МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

«До захисту допущена»

Зав. кафедри прикладної математики

д.т.н., професор Турбал Ю.В.

«____» _____ 20_ р.

КВАЛІФІКАЦІЙНА РОБОТА

“ ПРОЕКТУВАННЯ ТА РОЗРОБКА WEB-ДОДАТОКУ ДЛЯ ПРОДАЖУ ПИТНОЇ ВОДИ ”

Виконав:

Студент навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки

Група КН41 Паридубець Арсен Ігорович _____

підпис

Керівник: доц. Гладун Л.В. _____

підпис

Рівне-2023

Зміст

Національний університет водного господарства та природокористування

(повне найменування вищого навчального закладу)

НН інститут *автоматики, кібернетики та обчислювальної техніки*

Кафедра *Комп'ютерних наук та прикладної математики*

Освітньо кваліфікаційний рівень *Бакалавр*

Спеціальність *112 Комп'ютерні науки*

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

_____

"____"_____20___року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Паридубцю Арсену Ігоровичу

1. **Тема роботи** «***Проектування та розробка WEB-додатоку для продажу питної води***»

Керівник роботи: Гладун Любомир Володимирович, к.ф.-м. н., доцент

затверджені наказом вищого навчального закладу № С449 від 19.04.2023

2. **Термін здачі студентом закінченої роботи:** 22 червня 2023 року

3. **Вихідні дані до роботи:** наукові джерела з питань моделювання, методів обчислень; результати власних досліджень.

4. **Зміст розрахунково-пояснювальної записки:** дослідження сучасних підходів до проектування WEB-додатків.

5. **Перелік графічного матеріалу:** мультимедійна презентація.

6. **Консультанти розділів роботи**

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

Розділ 1	*доц. Гладун Л.В.*	*3.10.22*	*3.10.22*
Розділ 2	*доц. Гладун Л.В.*	*18.11.22*	*18.11.22*
Розділ 3	*доц. Гладун Л.В.*	*14.01.23*	*14.01.23*
Розділ 4	*доц. Гладун Л.В.*	*10.03.23*	*10.03.23*

7. **Дата видачі завдання** *01 жовтня 2022 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	**Назва етапів кваліфікаційної роботи**	**Строк виконання етапів роботи**	**Примітка**
1	*Вивчення літератури за обраною тематикою*	*3.10.22 -14.10.22*	*виконав*
2	*Формулювання завдання*	*15.10.22-28.10.22*	*виконав*
3	*Розробка алгоритму розв'язку поставленого завдання*	*28.10.22-14.01.22*	*виконав*
4	*Здійснення програмної реалізації*	*15.01.22-15.02.23*	*виконав*
5	*Тестування програми*	*16.02.23-9.04.23*	*виконав*
6	*Аналіз отриманих результатів*	*10.04.23-23.04.23*	*виконав*
7	*Загальні висновки до роботи*	*24.04.23-5.05.23*	*виконав*
8	*Підготовка звіту кваліфікаційної роботи*	*13.05.23-23.05.23*	*виконав*
9	*Підготовка мультимедійної презентації*	*23.05.23-10.06.23*	*виконав*
10	*Підготовка до захисту*	*11.06.23-23.06.23*	*виконав*

Студент _____ _____

(підпис) (прізвище та ініціали)

Керівник роботи _____ _____

(підпис) (прізвище та ініціали)

Вступ

У світі швидко розвивається електронна комерція, і продаж питної води через Інтернет стає все більш популярним. Без сумніву, вода - це основа життя людини. Питна вода є життєво важливим ресурсом для людства, і забезпечення її доступністю та якістю є критично важливою задачею. Відтак, створення веб-сайту для продажу питної води може виявитися винятково корисним та ефективним способом задовольнити потреби споживачів у воді.

Ця робота присвячена детальному аналізу та оцінці впливу сайтів продажу питної води на сучасну торгівлю та споживання води. Основною метою є розкриття потенціалу та переваг інтернет-платформ для забезпечення доступності високоякісної питної води, а також дослідження впливу таких сайтів на споживачів та суспільство в цілому.

У рамках дослідження будуть розглянуті наступні аспекти:

- Аналіз сучасного стану ринку питної води та виявлення тенденцій її споживання.
- Вивчення та порівняння різних підходів до продажу питної води через Інтернет.
- Оцінка переваг та обмежень веб-сайтів продажу питної води для споживачів, виробників та середовища.
- Дослідження впливу маркетингових стратегій та функціональності веб-сайтів на споживачів та їх сприйняття продукту.
- Визначення ключових чинників успіху та факторів ризику для сайтів продажу питної води.
- Застосування наукового методу, а також аналіз доступних статистичних даних та наукових досліджень дозволить підтвердити гіпотези та надати вагомі висновки та рекомендації для розвитку та оптимізації веб-сайтів продажу питної води.

Вивчення та розуміння цих аспектів не тільки допоможуть підприємцям та фахівцям з водопостачання покращити ефективність своїх бізнес-стратегій, але

й сприятимуть зростанню свідомості споживачів щодо важливості якості та доступності питної води. Отже, тема роботи є дуже актуальною.

На даний момент високорівнева мова програмування Python, зокрема його фреймворк Django є дуже популярною при розробці додатків. Великий набір бібліотек, інструментів, шаблонів, компактність коду та структурованість – ось причина популярності цієї мови програмування. А тому при розробці WEB-додатку будемо використовувати саме Django.

Об’єкт дослідження: сучасні технології розробки WEB-додатків

Предмет дослідження: методи розробки сайтів електронної комерції на основі фреймворку Django на прикладі WEB-додатку для продажу питної води.

Мета роботи: здійснити вивчення фреймворку Django та розробити WEB-додаток для продажу питної води.

Розділ 1. Огляд існуючих методів та розробок

1.1 Аналіз предметної області

Інтернет-магазин – це сайт, що містить докладний каталог товарів з описом та зображенням. Основна відмінність від звичайного інтернет-каталогу полягає в тому, що товари, представлені в інтернет-магазині, можна не тільки побачити, але й замовити не виходячи з дому. Для всіх інтернет-магазинів властивий певний обов'язковий набір елементів, таких як: Спеціалізований каталог з підрозділами, в яких представлені всі товари. Зовнішній вигляд каталогу може бути різним – дерево, списки меню, що випадають або вкладені. Система реєстрації користувача, яка створює для кожного нового клієнта його власний «кошик», в який можна «покласти» вибраний товар та згодом замовити. У міру пересування клієнта каталогом система також відстежує переваги клієнта, на основі яких у майбутньому може будуватися не лише асортимент магазину, а й структура видачі супутньої інформації каталогу. Система оплати товару: покупцю пропонується використовувати різні способи оплати – кредитні картки, електронні гроші, оплата готівкою (кур'єру або при отриманні поштою). Система доставки товару: тут також широкий вибір можливостей: пересилання електронною поштою (програмне забезпечення), доставка кур'єрською службою, звичайна пошта. Однак, незважаючи на загальні риси, Інтернет-магазини все ж таки відрізняються один від одного. Власник кожного магазину прагне зробити свій сайт максимально зручним для відвідувача, удосконалюючи систему замовлення та способи переходу від одного розділу до іншого. Як і у звичайному магазині, в Інтернет-магазині можуть влаштовуватись розпродажі та знижки. Головна відмінність Інтернет-магазину від звичайного магазину – це не тільки можливість купити щось , не виходячи з дому чи офісу, а також можливість витратити менші кошти. За рахунок чого виходить так, що покупка в Інтернет-магазині стає кращою. Для створення Інтернет-магазину не потрібно купувати або орендувати приміщення під магазин, ремонтувати та оформляти його, наймати штат продавців та охорону – а значить знижуються початкові витрати, а з ними і ціна товару. Тепер навіть з урахуванням доставки товар коштуватиме дешевше лише тому, що клієнту не доведеться платити ту частину ціни, за

допомогою якої продавець намагається відшкодувати витрати на щомісячне утримання магазину та штату співробітників. Інтернет магазин має такі переваги: допомагає швидко зорієнтуватися в асортименті та знайти потрібний товар чи послугу (за тематикою, назвою, ціною тощо); є можливість розглянути товар «з усіх боків», порівняти його характеристики, ціну, зовнішній вигляд з іншими товарами; є можливість переглянути інформацію про знижки, подарунки та подібні заходи; можна розрахувати точну вартість замовлення; можна відібрати товар в кошик, оформити замовлення онлайн, оформити доставку додому; – підтримка контактів продавець-покупець, наприклад, перегляд історії раніше зроблених замовлень, переглядати інформації на поточне замовлення.

1.2 Аналіз популярних інтернет магазинів

Перш, ніж проектувати веб-додаток прогнозування та продажу товарів, необхідно проаналізувати та виявити загальні принципи побудови та інструменти роботи вже існуючих та успішних веб-додатків такої ж спрямованості. Це важливо насамперед для виявлення архітектури та розуміння принципу роботи веб-додатка, що розробляється.

Розглінемо принципи побудови інтернет-магазину «DNS» Відразу після заходу на сайт магазину «DNS» він автоматично визначає ваше місто, так само є можливість змінити місто за необхідності. Особливістю даного магазину є можливість порівняння товарів за характеристиками, при цьому достатньо додати товари однієї категорії для порівняння спеціальною кнопкою на панелі товару. Якщо покупець знайшов необхідний товар, він може розпочати формування замовлення. Поруч із описом товару він завжди знайде або текст «Додати до кошика», або кнопку «Додати товар до кошика», натиснувши на які його товар потрапляє до кошика. У верхньому правому куті вікна браузера відображається стан кошика клієнта. Натиснувши на текст «Кошик покупця», клієнт потрапляє до кошика, де показано всі набрані товари. Нижче кошики представлені товари, які часто купуються у місці з вибраними вами товарами. Підбір таких товарів здійснюється за допомогою асоціативних зв'язків. Щоб приступити до оформлення замовлення, потрібно зареєструватися, якщо це не

зроблено раніше. Далі потрібно вибирати спосіб доставки товару із запропонованого списку (варіанти доставки залежать від місця, куди потрібно відправити замовлення) та спосіб оплати, який вже залежить від обраного способу доставки. Після цього покупець перебуває на сторінці оформлення замовлення. Потрібно уважно переглянути всі параметри замовлення та, якщо все правильно, натиснути кнопку «Замовити». Далі можна переглянути зразки документів, які допоможуть правильно оплатити замовлення. Потрібно обов'язково отримати підтвердження електронною поштою про те, що замовлення прийнято. У надісланому листі будуть посилання на підтвердження замовлення або відмову від нього. Замовлення буде оброблено тільки після його підтвердження, тобто коли покупець натисне відповідне посилання в листі.

Проаналізувавши існуючі інтернет-магазини, можна виявити загальні структурні елементи та функції, які повинні бути присутніми у будь-якому веб-додатку, що здійснює продаж. Типовий варіант інтернет-магазину складається з таких функціональних елементів:

- каталог товарів;
- пошукова система;
- користувальницький кошик;
- реєстраційна форма;
- форма відправлення замовлення.

Каталог товарів є складною і багаторівневою структурою даних, яка повинна простим і зрозумілим способом проводити впорядкування товарів. Найпростіше такий каталог представити у вигляді дерева об'єктів, верхній рівень якого складається із списку розділів. Розділи можуть містити підрозділи чи посилання конкретний товар тощо. Таке впорядкування просто необхідне для зручного та швидкого пошуку та замовлення товарів.

Пошукова система є обов'язковим елементом динамічного каталогу та реалізується на стороні сервера. Незважаючи на те, що каталог забезпечує впорядкування та групування даних, пошукова система дає користувачеві можливість швидкого пошуку інформації, що особливо важливо в тому випадку, коли каталог є досить розгалуженою структурою даних з великою кількістю

розділів, підрозділів і товарів, користувач погано представляє в якому розділі може знаходитися товар, що його цікавить і чи є він у каталозі взагалі. Пошукова система в деяких випадках дозволяє значно скоротити кількість переходів між сторінками каталогу для доступу до інформації [1]. Особливість реалізації пошуку в Інтернеті полягає в тому, що тут відбувається вибірка всіх записів, які відповідають умовам запиту. У разі великої вибірки даних виведення результатів пошуку здійснюється посторінково для того, щоб відвідувачам не доводилося довго чекати завантаження всієї вибірки, яка може включати сотні, тисячі і більше записів. Як правило, відвідувачі не переглядають усі сторінки вибірки, обмежуючись двома чи трьома. Тому цей механізм пошуку у багатьох випадках працює вкрай повільно та неефективно. Однак він дозволяє здійснити вибірку однакових товарів від різних постачальників, порівняти їх параметри між собою та вибрати оптимальний варіант.

Користувальницький кошик є деяким масивом даних, який служить для зберігання замовленого користувачем товару.

Реєстраційна форма необхідна для введення персональних даних користувачів. Надалі ця інформація використовується для їхньої ідентифікації між сеансами роботи з інтернет-магазином. Ця інформація може зберігатись як на стороні сервера, так і на стороні клієнта.

Форма відправлення замовлення служить для введення контактної інформації замовника та відправлення її та замовлення на електронну скриньку організації [1]. Отримані дані будуть використовуватися для проектування та розробки архітектури та інтерфейсу веб-додатку.

Реєстраційна форма необхідна для введення персональних даних користувачів. Надалі ця інформація використовується для їхньої ідентифікації між сеансами роботи з інтернет-магазином. Ця інформація може зберігатись як на стороні сервера, так і на стороні клієнта.

1.3 Огляд веб-фреймворку Django

Оскільки Django народився в середовищі новин, він надає деякі засоби (такі як адміністративний інтерфейс), які добре підійдуть для контент-орієнтованих сайтів, які надають динамічну інформацію з бази даних.

Походження Django сформувало культуру його спільноти відкритого вихідного коду. Так як Django був отриманий з реального коду, а не був академічною розробкою або комерційним продуктом, він повністю сфокусований на вирішення проблем розробки, з якими стикалися його автори. В результаті Django постійно вдосконалюється. Засновники середовища мають свій інтерес у тому, щоб Django заощаджував їх час, створював додатки легкі в обслуговуванні та добре працював під навантаженням. За відсутності інших чинників, розробники мотивуються своїми власними егоїстичними бажаннями заощадити свій час і насолоджуватися своєю работой.[7]

Відмінні риси Джанго:

1) будь-який запит обробляється програмно та перенаправляється на свою адресу (url);

2) поділ контенту та подання за допомогою шаблонів;

3) абстрагування від низького рівня баз даних.

Розділ 2. Особливості розробки інтернет-магазину та задачі керування

2.1 Принципи розробки

Django-додаток описує безліч дій, можливих при взаємодії сервера і клієнта. Ці дії можуть містити відображення веб-сторінки, виконання запитів до бази даних, запис даних на диск та багато іншого. Django використовує паттерн MVC, тому структура програми буде виглядати так.

У файлі models.py описуються моделі даних у вигляді класів мовою Python. Дані будуть зберігатися в реляційній базі даних, проте завдяки технології ORM, дані використовуються як об'єкти моделей.

Інтерфейс є взаємозв'язок шаблонів Django. Структура сторінок є базовим успадкованим шаблоном, який завжди використовується і множиною шаблонів з різними сторінками, які замінюють один одного при переході з розділу в розділ. Це досягається завдяки системі успадкування шаблонів. Дані, які встановлюються в шаблони беруться з моделі даних.

У views.py містяться всі можливі керуючі функції. Вони контролюють, який шаблон слід використовувати і які дані та методи моделі слід використовувати. Вся логіка додатків та всі можливі підрахунки зосереджені у цих функціях. Діаграма компонентів представлена на Рис.2.1.

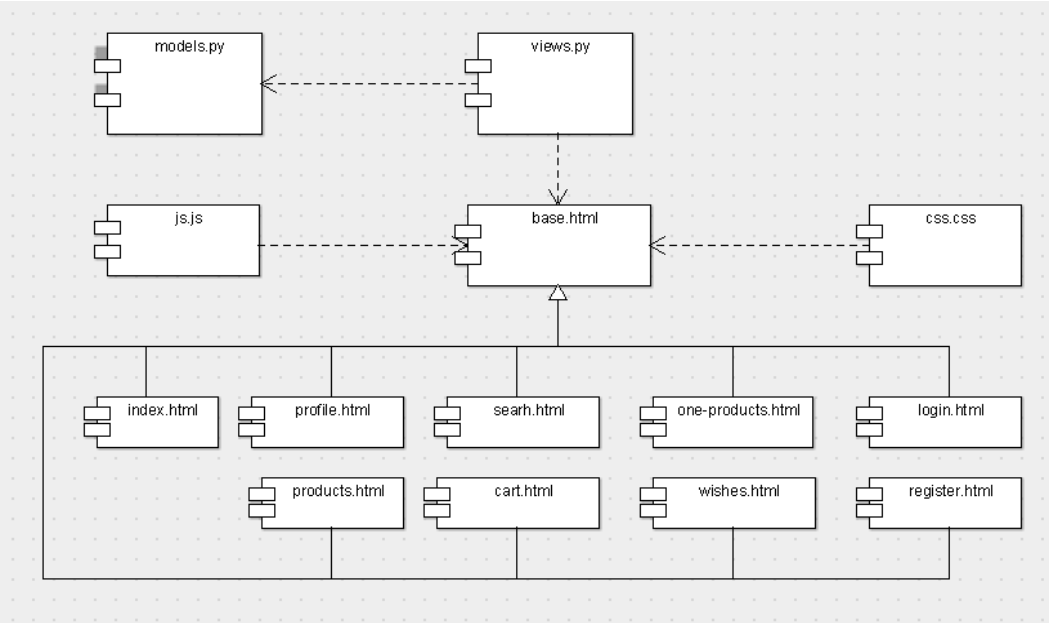

Рисунок 2.1 – Діаграма компонентів

Веб-додаток прогнозування та продажу товарів можна поділити на дві частини. Перша клієнтська частина відповідає за взаємодію з покупцями. Вона є інтернет-магазином і надає користувачам такі можливості.

Незареєстровані користувачі можуть переглядати каталог товарів (всі товари, новинки, лідери продажів), сортувати товари (за рейтингом, вартістю, переглядами, відгуками), користуватися вибіркою за характеристиками, пошуком товарів.

Пошук товарів здійснюється в такий спосіб. Спочатку шукається повний збіг пошукового запиту з назвою товару, якщо такий товар знайдено, то пошук завершується, якщо ні, пошуковий запит розбивається на окремі слова та шукаються входження цих слів у назву товару, чим більше входження, тим вірогідніше, що пошук завершиться успішно. Якщо входження недостатньо, то результати пошукового запиту виводяться можливі варіанти товарів.

Щоб придбати товари необхідно авторизуватися або зареєструватися на сайті. Після цього користувачі можуть додавати товари в кошик, де можна встановити потрібну кількість товару для покупки або видалити непотрібний та оформити покупку. Для оформлення замовлення слід заповнити форму реквізитами для оплати та підтвердити придбання поштою. У профілі є історія покупок, а також можливість змінити особисті дані або пароль. Якщо не вистачає грошей на покупку, можна додати товар, що сподобався, до списку бажань, щоб його не втратити. Діаграма діяльності для реєстрації користувача представлена на Рис.2.2.

Друга частина є адміністративною панеллю. Доступ до панелі дозволено лише користувачам із правами адміністратора. Адміністратор може створювати нових користувачів та розподіляти права доступу. За допомогою панелі можна заповнювати моделі даних необхідною інформацією, а також редагувати чи видаляти наявну. У розділі зі статистикою представлені графіки за обсягами продажу, виручки та відвідуваності магазину.

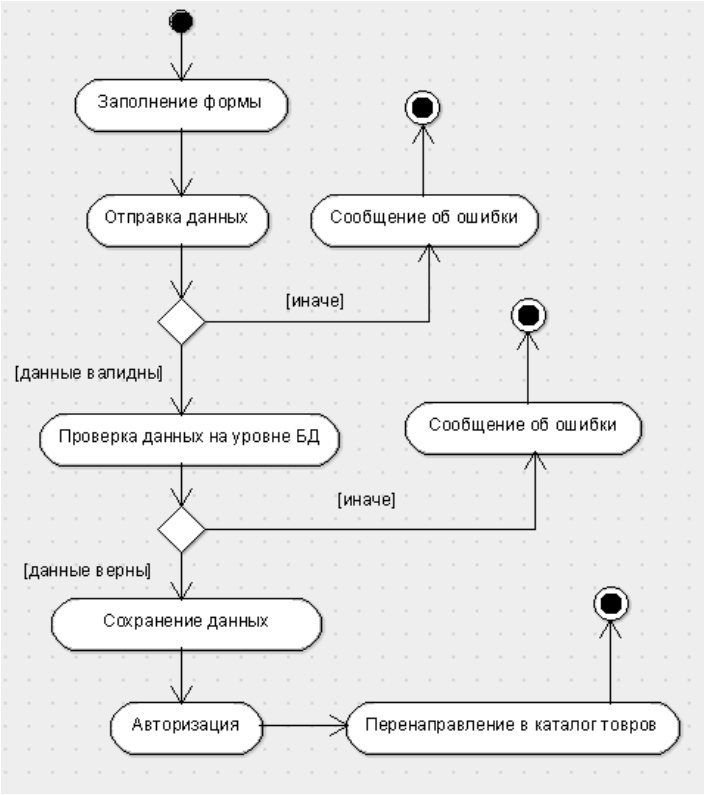

Рисунок 2.2 – Діаграма діяльності

У розділі з прогнозуванням можна зробити прогноз продажу будь-якого товару на наступний період часу, за умови якщо магазин функціонує деякий час і в базі зберігається достатньо інформації про продані товари. Прогноз був реалізований на основі мультиплікативної моделі, аналіз та тестовий розрахунок якої було проведено на стадії проектування.

2.2 Інструментальні засоби проектування

Для проектування веб-програми прогнозування та продажу товарів використовувалися діаграми нотації UML, а також CASE засіб ERwin. Мова графічного опису UML є загальноцільовою мовою графічного опису моделей, яка розроблена для специфікації, візуалізації, проектування та документування компонентів програмного забезпечення, бізнес-процесів, взаємодій об'єктів системи та інших систем. UML покликаний підтримувати процес моделювання програмних засобів на основі об'єктно-орієнтованого підходу, організовувати взаємозв'язок концептуальних та програмних понять та відображати проблеми масштабування складних систем. Моделі мови UML використовуються на всіх етапах життєвого циклу ПС, починаючи з бізнес-аналізу та закінчуючи

супроводом системи. Різні організації можуть застосовувати UML на власний розсуд залежно від своїх проблемних областей і технологій [2]. З погляду методології об'єктно-орієнтованого аналізу та проектування досить повна модель складної системи є певною кількістю взаємопов'язаних уявлень, кожне з яких адекватно відображає аспект поведінки або структури системи. Принцип ієрархічної побудови моделей складних систем передбачає розгляд процес побудови моделей на різних рівнях абстрагування або деталізації в рамках фіксованих уявлень. Будь-яка мова складається зі словника та правил комбінування слів для отримання осмислених конструкцій. Відмінною його особливістю є те, що словник утворюють графічні елементи. Кожному графічному символу 18 відповідає конкретна семантика, тому модель, створена одним розробником, може бути однозначно зрозуміла іншим, а також програмним засобом, що інтерпретує UML. Словник мови UML включає три види блоків: – сутність – це абстракції, які є основними елементами моделі; - Відносини - частини, що пов'язують різні сутності; - діаграми - блоки, що групують які представляють інтерес сукупності сутностей. Діаграми UML підвищують супровідність проекту та полегшують розробку документації до програмної системи. UML може бути застосований на всіх етапах життєвого циклу аналізу бізнес-систем та розробки додатків [2]. Для проектування бази даних веб-застосунку було використано CASE засіб ERwin. CASE - засіб ERwin - засіб розробки структури бази даних (БД). ERwin поєднує графічний інтерфейс Windows, інструменти для побудови ER-діаграм, редактори для створення логічного та фізичного опису моделі даних та прозору підтримку провідних реляційних СУБД та настільних баз даних. За допомогою ERwin можна створювати чи проводити зворотне проектування баз даних. Сімейство продуктів ERwin є набір засобів концептуального моделювання даних, що використовують метод IDEF1X. ERwin реалізує проектування схеми БД, генерацію її опису мовою цільової СУБД (Oracle, Informix, Sybase, DB2, Microsoft SQL Server та ін.) та реверсний інжиніринг існуючої БД. Випускається у кількох конфігураціях, орієнтованих найбільш поширені кошти розробки додатків 4GL. Інтегрується з

популярними засобами розробки клієнтської частини програм PowerBuilder, Visual Basic, Delphi, що дозволяє автоматично генерувати код програм [3].

2.3 Аналіз вимог на розробку та проектування

2.3.1 Аналіз варіантів використання

У рамках уніфікованого процесу функціональні вимоги досліджуються та формулюються у моделі варіантів використання.

Варіант використання - це незалежне від реалізації високорівневе уявлення конкретної функції системи, що розробляється. Варіант використання є послідовністю дій (транзакцій), що виконуються системою у відповідь на подію, що ініціюється зовнішнім об'єктом (дійовою особою, актором) [9].

Діаграма варіантів використання показує взаємодію між варіантами використання та дійовими особами, відображаючи функціональні вимоги до системи з погляду користувача.

Складання варіантів використання здійснюється на основі аналізу вимог до інтернет-магазинів.

Головними акторами в контексті програмного засобу є адміністратор та користувач цієї програми.

Користувачі мають безліч можливостей щодо пошуку, сортування та категоризації потрібних їм товарів, перед безпосередньою покупкою. Однак перед тим, щоб користувач повною мірою міг здійснювати будь-які дії на сайті, необхідно зареєструватися.

Перш ніж відкривати доступ користувачам адміністратору, необхідно заповнити каталог товарів. Адміністратор має право видалити або змінити вже існуючий товар. По мірі здійснення продажів адміністратор може відстежувати статистику по продажам, відвідуванню, активності і т.д. Якщо продаж здійснюються тривалий час і є накопичена статистика, адміністратор може спрогнозувати продажі на деякий період часу. На Рис.2.3 зображено діаграму варіантів використання у нотації UML.

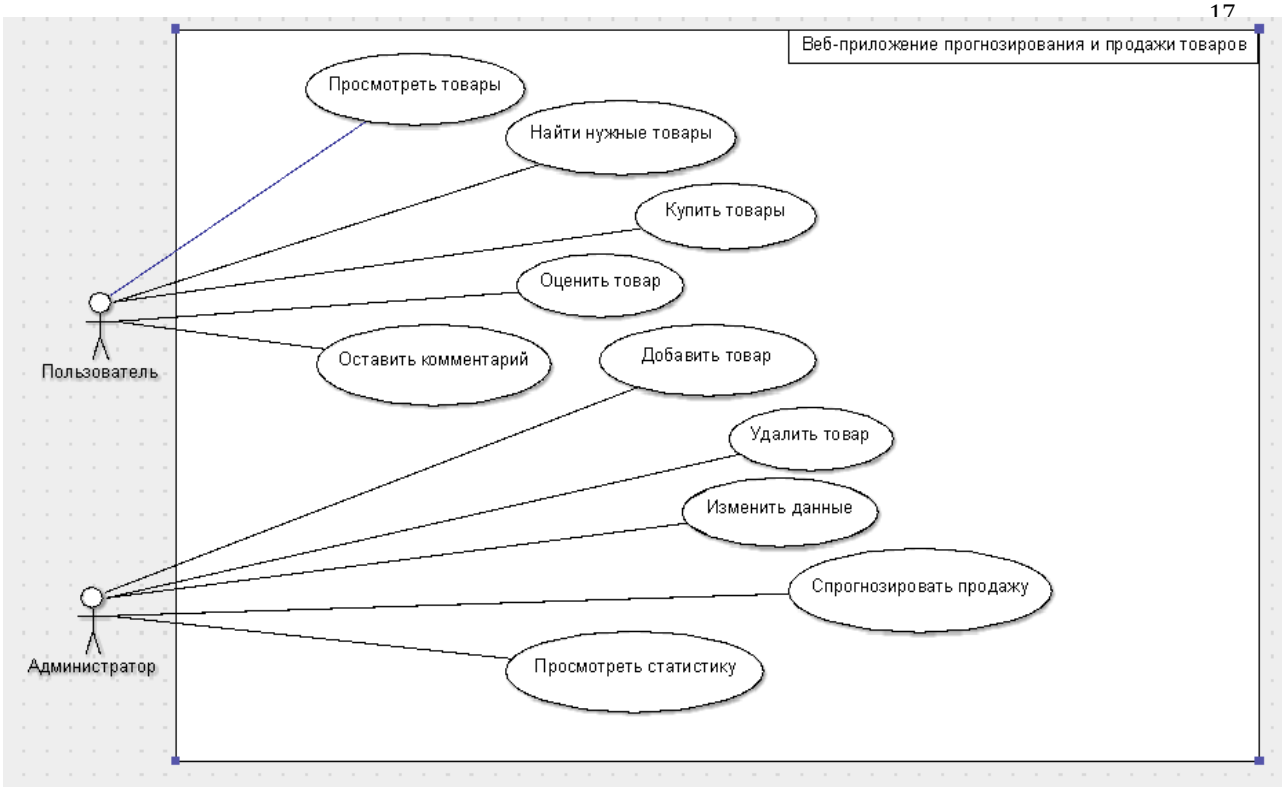

Рисунок 2.3 Діаграма варіантів використання

Виділення варіантів використання полегшить процес подальшого проектування.

2.3.2 Проектування моделі бази даних

Для розробки моделей бази даних використовувався Case засіб ERwin та його методологію IDEF1X. У ERwin існують два рівні представлення та моделювання – логічний та фізичний. Логічний рівень означає пряме відображення фактів із реального життя. На логічному рівні не розглядається використання конкретної СУБД, не визначаються типи даних (наприклад, ціле чи речове число) та не визначаються індекси для таблиць [3].

Розробимо логічну модель бази даних веб-додатку прогнозування та продажу товарів. Ця модель міститиме 11 таблиць пов'язаних між собою логічними зв'язками. Для того, щоб зберігати дані в базі необхідно створити такі таблиці:

«Категорії товарів» – для зберігання даних про категорії: Ідентифікатор категорії, Найменування;

«Бренди» – для зберігання даних про бренди: Ідентифікатор бренду, Найменування;

«Категорія характеристик» – для зберігання даних про категорію окремо взятої характеристики: Ідентифікатор категорії характеристик, Найменування;

«Характеристики товарів» – зберігання даних про властивості різних товарів: Ідентифікатор характеристики, Ідентифікатор категорії характеристик;

«Товари» – для зберігання даних про різні товари: Ідентифікатор товару, Найменування, Ціна, Залишок, Дата додавання, Зображення, Гарантія, Відображення, Перегляди, Опис, Новинка, Ідентифікатор бренду, Ідентифікатор категорії;

«Товар-Характеристики товарів» – допоміжна таблиця для зберігання даних про властивості властиві окремому товару, що дозволяє уникнути зв'язку багато до багатьох: Ідентифікатор товару, Ідентифікатор характеристики, Значення;

«Профіль користувача» – успадковує та розширює базовий клас django «Користувачі»: Телефон, Стать, Дата народження;

«Список бажань» – допоміжна таблиця для зберігання даних про товари, які відкладені для подальшої купівлі, що дозволяє уникнути зв'язку багато до багатьох: Ідентифікатор товару, Ідентифікатор користувача.

«Історія покупок» – допоміжна таблиця для зберігання даних про покупки здійснених користувачами, що дозволяє уникнути зв'язку багато до багатьох: Ідентифікатор товару, Ідентифікатор користувача, Кількість, Дата, Вартість;

«Кошик» – допоміжна таблиця для зберігання даних про товари, які додані для безпосередньої купівлі, що дозволяє уникнути зв'язку багато до багатьох: Ідентифікатор товару, Ідентифікатор користувача, Кількість;

"Коментарі" - допоміжна таблиця для зберігання коментарів про товари, що дозволяє уникнути зв'язку багато до багатьох: Ідентифікатор

товару, Ідентифікатор користувача, Дата, Оцінка, Плюси, Мінуси, Текст коментаря. Логічна модель має такий вигляд (рисунок 2):

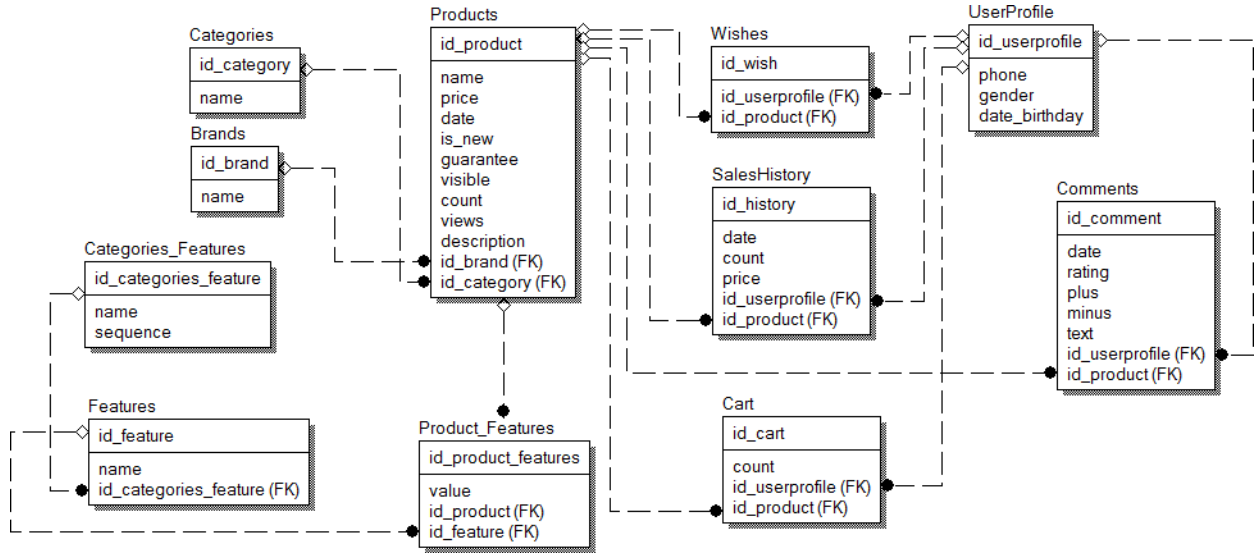

Рисунок 2.4 – Логічна модель даних

Фізичний рівень моделі ERwin становлять імена об'єктів та типи даних, індекси. Фізична модель представлена малюнку 3.

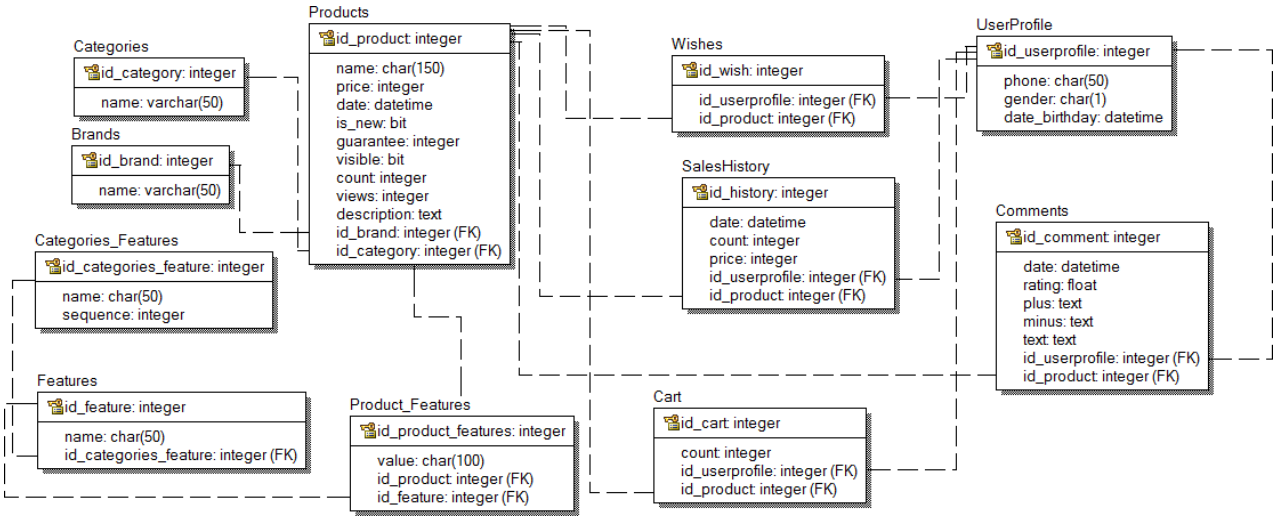

Рисунок 2.5 – Фізична модель даних

Фізична та логічна моделі дозволяють розглянути всі зв'язки та типи даних перед реалізацією, що полегшує створення бази даних.

2.3.3 Проектування інтерфейсу користувача

Інтерфейс користувача є своєрідним комунікаційним каналом, яким здійснюється взаємодія користувача і комп'ютера.

Кращий інтерфейс користувача – це такий інтерфейс, якому користувач не повинен приділяти багато уваги і практично не помічати його. Такий інтерфейс називають прозорим – користувач ніби дивиться крізь нього свою роботу [10].

Відвідувачі, як правило, приходять на сайт з певною метою, і їх цікавить швидкий і зручний доступ до необхідної інформації або товарів, тому важливо організувати простір так, щоб врахувати всі особливості сприйняття та потреби покупців.

Загальна структура сторінки має відповідати деяким очікуванням відвідувачів. При тривалому спілкуванні з інтернет-простором у користувачів складається якийсь стереотип, який є певною напрацьованою системою умовних інстинктів. Це означає, що заходячи на сайт, користувач вже заздалегідь припускає, в якій частині сторінки буде потрібна інформація або елемент меню.

Перший принцип, який є основою загальної композиції сторінки: схема сприйняття текстової інформації зліва направо і зверху вниз. Таким чином, інформація та елементи меню, що найбільш важливі або потребують реагування, повинні знаходитися в зоні початкової уваги - у лівій та лівій верхній частинах сторінки. Найбільш змістовна інформація зазвичай розташовується у центральній частині сторінки. Менш важлива, допоміжна інформація повинна знаходитися у правій та правій нижній частинах сторінки. p align="justify"> Однотипні дані на різних сторінках повинні розташовуватися в одній і тій же області [10].

Привернення уваги користувача, крім просторового розташування, здійснюється й іншими способами: виділення розміром (великий об'єкт привертає більше уваги та сприймається раніше дрібних об'єктів), освітленням (затемнені деталі сприймаються гірше за яскраві), кольором, шрифтом (оформлення тексту відмінної від стандартної гарнітури, накреслення та жирності). Під час створення інтерфейсу слід враховувати представлені особливості.

2.4 Математична оцінка деяких аспектів діяльності інтернет-магазину

2.4.1 Основні поняття та визначення

Нині активно розвивається роздрібна торгівля [1], у зв'язку з чим виникає необхідність удосконалювати управління магазинах. З погляду розуміння та описи конкурентоспроможності фірм, неокласична економічна теорія [2, 3] значною мірою поступається еволюційній економіці [4]. У роботах [5, 6] наведено побудову моделі ринку. Однак у цих моделях конкурентоспроможність фірм визначається конкурентоспроможністю вироблених ними товарів (ціна, різні характеристики). Магазин має певну специфіку, тобто. його дохід залежить від його цінової та асортиментної політики, викладення товару тощо, тому за допомогою однієї лише конкурентоспроможності товару достовірно описати роздрібний ринок навряд чи вдасться. Отже, при моделюванні роздрібного ринку особливу увагу слід приділяти не конкурентоспроможності товару,

У цій роботі ми розглянемо модель ринку одного виду товару (питної води), на основі якої можна виробляти рекомендації менеджерам магазину з управління процесом продажу з метою підвищення конкурентоспроможності. У ній використовується як внутрішня інформація про сам магазин, так і зовнішня інформація про споживачів та конкурентів [7–10]. Основна ідея у тому, що виручка магазину визначається переважно прихильністю споживачів до цього магазину. Така залежність погано формалізується, тому її можна будувати на основі логічних міркувань, апроксимуючи у разі потреби найпростішими функціями.

Під ринком слід розуміти сукупність n магазинів і сукупність споживачів, які у контакт з метою купівлі-продажу товару. Np – це потенціал ринку, сума коштів, яку споживачі готові витратити певний вид товару, за певний проміжок часу Tp.

Під характеристиками ринку розумітимемо ціну $P_{ср}$ (середня ціна товару по магазинах $P_{ср} = \frac{1}{n}\sum_{i=1}^{n} P_i$, де P_i – ціна товару і-го магазину, $i = \overline{1,n}$ та асортимент Amax (кількість можливих підвидів товару на даному ринку).

Припустимо, частка і-го магазину залежить від цього, наскільки споживачам привабливіше купувати товар у цьому магазині, ніж у інших. Привабливість і-го магазину обумовлена багатьма факторами; ціна, асортимент, викладення товару, відстань до магазину, якість обслуговування тощо. Розглянемо залежність привабливості і-го магазину від ціни та асортименту товару, оскільки ці фактори мають найбільш важливе значення для споживачів. Тоді характеристиками і-го магазину будуть: ціна Pi - середня ціна підвидів товару, асортимент Asi - кількістьпідвидів товару.

Люди при виборі магазину, в якому купуватимуть товар, як правило, насамперед звертають увагу на ціну товару, асортимент, якість обслуговування тощо, при цьому вони можуть скористатися аналогічним досвідом свого знайомого (чутки) чи рекламою, що дозволяє їм придбати знання про характеристики ринку, такі як середня ринкова ціна $P_{ср}$, можливий асортимент товару на даному ринку A_{max}. Таким чином, крім ціни P_i та асортименту As_i товару і-й магазин має наступні характеристики: $\overline{P_i} = \frac{P_{ср}}{P_i}$ – цінова характеристика та $\overline{As_i} = \frac{As_i}{A_{max}}$ - асортиментна характеристика. Передбачається, що саме з цих характеристик відбувається вибір і-го магазину.

2.4.2 Привабливість магазину.

Визначимо, як змінюється привабливість і-го магазину в залежності від цінової характеристики $\overline{P_i}$. Спочатку встановимо відомі точки цієї залежності. Так, за значного зменшення цінової характеристики товару $\overline{P_i}$, тобто. при значному збільшенні ціни P_i привабливість Fp_i (інтерес) споживачів до і-го магазину за ціною впаде до нуля $Fp_i = 0$. Однак, якщо і-й магазин значно підвищить свою цінову характеристику $\overline{P_i}$, тобто. значно зменшить ціну, то й привабливість (інтерес) споживачів різко зросте. Очевидно, що при подальшому зменшенні ціни привабливість також різко зростатиме. Теоретично магазин може віддавати товар безкоштовно і при цьому ще доплачувати споживачеві, тоді привабливість і-го магазину зросте дуже високо. Припустимо, що зі

збільшенням цінової характеристики, тобто. при зменшенні ціни підвищується привабливість магазину. Припустимо також, що залежність монотонно зростаюча і неперервна. Якісний вид залежності показано на рис. 1.

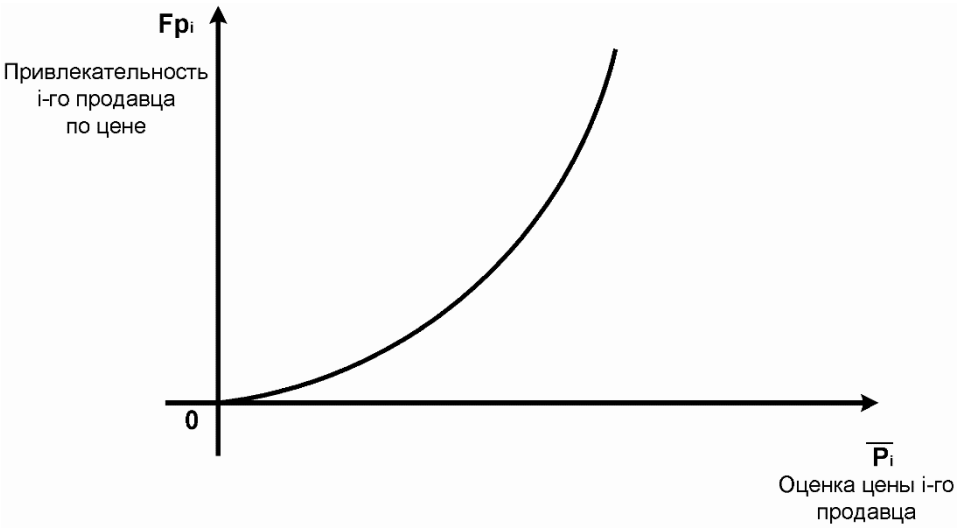

Рис. 2.6. Залежність привабливості Fp_i від цінової характеристики $\overline{P_i}$

Швидкість зміни привабливості і-го магазину за ціною Fp_i зі зростанням цінової характеристики $\overline{P_i}$ збільшується, тобто. $\frac{dFp_i}{d\overline{P_i}} = f(\overline{P_i})\frac{1}{2}$, де $f(\overline{P_i})$ – функція від характеристики ціни, для простоти вважатимемо її лінійною, тоді $\frac{dFp_i}{d\overline{P_i}} = k_1' \cdot \overline{P_i} + k_2$. Вирішуючи це рівняння та підставляючи нульову точку, отримуємо залежність

$$Fp_i = \overline{P_i} \cdot \left(k_1 \cdot \overline{P_i} + k_2\right) = \frac{P_{cp}}{P_i} \cdot \left(k_1 \cdot \frac{P_{cp}}{P_i} + k_2\right), \quad i = \overline{1,n}, \tag{2.1}$$

де k_1', $k_1 = \frac{1}{2} \cdot k_1'$, k_2 – деякі константи.

Зі спостережень видно, що привабливість за ціною магазину швидко прямує до нуля при значному збільшенні ціни над середньою ринковою Pcp. Приймемо, що за ціною, що дорівнює $P_i = P_{cp}(1+\gamma)$ значення привабливості дорівнює нулю: $Fp_i = 0$, $\gamma \in (0,1)$. Підставляючи в залежність (1), отримаємо

$$Fp_i = k_1 \cdot \frac{P_{cp}}{P_i} \cdot \left(\frac{P_{cp}}{P_i} - \frac{1}{1+\gamma}\right), \quad i = \overline{1,n}. \tag{2.2}$$

Визначимо, як змінюється привабливість магазину в залежності від асортиментної характеристики $\overline{As_i}$, тобто. оскільки гарний асортимент магазину для споживача. Спочатку встановимо відомі точки цієї залежності. Так, за

нульового значення асортиментної характеристики $\overline{As_i} = 0$ привабливість по асортименту Fai (інтерес) споживачів також дорівнюватиме нулю $Fa_i = 0$, так як товар відсутній в магазині. Однак, якщо в магазині є повний асортимент $As_i = A_{max}$, то і привабливість по асортименту $\overline{As_i}$ магазину також висока. Припускаємо, що зі збільшенням асортиментної характеристики $\overline{As_i}$, тобто. зі збільшенням асортименту As_i, зменшується приріст привабливості по асортименту Fa_i. Припустимо, що залежність монотонно зростає та неперервна. Якісний вид залежності представлений на рис. 2.6.

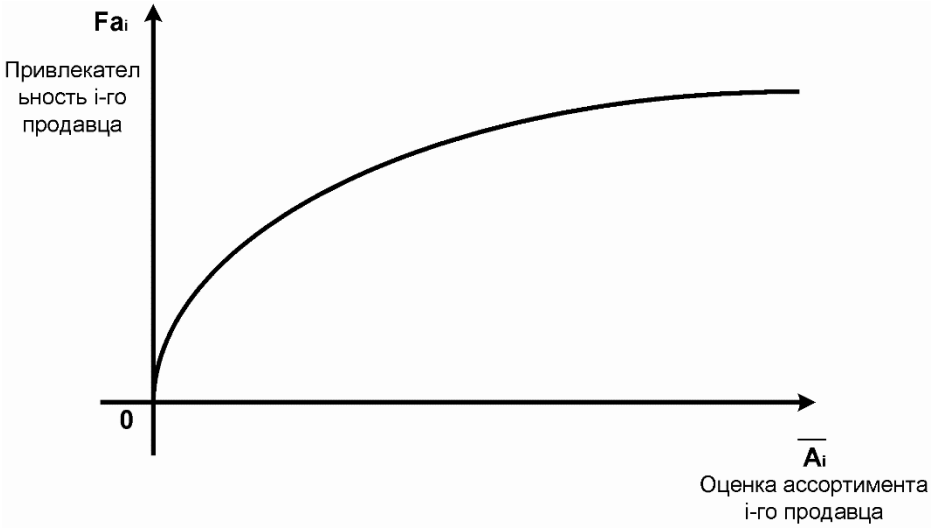

Рис. 2.7 Залежність привабливості Fa_i за асортиментом і-го магазину від асортиментної характеристики $\overline{As_i}$

Швидкість зміни привабливості і-го магазину за асортиментом Fa_i зі зростанням асортиментної характеристики $\overline{As_i}$ зменшується, тобто. $\frac{dFa_i}{d\overline{A_i}} = \frac{k'_3}{f(\overline{A_i})}$, де $f(\overline{A_i})$ – функція від властивості асортименту, для простоти вважатимемо її лінійної, тобто. $\frac{dFa_i}{d\overline{A_i}} = \frac{k'_3}{k'_2 \cdot \overline{A_i} + k'_4}$, де k'_2, k'_3, k'_4 - Деякі коефіцієнти. Вирішуючи це рівняння та підставляючи нульову початкову умову, отримуємо

$$Fa_i = k_3 \cdot \ln\left(1 + k_2 \cdot \overline{A_i}\right) = k_3 \cdot \ln\left(1 + k_2 \cdot \frac{A_i}{A_{max}}\right), \quad i = \overline{1,n}, \qquad (2.3)$$

де $k_2 = \frac{k'_2}{k'_4}, k_3 = \frac{k'_3}{k'_2}$ - деякі коефіцієнти.

Привабливість магазину залежить від ціни P_i, асортименту As_i, якості обслуговування, розміщення товару і т.д., тому визначатимемо загальну

привабливість $\overline{F_i}$ і-го магазину як добуток привабливості за ціною Fp_i на привабливість за асортиментом Fa_i та на невраховану привабливість α_i:

$$\overline{F_i} = Fp_i \cdot Fa_i = \alpha_i \cdot \ln\left(1 + k_2 \cdot \frac{A_i}{A_{max}}\right) \cdot \frac{P_{cp}}{P_i} \cdot \left(\frac{P_{cp}}{P_i} - \frac{1}{1+\gamma}\right), \quad i = \overline{1,n}, \tag{2.4}$$

де $\alpha_i = k_1 \cdot k_3$ - константа, що відображає невраховану привабливість (якість обслуговування, розміщення товару в і-му магазині і т.д.).

Вираз (2.4) запишемо у вигляді

$$\overline{F_i} = \alpha_i \cdot F_i, \quad i = \overline{1,n}, \tag{2.5}$$

де

$$F_i = \ln\left(1 + k_2 \cdot \frac{A_i}{A_{max}}\right) \cdot \frac{P_{cp}}{P_i} \cdot \left(\frac{P_{cp}}{P_i} - \frac{1}{1+\gamma}\right).$$

2.4.3 Виторг магазину $\overline{N_i}$ та ефективність реклами

Розподіл ринку має значення і розглядається з різних точок зору [2, 3, 11]. Вважатимемо, що частка виручки і-го магазину з Np дорівнює відношенню привабливості і-го магазину до суми привабливості всіх магазинів:

$$\overline{N_i} = Np \cdot \frac{\overline{F_i}}{\sum_{j=1}^{n} \overline{F_j}} = Np \cdot \frac{\alpha_i \cdot F_i}{\sum_{j=1}^{n} (\alpha_j \cdot F_j)}, \quad i = \overline{1,n}. \tag{2.6}$$

З виразу (2.6) видно, що з монополії n = 1 виручка магазину дорівнює потенціалу ринку. У межі, у міру того, як число магазинів наближається до нескінченності $n \to \infty$, вплив кожного магазину дуже мало.

Вочевидь, що з різкій зміні привабливості і-го магазину виручка зміниться відразу. Це пов'язано з тим, що має пройти час, перш ніж споживачі дізнаються про це з реклами та в результаті чуток (поширення інформації). Питання поширення інформації має значення [12]. Вважатимемо, що модель поширення інформації аналогічна моделям з робіт [13, 14]. У нашому випадку люди співвідносяться з їх грошима, а отже, іу моделі розглядатиметься не кількість людей, а їх кошти. При цьому модель визначає вплив реклами та чуток серед того сегмента споживачів, які є потенційними споживачами і-го магазину:

$$\frac{dN_i}{dt} = (a_i + b \cdot N_i) \cdot (\overline{N_i} - N_i), \quad i = \overline{1,n}, \tag{2.7}$$

де $\overline{N_i}$ - виручка і-го магазину в режимі (2.6); Nt - виручка і-го магазину в момент часу t (час, наприклад, день); b – коефіцієнт, що відображає вплив чуток, що поширюються для людей, на виручку магазину; аі – частка грошей, залучених рекламою і-го магазину (тобто ефективність реклами), $a_i \in [0,1]$.

Встановимо залежність ефективності реклами аі від фінансових вкладень і-го магазину. Нехай h_i – частка коштів, вкладених у рекламу і-м магазином, від коштів, необхідні залучення всіх споживачів. Вважатимемо, що, з одного боку, швидкість зміни a_i в залежності від h_i прямо пропорційна, з іншого – при досягненні значення аі одиниці реклама охоплює всіх споживачів ринку, тоді і $\frac{da_i}{dh_i}$ пропорційна 1 - a_i. Отже, остаточно вважаємо

$$\frac{da_i}{dh_i} = k \cdot h_i \cdot (1 - a_i).$$

Вирішуючи рівняння з врахуванням початкової умови, отримаємо

$$a_i = 1 - e^{-\frac{k}{2} \cdot h_i^2}, \quad i = \overline{1,n}, \tag{2.8}$$

де k – деякий коефіцієнт.

2.4.4 Ухвалення управлінських рішень

Таким чином, пропонована модель має вигляд:

$$\frac{dN_i}{dt} = (a_i + b \cdot N_i) \cdot (\overline{N_i} - N_i), \quad i = \overline{1,n}, \tag{2.9}$$

$$a_i = 1 - e^{-\frac{k}{2} \cdot h_i^2}, \quad i = \overline{1,n},$$

$$\overline{N_i} = Np \cdot \frac{\alpha_i \cdot F_i}{\sum_{j=1}^{n} (\alpha_j \cdot F_j)}, \quad i = \overline{1,n},$$

$$F_i = \ln\left(1 + k_2 \cdot \frac{A_i}{A_{max}}\right) \cdot \frac{P_{cp}}{P_i} \cdot \left(\frac{P_{cp}}{P_i} - \frac{1}{1+\gamma}\right), \quad i = \overline{1,n}.$$

Для цілей управління ціновою та асортиментною політикою припустимо, що менеджери і-го магазину в момент прийняття рішення щодо ціни , асортименту товару, грошових вкладень у рекламу вважають, що їх конкуренти не приймають у цей час жодних рішень. При цьому вважаємо, що самі

менеджери і-го магазину приймають оптимальні для себе рішення щодо ціни, асортименту товару, вкладень у рекламу за рахунок своїх грошових ресурсів R_i за деякий проміжок часу T_i.

Для прийняття управлінського рішення в і-му магазині часто використовується як критерій прибуток Ii за деякий проміжок часу Ti Ця залежність має вигляд

$$I_i = \int_{T_0}^{T_i} \frac{N_i}{P_i}\left(P_i - P_{per\,i}\right)dt, \quad i = \overline{1,n}, \tag{2.10}$$

де Pper i - закупівельна ціна; Ni - виручка в даний момент часу t, Ti - кінцевий момент часу і-го магазину, T0 - початковий момент часу.

Тоді $\frac{N_i}{P_i}$ - кількість проданого товару за час t, а $\left(P_i - P_{per\,i}\right)$ - торгова націнка. Перемножуючи кількість проданого товару на торгову націнку, отримуємо прибуток і магазину за час Ti.

На прийняття управлінського рішення також впливає обмеження ресурсів і-го магазину, яке має вигляд

$$Z_{P\,i} + Z_{As\,i} + h_i \le R_i, \quad i = \overline{1,n}, \tag{2.11}$$

де Z_{Pi} - витрати на зміни ціни, Z_{Asi} - витрати на зміни асортименту (при збільшенні асортименту потрібно закуповувати новий товар) $ hi - витрати на рекламу.Витрати зниження ціни товару рівні:

$$Z_{P\,i} = \int_{T_0}^{T_i} \frac{N_i}{P_{new\,i}}\left(P_{old\,i} - P_{new\,i}\right)dt, \quad i = \overline{1,n}, \tag{2.12}$$

де Pold i та Pnew i – ціна товару до та після зміни відповідно в і-му магазині; $\frac{N_i}{P_{new\,i}}$ – кількість проданого товару; $\left(P_{old\,i} - P_{new\,i}\right)$ - втрачені гроші з однієї одиниці товару.

Перемножуючи кількість проданого товару на втрачені гроші з одиниці товару, отримуємо суму втрат через зниження ціни:

$$Z_{As\,i} = \int_{T_0}^{T_i} \frac{N_i - N_0}{P_i} \cdot P_{pri\,i}dt, \quad i = \overline{1,n}, \tag{2.13}$$

де N_0 - виручка і-го магазину в момент часу T_0.

Тоді Ni - N_0 - збільшення виручки i-го магазину в момент часу t за рахунок збільшення асортименту, $\frac{N_i - N_0}{P_i}$ - Кількість проданого товару за момент часу t за рахунок збільшення асортименту, а $\frac{N_i - N_0}{P_i} \cdot P_{pri\,i}$ – кількість коштів, необхідні збільшення асортименту.

Отже, запропонована модель ринку окреслила завдання оптимального управління. Керуючі параметри Pi, As_i, h_i, $i = \overline{1,n}$ мають наступну область зміни. Асортимент As_i може бути як нульовий, так і максимальний A_{max} в i-му магазині, а значить, $As_i \in [0, A_{max}]$. Вкладення коштів у рекламу може бути будь-яким, тому hi≥0. Ціна товару Pi строго більша за закупівельну: Pi>Ppri i – це нижня межа ціни, що стосується верхньої її межі, то вона має бути не більшою за середню ціну на ринку, помножену на деякий коефіцієнтγ: $P_i \le \gamma \cdot P_{cp}$, отже, $P_i \in \left(P_{pri\ i}, \gamma \cdot P_{cp}\right]$:

$$P_i \in \left(P_{pri\ i}, \gamma \cdot P_{cp}\right], As_i \in [0, A_{max}], h_i \ge 0, \quad i = \overline{1,n}. \tag{2.14}$$

Обмеження на фінансові ресурси i-го магазину Ri задані залежністю (2.11). Цільовий функціонал, максимум якого потрібно знайти, є прибутком:

$$I_i = \int_{T_0}^{T_i} \frac{N_i}{P_i}\left(P_i - P_{per\,i}\right)dt \to \max, \quad i = \overline{1,n}. \tag{2.15}$$

Це завдання можна класифікувати як завдання Лагранжа. Рівняння руху встановлено рівнянням (2.9) з урахуванням відповідних залежностей. Вирішуючи завдання оптимального керування, визначимо оптимальні значення керуючих параметрів Pi, Asi, hi тільки для i-го магазину, а інші магазини не приймають ніяких нових управлінських рішень.

Розглянемо для прикладу поведінка ринку із двома магазинами n = 2. І тому приймемо такі значення необхідних величин. Час t будемо вимірювати у днях. Магазини приймають деякі рішення з розрахунку на прибуток за проміжок часу, рівний T1 = 20 днів і T2 = 20 днів. Потенціал ринку дорівнює Np = 5000 грн.., у сфері інформування всіх споживачів ринку потрібно витратити реклами 5000 грн.. Характеристики ринку: Pcp = 56 грн.. (середня ціна товару), Amax = 12 (асортимент). Характеристики товарів у магазинах: P1 = 54 грн.., P2 = 58 грн.. (ціна), As1 = 7, As2 = 11. (асортимент товарів першого та другого магазину

відповідно при t = 0). Оптова ціна товару дорівнює: Ppri 1 = 45 грн.., Ppri 2 = 46,4 грн. Коефіцієнти, обумовлені з урахуванням маркетингового дослідження, приймають такі значения: γ= 0,5, тобто при збільшенні ціни на товар у півтора рази більша від середньоринкової ціни привабливість магазину дорівнює нулю; a1 = 0,1, a2 = 0,25, тобто залучення у кожний момент часу коштів становить у першого магазину 10%, а у другого 2,5% за рахунок реклами; b = 0,1 - коефіцієнт, що відображає поширення чуток;α_1= 1,α_2= 1 - коефіцієнти, що відображають невраховані фактори, що впливають на привабливість магазину; k = 1, k2 = 1 – деякі коефіцієнти.

Проведемо підрахунки. Привабливість магазинів у момент часу $t = 0$ відповідно до формули (2.9) дорівнює: F1 = 0,812, F2 = 1,025, тобто другий магазин привабливіший для споживачів, ніж перший, відповідно і виручка магазинів дорівнює: $\overline{N_1} = 2210$ грн.., $\overline{N_2} = 2790$ грн.. (Рис. 2.8).

Припустимо, що в момент часу t = 0 другий магазин змінює ціну товару, збільшуючи її на 21 грн..: P2 = 75 грн., середня ціна на ринку дорівнює: Pcp = 64,5 грн.. Тоді привабливість другого магазину падає з F2 = 1,025 до F2 = 0,687, а отже, і виторг також зменшиться з $\overline{N_2} = 2790$ грн.. до $\overline{N_2} = 2291$ грн.., тобто. через підвищення ціни споживачі перейшли від другої до першої крамниці, цей перехід відбувався протягом 20 днів.

Якщо, у час t = 20 перший магазин збільшив свій асортимент з As1 = 7 до As1 = 9 задля зміцнення своїх позицій, його привабливість підвищиться з F1 = 0,812 до F1 = 0,989, тобто споживачі перейдуть від другого магазину до першого і відповідно виручка через 20 днів складатиме: $\overline{N_1} = 2951$ грн. та 2049 грн.

Якщо на момент часу t = 40, другий магазин вклав кошти у рекламу відповідно до формул (2.9) у сумі 2000 грн.., тоді $h_2 = \frac{2000}{5000} = 0,4$, тобто. ефективність реклами у другого магазину підвищується: a2 = 0,025 + 0,077 = 0,102, і знижується вартість з P2 = 75 до P2 = 52 грн. . В результаті через 15 днів (зменшення терміну за рахунок реклами), другий магазин повертає свої початкові позиції, його привабливість дорівнює: F1 = 0,989; F2 = 1,222, виручка складає: $\overline{N_1} = 2237$ грн.., $\overline{N_2} = 2763$ грн..

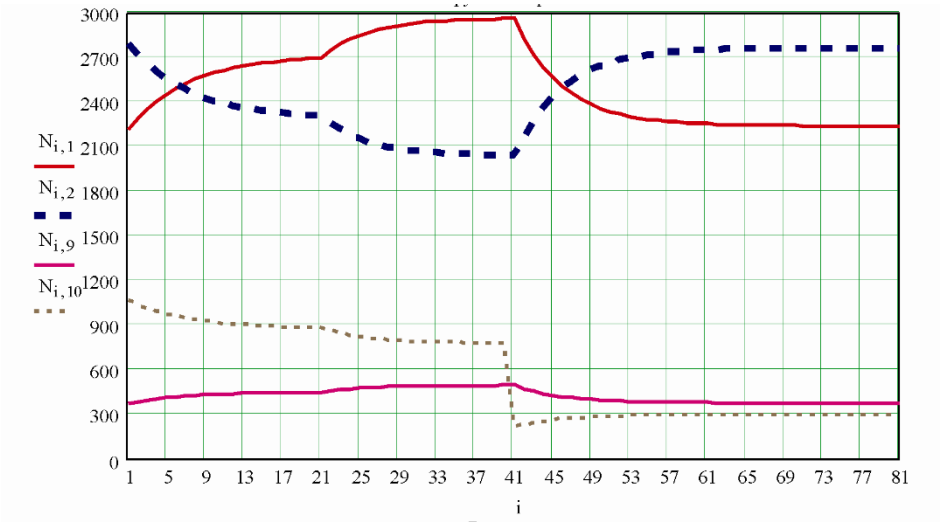

Рис.2.8 Поведінка виручки та прибутку в часі

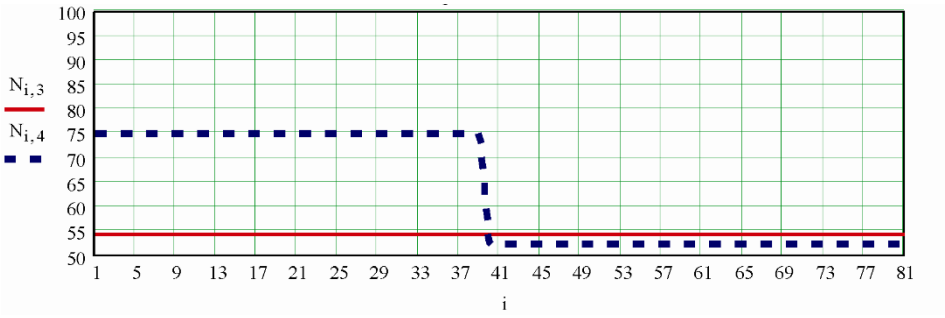

Рис.2.9 Управління ціною

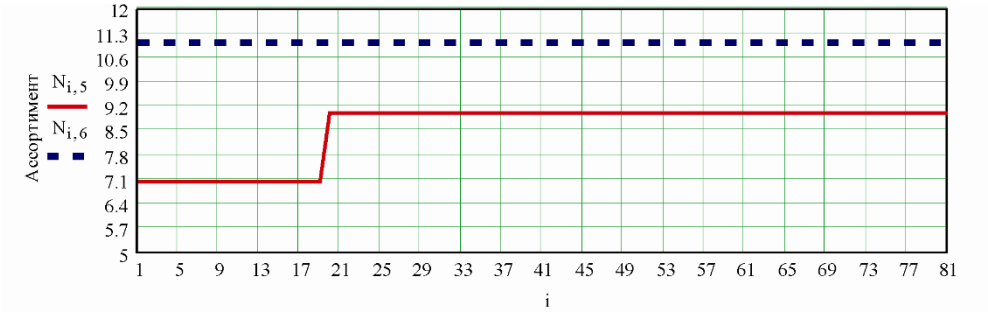

Рис.2.10 Управління асортиментом

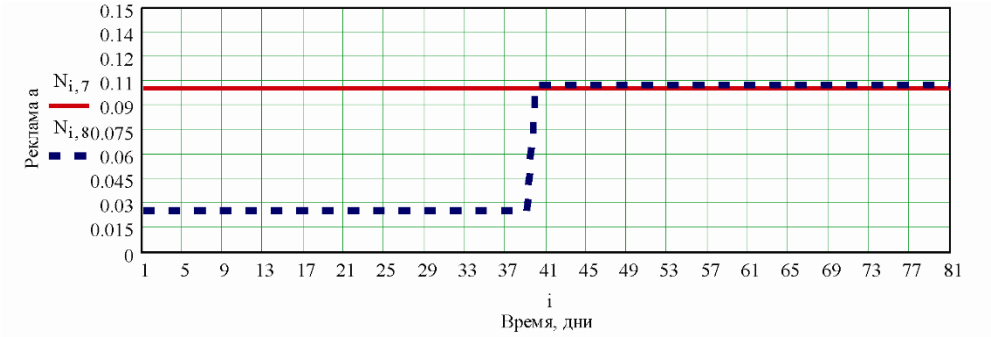

Рис.. 2.11. Управління рекламою

На Рис. 2.8-2.11 бачимо динаміку виручки магазинів та управління процесом продажів, суцільна лінія – динаміка першого магазину, пунктирна лінія – динаміка другого магазину.

Розділ 3. Розробка проекту

3.1 Уточнення задачі

Метою даної є розробка веб-додатка прогнозування та продажу товарів. Програма має бути спроектована з використанням клієнт-серверної архітектури. Умовно магазин можна розділити на дві частини: клієнтська частина, яка являє собою типовий інтернет магазин і адміністративна, в якій крім можливостей щодо додавання, зміни, видалення товарів, буде доступна функція прогнозування продажів на наступний рік, а також представлені різні діаграми та графіки, що інформують адміністратора про поточні продажі, виручку та інше. Виділимо основні вимоги до додатку. Для клієнтської частини необхідно реалізувати: – каталог товарів, з можливістю сортування та вибірки за необхідними параметрами; – форми для авторизації та реєстрації користувачів; - Пошук товарів за ключовими словами; - користувальницький кошик покупок; - форму оформлення замовлення. Для адміністративної частини додатка необхідно реалізувати розділ для прогнозування продажів товарів, а також графічний розділ, що відображає різні дані з продажу, що необхідно для контролю бізнес-процесів. Правильність роботи програмного засобу має перевірятися та тестуватися. Ручне тестування програмного засобу має здійснюватися шляхом виконання тестових прикладів та порівняння вихідних даних із очікуваними результатами. що відображає різні дані з продажу, що необхідно для контролю бізнес-процесів. Правильність роботи програмного засобу має перевірятися та тестуватися.

3.1 Структура Django-додатку

Джанго-додаток складається з чотирьох основних компонентів:

1) Модель даних: дані є серцевиною будь-якої сучасної Web-додатки. Модель – найважливіша частина програми, яка постійно звертається до даних за будь-якого запиту з будь-якої сесії. Будь-яка модель є стандартним пітонівським класом. Об'єктно-орієнтований мапер (ORM) забезпечує таким класам доступ безпосередньо до баз даних. Якби не було ORM, програмісту довелося писати запити безпосередньо на SQL. Модель забезпечує полегшений механізм доступу

до шару даних, інкапсулює бізнес-логіку. Модель не залежить від конкретної програми. Даними можна маніпулювати навіть із командного рядка, не використовуючи при цьому Web-сервер.

2) Подання (view): Подання у Джанго виконують різноманітні функції, у тому числі контролюють запити користувача, видають контекст залежно від його ролі. View - це звичайна функція, яка викликається у відповідь на запит якоїсь адреси (url) і повертає контекст.

3) Шаблони: вони є формою представлення даних. Шаблони мають свою власну просту метамову і є одним з основних засобів виведення на екран.

4) URL: це лише механізм зовнішнього доступу до уявлень (view). Вбудовані в url регулярні вирази роблять механізм досить гнучким. При цьому одне уявлення може бути налаштоване на кілька url, надаючи доступ різним додаткам. Тут підтримується філософія закладок: url-и стають як би самодостатніми і починають жити незалежно від уявлення.

У Django підтримується все те, що і в Python: тут є доступ до всіх стандартних, і не тільки бібліотек Python, плюс вбудований функціонал Джанго.

3.2 Базові налаштування

Створимо тепер додаток під назвою journal

```
(myvenv) C:\Projects>cd journal_proj

(myvenv) C:\Projects\journal_proj>python manage.py startapp journal

(myvenv) C:\Projects\journal_proj>
```

Рис. 3.1 Створення програми

Поруч із папкою проекту з'явилася папка з назвою програми

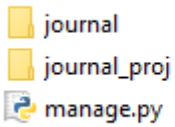

Рис.3.2 Папки додатоку

У папці такі файли:

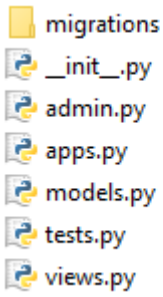

Рис.3.3 – Файли програми

Models.py – необхідний для створення моделей даних

Admin.py – для прописування параметрів адміністрування

Views.py - уявлення. Цей файл викликається у відповідь на запит URL-адреси.

Саме час створити базу даних нашого проекту. Робити це ми будемо у PostgreSQL. Для цього відкриємо PgAdmin 4.

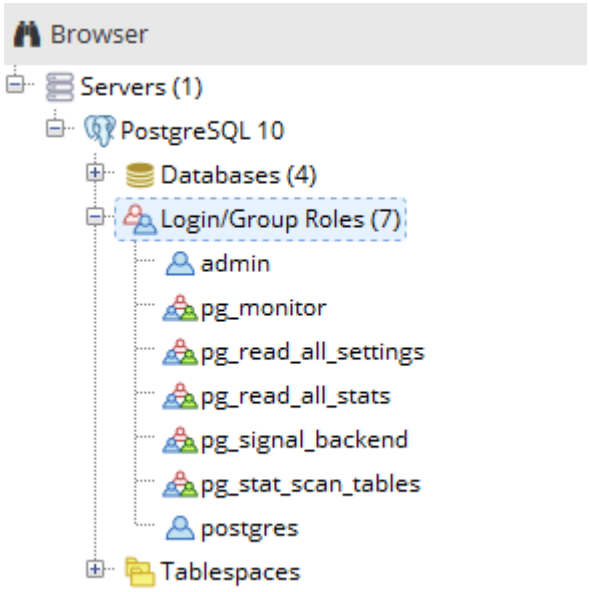

Рис.3.4 Користувачі бази даних

Для початку створимо суперкористувача, якого назвемо admin1 і задамо такий пароль.

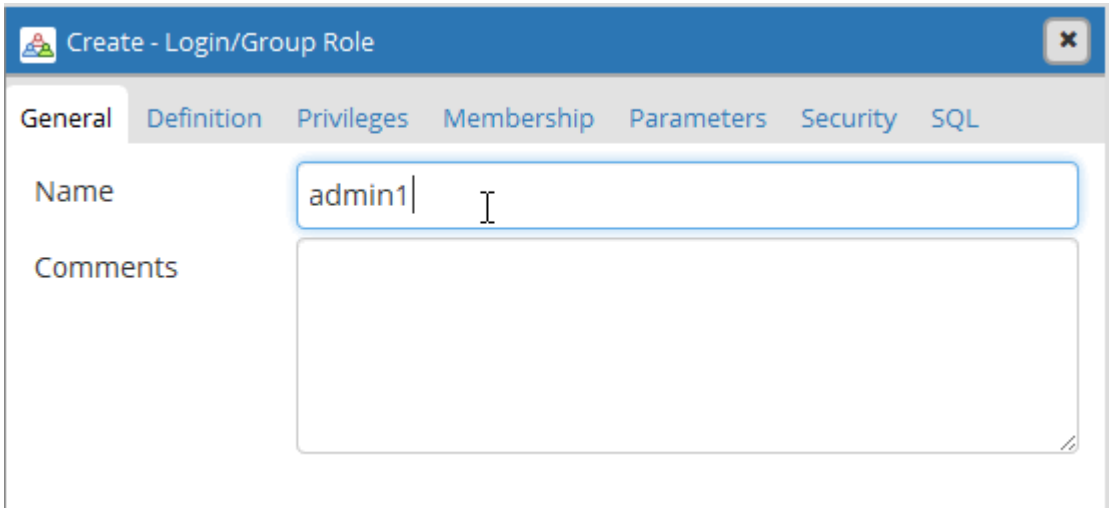

Рис.3.5 Створення суперкористувача

Задамо йому всі права.

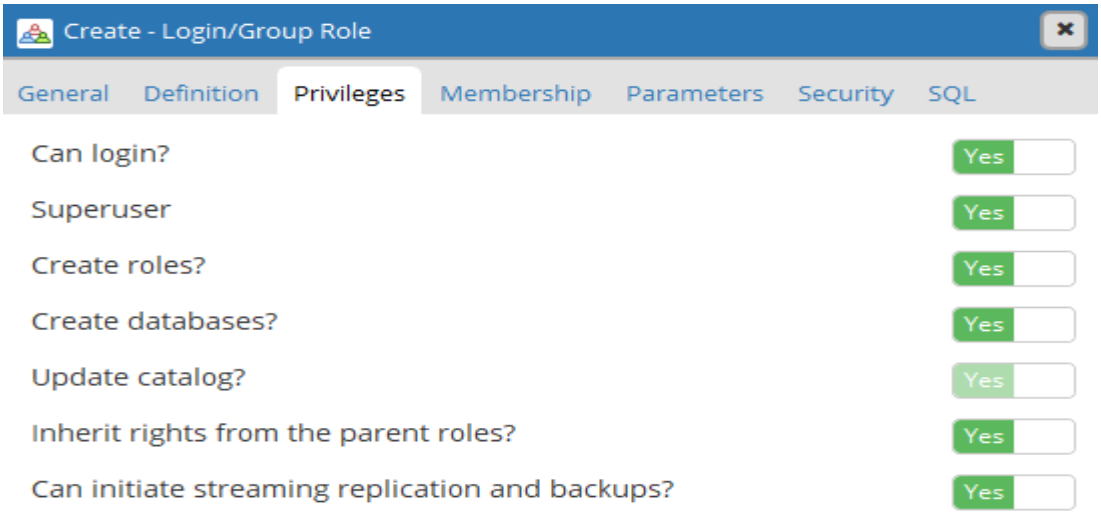

Рис.3.6 Завдання прав

Тепер створимо базу даних, яку назвемо jour_db і вкажемо як власник нами створеного суперкористувача admin1.

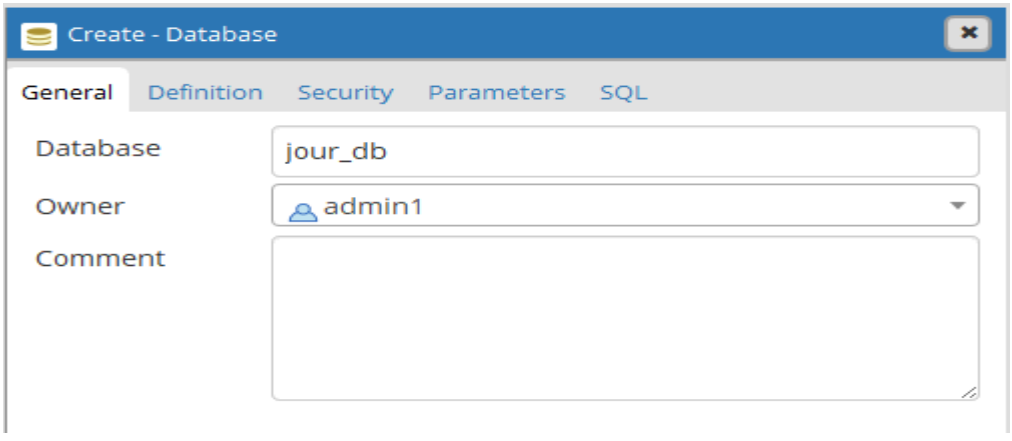

Рис.3.6 Створення бази даних

Зайдемо тепер назад у налаштування проекту та вкажемо туди базу даних та дані для суперкористувача.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'jour_db',
        'USER': 'admin1',
        'PASSWORD': 'admin1',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

Рис. 3.7 – Додавання бази у налаштування проекту

3.3 Налаштування urls.py

Перейдемо тепер в папку нашого проекту та відкриємо файл urls.py

```
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

Рис.3.7 Файл urls.py

urls – це механізм доступу до уявлень. У списку urlpatterns містяться шаблони адрес і переадресація на інший файл urls або відображення відповідних їм уявлень.

Оскільки запити йдуть через urls файл проекту, нам потрібно зробити так, щоб він посилався на urls файл нашої програми. Для цього імпортуємо функцію include, яка дозволить це зробити та запишемо до списку необхідний код.

```
from django.contrib import admin
from django.urls import path
from django.conf.urls import include

urlpatterns = [
    path('admin/', admin.site.urls),
    path(r'', include('journal.urls')),

]
```

Рис. 3.7 Додавання urls створеної програми

Django зіставляє URL-адреси та подання. URL-адреса записується за допомогою регулярних виразів, що виділяються лапками. Якщо в них нічого немає, то мається на увазі адресу http://127.0.0.1:8000.

Перед регулярними виразами бажано ставити літеру r. Це натяк для Python, що рядок може містити спеціальні символи, призначені не для Python,

а для регулярного вираження. Перейдемо в папку нашої програми та створимо в ній аналогічний файл з адресами. Структура буде такою самою.

Початковій адресі http://127.0.0.1:8000 відповідатиме функція представлення reg – вікно з реєстрацією користувача.

Адреса http://127.0.0.1:8000/auth- функція authoriz – вікно з авторизацією.

Адреса http://127.0.0.1:8000/main- функція main – головне меню.

Адреса http://127.0.0.1:8000/write- функція writing – вікно для заповнення журналу.

Адреса http://127.0.0.1:8000/jour- функція read_jour – вікно з вмістом журналу.

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

urlpatterns = [
    path(r'', views.reg),
    path(r'auth/', views.authoriz),
    path(r'reg/', views.reg),
    path(r'main/', views.main),
    path(r'write/', views.writing),
    path(r'jour/', views.read_jour),
]
```

Рис. 3.8 Створення файлу з адресами для програми

3.4 Створення моделей

Тепер створимо моделі – таблиці у нашій базі даних. Для цього відкриємо файл models.py та додамо туди текст, зображений на Рис.3.8.1. Таким чином, у нас буде 3 таблиці в базі даних. Перша називається users – список користувачів, яка зберігатиме у собі логіни та паролі. Тип CharField - Текстові значення. Максимальною довжиною вкажемо 200 символів.

Друга – trade – список відкритих угод. Містить номер авторизованого користувача, який є зовнішнім ключем. При видаленні користувача з системи буде видалено всі відкриті угоди, пов'язані з ним (завдяки параметру on_delete = models.CASCADE). Також ця таблиця міститиме в собі назву акції, ціну, за якою було відкрито правочин, кількість акцій та дату з часом.

Третя – trade_close – список закритих угод. Містить у собі номер відкритої угоди як зовнішній ключ, ціну закриття, кількість акцій та дату з часом.

```
from django.db import models
import datetime
from django.utils import timezone

class users(models.Model):
    login=models.CharField(max_length=200)
    password=models.CharField(max_length=200)


class trade(models.Model):
    login=models.ForeignKey(users, on_delete=models.CASCADE)
    secur_name=models.CharField(max_length=200)
    price_open=models.DecimalField(max_digits=8, decimal_places=2)
    quantity = models.IntegerField(default=0)
    date = models.DateTimeField('date open')


class trade_close(models.Model):
    open_trade=models.ForeignKey(trade, on_delete=models.CASCADE)
    price_close=models.DecimalField(max_digits=8, decimal_places=2)
    quantity = models.IntegerField(default=0)
    date = models.DateTimeField('date closed')
```

Рис.3.8.1 Фрагмент коду

Необхідність створення двох таблиць замість однієї виникла через те, що угоди можуть закриватися частково, з часом. Або взагалі не закриватиметься довгий період часу.

Далі за допомогою команди Python manage.py makemigrations journal фіксуємо список змін у базі даних, а потім за допомогою команди migrate записуємо таблиці до бази.

3.5 Створення представлень та шаблонів

Перед створенням уявлення створимо файли шаблонів - клієнтську частину. Усього їх буде 5. Зробимо ми це мовою html.

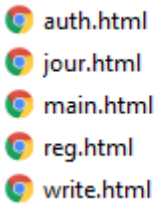

Рис.3.9 Шаблони html

У нас буде п'ять функцій уявлень – стільки ж, скільки шаблонів, які ми вказали у файлі urls.py. Насамперед імпортуємо наші моделі з файлуmodels.py.

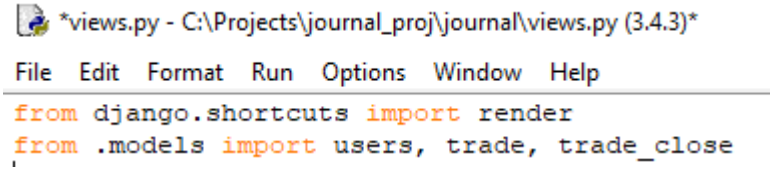

Рис.3.10 Імпорт моделей

Створимо функцію reg, яка буде відповідати за реєстрацію нових користувачів. Як аргумент вона прийматиме POST-запит, який виходитиме з клієнтської частини. Потім ми надаємо змінним значення, які ми отримуємо із запиту методом get(). У лапках при цьому вказуємо назву форми введення (у нас буде 2 форми введення: одна для введення логіну, яку ми назвемо login, а інша для пароля, яку назвемо password).

Далі робимо перевірку - якщо вхідний запит не порожній, то потім перевіряємо чи логін і пароль більше 5 символів чи ні. Якщо ця перевірка проходить, тоді за допомогою циклу for перевіряємо, чи є в таблиці users(усі записи якої ми на самому початку присвоюємо змінної f) логіни з такою ж назвою. Якщо так, то виведемо повідомлення про те, що такий логін вже використовується, інакше – виведемо повідомлення про успішну реєстрацію та зробимо запис до таблиці users.

Повідомлення буде передаватися до змінної s, яка в шаблоні матиме ім'я success. І вкажемо, що ця функція уявлення пов'язана із шаблоном reg.html, який ми зараз заповнимо.

Відкриємо reg.html, де створимо 2 форми для введення, які назвемо login та password, а також кнопку для надсилання даних на сервер. У самому верху ми вказуємо метод POST, трохи нижче додаємо змінну success, що пересилається, уклавши її в подвійні фігурні дужки. Також додамо кнопку «До вікна авторизації», яка перемістить нас за адресою, яка міститиме це вікно.

Оскільки ми ще не запустили сервер, змінні, що передаються, не сприймаються браузером, як теги Джанго.

Тепер створимо функцію уявлення для авторизації.

У змінні log і pas ми, як і у разі реєстрації записуємо значення полів для введення логіну і пароля. Створюємо булевську змінну auth_f, в якій буде міститися стан авторизації - авторизований користувач чи ні. Далі також ми перевіряємо, чи запит порожнім. Якщо ні, то намагаємося знайти в таблиці users елемент, який містить таку комбінацію логіна-паролю, якщо все відбулося успішно, то виводимо повідомлення про успішну авторизацію та змінну auth_f

присвоюємо значення True. Якщо раптом така комбінація не була знайдена в базі, виведеться відповідне повідомлення.

```
def reg(request):
    log=request.POST.get('login')
    pas=request.POST.get('password')
    print(log,'      ', pas)
    f = users.objects.all()

    if log != None:
        if len(str(log)) >= 5 and len(str(pas))>=5:
            uniq=True
            kolvo=f.count()
            for i in range(0,kolvo):
                if str(log)==str(f[i].login):
                    uniq=False
            if uniq == False:
                s='Данный логин уже используется '
            else:
                s='Успешная регистрация'
                new_user=users(login=log, password = pas)
                new_user.save()
        else:
            s='Длина логина и пароля должна быть больше 6 символов'
    else:
        s= ''
 #   print('f =  ', f.password)
 #   print(f[1].login)
    return render(request, 'reg.html',{'success':s})
```

Рис.3.11 Функція представлення reg

Повідомлення про успіх/невдачу передаємо в клієнтську частину змінної s1, яка там буде називатися succ_auth, а змінну з результатом передамо з таким же ім'ям. Також вкажемо шаблон, який відповідає цій функції – auth.html, який ми зараз заповнимо.

Шаблон буде аналогічним до шаблону з реєстрацією. Єдина відмінність – додамо кнопку «Зареєструватися», яка направить нас до вікна з реєстрацією. Також додамо кнопку головне меню, яка стане доступною для користувача тільки в тому випадку, якщо він буде авторизований - змінна стану auth_f, яку ми передаємо з функції представлення, дорівнюватиме True.

Тепер зробимо функцію головного меню, яку назвемо main. Складатиметься з одного рядка, де вкажемо, що він відповідає шаблону main.html:

```
def main (request):
    return render(request, 'main.html')
```

```
<h2>Регистрация</h2>

<form method="POST"> {% csrf_token %}
{{success}}

<br /><br />
 <label>
   Логин:<br />
   <input name="login"
     value="введите логин" />
 </label><br /><br />

 <label>
   Пароль:<br />
   <input name="password"
     value="введите пароль" />
 </label><br /><br />

 <input type="submit" value="Зарегистрироваться" />
  </form>

 <form action="http://127.0.0.1:8000/auth" method="GET">
  <input type="submit" value="К окну авторизации" />

  </form>
```

Рис.3.12 Шаблон для функції reg

```
def authoriz(request):
    log=request.POST.get('login')
    pas=request.POST.get('password')
    print(log,'      ', pas)
    auth_f=False
    if log != None:
        try:
            global user
            user = users.objects.get(login = log, password = pas)
            sl = 'Успешная авторизация. Пройдите в главное меню'
            auth_f = True
        except:
            sl='Неверный логин или пароль'
    else:
        sl=''

    return render(request, 'auth.html', {'auth_f':auth_f, 'succ_auth': sl})
```

Рис. 3.13 Функція подання для авторизації

```
<form action="http://127.0.0.1:8000/reg" method="GET">
   <input type="submit" value="Зарегистрироваться" />
   </form>

<form action="http://127.0.0.1:8000/main" method="GET">
 {% if auth_f == True %}
   <input type="submit" value="Главное меню" />
    {% endif %}
```

Рис.3.14 Шаблон для авторизації

Зробимо шаблон головного меню. Воно складатиметься з трьох кнопок – Заповнити журнал, Подивитися журнал, Вийти.

```
<form action="http://127.0.0.1:8000/write"
 method="GET">

 <input type="submit" value="Заполнить журнал" />

  </form>

   <form action="http://127.0.0.1:8000/jour"
 method="GET">

 <input type="submit" value="Посмотреть журнал" /

  </form>

   <form action="http://127.0.0.1:8000/auth"
 method="GET">

 <input type="submit" value="Выйти" />
```

Рис.3.15 Шаблон головного меню

Кнопка виходу посилатиметься на вікно з авторизацією.

Повернемося до уявлень та створимо функцію для заповнення журналу. Оскільки у нас на одній сторінці буде форма для заповнення як відкритих угод, так і ні, то розіб'ємо нашу функцію на дві частини.

У першій частині ми будемо обслуговувати запити для нових угод. Тут так само, як і у випадку з реєстрацією – якщо запит не порожній, вичленовуємо з нього назву акції, ціну, за якою було відкрито правочин, кількість куплених угод, дату та час (їх ми підсумовуємо, тому що у нас в таблиці поле типуdatetime - приймає і дату і час відразу, тобто поділ не потрібно). Потім записуємо дані до таблиці trade.

```
def writing(request):
    if request.POST.get('price_open') != None:
        s_name=request.POST.get('secur_name')
        price_op=request.POST.get('price_open')
        quan_op=request.POST.get('quantity_open')
     #   date_op=(request.POST.get('calendar_open')).strftime('%Y-%m-%d')+(reques
        date_op=(request.POST.get('calendar_open'))+' '
                                    +(request.POST.get('calendar_time_open'))
     #   print(price_op, ' quan  ',quan_op,' dat   ',date_op)
        trade_op=user.trade_set.create(secur_name = s_name, price_open=price_op,
                                       quantity = quan_op, date= date_op)
        trade_op.save()
```

Рис.3.16 Функція заповнення журналу

У другій частині ми опрацьовуємо запити для форми, де закривається відкрита раніше угода.

Тут все аналогічно до першої частини. Тільки змінній tr_list надається список усіх відкритих угод, що належать авторизованому користувачеві, який передається в клієнтську частину, де користувач зможе вибрати яку угоду закрити. Результат повернеться у запиті до змінної id_trade, значення якої ми надамо змінній id_tr.

```
  tr_list=user.trade_set.all()
  if  request.POST.get('price_close') != None:
      id_tr=request.POST.get('id_trade')
      price_cl=request.POST.get('price_close')
      quan_cl=request.POST.get('quantity_close')
      date_cl=(request.POST.get('calendar_close'))+' '
                      +(request.POST.get('calendar_time_close'))
#      date_cl = datetime.strptime(date_cl, '%Y-%m-%d %H:%M')
      print(price_cl, ' quan  ', quan_cl, ' dat   ',date_cl)
      trade_op=user.trade_set.get(pk=id_tr)

      trade_cl=trade_op.trade_close_set.create(price_close=price_cl,
                                               quantity = quan_cl, date= date_cl)
      trade_cl.save()
  return render(request, 'write.html', {'op_trade_list': tr_list})
```

Рис. 3.17 Друга частина функції для заповнення журналу

Цю функцію ми зв'яжемо із шаблоном write.html.

Для форми відкриття угоди потрібно 5 форм для введення інформації – назва акції, ціна, кількість, дата та час. Також кнопка «Надіслати» - для передачі введених даних на сервер.

```
<h2>Форма для заполнения журнала</h2>
  <form method="POST"> {% csrf_token %}
  <h3>Открыть сделку</h3>

<label>
      Название акции:<br />
      <input name="secur_name"
        value="" />
   </label><br />
  <label>
      Цена акции на момент покупки:<br />
      <input name="price_open"
        value="" />
   </label><br />
    <label>
      Количество акций:<br />
      <input name="quantity_open"
        value="" />
   </label><br />
      <label>
      Выберите дату и время:<br />
      <input type="date" name="calendar_open"
        value="" max="today" />

       <input type="time" name="calendar_time_open"
        value="" max="today" />
   </label><br />

   <input type="submit" value="Отправить" />
    </form>
```

Рис.3.18 Шаблон заповнення журналу

Далі створимо форму для закриття угоди. Тут все аналогічно, тільки ця форма приймає список відкритих угод, у тому числі з допомогою циклу формується набір номерів відкритих угод для авторизованого користувача. Робиться це за допомогою html-тегу <select>. Кожен варіант пишеться у вкладений тег <option>.

```
    <br/>
    <form method="POST"> {% csrf_token %}
     <h3>Закрыть сделку</h3>

  Выберите номер открытой сделки<br />
<select name = "id_trade">
    {% for i in op_trade_list %}
  <option>{{ i.id }}</option>
  {% endfor %}
    </select>
<br />
    <label>
        Цена акции на момент продажи:<br />
        <input name="price_close"
          value="" />
      </label><br />
       <label>

        Количество акций:<br />
        <input name="quantity_close"
          value="" />
      </label><br />

        <label>
        Выберите дату и время:<br />
        <input type="date" name="calendar_close"
          value="" max="today" />

         <input type="time" name="calendar_time_close"
          value="" max="today" />
      </label><br />

      <input type="submit" value="Отправить" />
```

Рис.3.19 Шаблон заповнення журналу

Створимо тепер останню функцію , яка дозволить нам прочитати журнал. Зв'яжемо її з шаблоном jour.html, куди відправлятимемо змінну tr_op, якій ми надамо всі угоди, що належать авторизованому користувачеві.

```
def read_jour(request):
    tr_op=user.trade_set.all()
    return render(request, 'jour.html', {'trade_op': tr_op})
```

Рис. 3.20 Функція для читання журналу

Заповнимо тепер шаблон jour.html. Виводити вміст журналу ми будемо у вигляді таблиці. Таблиця задається за допомогою тега <table>, після якого задаємо параметри таблиці, її ширину.

Рядки таблиці задаються за допомогою тега <tr>, а стовпці за допомогою <th> (виділяє вміст жирним) або <td>. Перший рядок у нас буде «константою» - у ньому будуть назви 8 стовпців нашої таблиці.

```
<table border="1" width="12%" cellpadding="2">
 <tr>
        <th>Номер сделки</th>
        <th>Название акции</th>
        <th>Цена открытия</th>
        <th>Количество</th>
        <th>Дата и время открытия</th>
        <th>Цена закрытия</th>
        <th>Количество</th>
        <th>Дата и время закрытия</th>
</tr>
```

Рис. 3.20 Шаблон читання журналу

Тепер за допомогою циклу здійснимо перебір відкритих угод. У кожному рядку виводитиметься вміст кожної відкритої угоди, там же буде здійснюватися перевірка, чи є у цієї відкритої угоди якісь відповідні їй закриті угоди. Якщо так, то виводиться перша відповідна угода (всі інші будуть виведені згодом, тому що для них потрібний новий рядок). Якщо ні, то стовпці, які відповідають закритим угодам, заповнюються порожнім для цього рядка.

```
{% for i in trade_op %}
            <tr class='success1'>
                <td>{{ i.id }}</td>
                <td>{{ i.secur_name }}</td>
                <td>{{ i.price_open }}</td>
                <td>{{ i.quantity }}</td>
                <td>{{ i.date }}</td>
                {% if i.trade_close_set.count > 0 %}
                <td>{{ i.trade_close_set.all.0.price_close}}</td>
                <td>{{ i.trade_close_set.all.0.quantity }}</td>
                <td>{{ i.trade_close_set.all.0.date }}</td>
                {% else %}
                <td></td>
                <td></td>
                <td></td>
                {% endif %}
                </tr>
```

Рис.3.21 Шаблон читання журналу

Далі здійснюється перевірка. Якщо у цієї відкритої угоди є більше однієї закритої, то з допомогою циклу здійснюється їх перебір. Причому заповнювати починаємо з другого елемента, т.к. перший заповнюється за допомогою коду. Це

реалізується завдяки перевірці лічильника циклу forloop.counter. Стовпці для відкритих угод ми заповнюємо порожніми значеннями.

```
                {% if i.trade_close_set.count > 1 %}

                {% for i1 in i.trade_close_set.all %}
                {% if forloop.counter > 1 %}
            <tr class='success2'>
                <td></td>
                <td></td>
                <td></td>
                <td></td>
                <td></td>
                <td>{{ i1.price_close }}</td>
                <td>{{ i1.quantity }}</td>
                <td>{{ i1.date }}</td>
            </tr>
                {% endif %}
                {% endfor %}
                {% endif %}
    {% endfor %}
</table>
```

Рис.3.22 Шаблон читання журналу

Розділ 4. Осибливості програмної реалізації

4.1 Опис розроблених частин проекту

4.1.1 Модель даних

У таблиці Django бази даних описуються у вигляді моделей класів. Модель – це джерело даних. Він містить набір полів та поведінку даних, які зберігаються в базі. Так як Django слідує принципу DRY всі моделі визначаються в одному місці.

Django використовує принцип ORM, тому підтримуються такі принципи ОВП як успадкування, поліморфізм та інкапсуляції тощо. Опис моделі даних є аналогом SQL CREATE TABLE та має особливості:

– назви таблиць створюються автоматично з назви програми та назви моделі в нижньому регістрі:

– первинні ключі автоматично додаються;

– Django додає "_id" до назви зовнішнього ключа.

Створена в ERwin модель бази даних описується за допомогою класів Python. Кожен клас успадковує базовий клас Django «Model», який у свою чергу містить методи додавання, видалення та зміни даних.

Діаграма класів представлена на Рис.4.1.

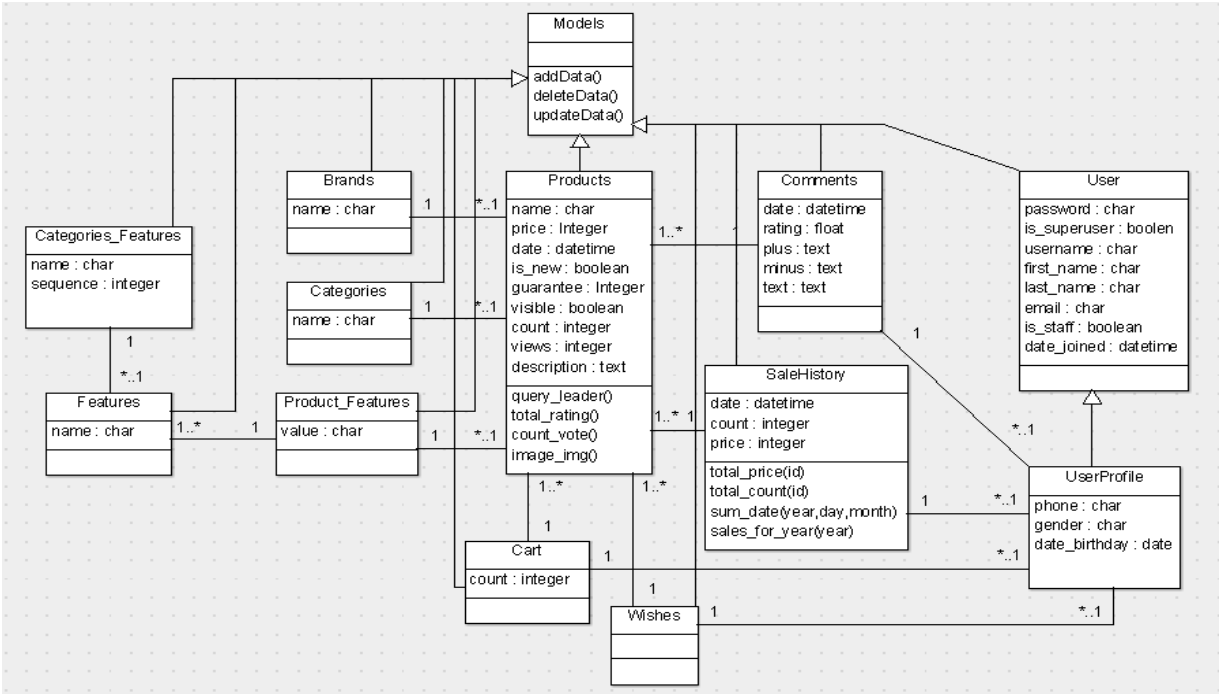

Рис.4.1 – Діаграма класів

Представлені класи:

– class Categories (models.Model) - описує таблицю «Категорії товарів»;

– Brands(models.Model) – описує таблицю «Бренди»;

– Class Categories_Features (models.Model) - описує таблицю "Характеристики товарів";

– class Features (models.Model) – визначає таблицю «Характеристики товарів»;

– class Products(models.Model) – описує таблицю "Товари";

– class Product_Features(models.Model) - описує таблицю «Товар-Характеристики товарів»;

class UserProfile(User) описує таблицю «Профіль користувача»;

class Wishes(models.Model) описує таблицю "Перелік бажань»;

class SalesHistory(models.Model) – описує таблицю «Історія покупок»;

class Cart(models.Model) – описує таблицю «Кошик»;

class Comments(models.Model) описує таблицю

"Коментарі";

Крім моделей даних для роботи веб-додатка знадобляться методи класів цих моделей. Виділимо необхідні методи:

query_leader(cls), метод класу Products, визначає топ 5 товарів, що є лідерами з продажу;

total_rating(self), метод об'єкта класу Products, обчислює загальний рейтинг певного товару;

count_vote(self), метод об'єкта класу Products, визначаєкількість голосів певного товару;

image_img(self), метод об'єкта клас Products, повертає шлях до зображення у необхідному для html вигляді;

total_price(cls, id), метод класу SalesHistory, визначає загальну суму покупок певного користувача;

total_count(cls, id), метод класу SalesHistory, визначає загальну кількість

куплених товарів певного користувача;

sum_date(cls, year, day, month), метод класу SalesHistory, визначає загальну вартість проданих товарів у визначений день;

sales_for_year(cls, year), метод класу SalesHistory, повертає словник даних із загальною виручкою по місяцях протягом року;

4.1.2 Структура сторінок веб-програми

Базовим шаблоном є base.html. Він містить у собі шапку сайту, з логотипом, пошуковим блогом та навігацією. Доступ до цих елементів необхідний з будь-якої частини сайту, тому базовий шаблон завжди використовуватиметься при рендерингу сторінки. Його успадковують інші шаблони (Рис.4.2).

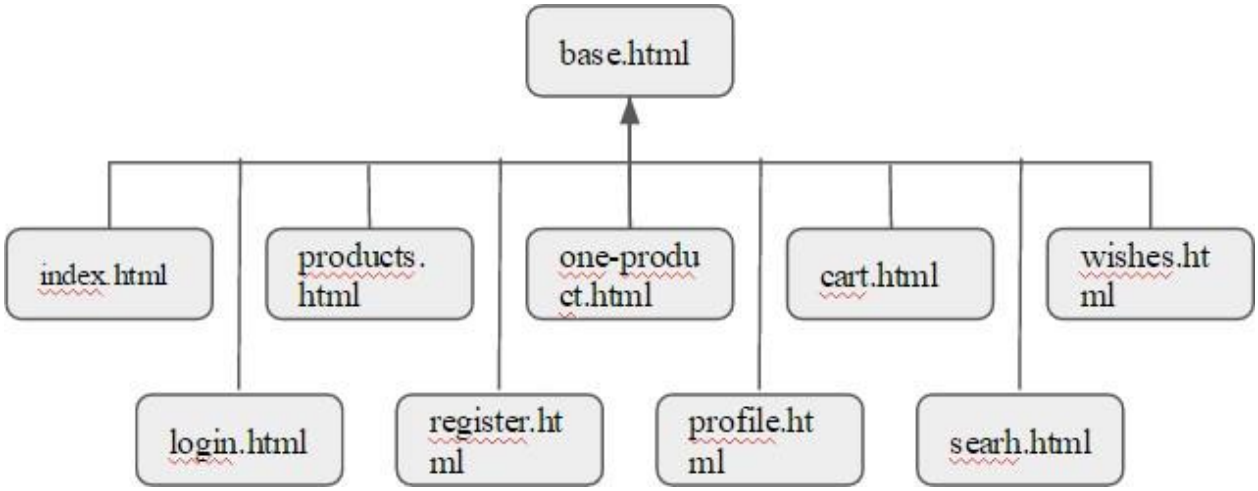

Рис.4.2 Успадкування шаблонів

На Рис.4.2 представлені такі шаблони:

– index.html, містить у собі структуру головної сторінки;

– products.html, є каталогом товарів, в залежності відпараметрів вибірки відображає потрібні товари;

– one-product.html, сторінка з детальною інформацією про конкретний товар, характеристики, рейтинг, відгуки і т.д.;

– cart.html, користувальницькакошик з можливістю змінюватикількість товарів, видаляти чи оформити купівлю;

– wishes.html, список бажаних товарів користувача;

– login.html, сторінка авторизації користувача;

– register.html, сторінка реєстрації нового користувача;

– profile.html, профіль користувача, є можливість змінювати інформацію, змінити пароль, переглянути історію покупок;

– search.html, сторінка з результатами пошуку.

4.1.3 Функціонал та блоки сайту

Проект можна розділити на кілька тематичних елементів, які в сукупності становлять одне ціле. Далі розглянемо докладно функціонал та блоки сайту.

Найважливішим елементом будь-якого інтернет-магазину є каталог товарів. Натисканням кнопки «до кошика» товар автоматично потрапляє у кошик покупця. Завдяки технології AJAX це відбувається без перезавантаження сторінки. Ліворуч можна спостерігати меню, в якому товари розділені на категорії та бренди. При натисканні на один із пунктів на сторінці будуть показані відповідні товари. Товарів може бути багато, у такому разі вони будуть поділені на сторінки.

```
class Product(models.Model):
    category = models.ForeignKey(Category,
                                 related_name='products',
                                 on_delete=models.CASCADE)

    name = models.CharField(max_length=150, db_index=True)
    slug = models.CharField(max_length=150, db_index=True, unique=True)
    image = models.ImageField(upload_to='product/%Y/%m/%d', blank=True)
    description = models.TextField(max_length=1000, blank=True)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    available = models.BooleanField(default=True)
    created = models.DateTimeField(auto_now_add=True)
    uploaded = models.DateTimeField(auto_now=True)
```

Такий функціонал досягається використанням модулів "paginate" та "bootstrap-paginate", які вбудовані в Laravel. Другий модуль використовується для налаштування модуля will_paginate на використання Bootstrap – фреймворку, що відповідає за адаптивну верстку сторінок. 40 Рисунок 4 – Каталог товарів Також на головній сторінці розташований слайдер (Малюнок 5), який в автоматичному режимі показує картинки з текстом для презентації магазину та розповіді про його переваги. Малюнок 5 - Слайдер На стартовій сторінці розташований блок із рекомендованими товарами, який дублює функцію каталогу.

```
class Category(models.Model):
    name = models.CharField(max_length=100, db_index=True)
    slug = models.SlugField(max_length=100, unique=True)

    class Meta:
        ordering = ('name',)
        verbose_name = 'Категория'
        verbose_name_plural = 'Категории'

    def __str__(self):
        return self.name
    def get_absolute_url(self):
        return reverse('myshop:product_list_by_category', args=[self.slug])
```

Кошик використовується для додавання до нього товарів, які користувач планує придбати. У самому кошику є можливість збільшити чи зменшити кількість товарів конкретної продукції та видалити товар із кошика повністю. Кошик з товарами можна спостерігати на малюнку 6. Малюнок 6 – Кошик з товарами У каталозі кожного товару є кнопка «У кошик», саме він дозволяє додати його в кошик. Здійснюється ця опція без повного перезавантаження сторінки за рахунок застосування технології AJAX.

Опишемо клас Cart та відразу задамо ініціалізацію кошика.

```
class Cart(object):
    def __init__(self, request):
        self.session = request.session
        cart = self.session.get(settings.CART_SESSION_ID)
        if not cart:
            # зберігаємо ПУСТИЙ кошик у сесії
            cart = self.session[settings.CART_SESSION_ID] = {}
        self.cart = cart
```

Перебираємо товари в кошику та отримуємо товари з бази даних. Потім отримуємо товари та додаємо їх у кошик. Це все реалізовано в наступному коді.

```
    def __iter__(self):
        product_ids = self.cart.keys()
        products = Product.objects.filter(id__in=product_ids)
        cart = self.cart.copy()
        for product in products:
            cart[str(product.id)]['product'] = product
        for item in cart.values():
            item['price'] = Decimal(item['price'])
            item['total_price'] = item['price'] * item['quantity']
            yield item
```

Окремою функцією додаємо товар у кошик чи оновлюємо його кількість:

```
    def add(self, product, quantity=1, update_quantity=False):
        product_id = str(product.id)
        if product_id not in self.cart:
            self.cart[product_id] = {'quantity': 0,
                                     'price': str(product.price)}
        if update_quantity:
            self.cart[product_id]['quantity'] = quantity
        else:
            self.cart[product_id]['quantity'] += quantity
```

```
            self.save()
```

Для збереження товару описуємо функцію:

```
        def save(self):
            self.session.modified = True
```

Видаляємо товар:

```
        def remove(self, product):
            product_id = str(product.id)
            if product_id in self.cart:
                del self.cart[product_id]
                self.save()
```

Отримуємо загальну вартість:

```
        def get_total_price(self):
            return sum(Decimal(item['price']) * item['quantity'] for item in self.cart.values())
```

Очищаємо кошик у сесії наступним чином:

```
        def clear(self):
            del self.session[settings.CART_SESSION_ID]
            self.save()
```

4.1.4 Реєстрація користувача

У меню є кнопка "Вхід", яка перенаправляє користувача до форми авторизації . Тут є кілька опцій – стандартний вхід через електронну пошту, вказану при реєстрації, та швидкий вхід через один із представлених сервісів. При авторизації у веб-застосунку використовується перевірка правильності заповнення полів. Наприклад, пароль не може бути меншим за шість символів, а поле «email» не можна залишити порожнім. Email є унікальним ім'ям, тому зареєструвати двічі одну електронну пошту не вийде. Для безпеки паролів у Laravel застосовується метод hash. При реєстрації потрібно активувати свій обліковий запис і підтвердити електронну пошту.

Шаблон Laravel включає клас Auth\VerificationController, який містить необхідну логіку для надсилання посилань підтвердження email та перевірки

електронної пошти. Щоб зареєструвати потрібні роути для цього контролера, потрібно передати опцію verify методу Auth::routes. Можна використовувати посередників роуту (Route middleware), щоб дозволити доступ тільки перевіреним користувачам до цього роуту. Laravel поставляється з посередником verified, який визначений Illuminate\Auth\Middleware\EnsureEmailIsVerified. Оскільки цей посередник вже зареєстрований у app/Http/Kernel.ph, потрібно підключити посередника до визначення роуту. Щоб зареєструвати потрібні роути для цього контролера, потрібно передати опцію verify методу Auth::routes. Можна використовувати посередників роуту (Route middleware), щоб дозволити доступ тільки перевіреним користувачам до цього роуту. Laravel поставляється з посередником verified, який визначений Illuminate\ Auth\Middleware\EnsureEmailIsVerified. Оскільки цей посередник вже зареєстрований у app/Http/Kernel.ph, потрібно підключити посередника до визначення роуту. Щоб зареєструвати потрібні роути для цього контролера, потрібно передати опцію verify методу Auth::routes. Можна використовувати посередників роуту (Route middleware), щоб дозволити доступ тільки перевіреним користувачам до цього роуту. Laravel поставляється з посередником verified, який визначений Illuminate\Auth\Middleware\EnsureEmailIsVerified. Оскільки цей посередник вже зареєстрований у app/Http/Kernel.ph, потрібно підключити посередника до визначення роуту.

Також адміністратор має можливість додавання статей на сайт. Ця функція допоможе не тільки урізноманітнити функціонал та контент веб-додатку, але й дасть можливість просувати товари, допомагати користувачеві вибрати потрібний товар. Даний функціонал також допоможе зробити із звичайного інтернет-магазину сайт, де користувачі зможуть не лише купувати товари, а й отримувати корисну інформацію.

4.2 Розробка інтерфейсу користувача

Взявши до уваги вимоги, представлені в розділі проектування, був розроблений інтерфейс користувача для клієнтської частини програми. Розглянемо основні сторінки веб-програми.

Інтерфейс головної сторінки представлений на Рис.4.3.

Рис.4.3. Головна сторінка сайту

Інтерфейс умовно можна поділити на три частини. Верхня – шапка сайту, центральна – слайдер та панель вибору категорії товару та нижня – блок із лідерами продажів. Верхня частина сайту включає основні функції та елементи навігації веб-програми. Там міститься інформація про товари, що знаходяться в кошику та їх сумарна вартість. Далі йде форма пошуку необхідного товару. Під нею знаходяться посилання на «Кошик», «Список бажань», "Профіль", а також кнопка виходу зі свого профілю. Також тут розташовуються кнопки переходу до каталогу товарів з можливістю подивитися новинки чи лідери продажу магазину.

Центральна частина сайту представлена анімованим слайдером з цікавими пропозиціями та знижками. Правіше знаходиться панель вибору категорії товарів, яка є посиланням на каталог товарів.

Нижня частина представлена панеллю з товарів магазину, що найбільше продаються. Є можливість відразу додати товар у кошик або ж перейти до їхнього детального опису.

Інтерфейс каталогу товарів представлений на Рис.4.4.

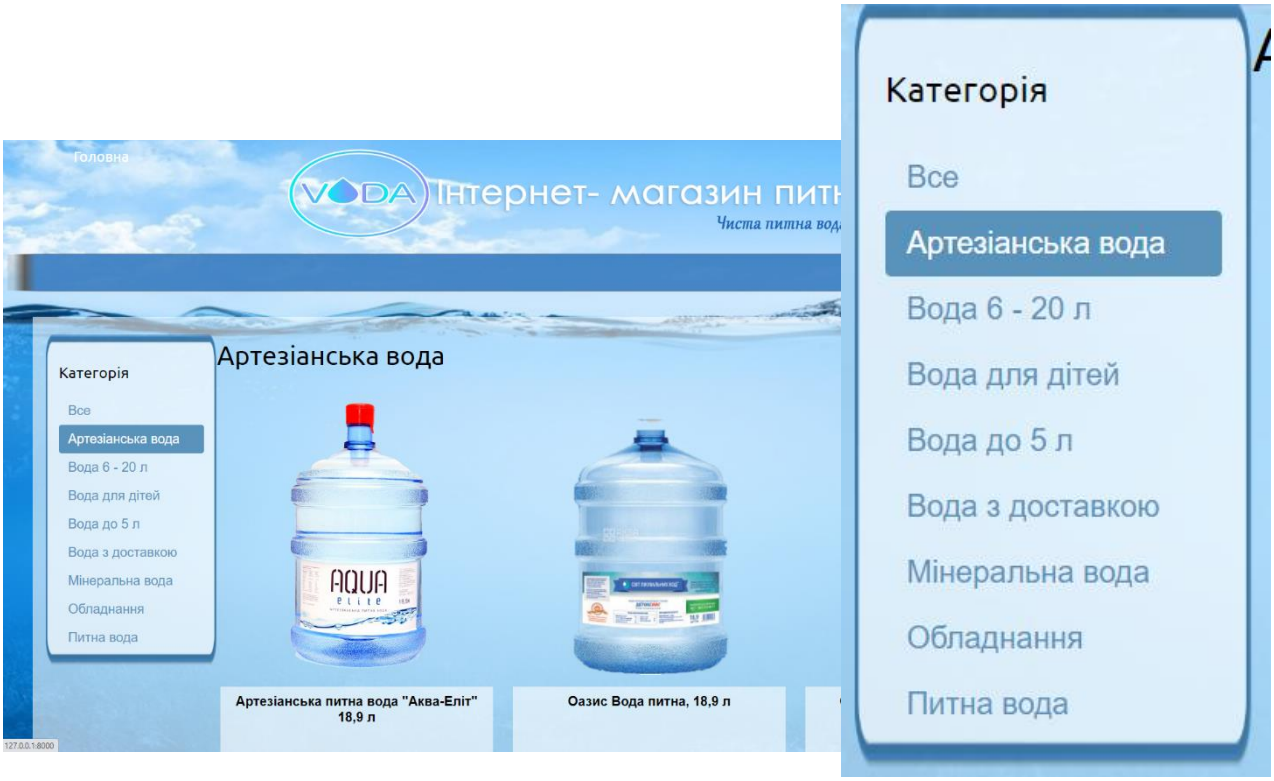

Рис.4.4. Каталог товарів

Тут верхня частина інтерфейсу залишається незмінною. Панель із категоріями товарів поповнюється параметрами для вибірки, а центральна частина є каталогом товарів.

Товари виводяться посторінково. Є меню вибору сортування товарів. Спеціальними маркерами відзначаються новинки або лідери продажу магазину. Є іконки, що свідчать про перегляди та коментарі конкретного товару. Товар, який було оцінено користувачами має рейтинг.

Інтерфейс сторінки певного товару представлений на Рис.4.5.

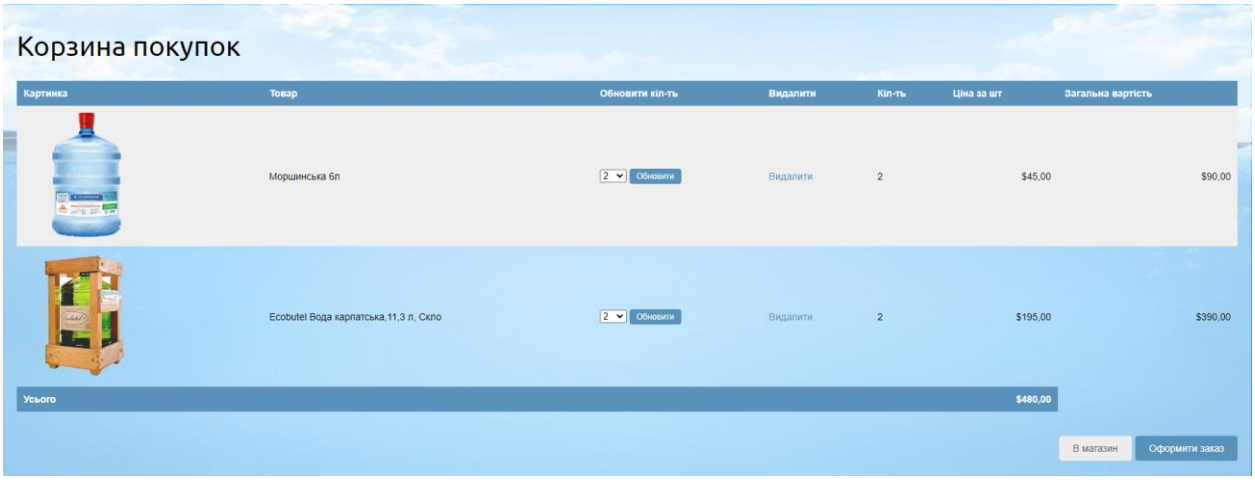

Рис.4.5 Сторінка товару

Сторінка товару надає детальну інформацію про конкретний товар. У верхній частині представлено найменування, зображення, вартість, рейтинг, гарантія товару, а також кнопки додавання до кошика та до списку бажань.

Далі розташована панель із перемикачем. У розділі «Опис» представлені властивості та характеристики товару, а в розділі «Коментарі» можна переглянути відгуки інших покупців або залишити свій.

4.3 Тестування

Тестування програмного забезпечення – процес виявлення помилок у програмному забезпеченні (ПЗ). Існуючі на сьогоднішній день методи тестування ПЗ не дозволяють однозначно та повністю усунути всі дефекти та помилки та встановити коректність функціонування аналізованої програми особливо у закритих приватних програмах. Тому всі існуючі методи тестування діють у рамках формального процесу перевірки досліджуваного або програмного забезпечення, що розробляється [13].

Такий процес формальної перевірки чи верифікації може довести, що дефекти відсутні, з погляду використовуваного методу – тобто немає можливості точно встановити чи гарантувати відсутність дефектів у

програмному продукті з урахуванням людського чинника, присутній всіх етапах життєвого циклу ПЗ.

Існує безліч підходів до вирішення завдання тестування та верифікації ПЗ, але ефективне тестування складних програмних продуктів – це процес надзвичайно творчий, що не зводиться до дотримання суворих і чітких процедур або створення таких.

Одним із підходів до тестування є методологія функціонального тестування.

Функціональне тестування - це тестування ПЗ з метою перевірки реалізованості функціональних вимог, тобто проведення функціонального тестування дозволяє перевірити здатність інформаційної системи в певних умовах вирішувати завдання, потрібні користувачам [13].

Функціональне тестування є одним із ключових видів тестування, завдання якого – встановити відповідність розробленого програмного забезпечення вихідним функціональним вимогам замовника.

Основні переваги функціонального тестування:

– функціональне тестування ПЗ повністю імітує фактичне використання системи;

– дозволяє своєчасно виявити системні помилки ПЗ і тим самим уникнути безлічі проблем при роботі з ним надалі:

– економія з допомогою виправлення помилок більш ранньому етапі життєвого циклу ПО.

Виконаємо функціональне тестування веб-додатки прогнозування та продажу товарів.

Таблиця 9 - Тестування клієнтської частини

№ п/п	Призначення тесту	Опис тесту

1	Тестування форми «Реєстрація»	Після заповнення форми коректними даними та натискання на кнопку «Реєстрація» відбувається реєстрація нового користувача
2	Тестування форми «Авторизація»	Після заповнення форми коректними даними та натискання на кнопку «Авторизація» відбувається авторизація користувача
3	Тестування форми «Параметри вибірки»	Після заповнення форми необхідними параметрами та натисканням на кнопку «Знайти» відбувається вибірка товарів, що відповідають вимогам користувача
4	Тестування форми «Пошук»	Після введення пошукового запиту в поле для пошуку та натисканням на кнопку «Знайти» відбувається вибірка товарів відповідних пошуковому запиту
5	Тестування відкриттясторінки товарів	Залежно від обраного критерію товарів відкривається каталог товарів відповідний вибраному варіанту
6	Тестування сортування	Після вибору необхідного параметра сортування товар у каталозі повинен упорядкуватися відповідно до вибраного параметра
№ п/п	Призначення тесту	Опис тесту
7	Тестування додавання товару в кошик	Після натискання кнопки «До кошика» товар повинен бути доданий до корзини користувача
8	Тестування додавання товару в бажане	Після натискання кнопки «В бажане» товар має бути доданий до списку бажаних товарів користувача

9	Тестування форми «Зміна данихпрофілю»	Після заповнення форми коректними даними та натискання на кнопку «Ок» відбувається зміна даних користувача
10	Тестування форми "Зміна пароля"	Після заповнення форми коректними даними та натискання на кнопку «Ок» відбувається зміна пароля користувача
11	Тестування оформлення покупки	Після додавання товару до кошика та натискання кнопки «Оформлення покупки»відбувається перехід на сторінку для введення реквізитів оплати
12	Тестування виходу зсайту	Після натискання кнопки «Вийти» користувач має вийти із сайту

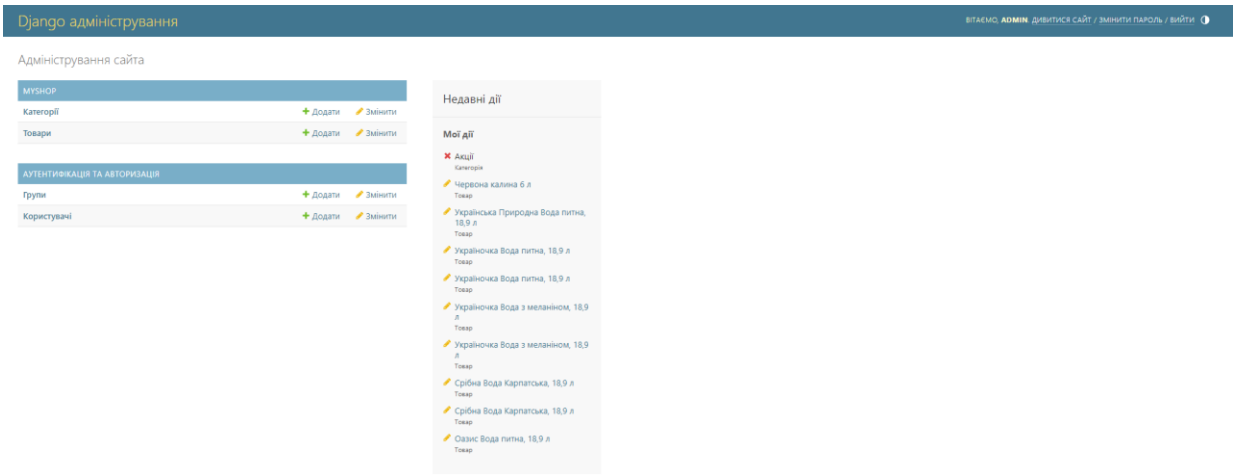

Рис.4.6. Тестування адміністративної панелі

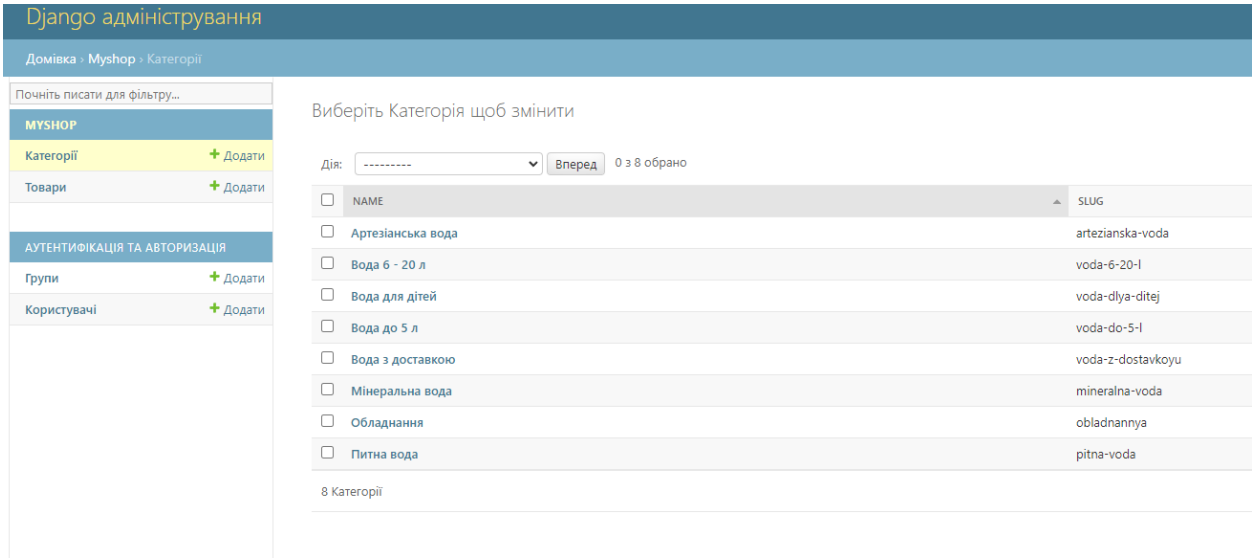

Рис.4.6. Зміна категорії

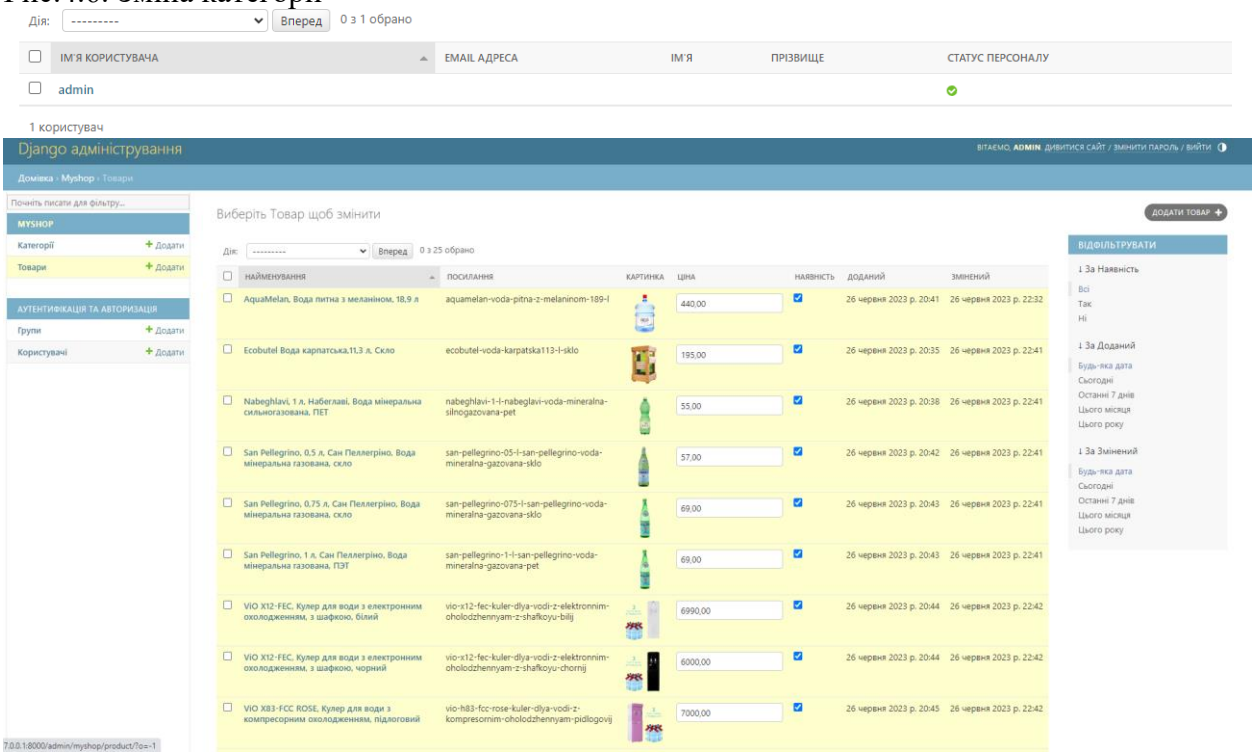

Рис.4.7. Зміни теми

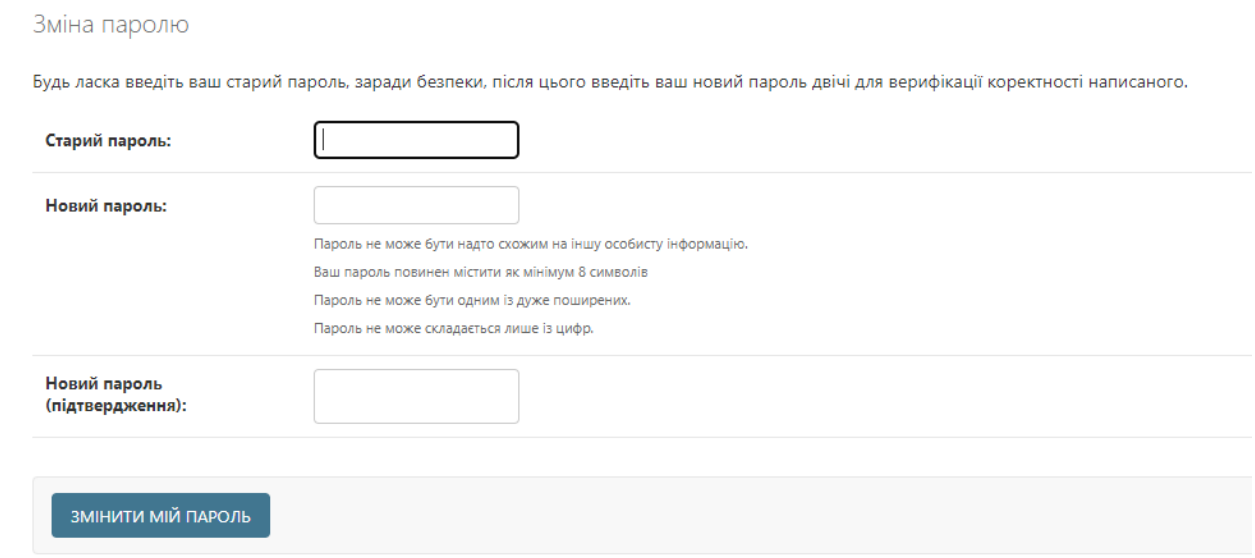

Рис.4.8. Зміна паролю

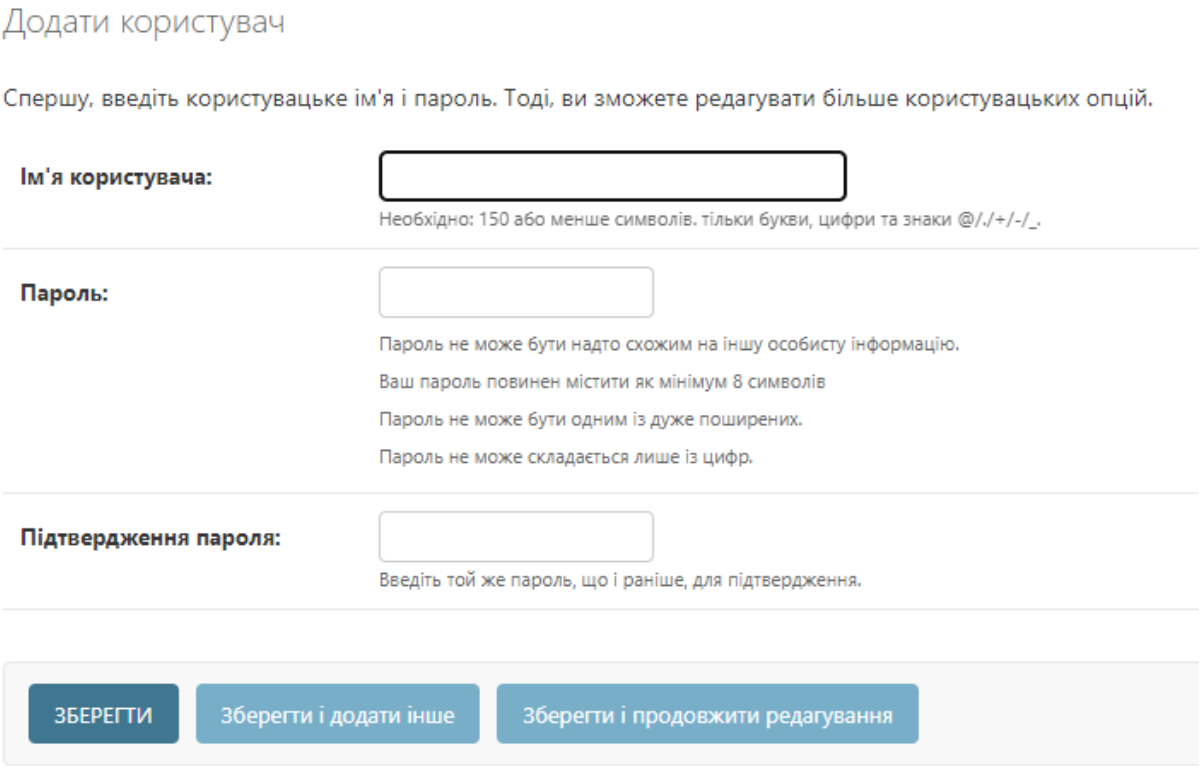

Рис.4.9. Додавання користувача

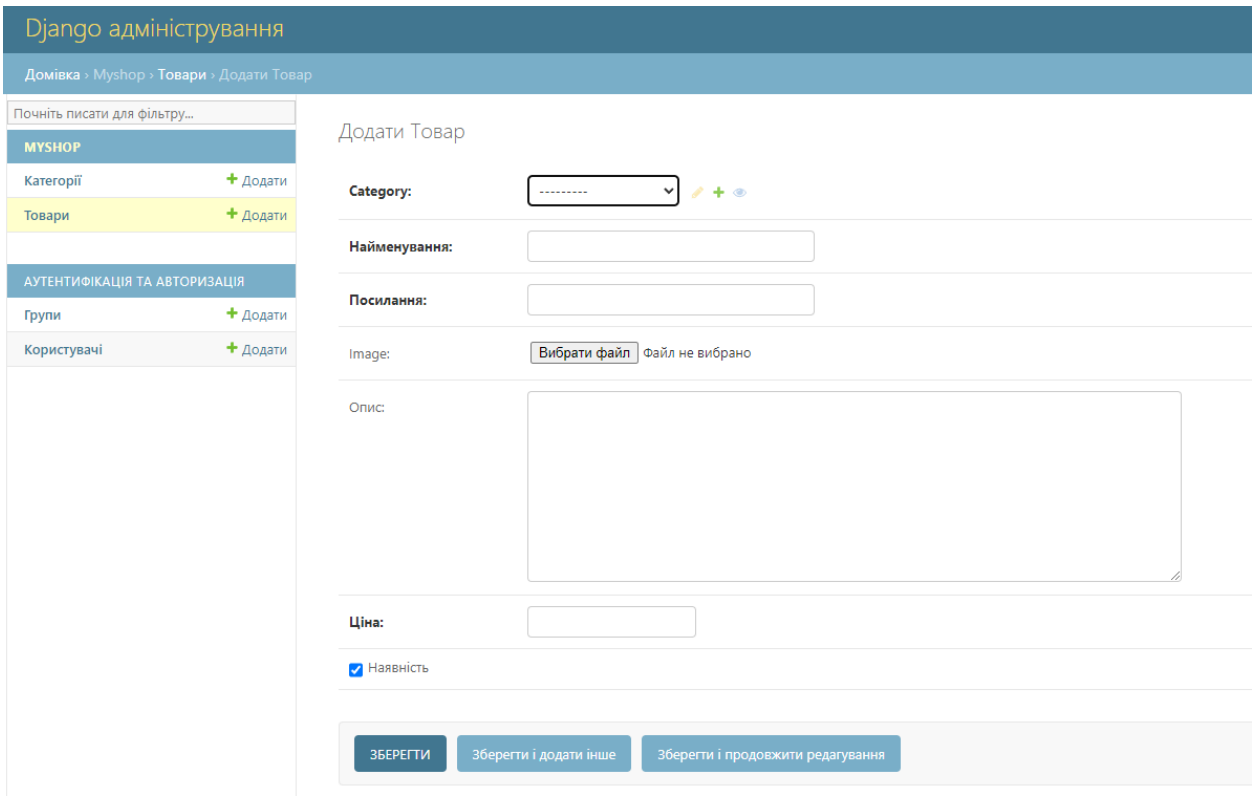

Рис.4.10 Додавання товару

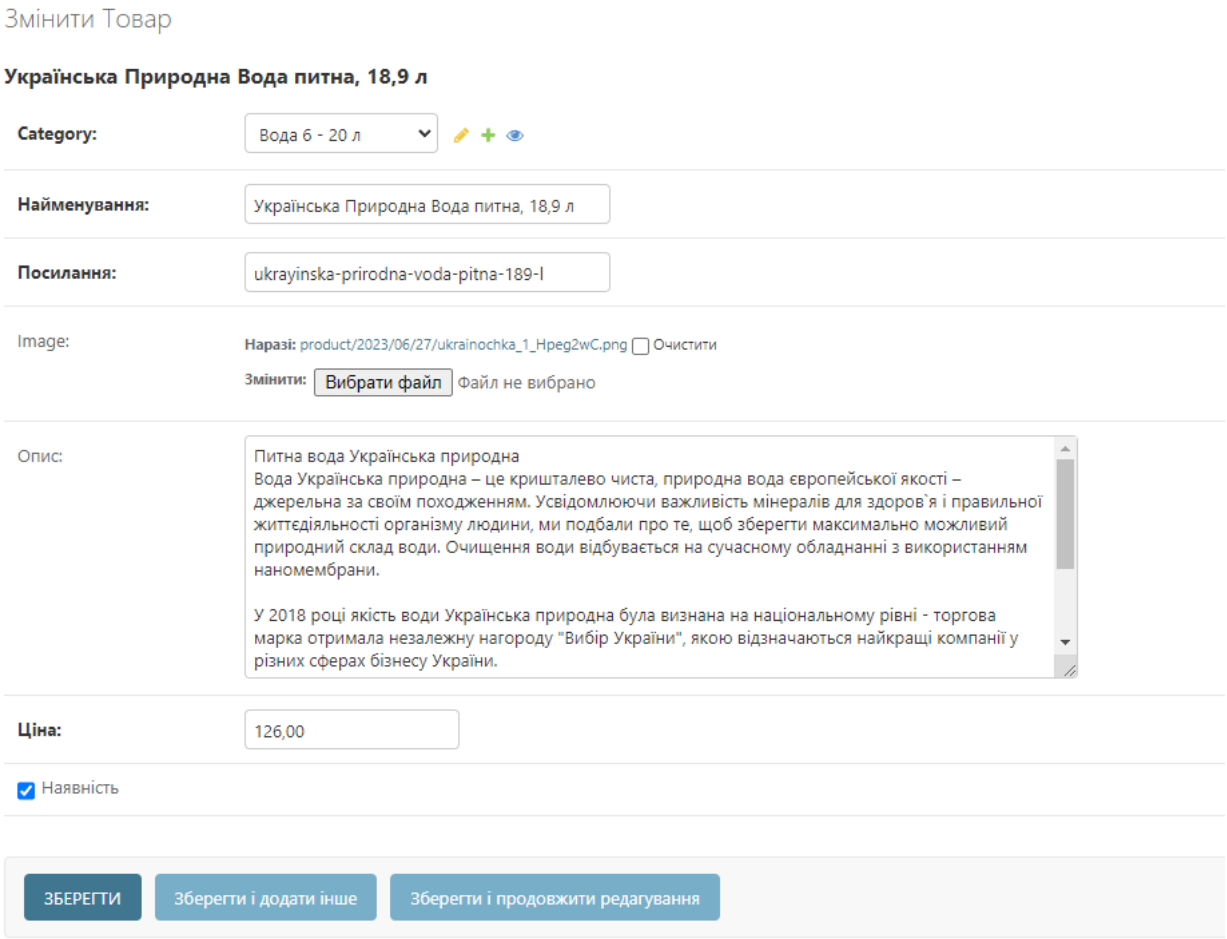

Рис.4.11 Зміна товару

Таблиця 10 – Тестування адміністративної панелі

№ п/п	Призначення тесту	Опис тесту
1	Тестування форми «Авторизація»	Після заповнення форми коректними даними та натисканняна кнопку «Авторизація» відбувається авторизація користувача
2	Тестування форми "Зміна пароля"	Після заповнення форми коректними даними та натискання на кнопку «Ок» відбувається зміна пароля користувача

№ п/п	Призначення тесту	Опис тесту
3	Тестування додаванняданих у базу даних	Після заповнення форми коректними даними та натискання на кнопку «Додати» відбувається додавання даних до бази даних
4	Тестування зміни даних у базі даних	Після заповнення форми коректними новими даними та натисканням на кнопку «Змінити» відбувається зміна даних до бази даних
5	Тестування видалення даних із бази даних	Після вибору даних длявидалення і натискання кнопки «Видалити» відбувається видалення даних із бази даних
6	Тестування форми "Створення нового користувача»	Після заповнення форми коректними даними та натискання на кнопку «Створити» відбувається створення нового користувача
7	Тестування зміни статусу персоналу	Після зміни статусу користувача у формі та натискання на кнопку «Застосувати» відбувається зміна статусу користувача
8	Тестування прогнозування продажу товарів	Після натискання на кнопку «Прогнозувати» відбуватися розрахунок даних та відображення графіка з прогнозом продажів на наступний період часу
9	Тестування виходу користувача з панелі	Після натискання кнопки «Вийти» користувач має вийти із сайту

	адміністратора	

За результатами тестування веб-додаток прогнозування та продажу товарів функціонує коректно.

Список використаних джерел

1 Коротка історія мов програмування. URL: http://younglinux.info/python/programminglanguage.php (Дата звернення: 12.06.2018).

2 Wikipedia, Python. URL: https://ua.wikipedia.org/wiki/Python (Дата звернення: 12.06.2018).

3 A Brief Timeline of Python. URL: http://python–history.blogspot.ru/2009/01/brief–timeline–of–python.html (Дата звернення: 12.06.2022).

4 Wikipedia, Історія мови програмування Python. URL: https://ua.wikipedia.org/wiki/Історія_програмування_Python (Дата звернення: 12.06.2022).

5 Короткий огляд мови Python. URL: http://www.helloworld.ru/texts/comp/lang/python/python2/index.htm(Дата звернення: 13.06.2022).

6 Форсьє Дж., Django. Розробка веб-додатків на Python, 2009. -456 с.

7 Історія Django. URL: https://djbook.ru/ch01s03.html (Дата звернення: 13.06.2022).

8 Python Django. URL:http://ep–z.ru/stroitelstvo/sayt/python/python–django (Дата звернення: 14.06.2022).

9 Переваги PostrgeSQL. URL: https://habr.com/post/282764/ (Дата звернення: 14.06.2022).