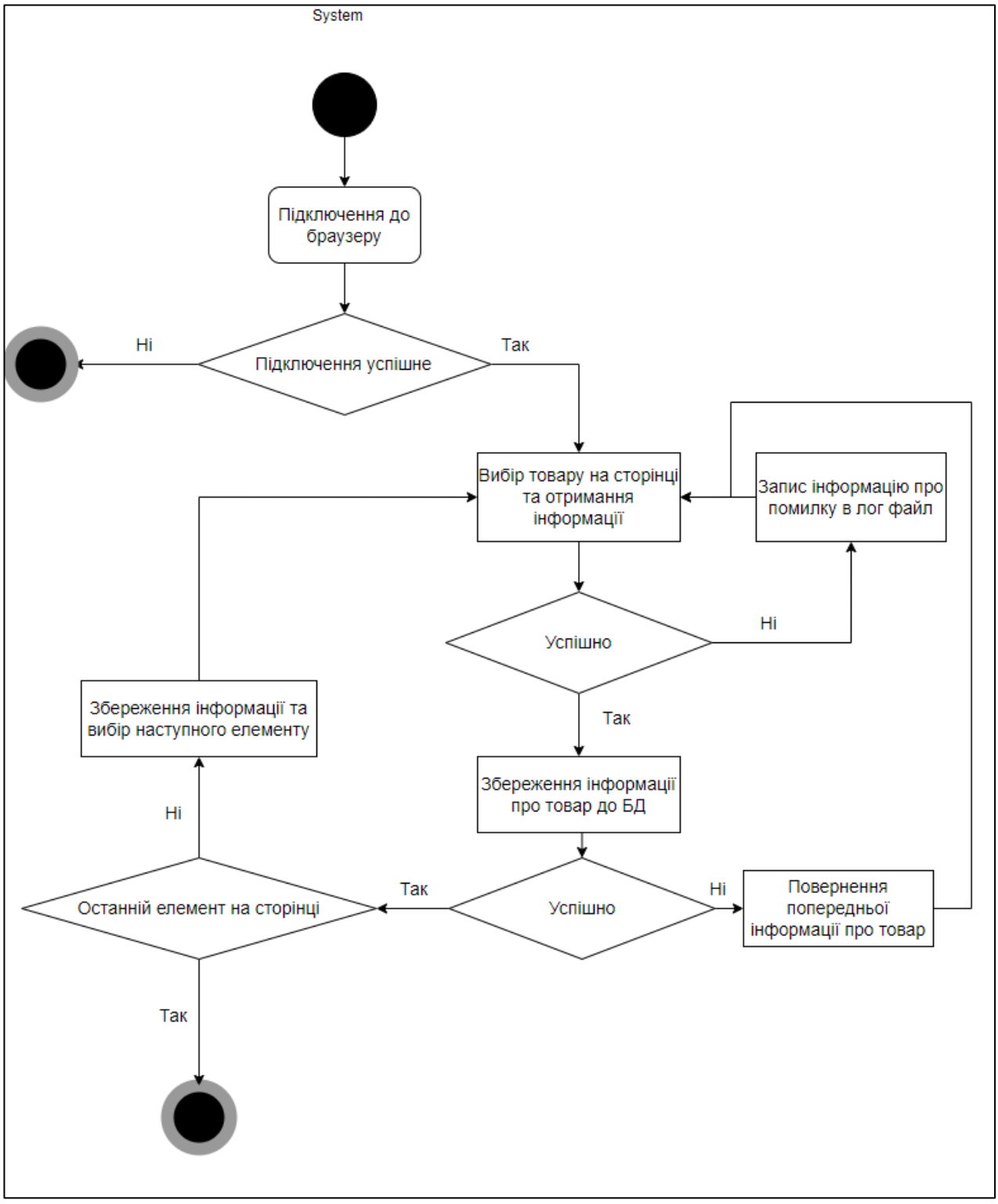



Рисунок 3.3 – UML діаграма активності роботи скрапера

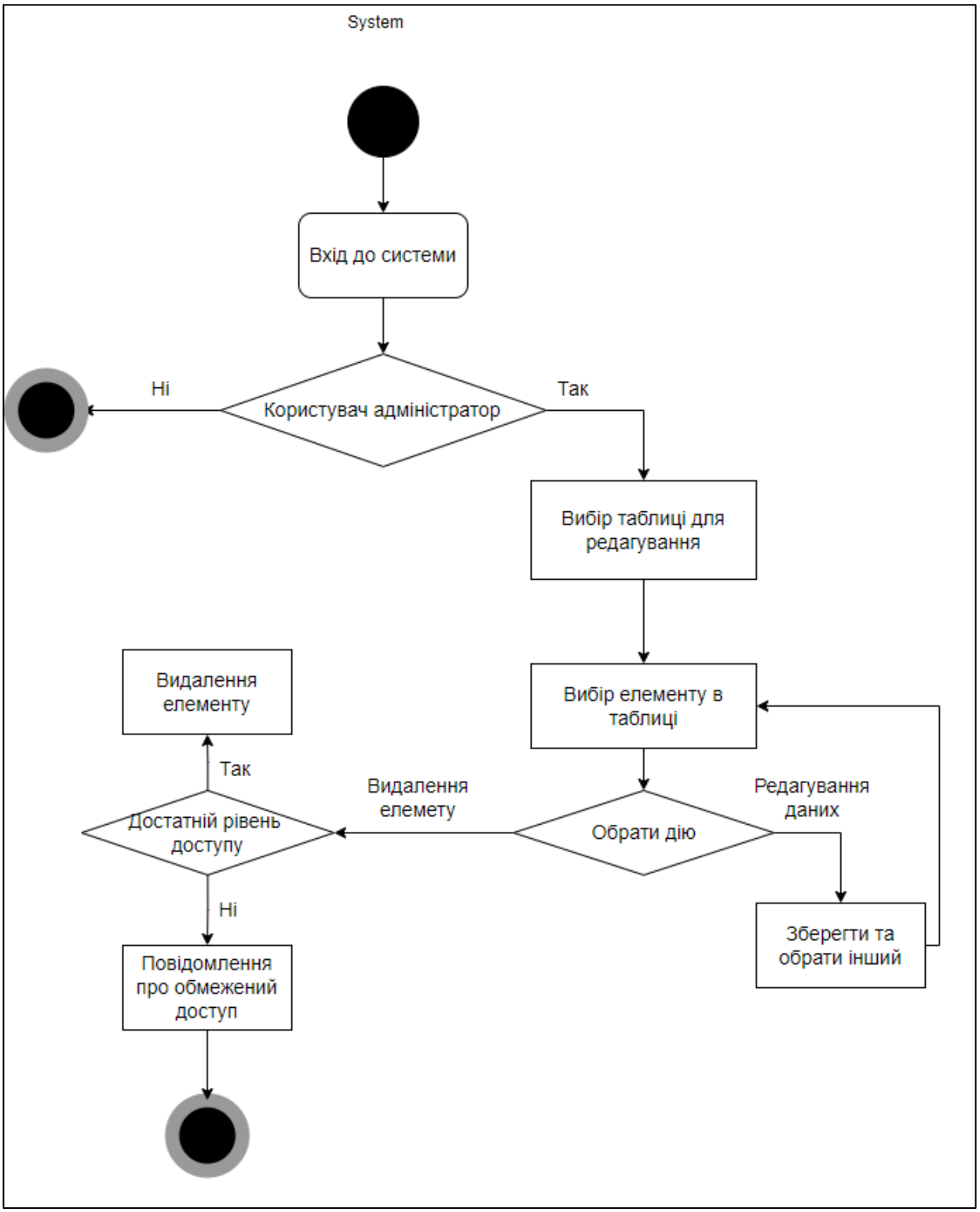



Рисунок 3.4 – UML діаграма активності редагування даних

## 3.2 Моделювання структури даних

На даному етапі розробки постає завдання у виборі інформаційних сутностей. Проаналізувавши потрібні запити, в базу даних потрібно включити

атрибути, які описують наступні сутності:

- категорії товарів;
- дописувачі (список товарів);
- інформація про товари.

Наступним кроком розробки є визначення атрибутів для опису вищеперечислених сутностей:

– категорії (id, назва категорії, slug (назва категорії для використання в URL), зображення);

- дописувачі (id, id категорії, виробник, модель, посилання на інтнет магазини, зображення товару);

- товар (id, id категорії, id дописувача, характеристики товару та посилання на товар в інтернет магазинах).

Представлення ER-моделі бази даних для збереження та обробки даних зображено на рисунку 3.5.

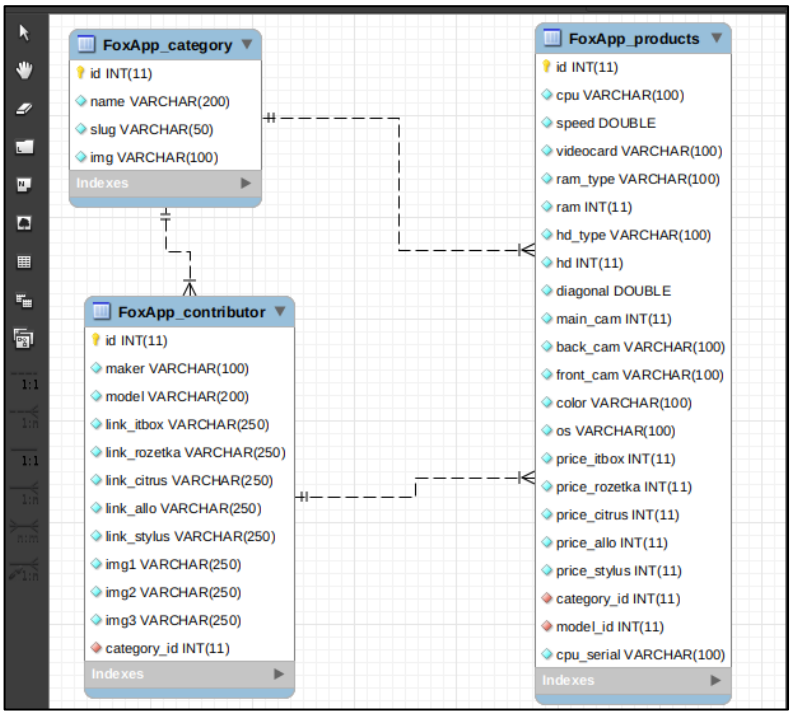



Рисунок 3.5 – ER-модель предметної області бази даних

В подальшому для зручності оперування базою даних за допомогою засобів Django було налаштовано адмін-панель, за допомогою якої користувач з

певними правами зможе оперувати базою даних напряму з веб-додатку.

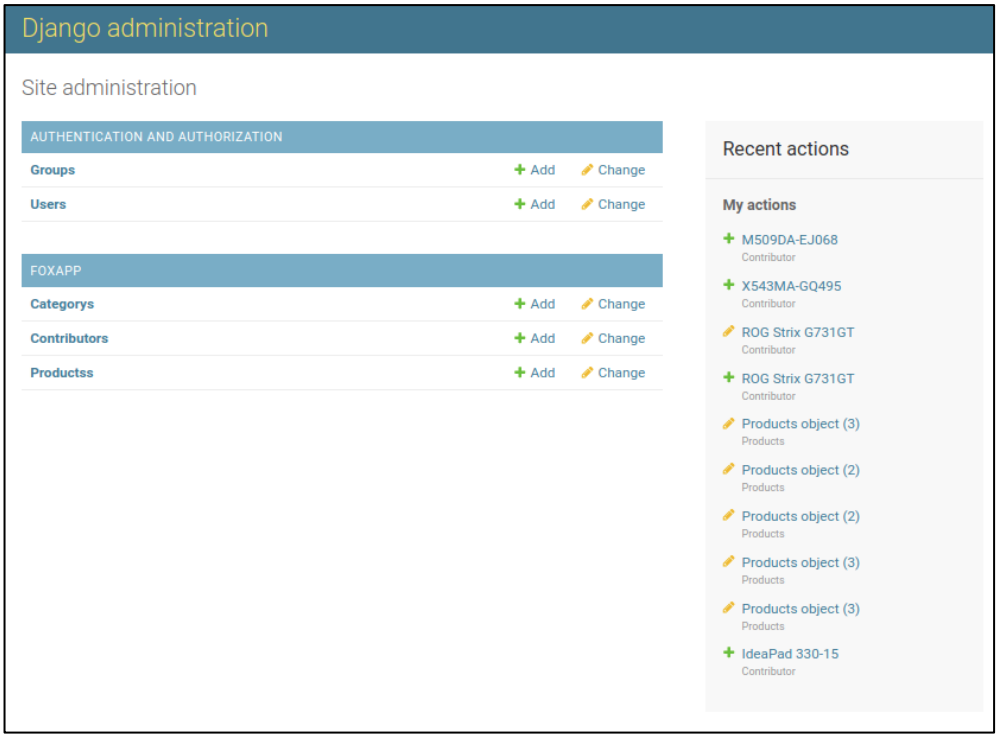



Рисунок 3.6 – Головне вікно адмін-панелі

На рисунку 3.6 представлена головна сторінка адмін-панелі, яка відображає бази даних які підключені до веб-додатку, а також таблиці які відносяться до потрібних БД, з цієї сторінки можна змінити налаштування потрібних таблиць, або ж перейти до потрібної таблиці для перегляду та редагування даних, що представлена на рисунку 3.7.

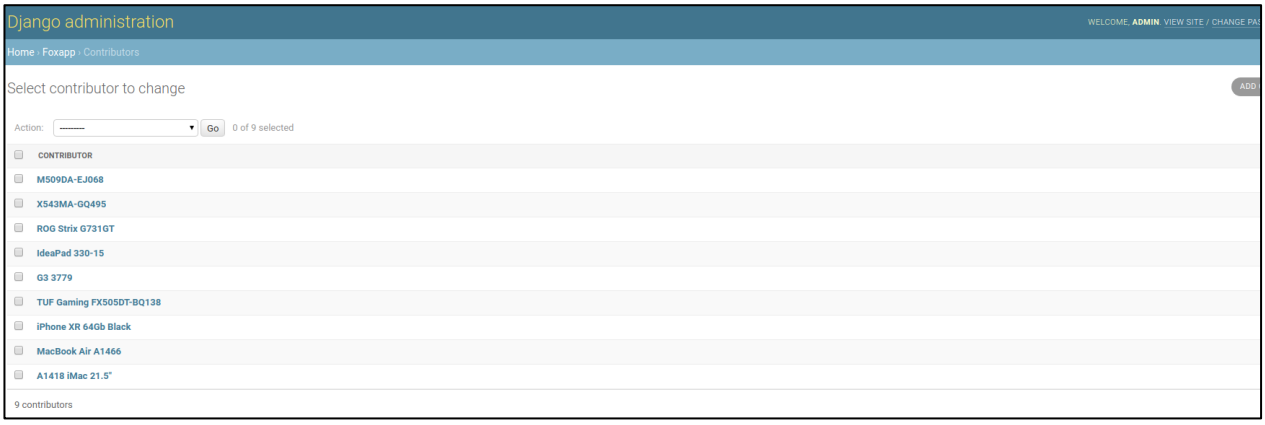



Рисунок 3.7 – Таблиця «Товари»

Перейшовши до потрібної траблиці з`являється можливість перегляду детальної інформації певного поля, видалення, редагування або додавання нової інформації. Детальніше представлено на рисунку 3.8.

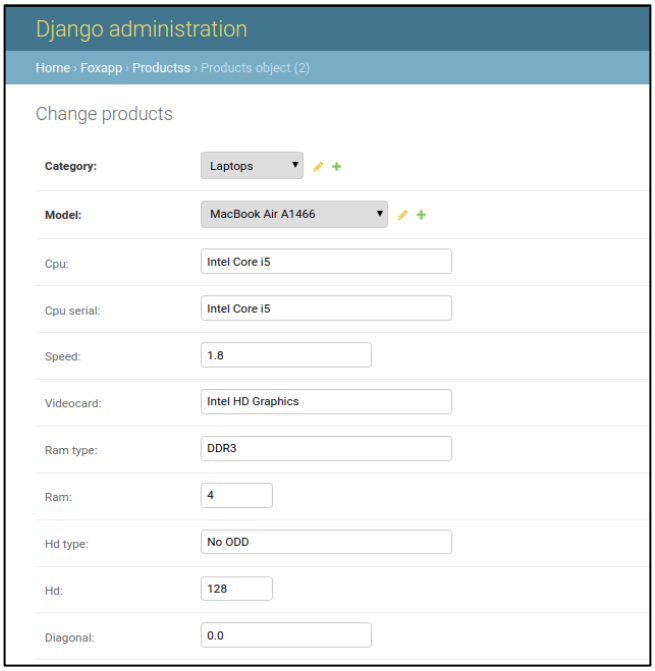



Рисунок 3.8 – Вікно редагування товару

Django має в собі певну структуру моделі користувачів «з коробки», але в нашому випадку вбудованого функціоналу недостатньо для повноцінної роботи з сайтом. Тому було прийняте рішення повністю перевизначити модель користувача, а також зв`язані з нею моделі (групи користувачів та доступ).

Структура | Данные | Ограничения | Индексы | Триггеры | DDL

mydatabase | Имя таблицы: users_customuser | WITHOUT ROWID

| | Имя | Тип данных | Первичный ключ | Внешний ключ | Уникальность | Проверка | Не NULL | Сравнение | Generated |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | integer | | | | | | | |
| 2 | password | varchar (128) | | | | | | | |
| 3 | last_login | datetime | | | | | | | |
| 4 | is_superuser | bool | | | | | | | |
| 5 | email | varchar (255) | | | | | | | |
| 6 | first_name | varchar (255) | | | | | | | |
| 7 | last_name | varchar (255) | | | | | | | |
| 8 | avatar | varchar (100) | | | | | | | |
| 9 | last_time_visit | datetime | | | | | | | |
| 10 | is_active | bool | | | | | | | |
| 11 | is_admin | bool | | | | | | | |
| 12 | login | varchar (100) | | | | | | | |
| 13 | is_staff | bool | | | | | | | |
| 14 | order | varchar (1000) | | | | | | | |
| 15 | phone_number | integer | | | | | | | |
| 16 | saved_products | varchar (100) | | | | | | | |

Рисунок 3.9 – Структура моделі користувача

Розширивши модель користувача у нас з`явилось ширші можливість налаштування користувачів-адміністраторів. За допомогою полів is_superuser, is_admin та is_staff в нас є можливість тонкого налаштування доступу до того чи іншого функціоналу веб-додатку. Для прикладу якщо «суперкористувач» має доступ до налаштування будь-якої частини сайту, та видимість всіх елементів налаштування, то користувач рівень доступу якого «персонал» має доступ лише до редагування деяких моделей, а також для кожного користувача з таким рівнем можна налаштовувати окремо рівень доступу до тієї чи іншої функціональності.

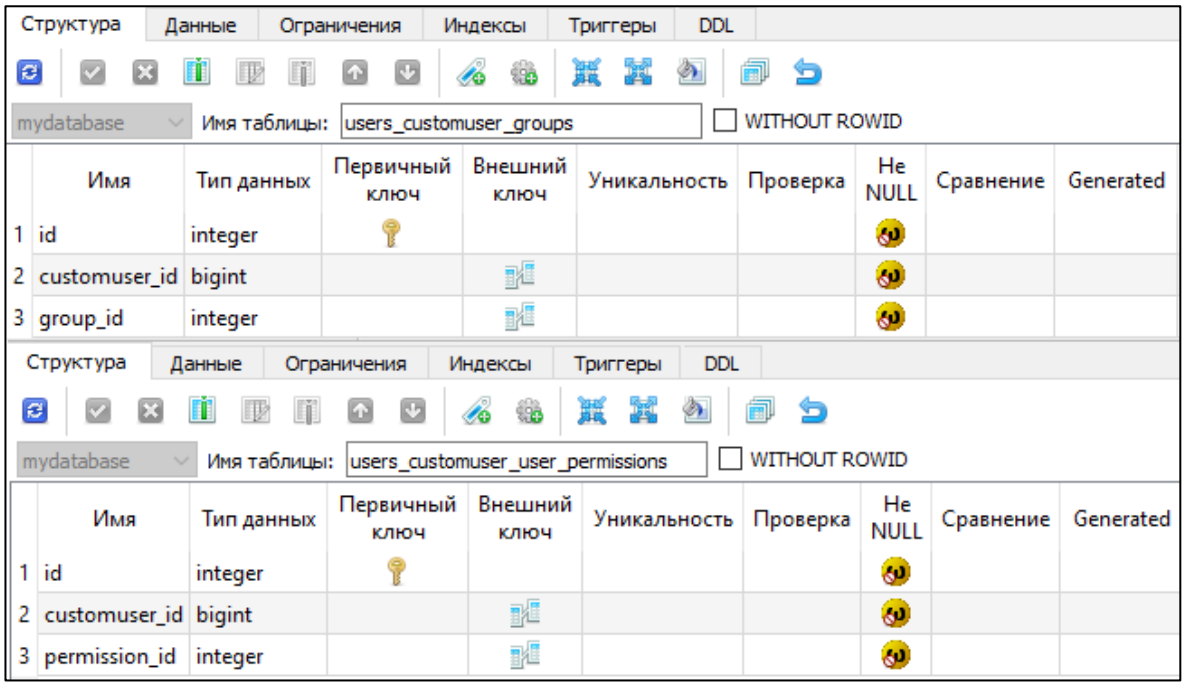



Рисунок 3.10 – Структура моделей доступу користувачів

Також в новій моделі додані поля для збору більш детальної інформації користувача такої як – ім`я користувача, прізвище, пошта та номер телефону. А для розширення функціоналу сайту поля – order та saved_products за допомогою яких користувач зможе додавати товари в збережені, або для порівняння характеристик між собою.

## 3.3 Розробка інтерфейсу користувача

Далі чи не найважливіша частина веб-додатку для користувача – інтерфейс. Інтерфейс веб-додатку буде складатись з 3 сторінок (головна сторінка, каталог товарів та сторінка перегляду товарів) і в подальшому за допомогою методів Django буде генеруватись контент відображуваний на представлених сторінках. Важливою частиною інтерфейсу сайту є “шапка” та “підвал” код створення цих елементів відображено на рисунках 3.10-3.11.

```
<div class="collapse navbar-collapse" id="navbarSupportedContent">
  <ul class="navbar-nav mr-auto">
    <li class="nav-item">
      <a class="nav-link" href="{% url 'index' %}">Home</a>
    </li>
    <li class="nav-item dropdown">
      <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="
        Categories
      </a>
      <div class="dropdown-menu" aria-labelledby="navbarDropdown">
          {% for all_category in All_categories %}
        <a class="dropdown-item" href="{{ all_category.get_absolute_url }}">{{ all_category.name }}</a>
          <div class="dropdown-divider"></div>
          {% endfor %}
      </div>
    </li>
  </ul>
  <form class="form-inline my-2 my-lg-0" action="{% url 'search' %}" method="get">
    <input class="form-control mr-sm-2" type="search" placeholder="search" name="search" aria-label="search">
    <button class="btn btn-outline-success my-2 my-sm-0 margin-right-25" type="submit">Search <span class="fas fa-search"></span></button>
  </form>

      <ul class="nav navbar-nav" >
          <li><a href="#"><span class="fas fa-bookmark margin-right-25 " style="..."></span></a></li>
          <li><a href="#"><span class="fas fa-shopping-cart margin-right-25" style="..."></span></a></li>
          {% if request.user.is_authenticated %}
          <li><a href="{% url 'user' %}"> <span class="fas fa-user-alt margin-right-25" style="font-size: 30px" ></span>{{ user.first_name }}</a
          {% else %}
          <li><a href="{% url 'user' %}"> <span class="fas fa-user-alt margin-right-25" style="font-size: 30px" ></span></a></li></a>
          {% endif %}
      </ul>
</div>
```

Рисунок 3.10 – Код «шапки» сторінки

```
    <div class="footer">
        <p> by: @gRow09 <br> Contact Information:</p>
        <a href="https://www.instagram.com/your_homie09/"><button class="socbutton">
            <span class="fab fa-instagram fa-2x"></span>
            </button></a>
        <a href="https://www.facebook.com/profile.php?id=100006172983195"><button class="socbutton">
            <span class="fab fa-facebook fa-2x"></span>
            </button></a>
        <a href="https://www.t.me/grow09"><button class="socbutton">
            <span class="fab fa-telegram-plane fa-2x"></span>
            </button></a>
        <a href="#"><button class="socbutton">
            <span class="fab fa-twitter fa-2x"></span>
            </button></a>
            <br><br>
    <p>© 2020 gRow09, all rights reserved. Made with <span class="far fa-heart"></span> for a better mark.</p>

    </div>
  <!-- Latest compiled and minified JavaScript -->
          <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin=
        <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+
        <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.3/Chart.js" ></script>
        <script src="{% static 'js/slider.js' %}"></script>
        <script src="{% static 'js/chart.js' %}"></script>
</body>
</html>
```

Рисунок 3.11 – Код «підвалу» сторінки

Також на головній сторінці відображено каталог категорій продуктів. Код відображений для створення однієї категорії, але залежно від кількості категорій в базі даних додатку буде генеруватись кількість категорій зоображених в інтерфейсі сторінки. Відповідний код представлений на рисунку 3.12.

```
{% block content %}
    {% load static %}
  <img src="{% static 'Resource/sozdanie-internet-magazina.png' %}" height="15%" width="35%" style="...">
    <div class="content" style="margin-top: 50px">

        <div class="card-columns" style="margin-left: 10%;">
            <div class="col-sm-10">
                {% for all_category in All_categories %}
      <a href="{{ all_category.get_absolute_url }}">
          <div class="card border-success  card-big">
  <div class="card-header border-success">{{ all_category.name }}</div>
  <div class="card-body text-success">
    <img src="{{ all_category.img }}" class="card-img-top" style="margin-top: 5px;" alt="...">
  </div>
</div>
          </a>
                {% endfor %}
          </div>
```

Рисунок 3.12 – Код каталогу категорій сторінки

Результат готової сторінки зображено на рисунку 3.13.

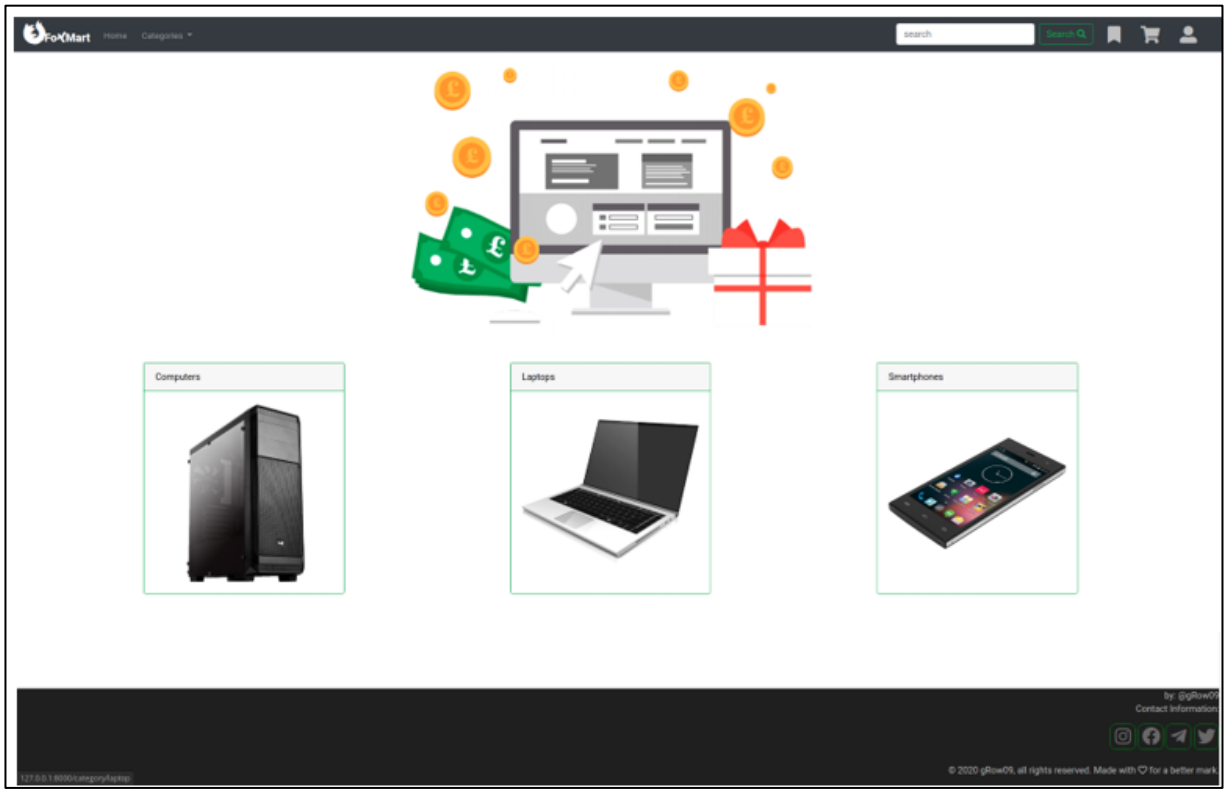



Рисунок 3.13 – Результат готової сторінки

Далі обравши категорію користувач потрапляє на сторінку каталогу з товарами які відносяться до цієї категорії. Дана сторінка складається з колонки фільтрів, які генеруються автоматично виходячи з параметрів товарів на сторінці та самого каталогу товарів (рис. 3.14).

```
<form action="{% url 'filtered' %}" method="get" name="filter">
    <div class="filters">
   <div class="form-group">
  <label for="sel1">Manufacturer:</label>
  <select class="form-control" name="maker" id="sel1">
      <option>All</option>
      {% for makers in Makers%}
    <option name="maker" value="{{ makers }}">{{ makers }}</option>
      {% endfor %}
  </select>
       <hr>
       <div class="panel-group" id="accordion">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">
        <a data-toggle="collapse" data-parent="#accordion" href="#collapse1" style="color: black;">Processor:</a>
      </h4>
    </div>
       {% for cpu_serial in Cpu_serial %}
    <div id="collapse1" class="panel-collapse collapse in">
      <ul class="list-group">
        <div class="form-check">
  <label class="form-check-label">
    <input type="checkbox" class="form-check-input" name="cpu" value="{{ cpu_serial }}">{{ cpu_serial }}
  </label>
</div>
      </ul>
    </div>

              <div id="collapse3" class="panel-collapse collapse in">
                 {% for diagonal in Diagonal %}
      <ul class="list-group">
        <div class="form-check">
  <label class="form-check-label">
     <input type="checkbox" class="form-check-input" name="diagonal" value="{{ diagonal }}"><p>{{ diagonal }}</p>
  </label>
</div>
       </ul>
      {% endfor %}
   </div>
  </div>
{% endif %}
<input type="hidden" name="category" value="{{ category.pk }}">
          <hr>
          <button type="submit" class="btn btn-success">Search <span class="fas fa-search"></span></button>
      </div>
      </div>
       </div>
{% endfor %}
  </form>
```

Рисунок 3.14 – Частина коду, який відповідає за створення форми фільтрації

Представлений код відповідає за генерацію фільтрації товарів для звуження каталогу товарів по потрібним користувачу параметрам. Параметри фільтрації генеруються автоматично виходячи з важливих параметрів товару в представленій категорії (рис. 3.15).

```
    <div class="catalogue">
        {% for products in Products %}
        {% for category in Categories %}
        {% if products.category.slug  == category.slug %}
    <div class="product" style="margin-right: 30px;">
        {% for contributor in Contributor %}
        {% if contributor.pk == products.pk %}
        <div class="imgbox">
        <img src="{{ contributor.img1 }}">
        </div>
        <div class="specifice">
        <h2>{{ contributor.maker }}<br><span>{{ products.model }}</span></h2>
        <div class="price">From {{ products.minimal }}UAH<br>To {{ products.maximal }}UAH</div>
         <hr>
        <label>CPU: <p>{{ products.cpu }}</p></label>
        <label>RAM: <p>{{ products.ram }} GB</p></label>
        {% if products.videocard != "0"  %}
        <label>Videocard: <p>{{ products.videocard }}</p></label>
        {% else %}
        <label>Diagonal: <p>{{ products.diagonal }} Inch</p></label>
        {% endif %}
        <a href="{{ products.get_absolute_url }}">Buy it now</a>
        </div>
    </div>
        {% endif %}
        {% endfor %}
        {% endif %}
        {% endfor %}
        {% endfor %}
    </div>
    </div>
{% endblock %}
<br><br><br>
{% endblock %}
```

Рисунок 3.15 – Частина коду, який відповідає за генерацію товарів на сторінці

Результат готової сторінки каталогу товарів представлено на рисунку 3.16.

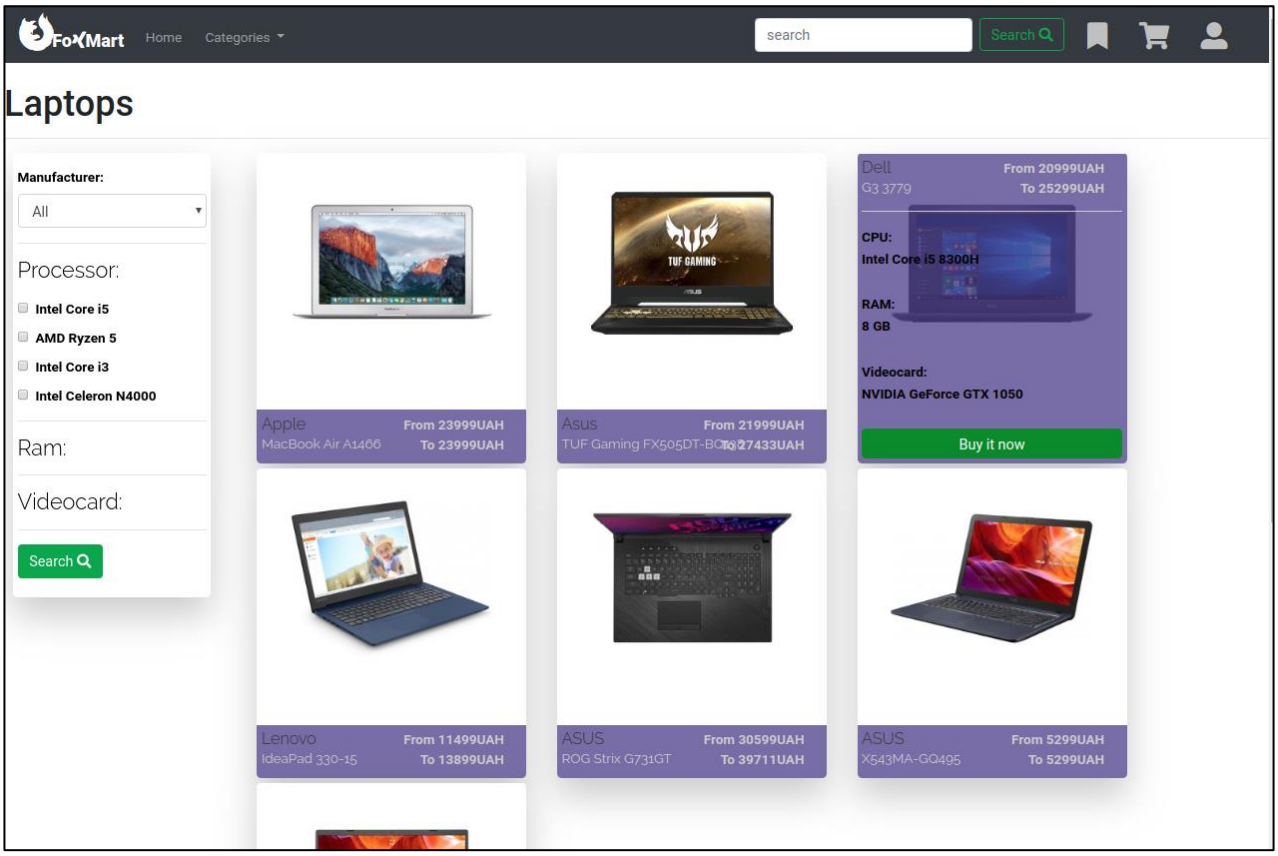
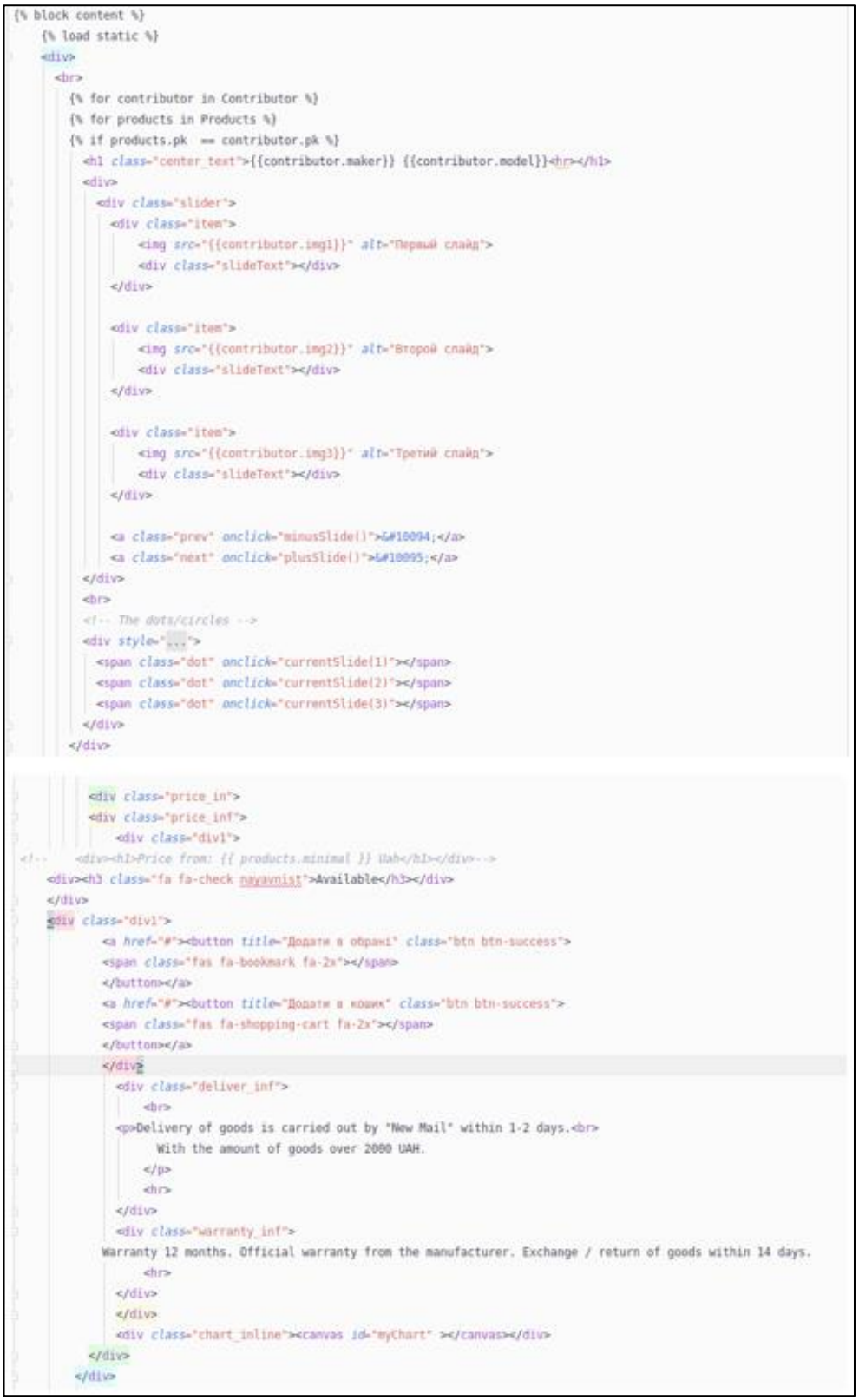



Рисунок 3.16 – Результат згенерованої сторінки з товарами

Обравши потрібний товар користувач потрабляє на сторінку з більш детальною інформацією про товар. Сторінка складається з чотирьох елементів, фото та короткої інформації про замовлення, графіку для оцінки пропозицій на ринку, таблиці порівняння інтернет-магазинів, та детальних характеристик товару. Код даної сторінки представлено на рисунку 3.17.



```
{% block content %}
    {% load static %}
    <div>
      <br>
        {% for contributor in Contributor %}
        {% for products in Products %}
        {% if products.pk  == contributor.pk %}
          <h1 class="center_text">{{contributor.maker}} {{contributor.model}}<hr></h1>
          <div>
            <div class="slider">
              <div class="item">
                  <img src="{{contributor.img1}}" alt="Перший слайд">
                  <div class="slideText"></div>
              </div>

              <div class="item">
                  <img src="{{contributor.img2}}" alt="Второй слайд">
                  <div class="slideText"></div>
              </div>

              <div class="item">
                  <img src="{{contributor.img3}}" alt="Третий слайд">
                  <div class="slideText"></div>
              </div>

              <a class="prev" onclick="minusSlide()">&#10094;</a>
              <a class="next" onclick="plusSlide()">&#10095;</a>
            </div>
            <br>
            <!-- The dots/circles -->
            <div style="...">
              <span class="dot" onclick="currentSlide(1)"></span>
              <span class="dot" onclick="currentSlide(2)"></span>
              <span class="dot" onclick="currentSlide(3)"></span>
            </div>
          </div>

              <div class="price_in">
              <div class="price_inf">
                  <div class="div1">
<!--    <div><h1>Price from: {{ products.minimal }} Uah</h1></div>-->
    <div><h3 class="fa fa-check nayavnist">Available</h3></div>
    </div>
    <div class="div1">
          <a href="#"><button title="Додати в обрані" class="btn btn-success">
          <span class="fas fa-bookmark fa-2x"></span>
          </button></a>
          <a href="#"><button title="Додати в кошик" class="btn btn-success">
          <span class="fas fa-shopping-cart fa-2x"></span>
          </button></a>
          </div>
            <div class="deliver_inf">
                <br>
            <p>Delivery of goods is carried out by "New Mail" within 1-2 days.<br>
                With the amount of goods over 2000 UAH.
                </p>
                <hr>
            </div>
            <div class="warranty_inf">
          Warranty 12 months. Official warranty from the manufacturer. Exchange / return of goods within 14 days.
                <hr>
            </div>
            </div>
            <div class="chart_inline"><canvas id="myChart" ></canvas></div>
          </div>
        </div>
```

Рисунок 3.17 – Код частини сторінки з короткою інформацією та фото товару

Представлена частина коду відповідає за головну сторінку товару на якій представлено інформацію про товар та порівняння цін в різним магазинах.

На даній сторінці є частина з «каруселлю» фото з циклічно відображеними зображеннями товару. Зображення товару отримуються з бази даних підключеної до веб-додатку та виводяться циклічно на детальній сторінці. Код генерації «каруселі» відображено на рисунку 3.18.

```
var slideIndex = 1;
showSlides(slideIndex);

function plusSlide() {
    showSlides(slideIndex += 1);
}

function minusSlide() {
    showSlides(slideIndex -= 1);
}

function currentSlide(n) {
    showSlides(slideIndex = n);
}

function showSlides(n) {
    var i;
    var slides = document.getElementsByClassName("item");
    var dots = document.getElementsByClassName("slider-dots_item");
    if (n > slides.length) {
      slideIndex = 1
    }
    if (n < 1) {
        slideIndex = slides.length
    }
    for (i = 0; i < slides.length; i++) {
        slides[i].style.display = "none";
    }
    for (i = 0; i < dots.length; i++) {
        dots[i].className = dots[i].className.replace(" active", "");
    }
    slides[slideIndex - 1].style.display = "block";
    dots[slideIndex - 1].className += " active";
}

function showDiv2(){
    var x = document.getElementById("div2");
    var y = document.getElementById("div3");
    x.style.display = "block";
    y.style.display = "none";
}

function showDiv3(){
    var x = document.getElementById("div2");
    var y = document.getElementById("div3");
    x.style.display = "none";
    y.style.display = "block";
}
```

Рисунок 3.18 – Скрипт який відповідає виведення зображень товару

Ціни отримані з різнимх магазинів вносяться до бази данних, в подальшому виводяться на сторінці певного товару, а також за допомогою скрипта на рисунку 3.19 будується графік цін.

```
onload = function()
{
    var ItBox  = document.getElementById("itbox").innerText
    var Rozetka = document.getElementById("rozetka").innerText
    var Citrus = document.getElementById("citrus").innerText
    var Allo = document.getElementById("allo").innerText
    var Stylus = document.getElementById("stylus").innerText



    var ctx = document.getElementById('myChart').getContext('2d');
    var chart = new Chart(ctx, {
        // The type of chart we want to create
        type: 'line',

        // The data for our dataset
        data: {
            labels: ['ItBox', 'Rozetka', 'Citrus', 'Allo', 'Stylus'],
            datasets: [{
                label: 'Price comparing',
                backgroundColor: 'rgb(77, 30, 84)',
                borderColor: 'rgb(30, 173, 20)',
                data: [ItBox, Rozetka, Citrus, Allo, Stylus]
            }]
        },
        // Configuration options go here
        options: {}
    });
}
```

Рисунок 3.19 – Скрипт який відповідає за побудову графіку

Також окрім графіку порівняння на сторінці представлена таблиця з магазинами продавцями даного товару та цінами для порівняння та вибору потрібного магазину. Частина коду даної таблиці представлений на рисунку 3.20.

```
<div class="div2" id="div2" style="...">
  <table class="table">
    <thead>
      <tr>
        <th scope="col">Seller:</th>
        <th scope="col">Model:</th>
        <th scope="col">Count of reviews:</th>
          <th scope="col">Price:</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td><img src="https://seeklogo.com/images/I/itbox-logo-F3D2D6730E-seeklogo.com.png" style="width: 120px;"></td>
        <td>{{contributor.maker}} {{contributor.model}}</td>
          <td>{{products.itbox_com_count}}</td>
        <td><span id="itbox">{{products.price_itbox}}</span> грн <a href="{{ contributor.link_itbox }}"><button class="b
      </tr>
      <tr>
        <td><img src="https://upload.wikimedia.org/wikipedia/commons/b/be/ROZETKA-Logo-L3-B-RGB.png" style="..."></td>
        <td>{{contributor.maker}} {{contributor.model}}</td>
          <td>{{products.rozetka_com_count}}</td>
        <td><span id="rozetka">{{products.price_rozetka}}</span> грн <a href="{{ contributor.link_rozetka }}"><button cl
      </tr>
      <tr>
        <td><img src="https://i.citrus.ua/pictures/all/svg/citrus_brand_new_logo.svg" style="..."></td>
        <td>{{contributor.maker}} {{contributor.model}}</td>
          <td>{{products.citrus_com_count}}</td>
```

Рисунок 3.20 – Код частини сторінки з порівнянням пропозицій на ринку

Описана вище таблиця складається з двох частин, перейшовши на іншу можна переглянути детальні характеристики певного товару, код для цієї частини представлено на рисунку 3.21.

```
  <thead>
    <tr>
      <th scope="col">Модель:</th>
      <th scope="col">{{products.category}}</th>
    </tr>
  </thead>
  {% if products.ram_type != "0"  %}
  <tbody>
    <tr>
      <td>Cpu</td>
      <td>{{products.cpu}} {{products.speed}}GHz</td>
    </tr>
    <tr>
      <td>Videocard</td>
      <td>{{products.videocard}}</td>
    </tr>
    <tr>
      <td>Ram type</td>
      <td>{{products.ram_type}}</td>
    </tr>
  <tr>
      <td>Ram </td>
      <td>{{products.ram}} GB</td>
    </tr>
  <tr>
      <td>Drive type</td>
      <td>{{products.hd_type}}</td>
    </tr>
  <tr>
      <td>Drive memory</td>
      <td>{{products.hd}} GB</td>
    </tr>
  </tbody>
</table>
  {% else %}
                <tbody>
    <tr>
      <td>Cpu</td>
```

Рисунок 3.21 – Код частини сторінки з характеристиками товару

На рисунку 3.22 зображено фронтенд детальної сторінки товару. Першим представлено основна інформація про товар та порівняння цін на товари, а також в нижній частині дві вкладки через яку можна перейти до детальних характеристик товару.

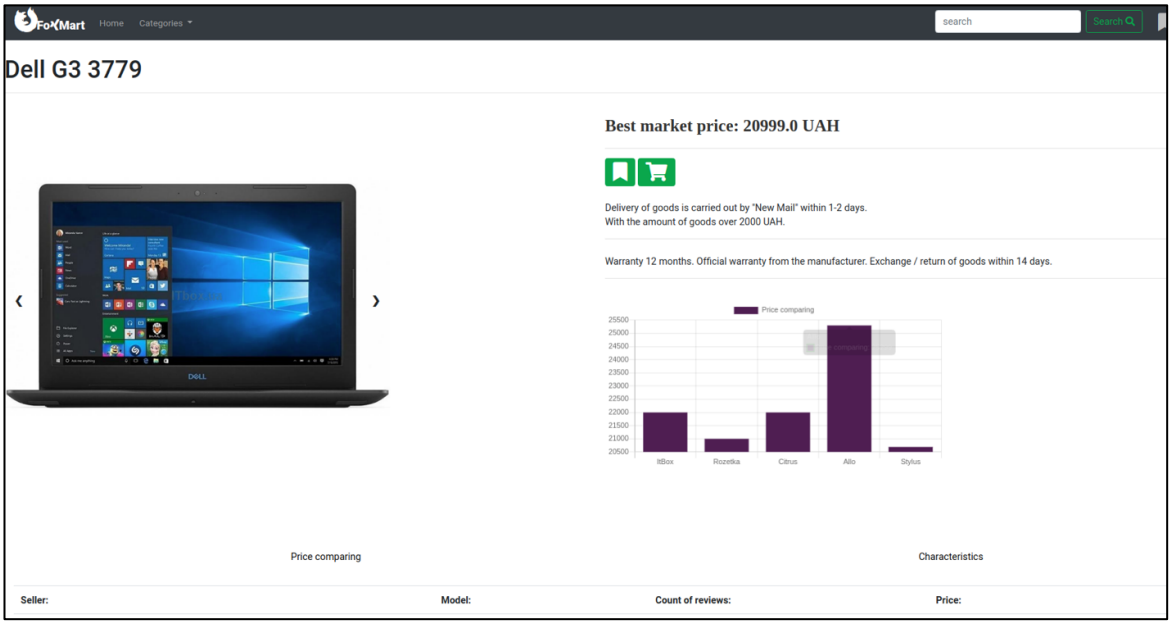



Рисунок 3.22 – Частина сторінки з фото товару та графіком порівняння цін

На рисунку 3.23 представлена таблиця порівняння цін в різних інтернет-магазинах.

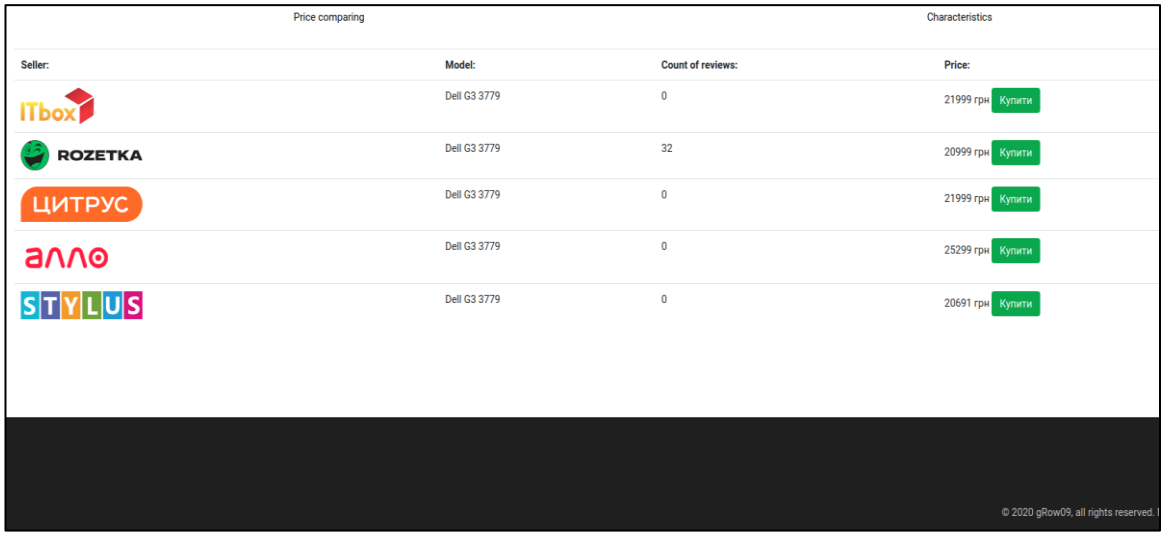



Рисунок 3.23 – Частина сторінки з таблицею порівняння цін

На рисунку 3.24 зображено другу частину таблиці з детальними характеристиками товару.

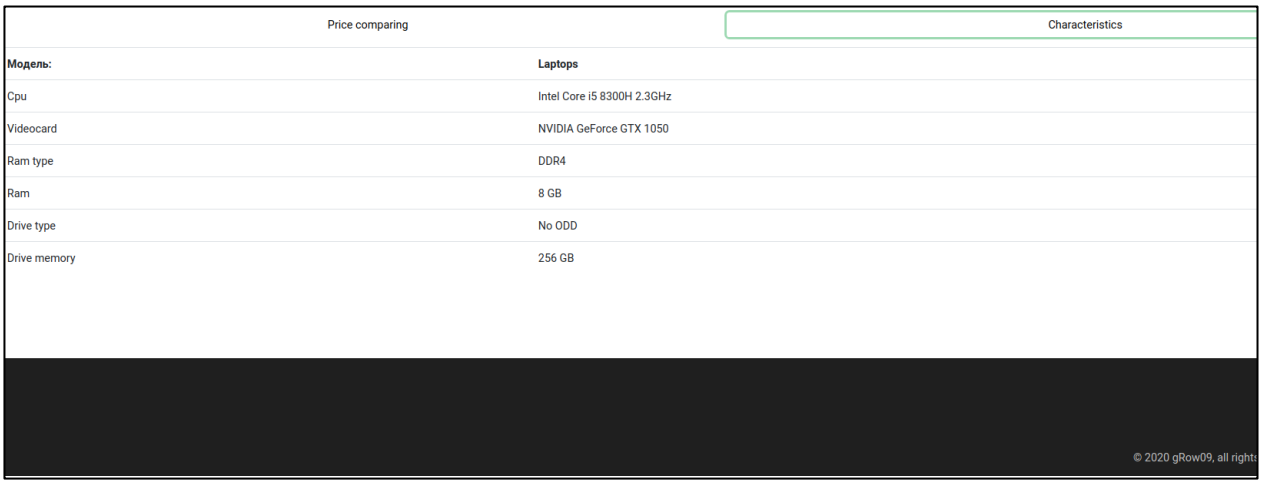



Рисунок 3.24 – Частина сторінки з характеристиками товару

Для зручності користуванням сайту та для входу адміністрації сайту не використовуючи адмін панель створені сторінки реєстрації та авторизації. На рисунку 3.25 код сторінки реєстрації користувача.

```
<form action="{% url 'registering' %}" method="post">
  <div class="form-group">
    {% csrf_token %}
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter email" 
    <small id="emailHelp" class="form-text text-muted">We'll never share your email with anyone else.</small>
  </div>
  <div class="form-group">
    <label for="exampleInputPassword">Name</label>
    <input class="form-control" id="firstname" placeholder="Name" name = "firstname">
  </div>
  <div class="form-group">
    <label for="exampleInputPassword">Surname</label>
    <input class="form-control" id="lastname" placeholder="Surname" name = "lastname">
  </div>
  <div class="form-group">
    <label for="exampleInputPassword">Phone number</label>
    <input class="form-control" id="phonenumber" placeholder="Your phone number" name = "phonenumber">
  </div>
  <div class="form-group">
    <label for="exampleInputPassword">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword" placeholder="Password" name = "password">
  </div>
  <div class="form-group">
      <label for="exampleRepeatPassword">Repeat your password</label>
      <input type="password" class="form-control" id="exampleRepeatPassword" placeholder="Password" name = "repeat">
    </div>
  <button type="submit" class="btn btn-primary">Register</button>
</form>
```

Рисунок 3.25 – Частина сторінки з характеристиками товару

Для безпеки відвідувачів сайту та для запобігання перехвату даних користувача використано «csrf_token». CSRF токен — це унікальне, секретне, непередбачуване значення, яке генерується програмою на стороні сервера і передається клієнту таким чином, що воно включається в наступний запит HTTP, зроблений клієнтом. Тобто якщо зловмисник спробує перехватити HTTP запит то замість даних які були введені на сторінці реєстрації будуть видані лише зашифровані хеш коди.

Далі на рисунку 3.26 представлена фронтенд реалізація сторінки реєстрації користувача.

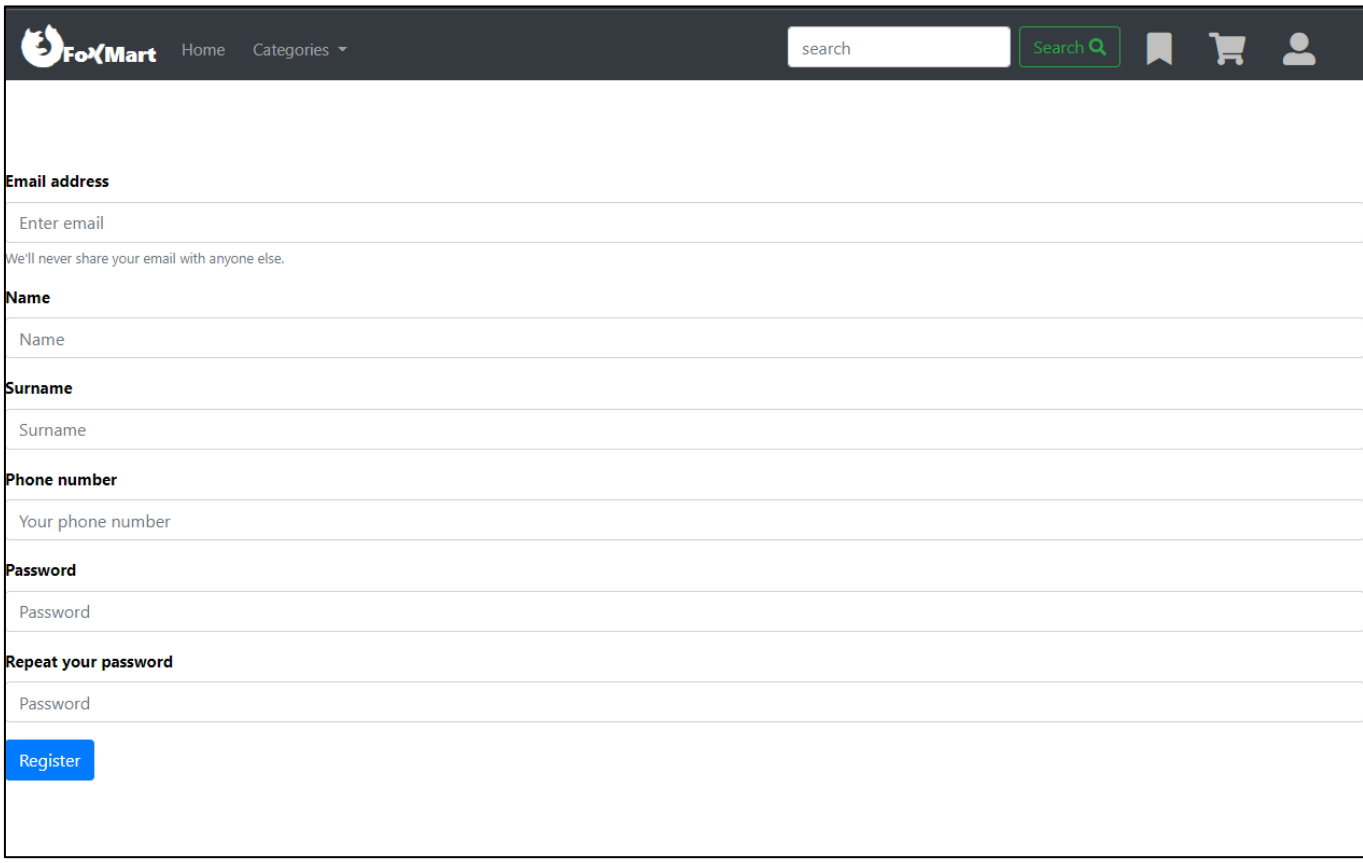



Рисунок 3.26 – Частина сторінки з характеристиками товару

Для зручного порівнювання різних електронних товарів для зареєстрованих користувачів розроблено функціонал порівняння товарів представлений на рисунках 3.27-3.28.

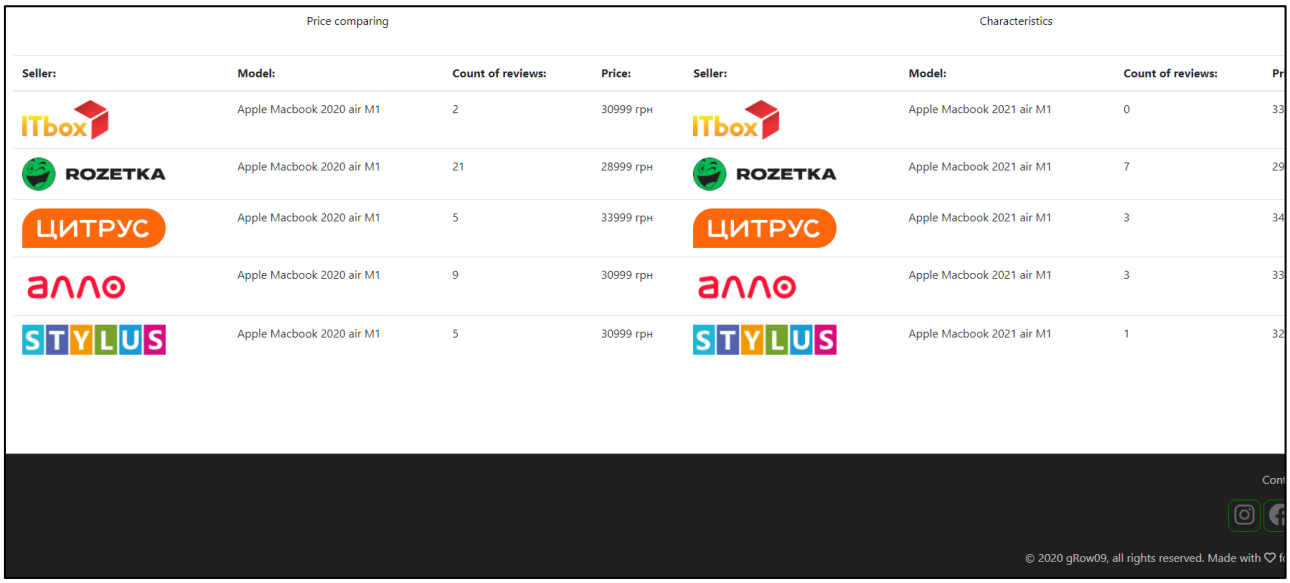



Рисунок 3.27 – Частина сторінки порівняння цін на різні товари

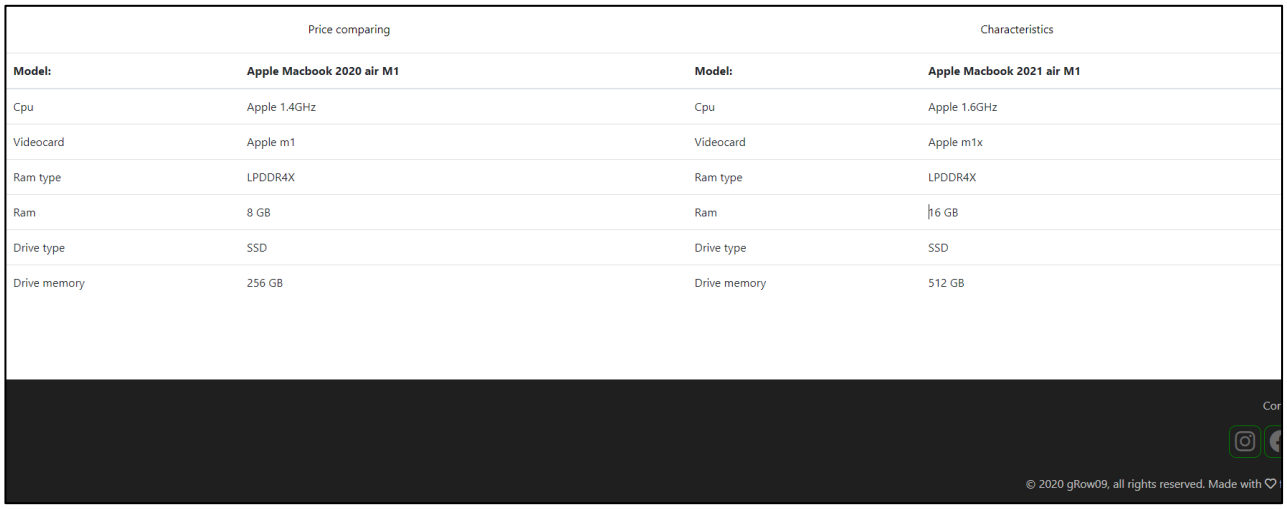



Рисунок 3.28 – Частина сторінки порівняння характеристик на різні товари

Розробивши інтерфейс користувача потрібно опрацювати програмно апаратну частину додатку, яка буде відповідати за обробку даних з бази даних та представлення її в інтерфейсі користувача.

Основна структура проекту складається з 2 додатків – перший це основний додаток який відповідає майже за весь функціонал сайту, інший це додаток для розширення моделі користувача та відповідає за весь функціонал сайту пов`язаний з користувачем.

Найбільш важлива частина додатку Users це файл models.py в якому визначено вигляд моделі користувачів в базі даних, та функціонал реєстрації і

надання прав доступу до функціоналу сайту певному користувачу. Код створення моделі користувача представлено на рисунку 3.29

```
class CustomUser(AbstractBaseUser, PermissionsMixin):
    email = models.EmailField(max_length=255, unique=True)
    first_name = models.CharField('Surname', max_length=255, blank=True, null=True)
    last_name = models.CharField('Name', max_length=255, blank=True, null=True)
    avatar = models.ImageField(null=True, blank=True, upload_to='avatars')
    last_time_visit = models.DateTimeField(default=timezone.now)
    is_active = models.BooleanField(default=True)
    is_admin = models.BooleanField(default=False)
    is_staff = models.BooleanField(default=False)
    login = models.CharField(verbose_name='Login', max_length=100)
    phone_number = models.IntegerField(max_length=10, verbose_name='Phone', blank=True, null
    saved_products = models.CharField(verbose_name='Saved', max_length=100, blank=True, null
    order = models.CharField(verbose_name='Order', max_length=1000, blank=True, null=True)

    objects = UserManager()

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = []

    def __str__(self):
        return self.email

    class Meta:
        verbose_name = 'User'
        verbose_name_plural = 'Users'
```

Рисунок 3.29 – Створення моделі CustomUser.

Функціонал реєстрації користувачів задано в класі UserManager його вигляд представлено на рисунку 3.30.

```python
class UserManager(BaseUserManager):

    def create_user(self, email, password=None,**kwargs):
        if not email:
            raise ValueError('Users must have an Email')

        user = self.model(
            email=email,**kwargs)

        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, email, password):
        """
        Creates and saves a superuser with the given email and password.
        """
        user = self.create_user(
            email,
            password=password
        )
        user.is_admin = True
        user.is_staff = True
        # user.is_superuser = True
        user.save(using=self._db)
        return user
```

Рисунок 3.30 – Створення моделі CustomUser.

Далі перейдемо до головного додатку де описано весь функціонал веб-сайту.

Основна структура проекту має такий вигляд (рис 3.31):

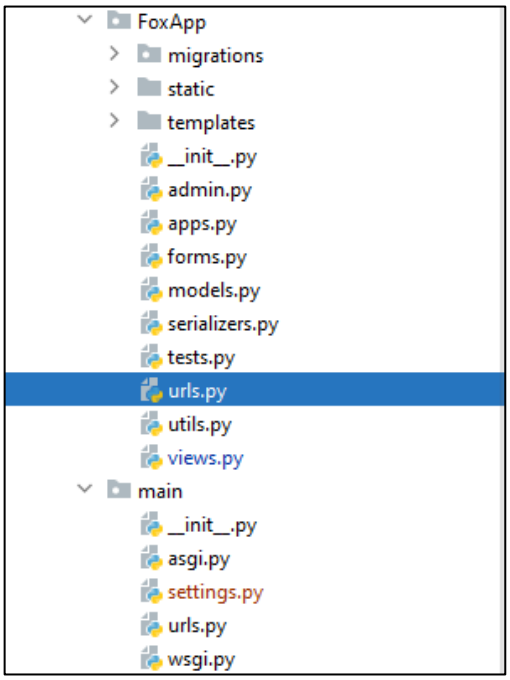



Рисунок 3.31 – Структура проекту

Папка migrations відповідає за всі міграції які проводились під час проектування та забезпечує можливість перенестись на більш стару гілку розробки. Папка static містить всі статичні файли використані на веб-сайті, це можуть бути як зображення, так css або js файли використані на фронтенд частині сайту. В папці templates зберігаються всі html шаблони сторінок (рис.3.32)

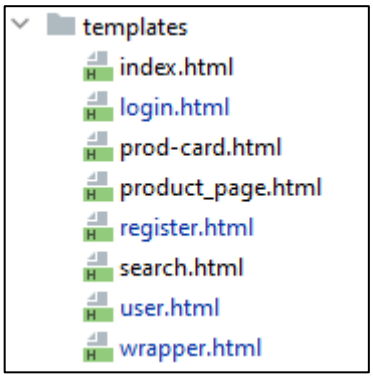



Рисунок 3.32 – Шаблони сторінок веб-додатку

Таким чином поглянувши на шаблони зрозуміла структура сайту з клієнтської сторони.

admin.py – це файл який відповідає за адмін-панель веб-додатку, яка в основному використовується для роботи з базою даних проекту та користувачами (рис 3.33).

```
from django.contrib import admin

# Register your models here.
from .models import Category, Contributor, Products


admin.site.register(Category)
admin.site.register(Contributor)
admin.site.register(Products)
```

Рисунок 3.33 – Код файлу admin.py

urls.py – відповідає за налаштування прямих посилань на сторінки додатку, та налаштування зв'язку між сторінками сайту (рис 3.34).

```python
from django.urls import path
from . import views
from .views import prod_card, product_page, filtered, search, ProdView, ProdViewSimple, user, register, login,r
from .models import *

urlpatterns = [
    path('', views.index, name='index'),
    path('category/<category_slug>', prod_card, name='category'),
    path('search', search, name='search'),
    path('filtered/', filtered, name='filtered'),
    path('products/<product_slug>', product_page, name='products'),
    path('user/', user, name='user'),
    path('register/', register_page, name='register'),
    path('registering/', register, name='registering'),
    path('login/', login, name='login'),
    path('api/<int:pk>/', ProdView.as_view()), ## API table products
    path('api_v1/', ProdViewSimple.as_view()), ## Simple Api (products)
]
```

Рисунок 3.34 – Код файлу urls.py

settings.py – це центральний конфігураційний файл для всього проекту. Він містить в собі такі налаштування проекту як – підключені додатки до проекту (як вбудовані, так і створені під час розробки), налаштування підключення до бази даних, та шлях де будуть зберігатись статичні файли.

models.py – цей файл містить моделі, які відображають інформацію про дані, з якими ви працюєте. Вони містять поля і поведінку ваших даних. Зазвичай одна модель представляє одну таблицю в базі даних. Код створення основних моделей проекту відображено на рисунках 3.35-3.37.

```python
class Category(models.Model):
    name = models.CharField(max_length=200)
    slug = models.SlugField(blank=True)
    img = models.ImageField(upload_to='./static/Resource')

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('category', kwargs={'category_slug': self.slug})
```

Рисунок 3.35 – Опис моделі «Category»

```python
class Contributor(models.Model):
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    maker = models.CharField(max_length=100)
    model = models.CharField(max_length=200)
    is_active = models.BooleanField
    link_itbox = models.CharField(max_length=250)
    link_rozetka = models.CharField(max_length=250)
    link_citrus = models.CharField(max_length=250)
    link_allo = models.CharField(max_length=250)
    link_stylus = models.CharField(max_length=250)
    img1 = models.CharField(max_length=250)
    img2 = models.CharField(max_length=250)
    img3 = models.CharField(max_length=250)

    def __str__(self):
        return self.model

    def get_absolute_url(self):
        return reverse('products', kwargs={'product_slug': self.model})
```

Рисунок 3.36 – Опис моделі «Contributor»

```python
class Products(models.Model):
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    model = models.ForeignKey(Contributor, on_delete=models.CASCADE)
    cpu = models.CharField(max_length=100, blank=True)
    cpu_serial = models.CharField(max_length=100, blank=True)
    speed = models.FloatField(max_length=25, blank=True)
    videocard = models.CharField(max_length=100, blank=True)
    ram_type = models.CharField(max_length=100, blank=True)
    ram = models.IntegerField(blank=True)
    hd_type = models.CharField(max_length=100, blank=True)
    hd = models.IntegerField(blank=True)
    diagonal = models.FloatField(max_length=25, blank=True)
    main_cam = models.IntegerField(blank=True)
    back_cam = models.CharField(max_length=100, blank=True)
    front_cam = models.CharField(max_length=100, blank=True)
    color = models.CharField(max_length=100, blank=True)
    os = models.CharField(max_length=100, blank=True)
    price_itbox = models.IntegerField(blank=True)
    price_rozetka = models.IntegerField(blank=True)
    price_citrus = models.IntegerField(blank=True)
    price_allo = models.IntegerField(blank=True)
    price_stylus = models.IntegerField(blank=True)

    def __int__(self):
        return self.model

    def get_absolute_url(self):
        return reverse('products', kwargs={'product_slug': self.model})


    def minimal(self):
        prices = [self.price_itbox, self.price_rozetka, self.price_citrus, self.price_allo, self.price_stylus]
        prices = [x for x in prices if x != 0]
        return min(prices)

    def maximal(self):
        prices = [self.price_itbox, self.price_rozetka, self.price_citrus, self.price_allo, self.price_stylus]
        return max(prices)
```

Рисунок 3.37 – Опис моделі «Products»

Найважливішою є модель Products в якій описана вся інформація про певний продукт, ціни з певного магазину та посилання на перших 2 моделі. Також в цій моделі описано функціонал для знаходження мінімальної та максимальної ціни певного продукту, під час виклику даних функцій вони

отримують всі ціни на представлений товар з бази даних та обчислюють найменшу чи найбільшу, в залежності від потреби.

Також в моделі Products описаний один із важливих алгоритмів для проведення аналізу цін та магазинів. Алгоритм вираховує найкращу ціну на ринку, яка вирахувується на основі відгуків в магазинів, які можна вважати коефіцієнтом популярності магазину та цін в представлених магазинах. Даний алгоритм представлено на рисунку 3.38.

```
def best_price(self):
    com = [self.itbox_com_count, self.rozetka_com_count, self.citrus_com_count, self.allo_com_count,
           self.stylus_com_count]
    prices = [self.price_itbox * com[0], self.price_rozetka * com[1], self.price_citrus * com[2],
              self.price_allo * com[3], self.price_stylus * com[4]]
    count = 0
    for i in range(len(prices)):
        if prices[i] == 0:
            del com[i - count]
            count = count + 1
    price = (sum(prices) / sum(com))
    round(price, 0)
    print(price)
    return price
```

Рисунок 3.38 – Алгоритм обрахування найкращої ціни на ринку

views.py – це файл який відповідає за функції представлення, або коротко представлення – це функція Python, яка приймає Web-запит і повертає Web-відповідь. Відповіддю може бути HTML-вміст сторінки, або перенаправлення, або 404 помилка, або XML-документ, або зображення... що завгодно. Представлення містить всю необхідну логіку для створення відповіді. Цей код може знаходитися де завгодно, головне, щоб він знаходився в PYTHON_PATH. Ніяких інших вимог немає – ніякої "магії". Незважаючи на можливість розташувати код представлення де завгодно, прийнято тримати його у файлі views.py, який знаходиться в каталозі проекту або програми.

На рисунку 3.39 представлено код представлення головної сторінки, яку користувач зустрічає вперше перейшовши на сайт.

```python
def index(request):
    contributor = Contributor.objects.all()
    products = Products.objects.all()
    all_category = CategoryModel.objects.all()
    context = {
        'Contributor': contributor,
        'Products': products,
        'All_categories': all_category
    }
    return render(request, "index.html", context)
```

Рисунок 3.39 – Код представлення для головної сторінки

Сторінка каталогу більш складна, так як на даній сторінці виводяться характеристики для кожного товару, а також меню фільтрації, яке генерується відносно до того які властивості мають товари представлені в даній категорій, код для обробки цих процесів представлено на рисунку 3.40.

```python
def prod_card(request, category_slug):
    contributor = Contributor.objects.all()
    products = Products.objects.all()
    all_category = CategoryModel.objects.all()
    category = CategoryModel.objects.filter(slug=category_slug)
    cat = (str(request).split('/')[2].split("'")[0])
    makers = Contributor.objects.filter(category__slug=category_slug).values_list('maker', flat=True).distinct()
    videocard = Products.objects.filter(category__slug=category_slug).values_list('videocard', flat=True).distinct()
    ram = Products.objects.filter(category__slug=category_slug).values_list('ram', flat=True).distinct()
    cpu_serial = Products.objects.filter(category__slug=cat).values_list('cpu_serial', flat=True).distinct()
    diagonal = Products.objects.filter(category__slug=cat).values_list('diagonal', flat=True).distinct()
    context = {
        'Contributor': contributor,
        'Products': products,
        'Categories': category,
        'cat': cat,
        'Makers': makers,
        'Ram': ram,
        'Videocard': videocard,
        'Cpu_serial': cpu_serial,
        'Diagonal': diagonal,
        'All_categories': all_category
    }
    return render(request, "prod-card.html", context)
```

Рисунок 3.40 – Код представлення для каталогу товарів

Код представлення каталогу товарів відповідає за представлення товарів в певній категорії. На даній сторінці отримуються всі активні продукти для певної категорії відсортовані за потрібністю, а також їх можна відфільтрувати за характеристиками товару, опис цієї функціональності буде приведено нижче.

В більшості випадків одній сторінці відповідає одне представлення, але в випадку з сторінкою каталогом товарів після фільтрації товарів використовується та ж сторінка, але кардинально міняється представлення, тому для каталогу товарів було прописано окремий вивід, який представлено раніше. Код для даної сторінки представлено на рисунку 3.41

```
def filtered(request):
    cat = request.GET.get("category")
    contributor = Contributor.objects.all()
    category = CategoryModel.objects.filter(pk=cat)
    makers = Contributor.objects.filter(category__pk=cat).values_list('maker', flat=True).distinct()
    videocard = Products.objects.filter(category__pk=cat).values_list('videocard', flat=True).distinct()
    diagonal = Products.objects.filter(category__pk=cat).values_list('diagonal', flat=True).distinct()
    ram = Products.objects.filter(category__pk=cat).values_list('ram', flat=True).distinct()
    cpu_serial = Products.objects.filter(category__pk=cat).values_list('cpu_serial', flat=True).distinct()
    all_category = CategoryModel.objects.all()
    # maker
    if request.GET.getlist("maker") == ['All']:
        maker = Products.objects.all().values_list('model', flat=True).distinct()
    elif request.GET.getlist("maker"):
        maker = Contributor.objects.filter(maker=request.GET.get('maker'))
    else:
        maker = Products.objects.all().values_list('model', flat=True).distinct()
    # checkbox ram
    if request.GET.getlist("ram"):
        ram1 = request.GET.getlist('ram')
    else:
        ram1 = ram
    # checkbox cpu
    if request.GET.getlist("cpu"):
        cpu_serial1 = request.GET.getlist('cpu')
    else:
        cpu_serial1 = cpu_serial
    # checkbox videocard
    if request.GET.getlist("videocard"):
        videocard1 = request.GET.getlist('videocard')
    else:
        videocard1 = videocard
    # checkbox diagonal
    if request.GET.getlist("diagonal"):
```

Рисунок 3.41 – Код представлення для фільтрованого каталогу

Представлення фільтрованої сторінки подібне до звичайного каталогу але на сторінці відсутні товари які не відповідають пошуковому запиту користувача.

Окрім пошуку по заданим характеристикам товарів також можливий пошук товарів по імені. Представлення такого пошуку відображено на рисунку 3.42.

```
132     def search(request):
133         contributor = Contributor.objects.filter(model__icontains=request.GET.get("s
134         cat = request.GET.get("search")
135         products = Products.objects.all()
136         print(contributor)
137         context = {
138             'Products': products,
139             'Contributor': contributor,
140             'Cat': cat,
141         }
142         return render(request, "search.html", context)
```

Рисунок 3.42 – Код представлення пошуку товару по назві

На рисунку 3.43 представлено код реалізації детальної сторінки товару.

```
def product_page(request, product_slug):
    contributor = Contributor.objects.filter(model=product_slug)
    products = Products.objects.all()
    category = CategoryModel.objects.all()
    all_category = CategoryModel.objects.all()
    context = {
        'Contributor': contributor,
        'Products': products,
        'Categories': category,
        'All_categories': all_category
    }
    return render(request, "product_page.html", context)
```

Рисунок 3.43 – Код представлення для сторінки товару

Для забезпечення реєстрації та авторизації користувачів також потрібні відповідні представлення які отримують хеш введені користувачем на фронтенд частині сторінки та проводять певні маніпуляції в базі даних (рис. 3.44-3.45).

```
165  def login(request):
166      success_url = reverse_lazy('user')
167
168      username = request.POST['username']
169      password = request.POST['password']
170      userdata = authenticate(request, email=username, password=password
171      if userdata is not None:
172          auth_login(request, userdata)
173          return redirect('index')
174      else:
175          return redirect('user')
```

Рисунок 3.44 – Код представлення для сторінки авторизації

```
182  def register(request):
183      mails = CustomUser.objects.all().values_list('email')
184      mails_list = []
185      for m in mails:
186          mails_list.append(m[0])
187
188      if request.POST['mail'] in mails_list:
189          return redirect('register')
190      else:
191          if request.POST['password'] == request.POST['repeat']:
192              CustomUser.objects.create_user(email=request.POST['mail'],
193                                             first_name=request.POST['firstname'],
194                                             last_name=request.POST['lastname'],
195                                             phone_number=request.POST['phonenumber']
196                                             password=request.POST['password']
197                                             )
198              return redirect('user')
199          else:
200              return redirect('register')
```

Рисунок 3.45 – Код представлення для сторінки реєстрації

Для меншого навантаження на обчислювальні машини та більш зручного перегляду товарів на сторінках, було створено утиліту для пагінації товарів на сторінках, даний функціонал представлено на рисунку 3.46.

```python
class DataMixin:
    paginate_by = 3

    def get_user_context(self, **kwargs):
        context = kwargs
        cats = cache.get('cats')
        if not cats:
            cats = Category.objects.annotate(Count('product'))
            cache.set('cats', cats, 60*15)

        user_menu = main_menu.copy()
        if not self.request.user.is_staff:
            user_menu.pop(1)

        context['main_menu'] = user_menu
        context['cats'] = cats
        if 'cat_selected' not in context:
            context['cat_selected'] = 0
        return context
```

Рисунок 3.46 – Код утиліти для пагінації

Таким чином виглядить структура веб-додатку, де в підсумку отримано сайт на якому можна знайти потрібний товар, переглянути статистику по магазинам, які продають представлений товар та рекомендовану ринкову ціну відштовхуючись від якої потрібно обирати потрібний товар.

Для актуалізації інформації про товар було розроблено скрипт для асинхронного оновлення потрібних даних на сторінці товару. Схема роботи скрипта:

– запит на отримання посилань на товар в різні магазини;

– отримання інформації про товар з веб-сторінки;

– оновлення інформації про товар в базі даних;

– вивід інформації на сторінку.

Скрипт для автоматичного оновлення цін на товар в базі даних представлено на рисунку 3.47.

```
def get_content(html, site):
    html = soup(html, 'html.parser')
    html.prettify(formatter=lambda s: s.replace(u'\xa0', ' '))
    if site.lower() == 'itbox':
        try:
            price = html.select_one("div.product-info strong.stuff-price__digits.no-product").get_text()
        except:
            price = html.select_one("div.product-info strong.stuff-price__digits").get_text().replace(' ','')
            print(site,price)
    elif site.lower() == 'rozetka':
        try:
            price = html.select_one("p.status-label.status-label--gray").get_text()
        except:
            price = html.select_one("p.product-prices__big").get_text().replace('₴','').replace(' ','').replace(
            print(site,price)
    elif site.lower() == 'citrus':
        availability = html.select_one("span.aic").get_text()
        if availability.lower() == 'есть в наличии':
            price = html.select_one("div.price").get_text().replace('₴','').replace(' ','').replace(u'\xa0','')
            print(site,price)
    elif site.lower() == 'stylus':
        try:
            price = html.select_one("div.center-part div.not-available").get_text()
        except:
            price = html.select_one("div.center-part div.regular-price").get_text().replace('грн','').replace('
            print(site,price)
    elif site.lower() == 'allo':
        try:
            price = html.select_one("span.p-trade__stock-label-icon").get_text()
            price = html.select_one("div.p-trade-price meta[itemprop='price']").get('content')
            print(site,price)
        except:
            pass
```

Рисунок 3.47 – Скрипт для автоматичного оновлення цін

Для забезпечення роботи скрипта без навантаження на обчислювальну машину користувача створено асинхронну чергу задач методами бібліотек Celery та Redis, для їх адекватної роботи також потрібно помістити розробку в контейнер Docker.

Нижче на рисунку 3.48 представлено основний docker файл який описує покроково запуск проекту всередені контейнера.

– встановлюється робоча директорія проекту;

– run requirements.txt відповідає за встановлення всіх залежностей та бібліотек, які використані в проекті та описані в окермому файлі;

– run entrypoint.sh налаштовує базу даних перед запуском проекту та створює «суперюзера» для можливості управління веб-додатком, а також відповідні кроки для запуску додатку.

```
FROM python:3

WORKDIR /usr/src/app

COPY requirements.txt .
COPY entrypoint.sh .

RUN pip install -r requirements.txt
RUN chmod +x entrypoint.sh

copy . .

ENTRYPOINT ["/usr/src/app/entrypoint.sh"]
```

Рисунок 3.48 – Основний docker файл

Також для налаштування контейнеру створено файл docker-yaml який містить налаштування окремих додатків всього проекту, таких як – база даних, сервер та сам веб-додаток, код налаштування представлено на рисунку 3.49.

```
services:
  postgresdb:
    build:
      context: ./docker/postgres
      dockerfile: Dockerfile
    environment:
      - POSTGRES_PASSWORD=admin1
      - POSTGRES_USER=admin1
      - POSTGRES_DB=django_docker
    volumes:
    - ./docker/postgres/init.sql:/docker-entrypoint-initdb.d/init.sql
    ports:
    - "5432:5432"

  webapp:
    build:
      context: ./
      dockerfile: Dockerfile
    volumes:
    - ./:/usr/src/app
    depends_on:
      - postgresdb
    ports:
      - "8000:8000"
  nginx:
    build:
      context: ./docker/nginx
      dockerfile: Dockerfile
    depends_on:
      - webapp
      - postgresdb
    ports:
    - "80:80"
    volumes:
    - ./static/:/static
```

Рисунок 3.49 – Налаштування додатків docker

## 3.4 Висновки

В цьому розділі розроблено та протестовано аналітичну систему моніторингу цін електронних товарів в різних інтернет-магазинів, а також розроблено архітектуру системи та побудовано необхідні для подальшої розробки UML- діаграми сценаріїв використання.

# 4 ЕКОНОМІЧНА ЧАСТИНА

## 4.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного та технологічного аудиту є оцінювання комерційного потенціалу розробки математичної моделі, яка дозволяє спрогнозувати рекомендовану ціну необхідного електронного товару та визначити найкращий інтернет-магазин для його купівлі.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів Вінницького національного технічного університету кафедри системного аналізу та комп'ютерної інженерії: к.т.н.,доц. Козачко О.М., к.т.н., доц. Крижановський Є.М., к.т.н., доц. Варчук І.В. Для проведення технологічного аудиту було використано таблицю 4.1 [20] в якій за п’ятибальною шкалою використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

| Критерії оцінювання та бали (за 5-ти бальною шкалою) | | | | | |
|---|---|---|---|---|---|
| Кри-терій | 0 | 1 | 2 | 3 | 4 |
| Технічна здійсненність концепції: | | | | | |
| 1 | Достовірність концепції не підтверджена | Концепція підтверджена експертними висновками | Концепція підтверджена розрахунками | Концепція перевірена на практиці | Перевірено роботоздатність продукту в реальних умовах |
| Ринкові переваги (недоліки): | | | | | |
| 2 | Багато аналогів на малому ринку | Мало аналогів на малому ринку | Кілька аналогів на великому ринку | Один аналог на великому ринку | Продукт не має аналогів на великому ринку |
| 3 | Ціна продукту значно вища за ціни аналогів | Ціна продукту дещо вища за ціни аналогів | Ціна продукту приблизно дорівнює цінам аналогів | Ціна продукту дещо нижче за ціни аналогів | Ціна продукту значно нижче за ціни аналогів |
| 4 | Технічні та споживчі властивості продукту значно гірші | Технічні та споживчі властивості продукту трохи гірші | Технічні та споживчі властивості продукту на рівні аналогів | Технічні та споживчі властивості продукту трохи кращі | Технічні та споживчі властивості продукту значно кращі |

Продовження табл. 4.1

| Критерії оцінювання та бали (за 5-ти бальною шкалою) | | | | | |
|---|---|---|---|---|---|
| Кри-терій | 0 | 1 | 2 | 3 | 4 |
| 5 | Експлуатаційні витрати значно вищі, ніж в аналогів | Експлуатаційні витрати дещо вищі, ніж в аналогів | Експлуатаційні витрати на рівні експлуатаційних витрат аналогів | Експлуатаційні витрати трохи нижчі, ніж в аналогів | Експлуатаційні витрати значно нижчі, ніж в аналогів |
| Ринкові перспективи | | | | | |
| 6 | Ринок малий і не має позитивної динаміки | Ринок малий, але має позитивну динаміку | Середній ринок з позитивною динамікою | Великий стабільний ринок | Великий ринок з позитивною динамікою |
| 7 | Активна конкуренція великих компаній на ринку | Активна конкуренція | Помірна конкуренція | Незначна конкуренція | Конкурентів немає |
| Практична здійсненність | | | | | |
| 8 | Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї | Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців | Необхідне незначне навчання фахівців та збільшення їх штату | Необхідне незначне навчання фахівців | Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї |
| 9 | Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні | Потрібні незначні фінансові ресурси. Джерела фінансування відсутні | Потрібні значні фінансові ресурси. Джерела фінансування є | Потрібні незначні фінансові ресурси. Джерела фінансування є | Не потребує додаткового фінансування |

| | | | | | |
|---|---|---|---|---|---|
| 10 | Необхідна розробка нових матеріалів | Потрібні матеріали, що використовуються у військово-промисловому комплексі | Потрібні дорогі матеріали | Потрібні досяжні та дешеві матеріали | Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві |
| 11 | Термін реалізації ідеї більший за 10 років | Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років | Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років | Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років | Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років |
| 12 | Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту | Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу | Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу | Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту | Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту |

Таблиця 4.2 – Рівні комерційного потенціалу розробки

| Середньоарифметична сума балів СБ, розрахована на основі висновків експертів | Рівень комерційного потенціалу розробки |
|---|---|
| 0-10 | Низький |
| 11-20 | Нижче середнього |
| 21-30 | Середній |
| 31-40 | Вище середнього |
| 41-48 | Високий |

В таблиці 4.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 4.3 – Результати оцінювання комерційного потенціалу розробки

| Критерії | Прізвище, ініціали, посада експерта | | |
|---|---|---|---|
| | Козачко О.М. | Крижановський Є.М. | Варчук І.В. |
| | Бали, виставлені експертами: | | |
| 1 | 4 | 4 | 3 |
| 2 | 1 | 1 | 2 |
| 3 | 3 | 4 | 4 |
| 4 | 2 | 2 | 3 |
| 5 | 3 | 3 | 4 |
| 6 | 3 | 3 | 3 |
| 7 | 2 | 2 | 2 |
| 8 | 3 | 1 | 2 |
| 9 | 4 | 3 | 3 |
| 10 | 3 | 4 | 4 |
| 11 | 4 | 3 | 3 |
| 12 | 2 | 4 | 3 |
| Сума балів | $СБ_1$=35 | $СБ_2$=34 | $СБ_3$=34 |
| Середньоарифметична сума балів $\overline{СБ}$ | $\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{35+34+34}{3} = 34$ | | |

Середньоарифметична сума балів, розрахована на основі висновків експертів склала 34 бали, що згідно таблиці 4.2 вважається, що рівень комерційного потенціалу проведених досліджень є вище середнього.

Програмний продукт, який розробляється в магістерській роботі може бути використаний у аналізі ринку електронних товарів України та для вибору найкращих магазинів для покупки товарів.

Порівняємо нову розробку з аналогами, які існують на ринку. В якості аналога було обрано https://ek.ua/. Основними недоліками аналога є представлення цін певних магазинів без аналізу конкурентності ціни. Також до недоліків можна віднести співпраця з магазинами представниками.

Проведемо оцінку якості і конкурентоспроможності нової розробки порівняно з аналогом. В таблиці 4.4 наведені основні техніко-економічні показники аналога і нової розробки.

Таблиця 4.4 – Основні параметри нової розробки та товару-конкурента

| Показник | Варіанти | | Відносний показник якості | Коефіцієнт вагомості параметра |
|---|---|---|---|---|
| | Базовий (товар-конкурент) | Новий (інноваційне рішення) | | |
| 1 | 2 | 3 | 4 | 5 |
| Швидкість обробки даних, мб/с | 10 | 13 | 1,3 | 35% |
| Точність представлених даних, % | 75 | 90 | 1,2 | 50% |
| Навантаження на сервер, % | 90 | 50 | 1,8 | 15% |

Проведемо оцінку якості продукції, яка є найефективнішим засобом забезпечення вимог споживачів та порівняємо її з аналогом.

Визначимо відносні одиничні показники якості по кожному параметру за формулами (4.1) та (4.2) і занесемо їх у відповідну колонку таблиці 4.5.

$$q_i = \frac{P_{Hi}}{P_{Бi}} \tag{4.1}$$

або

$$q_i = \frac{P_{Бi}}{P_{Hi}} \tag{4.2}$$

де $P_{Hi}$, $P_{Бi}$ – числові значення $i$-го параметру відповідно нового і базового виробів.

$$q_1 = \frac{13}{10} = 1{,}3;$$

$$q_2 = \frac{90}{75} = 1{,}2;$$

$$q_3 = \frac{90}{50} = 1{,}8.$$

Відносний рівень якості нової розробки визначаємо за формулою:

$$К_{я.в.} = \sum_{i=1}^{n} q_i \cdot \alpha_i, \quad (4.3)$$

$$К_{я.в.} = 1{,}3 \cdot 0{,}35 + 1{,}2 \cdot 0{,}5 + 1{,}8 \cdot 0{,}15 = 1{,}3.$$

Відносний коефіцієнт показника якості нової розробки більший одиниці, отже нова розробка якісніший базового товару-конкурента.

Наступним кроком є визначення конкурентоспроможності товару. Конкурентоспроможність товару є головною умовою конкурентоспроможності підприємства на ринку і важливою основою прибутковості його діяльності.

Однією із умов вибору товару споживачем є збіг основних ринкових характеристик виробу з умовними характеристиками конкретної потреби покупця. Такими характеристиками найчастіше вважають нормативні та технічні параметри, а також ціну придбання та вартість споживання товару.

В таблиці 4.5 наведено технічні та економічні показники для розрахунку конкурентоспроможності нової розробки відносно товару-аналога, технічні дані взяті з попередніх розрахунків.

Таблиця 4.5 – Нормативні, технічні та економічні параметри нової розробки і товару-виробника

| Показники | Варіанти | |
|---|---|---|
| | Базовий (товар-конкурент) | Новий (інноваційне рішення) |
| 1 | 2 | 3 |
| 1. Нормативно-технічні показники | | |
| Швидкість обробки даних, мб/с | 10 | 13 |
| Точність представлених даних, % | 75 | 90 |
| Навантаження на сервер, % | 90 | 50 |
| 2. Економічні показники | | |
| Ціна придбання, грн. | 135000 | 81000 |

Загальний показник конкурентоспроможності інноваційного рішення (К) з урахуванням вищезазначених груп показників можна визначити за формулою:

$$K = \frac{I_{m.n.}}{I_{e.n.}}, \tag{4.4}$$

де $I_{m.n.}$ – індекс технічних параметрів;

$I_{e.n.}$ – індекс економічних параметрів.

Індекс технічних параметрів є відносним рівнем якості інноваційного рішення. Індекс економічних параметрів визначається за формулою (4.5)

$$I_{e.n.} = \frac{\sum_{i=1}^{n} P_{Нei}}{\sum_{i=1}^{n} P_{Бei}}, \tag{4.5}$$

де $P_{Нei}$, $P_{Бei}$ – економічні параметри (ціна придбання та споживання товару) відповідно нового та базового товарів.

$$I_{е.п.} = \frac{81000}{135000} = 0,6;$$

$$К = \frac{1,3}{0,6} = 2,2.$$

Зважаючи на розрахунки, можна зробити висновок, що нова розробка буде конкурентоспроможніше, ніж конкурентний товар.

## 4.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов'язані з проведенням науково-дослідної роботи групуються за такими статтями: витрати на оплату праці, витрати на соціальні заходи, матеріали, паливо та енергія для науково-виробничих цілей, витрати на службові відрядження, програмне забезпечення для наукових робіт, інші витрати, накладні витрати.

1. Основна заробітна плата кожного із дослідників $З_О$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_О = \frac{М}{Т_р} *$$

$t$ (грн) (4.6)

де $M$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$T_P$ – число робочих днів в місяці; приблизно $T_P \approx 21...23$ дні;

$t$ – число робочих днів роботи дослідника.

Для розробки програмні засоби для формування та контролю індивідуальної програми фізичних тренувань необхідно залучити програміста з посадовим окладом 12000 грн. Кількість робочих днів у місяці складає 22, а кількість робочих днів програміста складає 21. Зведемо сумарні розрахунки до таблиця 4.6.

Таблиця 4.6 – Заробітна плата дослідника в науковій установі бюджетної сфери

| Найменування посади | Місячний посадовий оклад, грн. | Оплата за робочий день, грн. | Число днів роботи | Витрати на заробітну плату грн. |
|---|---|---|---|---|
| Керівник | 15000 | 681,8 | 5 | 3409 |
| Програміст | 12000 | 545,5 | 21 | 11455 |
| Всього | | | | 14864 |

2. Розрахунок додаткової заробітної плати робітників

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 – 12 % від основної заробітної плати робітників.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 10% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{дод}}{100\%} \quad (4.7)$$

$$З_д = 0{,}11 * 14864 = 1635 \text{ (грн)}$$

3. Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою (4.3):

$$Н_{ЗП} = (З_о + З_д) * \frac{\beta}{100} \text{ (грн)} \quad (4.8)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

$\beta$ – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді:

$$Н_{ЗП} = (14864 + 1635) * \frac{22}{100} = 3629{,}7 \text{ (грн)}$$

4. Витрати комлектуючі К, що були використані під час виконання даного етапу роботи, розраховуються по кожному виду матеріалів за формулою:

$$К = \sum_{i=1}^{n} Н_i \cdot Ц_i \cdot К_i, \quad (4.9)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.;

$Ц_i$ – покупна ціна комплектуючих *i*-го найменування, грн.;

$К_i$ – коефіцієнт транспортних витрат (1,1…1,15).

Таблиця 4.7 – Комплектуючі, що використані на розробку

| Найменування | Ціна за одиницю, грн. | Витрачено | Вартість витраченого матеріалу, грн. |
|---|---|---|---|
| Папір | 130 | 1 | 130 |
| Ручка | 15 | 1 | 15 |
| CD-диск | 12 | 1 | 12 |
| Флешка | 180 | 1 | 180 |
| Всього | | | 337 |
| З врахуванням коефіцієнта транспортування | | | 370,7 |

5. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час виконання даного етапу роботи

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

$$A = \frac{Ц \cdot T}{Ткор \cdot 12} \quad [грн], \tag{4.10}$$

де Ц – балансова вартість даного виду обладнання (приміщень), грн.;

$Т_{кор}$ – час користування;

Т – термін використання обладнання (приміщень), цілі місяці.

Згідно пункта 137.3.3 Податкового кодекса амортизація нараховується на основні засоби вартістю понад 2500 грн. В нашому випадку для написання магістерської роботи використовувався персональний комп'ютер вартістю 26000 грн.

$$\text{А} = \frac{26000 \cdot 1}{2 \cdot 12} = 1083{,}33$$

6. До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

$$\text{В}_\text{е} = \sum_{\text{i}=1}^{n} \frac{W_{yt} \cdot t_i \cdot \text{Ц}_\text{е} \cdot \text{К}_\text{впі}}{\eta_\text{і}} \tag{4.11}$$

де $W_{yt}$ – встановлена потужність обладнання на певному етапі розробки, кВт;

$t_i$ – тривалість роботи обладнання на етапі дослідження, год;

$\text{Ц}_\text{е}$ – вартість 1 кВт-години електроенергії, грн;

$\text{К}_\text{впі}$ – коефіцієнт, що враховує використання потужності, $\text{К}_\text{впі} < 1$;

$\eta_\text{і}$ – коефіцієнт корисної дії обладнання, $\eta_\text{і} < 1$.

Для написання магістерської роботи використовується персональний комп'ютер для якого розрахуємо витрати на електроенергію.

$$\text{В}_\text{е} = \frac{0{,}3 \cdot 170 \cdot 4{,}1 \cdot 0{,}5}{0{,}8} = 130{,}69$$

Витрати на службові відрядження, витрати на роботи, які виконують сторонні підприємства, установи, організації та інші витрати в нашому досліджені не враховуються оскільки їх не було.

Накладні (загальновиробничі) витрати Внзв охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати Внзв можна прийняти як (100…150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$В_{нзв} = \left(З_о + З_р\right) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.12)$$

де $Н_{нзв}$ – норма нарахування за статтею «Інші витрати».

$$В_{нзв} = 14864 \cdot \frac{100}{100\%} = 14864 \text{ грн}$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР

$$В = 14864+1635+3629,7+370,7+1083,3+130,69+14864= 36576,7$$

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою:

$$ЗВ = \frac{В}{\eta}, \quad (4.13)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт $\beta$ = 0,5. Звідси:

$$ЗВ = \frac{36576,7}{0,5} = 73153,79 \text{ грн.}$$

## 4.3 Розрахунок економічної ефективності науково-технічної розробки

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою

$$\Delta П_i = \sum_{1}^{n} \left( \Delta Ц_o \cdot N + Ц_o \cdot \Delta N \right)_i \cdot \lambda \cdot \rho \cdot \left( 1 - \frac{\upsilon}{100} \right) \quad (4.14)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

л – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт *л* = 0,8333.

р – коефіцієнт, який враховує рентабельність продукту. р = 0,25;

х – ставка податку на прибуток. У 2021 році – 18%.

Припустимо, що при впроваджені результатів наукової розробки покращується якість програмного продукту для формування індивідуальних тренувань. Припустимо, що ціна від зросте на 500 грн. Кількість одиниць

реалізованої продукції також збільшиться: протягом першого року на 150 шт., протягом другого року – на 75 шт., протягом третього року на 50 шт. Реалізація продукції до впровадження розробки складала 1 шт., а її ціна до 81000 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [500 \cdot 1 + (81000 + 500) \cdot 150] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 2088439{,}4 \text{ грн.}$$

$$\Delta П_2 = [500 \cdot 1 + (81000 + 500) \cdot (150 + 75)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 3133030{,}9 \text{ грн.}$$

$$\Delta П_3 = [500 \cdot 1 + (81000 + 500) \cdot (150 + 75 + 50)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 3829148{,}9 \text{ грн.}$$

## 4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \qquad (4.15)$$

$k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{\text{інв}} = 2…5$).

$$PV = 2 \cdot 73153{,}79 \ = 146306{,}77$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$E_{абс} = (ПП - PV) \tag{4.16}$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн.;

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \tag{4.17}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

$\tau$ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$ПП = \frac{2088439,4}{(1+0,2)^1} + \frac{3133030,9}{(1+0,2)^2} + \frac{3829148,9}{(1+0,2)^3} = 6142331,46 \text{ грн.}$$

$$Е_{абс} = (6142331,46 - 146306,77\ ) = 5996024,68 \text{грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.18}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$\mathrm{E_в} = \sqrt[3]{1 + \frac{5996024{,}68}{144630{,}06}} - 1 = 3{,}36 = 336\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{4.19}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d= (0,14…0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{\min} = 0{,}18 + 0{,}05 = 0{,}23.$$

Так як $E_в > \tau_{\min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в} \tag{4.20}$$

$$\mathrm{T_{ок}} = \frac{1}{3{,}36} = 0{,}3 \text{ роки}$$

Так як $T_{ок} \leq$ 3...5-ти років, то фінансування даної наукової розробки в принципі є доцільним.

## 4.5 Висновки

Було проведено оцінку програмного засобу, який дозволяє спрогнозувати рекомендовану ціну необхідного електронного товару та визначити найкращий інтернет-магазин для його купівлі, який є на вище середньому рівні. При порівнянні нової розробки з аналогом виявлено, що вона є якіснішою і конкурентоспроможнішою відносно аналога, а також краще по технічним і економічним показникам.

Прогнозування витрат на виконання науково-дослідної роботи по кожній з статей витрат складе 36576,7 грн. Загальна ж величина витрат на виконання та впровадження результатів даної НДР буде складати 73153,79 грн.

Вкладені інвестиції в даний проект окупляться через 3 місяці при прогнозованому прибутку 6142331,46 грн. за три роки.

# ВИСНОВКИ

В ході магістерської кваліфікаційної роботи було розроблено інформаційну технологію аналізу та прогнозування цін інтернет-магазинів для електронних товарів. Проведено розвідувальний аналіз даних про товари зібраних власноруч з популярних українських інтернет-магазинів. Проведено аналіз існуючих методів аналізу для розв’язання поставленої задачі. Розроблений веб додаток для прогнозування оптимальної ціни та оптимального інтернет-магазину для придбання певного товару.

Проведений аналіз вказав на параметри, які найкраще впливають на вибір необхідного магазину. Також було проведено розвідувальний аналіз цін на електронні товари в різних інтернет-магазинах, який показав досить широкий діапазон варіювання, що ускладнює процес купівлі товару.

Розроблено математичну модель прогнозування рекомендованої ціни на основі ціни та кількості відгуків що вказує на відповідність ціни до якості на товар в різних інтернет-магазинах.

Запропонована модель дозволяє визначити інтернет-магазин, в якому товар є оптимальним за ціною та якістю. Виконавши розробку та протестувавши аналітичну систему, було розроблено веб-додаток з дружнім користувацьким інтерфейсом, можливістю порівнювати товари або магазина між собою, та власним кабінетом користувача.

Виконано економічну частину кваліфікаційної роботи під час розрахунків було доведено:

– абсолютна ефективність вкладених інвестицій, яка дорівнює 6142331,46 грн, що є більшим 0 і вказує на те, що інвестор може бути зацікавленим у нашій розробці;

– відносна ефективність наукової розробки становить 336%, що є вищим за мінімальну ставку дисконтування, тому вкласти кошти у нашу розробку є вигідніше, ніж покласти кошти на депозит.

Поставлені завдання виконано в повному обсязі, мету роботи досягнуто.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Адамова І. З., Уграк М. І. Використання інтернет-технологій у навчальному процесі. КНТЕУ, Чернівці – 82с.

2. Bill Stewart. Internet history – One page summary. 2000.-113с.

3. Ющук І. О. Козачко О. М. Інформаційна тенологія аналізу та прогнозування цін інтернет-магазинів для електронних товарів. *Всеукраїнська науково-практична інтернет-конференція студентів аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2022)»*. 2021-2022.
URL: https://conferences.vntu.edu.ua/index.php/mn/mn2022/paper/view/14224

4. Chatfield, C. Problem Solving: A Statistician's Guide (2nd ed.). Chapman and Hall / ISBN 1995 – 978 с.

5. Encephalos Journal [Електронний ресурс]. URL: www.encephalos.gr

6. Build with AI | DeepAI [Електронний ресурс]. URL: https://deepai.org/machine-learning-glossary-and-terms/neural-network

7. Novak, V .; Perfilieva, I .; Močkoř, J. Mathematical principles of fuzzy logic, 1999.-143с

8. Посібник для розробників CSS. Митчелл Бейкер. [Електронний ресурс]. URL: https://developer.mozilla.org/uk/docs/Web/CSS

9. Що таке CSS?. Всесвітній веб-консорціум. [Електронний ресурс]. URL: http://www.css-class.com/what-is-the-world-wide-web-consortium/

10. David Flanagan JavaScript definitive guide.– 34с.

11. Kuhlman, Dave. A Python Book: Beginning Python, Advanced Python, and Python Exercises. – 76с.

12. Special method names. The Python Language Reference [Електронний ресурс]. URL: https://docs.python.org/3/reference/datamodel.html

13. Adrian Holovaty, Jacob Kaplan-Moss. The Django Book. 118с.

14. Дронов В. А. Django 3.0. Практика створення веб-сайтів на Python. – СПб.: БХВ-Петербург, 2021. – 704 с

15. A Brief History of PostgreSQL. [Електронний ресурс]. URL: https://www.postgresql.org/docs/current/history.html.

16. І. Панченко PostgreSQL: вчора, сьогодні, завтра. Відкриті системи. СУБД, 2015. – 48 с.

17. Dirk Merkel. Docker: lightweight Linux containers for consistent development and deployment (англ.) // Linux Journal. 2014. – 71 с.

18. Китаев Е. Л., Скорнякова Р. Ю. Скрепінг «на льоту» зовнішніх веб-ресурсів, / Препринти ІПМ, 2019. – 149 с.

19. Тютярев А. А., Соломатин Д. И. Розробка фреймворків для створення веб-скраперів 2016. – 98 с.

20. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

Додаток А

(обов'язковий)

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет комп'ютерних систем і автоматики

ЗАТВЕРДЖУЮ

Завідувач кафедри САІТ

_____ д.т.н., проф. Мокін В. Б.

«___» _____ 2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ЦІН ІНТЕРНЕТ-МАГАЗИНІВ ДЛЯ ЕЛЕКТРОННИХ ТОВАРІВ»

08-53.МКР.008.02.000.ТЗ

Керівник: к.т.н., доцент

_____ Козачко О. М.

«___» _____ 2021 р.

Розробив: студент гр. 2ІСТ-20м

_____ Ющук І. О .

«___» _____ 2021 р.

Вінниця 2021

1. Підстава для проведення робіт

Підставою для виконання роботи є наказ № __ по ВНТУ від «__» _____ 2021 р., та індивідуальне завдання на МКР, затверджене протоколом № __ засідання кафедри САІТ від «__» _____ 2021 р.

2. Джерела розробки:

- Дронов В. А. Django 3.0. Практика створення веб-сайтів на Python. – СПб.: БХВ-Петербург, 2021. – 704 с
- Тютярев А. А., Соломатин Д. И. Розробка фреймворків для створення веб-скраперів 2016. – 98 с.

3. Мета і призначення роботи:

Розробка інформаційної технології аналізу та прогнозування цін інтернет-магазинів для електронних товарів

4.Вихідні дані для проведення робіт:

Датасети з даними про товари зібрані з популярних українських інтернет магазинів.

5.Методи дослідження:

- розвідувальний аналіз;
- моделювання системи.

6. Етапи роботи і терміни їх виконання:

1. Аналіз предметної області .....____ – ____
2. Розробка моделі прогнозування рекомендованої ціни електронних товарів.....____ – ____
3. Розробка аналітичної системи моніторингу цін електронних товарів в інтернет-магазинах.....____ – ____
4. Оформлення пояснювальної записки. .....____ – ____

7. Очікувані результати та порядок реалізації:

Отримання інформаційної технології аналізу та прогнозування цін інтернет-магазинів для електронних товарів.

8. Вимоги до розробленої документації

Пояснювальна записка оформлена у відповідності до вимог «Методичних вказівок до виконання та оформлення магістерських кваліфікаційних робіт для студентів спеціальності 126 – «Інформаційні системи та технології» денної форми навчання».

9. Порядок приймання роботи

Публічний захист ..... «___» _____.2021 р.
Початок розробки..... «___» _____ 2021 р.
Граничні терміни виконання МКР ..... «___» _____ 2021 р.

Розробив студент групи 2ІСТ-20м _____ Ющук І.О.

ДОДАТОК Б

(обов'язковий)

Протокол перевірки кваліфікаційної роботи

Назва роботи: «Інформаційна технологія аналізу та прогнозування цін інтернет-магазинів для електронних товарів»
Тип роботи: магістерська кваліфікаційна робота
Підрозділ: кафедра САІТ
Науковий керівник: Козачко О.М. к.т.н., доцент

Показники звіту подібності

| Unicheck | |
|---|---|
| Оригінальність | 88,5 % |
| Схожість | 11,5 % |

Аналіз звіту подібності (відмітити потрібне)

- ☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений з повним звітом подібності, який був згенерований системою щодо роботи

Автор _____ Ющук І.О.
(підпис)

Опис прийнятого рішення

Робота допускається до захисту

Особа, відповідальна за перевірку _____ Жуков С. О.
(підпис)

Додаток В

(довідниковий)

Лістинг програмного коду.

```
from selenium import webdriver
import mysql.connector
import time

# Підключення до браузера
executable_path = "/home/grow/Documents/Diplom/Pars_eShop/chromedriver"
chromeOptions = webdriver.ChromeOptions()
driver = webdriver.Chrome(executable_path=executable_path)
driver.implicitly_wait(5)

# з'єднання з базою даних
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="ei7veeChu4bo",
    database="eShop"
)
cursor = mydb.cursor()

def cycle(id):
    print(id)
    query = "select count(id) from FoxApp_contributor;"
    cursor.execute(query)
    counter = cursor.fetchone()
```

```
    while id <= counter[0]:
        query = "select category_id from FoxApp_contributor where id = '" + str(id) +
"';"
        cursor.execute(query)
        type_ = cursor.fetchone()
        try:
            if type_[0] == 1:
                pc(id)
            elif type_[0] == 2:
                laptop(id)
            elif type_[0] == 3:
                smartphone(id)
        except:
            id = id + 1
            cycle(id)
        id = id + 1


def pc(id):
    DEFAULT = 0
    id = """+str(id)+"""
    query = "select link_itbox, link_rozetka, link_citrus, link_allo, link_stylus  from
FoxApp_contributor where id = "+id+";"
    cursor.execute(query)
    link = cursor.fetchone()
    itbox = link[0]
    rozetka = link[1]
    citrus = link[2]
    allo = link[3]
    stylus = link[4]
```

```
    driver.get(itbox)
    driver.find_element_by_class_name("characteristic-toggle").click()
    elements1 = []
    info = driver.find_elements_by_tag_name("td")
    for elem in info:
        elements1.append(elem.text)
    i = 0
    for elem in elements1:
        i = i+1
        if elem == "Серия процессора":
            cpu = elements1[i]
        if elem == "Частота процессора, ГГц":
            speed = elements1[i]
        if elem == "Модель видеокарты":
            videocard = elements1[i]
        if elem == "Тип памяти":
            ram_type = elements1[i]
        if elem == "Объем установленной памяти":
            ram = elements1[i].split("ГБ")[0]
        if elem == "Типы внутренних накопителей":
            hd_type = elements1[i]
        if elem == "Объем HDD" or elem == "Объем SSD":
            hd = elements1[i].split(" ")[0]
    try:
        itbox_price = driver.find_element_by_class_name("stuff-
price").text.split("грн")[0].replace(' ', '')
        itbox_price = int(itbox_price)
    except:
```

```
        itbox_price = 0
    try:
        itbox_com_count = driver.find_element_by_class_name("header-comments-count").text.split(" ")[0]
    except:
        itbox_com_count = 0
    try:
        driver.get(rozetka)
        driver.implicitly_wait(5)
        rozetka_price = driver.find_element_by_class_name("product-prices__big").text.split("₴")[0].replace(' ', '')
        rozetka_price = int(rozetka_price)
    except:
        rozetka_price = 0
    try:
        rozetka_com_count = driver.find_element_by_class_name("product-tabs__link-text").text
    except:
        rozetka_com_count = 0
    try:
        driver.get(citrus)
        time.sleep(1)
        driver.implicitly_wait(5)
        citrus_price = driver.find_element_by_xpath("//div[@class='price']").text.split("₴")[0].replace(' ', '')
        citrus_price = int(citrus_price)
    except:
        citrus_price = 0
    try:
```

```
        citrus_com_count = driver.find_elements_by_class_name("el-tabs__item")[-
2].text.split(" ")[1]
        if type(int(citrus_com_count)) != int:
            citrus_com_count = 0
        else:
            citrus_com_count = citrus_com_count
    except:
        citrus_com_count = 0
    try:
        driver.get(allo)
        driver.implicitly_wait(5)
        allo_price =
driver.find_element_by_xpath("//meta[@itemprop='price']").get_attribute("content").
replace(' ', '')
        allo_price = int(allo_price)
    except:
        allo_price = 0
    try:
        allo_com_count =
driver.find_element_by_xpath("//a[@id='discussionTab']").text.split(" ")[3]
    except:
        allo_com_count = 0
    try:
        driver.get(stylus)
        driver.implicitly_wait(5)
        stylus_price = driver.find_element_by_class_name("regular-
price").text.split("грн")[0].replace(' ', '')
        stylus_price = int(stylus_price)
    except:
        stylus_price = 0
```

```
    try:
        driver.maximize_window()
        stylus_com_count = driver.find_element_by_class_name("count-of-reviews").text.split(" ")[0]
    except:
        stylus_com_count = 0
    if int(hd) < 10:
        hd = int(hd)*1000
    cpu_serial = cpu.split(' ')[0] + ' ' + cpu.split(' ')[1] + ' ' + cpu.split(' ')[2]
    category_id = 1
    print(id.split("""")[1], cpu, speed, videocard, ram_type, ram, hd_type, hd, itbox_price, rozetka_price, citrus_price, allo_price,
          stylus_price, category_id, cpu_serial, itbox_com_count, rozetka_com_count, citrus_com_count, allo_com_count, stylus_com_count)
    sql = "INSERT INTO FoxApp_products (id, cpu, speed, videocard, ram_type, ram, hd_type, hd, " \
          "diagonal, main_cam, back_cam, front_cam, color, os, price_itbox, price_rozetka, " \
          "price_citrus, price_allo, price_stylus, category_id, model_id, cpu_serial, itbox_com_count, rozetka_com_count, citrus_com_count, allo_com_count, stylus_com_count) " \
          "VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
    val = (id.split("""")[1], cpu, float(speed), videocard, ram_type, int(ram), hd_type, hd, DEFAULT, DEFAULT,
           DEFAULT, DEFAULT, DEFAULT, DEFAULT, itbox_price, rozetka_price,
           citrus_price, allo_price, stylus_price, category_id, id.split("""")[1], cpu_serial, itbox_com_count, rozetka_com_count, citrus_com_count, allo_com_count, stylus_com_count)
    print(val)
    cursor.execute(sql, val)
    mydb.commit()
```

```
def laptop(id):
    DEFAULT = 0
    id = """ + str(id) + """
    query = "select link_itbox, link_rozetka, link_citrus, link_allo, link_stylus  from FoxApp_contributor where id = " + id + ";"
    cursor.execute(query)
    link = cursor.fetchone()
    itbox = link[0]
    rozetka = link[1]
    citrus = link[2]
    allo = link[3]
    stylus = link[4]
    driver.get(itbox)
    driver.find_element_by_class_name("characteristic-toggle").click()
    elements1 = []
    ram_type = '-'
    info = driver.find_elements_by_tag_name("td")
    for elem in info:
        elements1.append(elem.text)
    i = 0
    for elem in elements1:
        i = i + 1
        if elem == "Процессор":
            proc = elements1[i]
            cpu = proc.split("(")[0]
            speed = proc.split("(")[1].replace(')', ').split("ГГц")[0]
```

```
            try:
                print(type(float(speed)))
            except:
                speed = proc.split("(")[1].replace(')', '').split("GHz")[0].split('-')[0]
        if elem == "Модель видеокарты" or elem == "Видеокарта":
            videocard = elements1[i]
        if elem == "Тип оперативной памяти":
            ram_type = elements1[i]
        if elem == "Объем оперативной памяти":
            ram = elements1[i].split("ГБ")[0]
        if elem == "Типы внутренних накопителей" or elem == "Оптический привод":
            hd_type = elements1[i]
        if elem == "Объем HDD" or elem == "Объем SSD":
            hd = elements1[i].split(" ")[0]
    try:
        itbox_price = driver.find_element_by_class_name("stuff-price").text.split("грн")[0].replace(' ', '')
        itbox_price = int(itbox_price)
    except:
        itbox_price = 0
    try:
        itbox_com_count = driver.find_element_by_class_name("header-comments-count").text.split(" ")[0]
    except:
        itbox_com_count = 0
    try:
        driver.get(rozetka)
        driver.implicitly_wait(5)
```

```
        rozetka_price = driver.find_element_by_class_name("product-
prices__big").text.split("₴")[0].replace(' ', '')
        rozetka_price = int(rozetka_price)
    except:
        rozetka_price = 0
    try:
        rozetka_com_count = driver.find_element_by_class_name("product-tabs__link-
text").text
    except:
        rozetka_com_count = 0
    try:
        driver.get(citrus)
        time.sleep(1)
        driver.implicitly_wait(5)
        citrus_price =
driver.find_element_by_xpath("//div[@class='price']").text.split("₴")[0].replace(' ', '')
        citrus_price = int(citrus_price)
    except:
        citrus_price = 0
    try:
        citrus_com_count = driver.find_elements_by_class_name("el-tabs__item")[-
2].text.split(" ")[1]
        if type(int(citrus_com_count)) != int:
            citrus_com_count = 0
        else:
            citrus_com_count = citrus_com_count
    except:
        citrus_com_count = 0
    try:
```

```
        driver.get(allo)
        driver.implicitly_wait(5)
        allo_price =
driver.find_element_by_xpath("//meta[@itemprop='price']").get_attribute("content").
replace(' ', '')
        allo_price = int(allo_price)
    except:
        allo_price = 0
    try:
        allo_com_count =
driver.find_element_by_xpath("//a[@id='discussionTab']").text.split(" ")[3]
    except:
        allo_com_count = 0
    try:
        driver.get(stylus)
        driver.implicitly_wait(5)
        stylus_price = driver.find_element_by_class_name("regular-
price").text.split("грн")[0].replace(' ', '')
        stylus_price = int(stylus_price)
    except:
        stylus_price = 0
    try:
        driver.maximize_window()
        stylus_com_count = driver.find_element_by_class_name("count-of-
reviews").text.split(" ")[0]
    except:
        stylus_com_count = 0
    if int(hd) < 10:
        hd = int(hd)*1000
    cpu_serial = cpu.split(' ')[0] + ' ' + cpu.split(' ')[1] + ' ' + cpu.split(' ')[2]
```

```
    category_id = 2

    print(id.split("""")[1], cpu, speed, videocard, ram_type, ram, hd_type, hd,
itbox_price, rozetka_price, citrus_price, allo_price,

        stylus_price, category_id, cpu_serial, itbox_com_count, rozetka_com_count,
citrus_com_count, allo_com_count, stylus_com_count)

    sql = "INSERT INTO FoxApp_products (id, cpu, speed, videocard, ram_type, ram,
hd_type, hd, " \

        "diagonal, main_cam, back_cam, front_cam, color, os, price_itbox,
price_rozetka, " \

        "price_citrus, price_allo, price_stylus, category_id, model_id, cpu_serial,
itbox_com_count, rozetka_com_count, citrus_com_count, allo_com_count,
stylus_com_count) " \

        "VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s,
%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"

    val = (id.split("""")[1], cpu, float(speed), videocard, ram_type, int(ram), hd_type, hd,
DEFAULT, DEFAULT,

        DEFAULT, DEFAULT, DEFAULT, DEFAULT, itbox_price, rozetka_price,

        citrus_price, allo_price, stylus_price, category_id, id.split("""")[1], cpu_serial,
itbox_com_count, rozetka_com_count, citrus_com_count, allo_com_count,
stylus_com_count)

    print(val)

    cursor.execute(sql, val)

    mydb.commit()



def smartphone(id):

    DEFAULT = 0

    id = """ + str(id) + """

    query = "select link_itbox, link_rozetka, link_citrus, link_allo, link_stylus  from
FoxApp_contributor where id = " + id + ";"

    cursor.execute(query)

    link = cursor.fetchone()
```

```
itbox = link[0]
rozetka = link[1]
citrus = link[2]
allo = link[3]
stylus = link[4]
driver.get(itbox)
driver.find_element_by_class_name("characteristic-toggle").click()
elements1 = []
info = driver.find_elements_by_tag_name("td")
for elem in info:
    elements1.append(elem.text)
i = 0
for elem in elements1:
    i = i + 1
    if elem == "Диагональ экрана":
        diagonal = elements1[i].split("")[0]
    if elem == "Процессор":
        cpu = elements1[i]
    if elem == "Оперативная память":
        ram = elements1[i].split("Gb")[0]
    if elem == "Основная камера":
        back_cam = elements1[i]
        main_cam = back_cam.split(" ")[0]
    if elem == "Фронтальная камера":
        front_cam = elements1[i]
    if elem == "Операционная система":
        os = elements1[i]
    if elem == "Цвет":
```

```
            color = elements1[i]
    try:
        itbox_price = driver.find_element_by_class_name("stuff-price").text.split("грн")[0].replace(' ', '')
        itbox_price = int(itbox_price)
    except:
        itbox_price = 0
    try:
        itbox_com_count = driver.find_element_by_class_name("header-comments-count").text.split(" ")[0]
    except:
        itbox_com_count = 0
    try:
        driver.get(rozetka)
        driver.implicitly_wait(5)
        rozetka_price = driver.find_element_by_class_name("product-prices__big").text.split("₴")[0].replace(' ', '')
        rozetka_price = int(rozetka_price)
    except:
        rozetka_price = 0
    try:
        rozetka_com_count = driver.find_element_by_class_name("product-tabs__link-text").text
    except:
        rozetka_com_count = 0
    try:
        driver.get(citrus)
        time.sleep(1)
        driver.implicitly_wait(5)
```

```
        citrus_price = driver.find_element_by_xpath("//div[@class='price']").text.split("₴")[0].replace(' ', '')
        citrus_price = int(citrus_price)
    except:
        citrus_price = 0
    try:
        citrus_com_count = driver.find_elements_by_class_name("el-tabs__item")[-2].text.split(" ")[1]
        if type(int(citrus_com_count)) != int:
            citrus_com_count = 0
        else:
            citrus_com_count = citrus_com_count
    except:
        citrus_com_count = 0
    try:
        driver.get(allo)
        driver.implicitly_wait(5)
        allo_price = driver.find_element_by_xpath("//meta[@itemprop='price']").get_attribute("content").replace(' ', '')
        allo_price = int(allo_price)
    except:
        allo_price = 0
    try:
        allo_com_count = driver.find_element_by_xpath("//a[@id='discussionTab']").text.split(" ")[3]
    except:
        allo_com_count = 0
    try:
        driver.get(stylus)
```

```
        driver.implicitly_wait(5)

        stylus_price = driver.find_element_by_class_name("regular-
price").text.split("грн")[0].replace(' ', '')

        stylus_price = int(stylus_price)

    except:

        stylus_price = 0

    try:

        driver.maximize_window()

        stylus_com_count = driver.find_element_by_class_name("count-of-
reviews").text.split(" ")[0]

    except:

        stylus_com_count = 0

    # cpu_serial = cpu.split(' ')[0] + ' ' + cpu.split(' ')[1] + ' ' + cpu.split(' ')[2]

    category_id = 3

    print(id.split("'")[1], cpu, float(diagonal), int(main_cam), back_cam, front_cam,
ram, color, os, itbox_price,

          rozetka_price, citrus_price, allo_price, stylus_price, category_id, cpu,
itbox_com_count, rozetka_com_count, citrus_com_count, allo_com_count,
stylus_com_count)

    sql = "INSERT INTO FoxApp_products (id, cpu, speed, videocard, ram_type, ram,
hd_type, hd, " \

          "diagonal, main_cam, back_cam, front_cam, color, os, price_itbox,
price_rozetka, " \

          "price_citrus, price_allo, price_stylus, category_id, model_id, cpu_serial,
itbox_com_count, rozetka_com_count, citrus_com_count, allo_com_count,
stylus_com_count) " \

          "VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s,
%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"

    val = (id.split("'")[1], cpu, DEFAULT, DEFAULT, DEFAULT, ram, DEFAULT,
DEFAULT, float(diagonal),

           int(main_cam), back_cam, front_cam, color, os, itbox_price,
```

```
        rozetka_price, citrus_price, allo_price, stylus_price, category_id,
id.split("")[1], cpu, itbox_com_count, rozetka_com_count, citrus_com_count,
allo_com_count, stylus_com_count)
    print(val)
    cursor.execute(sql, val)
    mydb.commit()


if __name__ == '__main__':
    try:
        cycle(1)
    except:
        pass
```

Файл admin.py:

```
from django.contrib import admin

# Register your models here.
from .models import Category, Contributor, Products


admin.site.register(Category)
admin.site.register(Contributor)
admin.site.register(Products)
```

Файл urls.py:

```
from django.urls import path
from . import views
from .views import prod_card, product_page, filtered, search
from .models import *

urlpatterns = [
    path('', views.index, name='index'),
    path('category/<category_slug>', prod_card, name='category'),
```

```
    path('search', search, name='search'),
    path('filtered/', filtered, name='filtered'),
    path('products/<product_slug>', product_page, name='products'),
]
```

Файл views.py

```
from django.shortcuts import render
from django.http import HttpResponse, JsonResponse
from .models import Contributor, Products, Category
from .models import Category as CategoryModel
from django.db.models import F, Q
from django.db.models import Avg, Count, Max, Min
# Create your views here.


def index(request):
    contributor = Contributor.objects.all()
    products = Products.objects.all()
    all_category = CategoryModel.objects.all()
    context = {
        'Contributor': contributor,
        'Products': products,
        'All_categories': all_category
    }
    return render(request, "index.html", context)


def prod_card(request, category_slug):
    contributor = Contributor.objects.all()
    products = Products.objects.all()
    all_category = CategoryModel.objects.all()
    category = CategoryModel.objects.filter(slug=category_slug)
    cat = (str(request).split('/')[2].split("'")[0])
    makers =
Contributor.objects.filter(category__slug=category_slug).values_list('maker',
flat=True).distinct()
    videocard =
Products.objects.filter(category__slug=category_slug).values_list('videocard',
flat=True).distinct()
    ram = Products.objects.filter(category__slug=category_slug).values_list('ram',
flat=True).distinct()
    cpu_serial = Products.objects.filter(category__slug=cat).values_list('cpu_serial',
flat=True).distinct()
```

```
    diagonal = Products.objects.filter(category__slug=cat).values_list('diagonal',
flat=True).distinct()
    context = {
        'Contributor': contributor,
        'Products': products,
        'Categories': category,
        'cat': cat,
        'Makers': makers,
        'Ram': ram,
        'Videocard': videocard,
        'Cpu_serial': cpu_serial,
        'Diagonal': diagonal,
        'All_categories': all_category
    }
    return render(request, "prod-card.html", context)


def product_page(request, product_slug):
    contributor = Contributor.objects.filter(model=product_slug)
    products = Products.objects.all()
    category = CategoryModel.objects.all()
    all_category = CategoryModel.objects.all()
    context = {
        'Contributor': contributor,
        'Products': products,
        'Categories': category,
        'All_categories': all_category
    }
    return render(request, "product_page.html", context)


def filtered(request):
    cat = request.GET.get("category")
    contributor = Contributor.objects.all()
    category = CategoryModel.objects.filter(pk=cat)
    makers = Contributor.objects.filter(category__pk=cat).values_list('maker',
flat=True).distinct()
    videocard = Products.objects.filter(category__pk=cat).values_list('videocard',
flat=True).distinct()
    diagonal = Products.objects.filter(category__pk=cat).values_list('diagonal',
flat=True).distinct()
    ram = Products.objects.filter(category__pk=cat).values_list('ram',
flat=True).distinct()
    cpu_serial = Products.objects.filter(category__pk=cat).values_list('cpu_serial',
flat=True).distinct()
```

```
all_category = CategoryModel.objects.all()
# maker
if request.GET.getlist("maker") == ['All']:
    maker = Products.objects.all().values_list('model', flat=True).distinct()
elif request.GET.getlist("maker"):
    maker = Contributor.objects.filter(maker=request.GET.get('maker'))
else:
    maker = Products.objects.all().values_list('model', flat=True).distinct()
# checkbox ram
if request.GET.getlist("ram"):
    ram1 = request.GET.getlist('ram')
else:
    ram1 = ram
# checkbox cpu
if request.GET.getlist("cpu"):
    cpu_serial1 = request.GET.getlist('cpu')
else:
    cpu_serial1 = cpu_serial
# checkbox videocard
if request.GET.getlist("videocard"):
    videocard1 = request.GET.getlist('videocard')
else:
    videocard1 = videocard
# checkbox diagonal
if request.GET.getlist("diagonal"):
    diagonal1 = request.GET.getlist('diagonal')
else:
    diagonal1 = diagonal
# --------------------------
product = Products.objects.filter(
    Q(ram__in=ram1),
    Q(cpu_serial__in=cpu_serial1),
    Q(videocard__in=videocard1),
    Q(diagonal__in=diagonal1),
    Q(model__in=maker),
)
context = {
    'Contributor': contributor,
    'Products': product,
    'Categories': category,
    'cat': cat,
    'Makers': makers,
    'Ram': ram,
    'Videocard': videocard,
    'Cpu_serial': cpu_serial,
```

```
        'Diagonal': diagonal,
        'All_categories': all_category
    }
    return render(request, "prod-card.html", context)

def search(request):
    contributor =
Contributor.objects.filter(model__icontains=request.GET.get("search"))
    cat = request.GET.get("search")
    products = Products.objects.all()
    print(contributor)
    context = {
        'Products': products,
        'Contributor': contributor,
        'Cat': cat,
    }
    return render(request, "search.html", context)
```

Файл models.py:

```
from django.db import models
from django.urls import reverse
from django.utils.text import slugify

# Create your models here.


class Category(models.Model):
    name = models.CharField(max_length=200)
    slug = models.SlugField(blank=True)
    img = models.ImageField(upload_to='./static/Resource')

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('category', kwargs={'category_slug': self.slug})


class Contributor(models.Model):
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    maker = models.CharField(max_length=100)
    model = models.CharField(max_length=200)
    is_active = models.BooleanField
```

```
    link_itbox = models.CharField(max_length=250)
    link_rozetka = models.CharField(max_length=250)
    link_citrus = models.CharField(max_length=250)
    link_allo = models.CharField(max_length=250)
    link_stylus = models.CharField(max_length=250)
    img1 = models.CharField(max_length=250)
    img2 = models.CharField(max_length=250)
    img3 = models.CharField(max_length=250)

    def __str__(self):
        return self.model

    def get_absolute_url(self):
        return reverse('products', kwargs={'product_slug': self.model})


class Products(models.Model):
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    model = models.ForeignKey(Contributor, on_delete=models.CASCADE)
    cpu = models.CharField(max_length=100, blank=True)
    cpu_serial = models.CharField(max_length=100, blank=True)
    speed = models.FloatField(max_length=25, blank=True)
    videocard = models.CharField(max_length=100, blank=True)
    ram_type = models.CharField(max_length=100, blank=True)
    ram = models.IntegerField(blank=True)
    hd_type = models.CharField(max_length=100, blank=True)
    hd = models.IntegerField(blank=True)
    diagonal = models.FloatField(max_length=25, blank=True)
    main_cam = models.IntegerField(blank=True)
    back_cam = models.CharField(max_length=100, blank=True)
    front_cam = models.CharField(max_length=100, blank=True)
    color = models.CharField(max_length=100, blank=True)
    os = models.CharField(max_length=100, blank=True)
    price_itbox = models.IntegerField(blank=True)
    price_rozetka = models.IntegerField(blank=True)
    price_citrus = models.IntegerField(blank=True)
    price_allo = models.IntegerField(blank=True)
    price_stylus = models.IntegerField(blank=True)
    itbox_com_count = models.IntegerField(blank=True)
    rozetka_com_count = models.IntegerField(blank=True)
    citrus_com_count = models.IntegerField(blank=True)
    allo_com_count = models.IntegerField(blank=True)
    stylus_com_count = models.IntegerField(blank=True)

    def __int__(self):
```

```
        return self.model

    def get_absolute_url(self):
        return reverse('products', kwargs={'product_slug': self.model})


    def minimal(self):
        prices = [self.price_itbox, self.price_rozetka, self.price_citrus, self.price_allo,
self.price_stylus]
        prices = [x for x in prices if x != 0]
        return min(prices)

    def maximal(self):
        prices = [self.price_itbox, self.price_rozetka, self.price_citrus, self.price_allo,
self.price_stylus]
        return max(prices)

    def best_price(self):
        com = [self.itbox_com_count, self.rozetka_com_count, self.citrus_com_count,
self.allo_com_count,
               self.stylus_com_count]
        prices = [self.price_itbox * com[0], self.price_rozetka * com[1], self.price_citrus
* com[2],
                  self.price_allo * com[3], self.price_stylus * com[4]]
        count = 0
        for i in range(len(prices)):
            if prices[i] == 0:
                del com[i - count]
                count = count + 1
        price = (sum(prices) / sum(com))
        round(price, 0)
        print(price)
        return price
```

Додаток Г

(обов’язковий)

**ІЛЮСТРАТИВНА ЧАСТИНА**

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ЦІН ІНТЕРНЕТ-МАГАЗИНІВ ДЛЯ ЕЛЕКТРОННИХ ТОВАРІВ

Виконав: студент гр. 2ІСТ-20м

_____ Ющук І. О.

«___» _____ 2021 р.

Керівник: к.т.н., доцент

_____ Козачко О. М.

«___» _____ 2021 р.

Нормоконтроль: к.т.н., доцент

_____ Жуков С. О.

«___» _____ 2021 р.

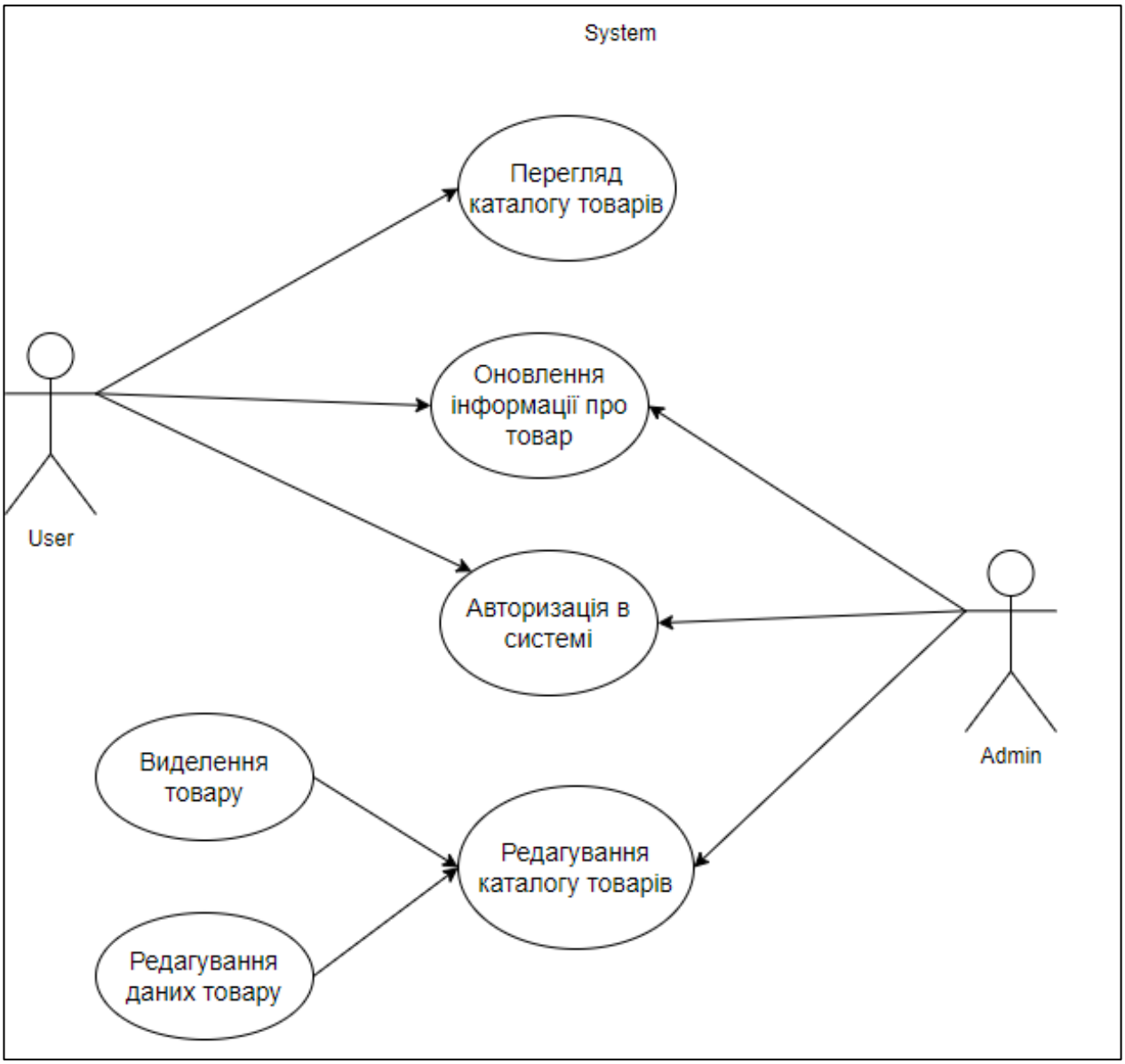



Рисунок Д.1 – UML діаграма прецедентів

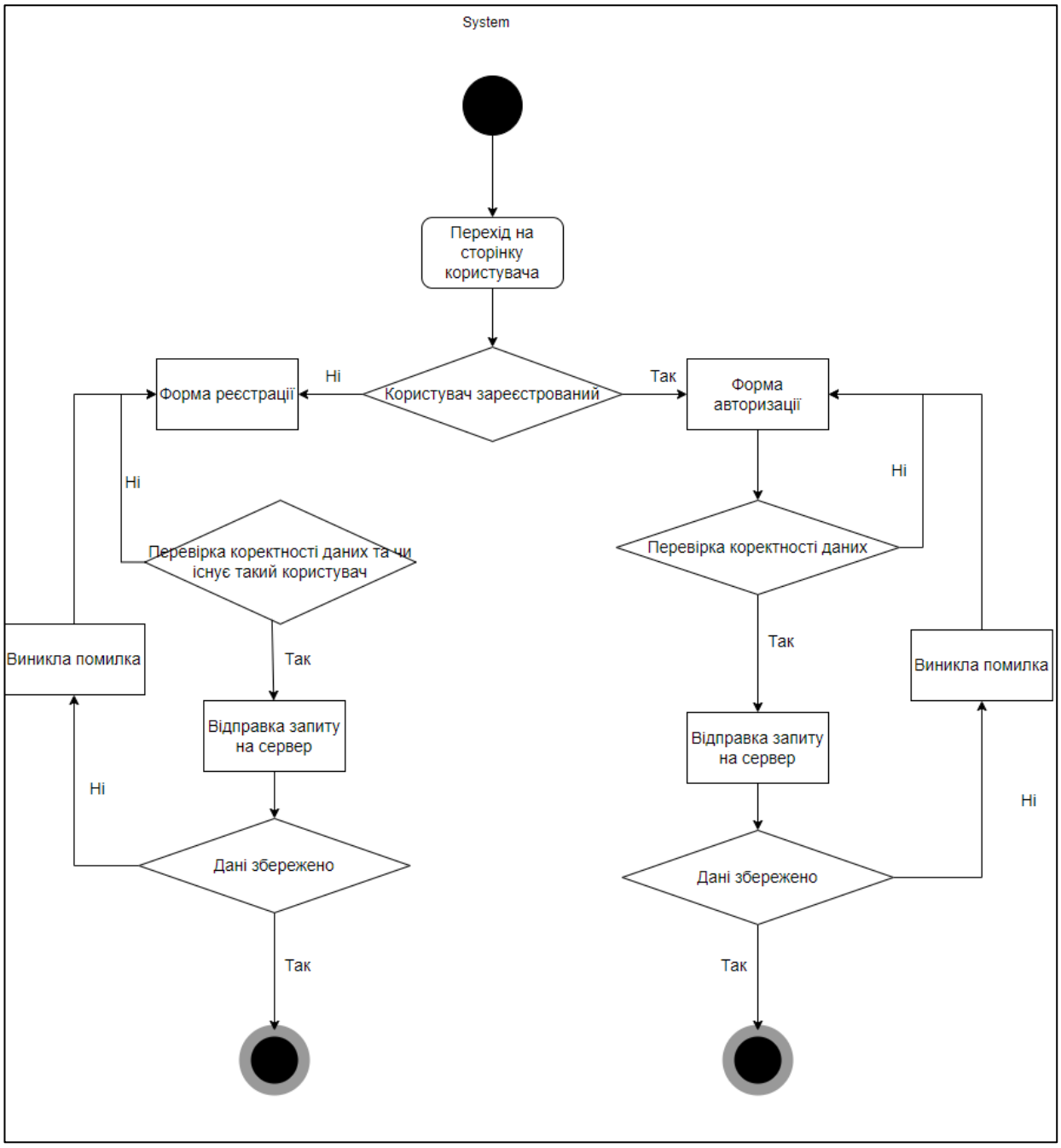



Рисунок Д.2 – UML діаграма активності додавання даних

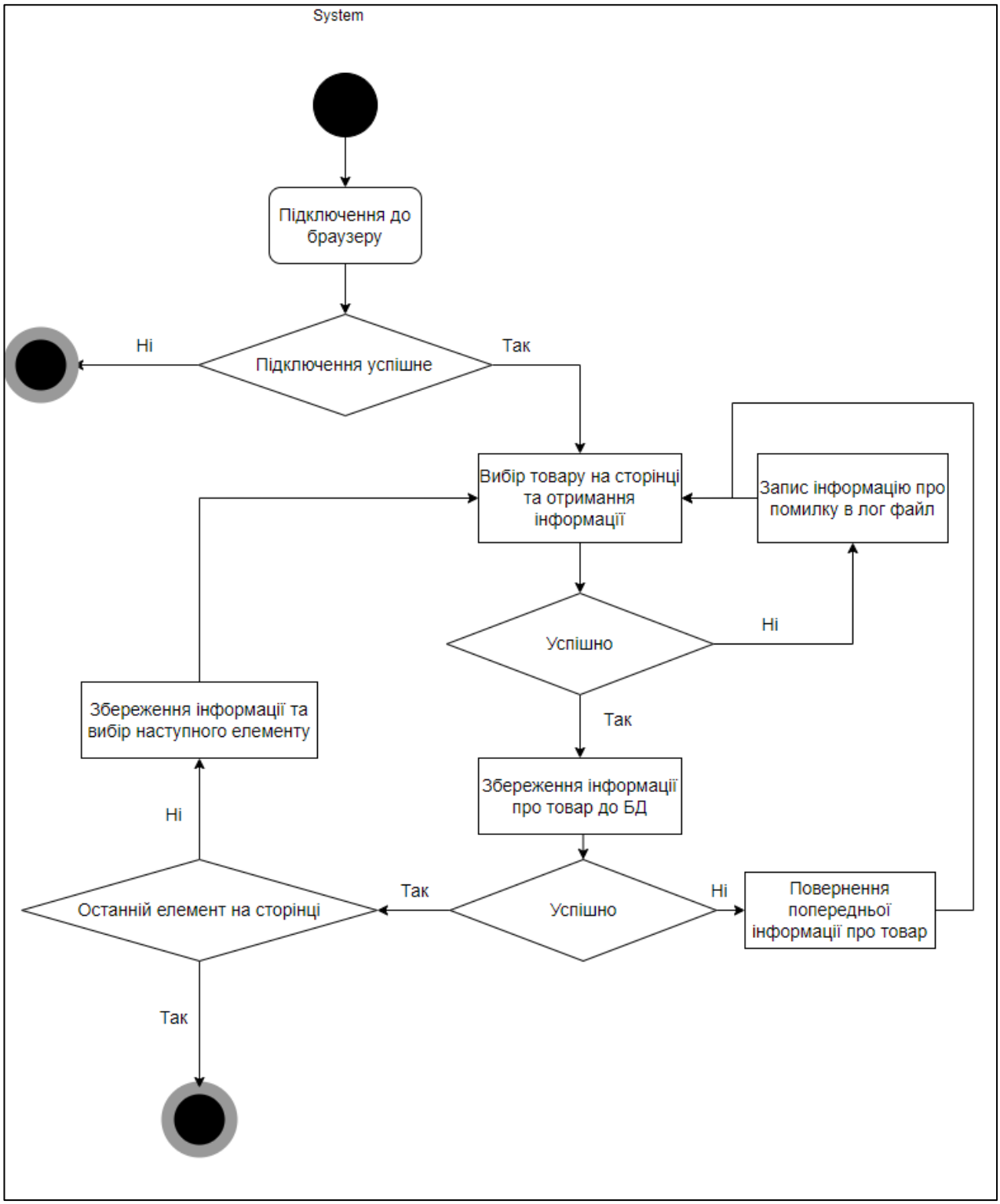



Рисунок Д.3 – UML діаграма активності роботи скрапера

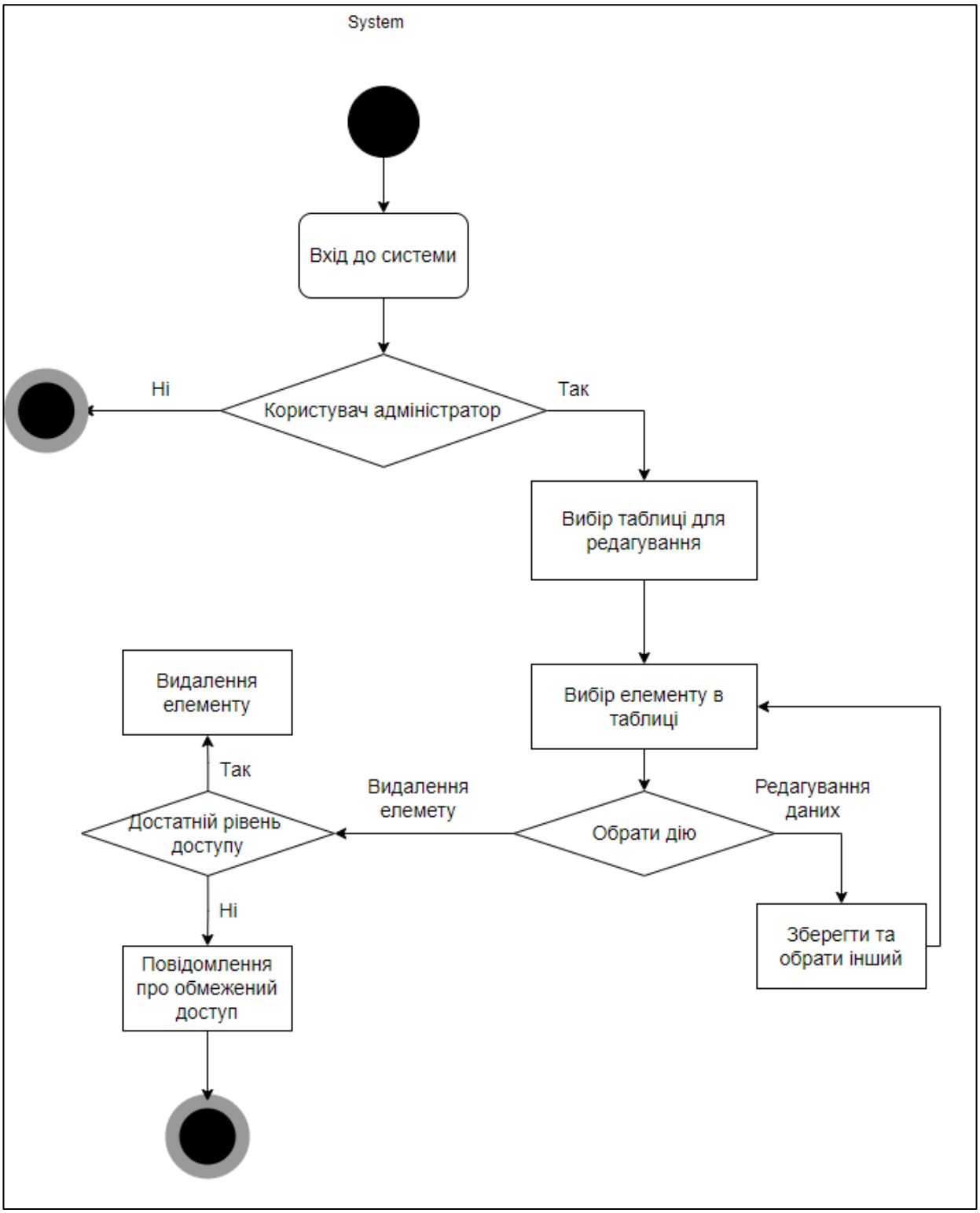



Рисунок Д.4 – UML діаграма активності редагування даних

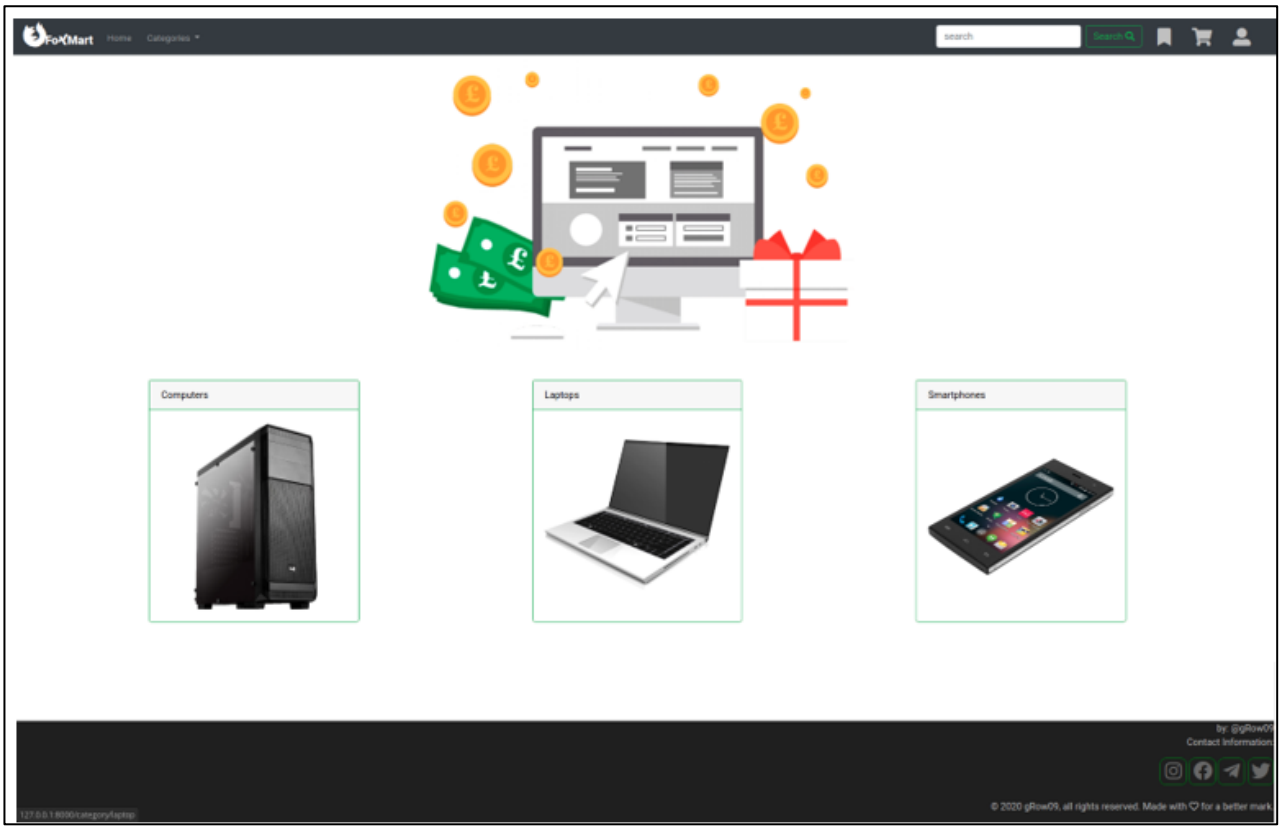



Рисунок Д.5 – Головна сторінка веб-додатку

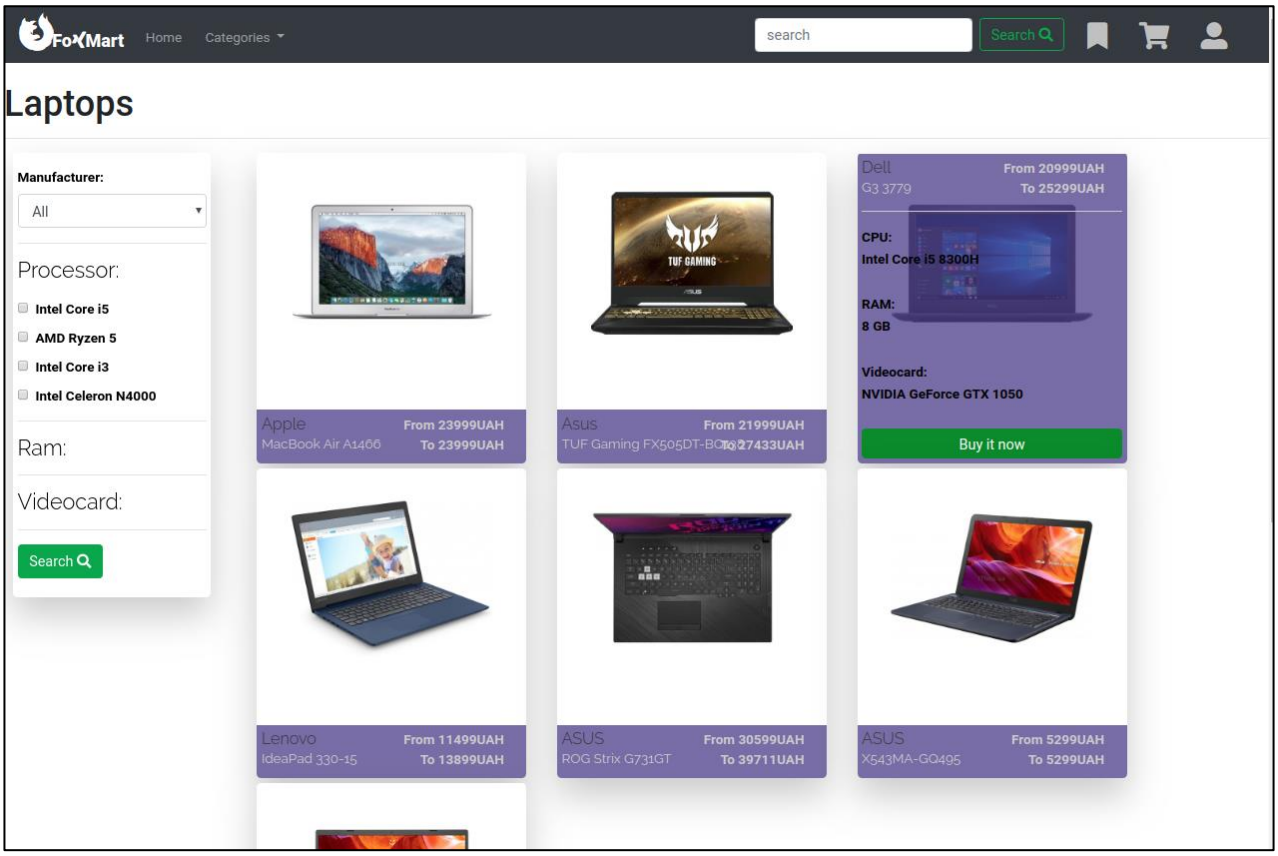



Рисунок Д.6 – Каталог товарів

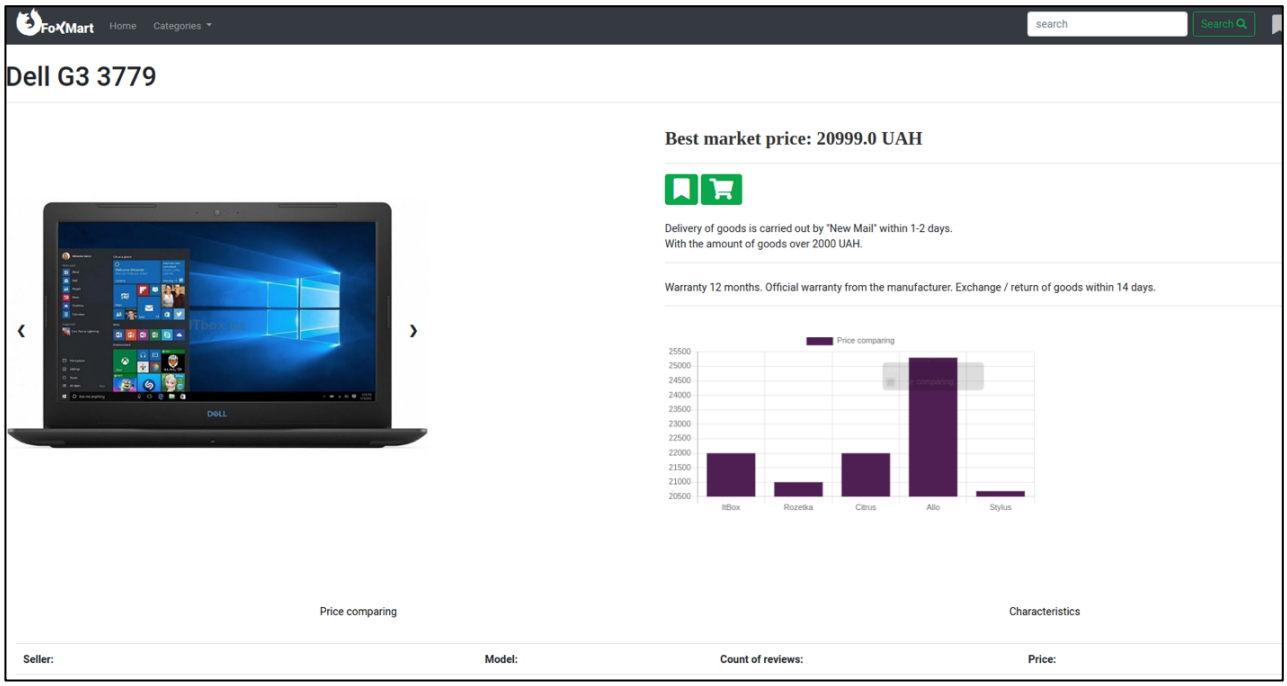



Рисунок Д.7 – Сторінка детальної інйормації товару