ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерно-інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету)

Кафедра прикладної математики і інформатики
(повна назва кафедри)

«До захисту допущено»
Завідувачка кафедри ПМІ


Ольга ДМИТРІЄВА
(підпис) (ім'я та прізвище)

"13" червня 2022 р.

Випускна кваліфікаційна робота
бакалавра
(освітній ступінь)

на тему «Розробка програмної системи для автоматизації торгівлі комп'ютерної техніки з можливістю конфігурування апаратної частини»
зі спецчастиною «Розробка програмних модулів, інтерфейсу користувача та бази даних засобами Python, JavaScript, HTML, CSS»

Виконав (-ла): студент (-ка) 4 курсу, групи КН-18
(шифр групи)

спеціальності (напряму підготовки) 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)

Усатий І. О. _____
(прізвище та ініціали) (підпис)

Керівник ст. викл. каф. ПМІ Євген ПАВЛОВСЬКИЙ _____
(посада, науковий ступінь, вчене звання, ім' я та прізвище) (підпис)

Рецензент зав. каф. КІ, к.т.н., доц. Сергій КОВАЛЬОВ _____
(посада, науковий ступінь, вчене звання, ім' я та прізвище) (підпис)

Нормоконтроль:


(підпис) к.т.н., доц. Ірина НАЗАРОВА

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Луцьк – 2022 р.

ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерно-інформаційних технологій та автоматизації
Кафедра прикладної математики та інформатики
Освітньо-кваліфікаційний рівень (освітній ступінь) бакалавр
Напрям підготовки (спеціальність) 122 Комп'ютерні науки
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувачка кафедри ПМІ

/Ольга ДМИТРІЄВА/
18 квітня 2022 року

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТУ

Усатому Ігорю Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка програмної системи для автоматизації торгівлі комп'ютерної техніки з можливістю конфігурування апаратної частини
Спецчастина Розробка програмних модулів, інтерфейсу користувача та бази даних засобами Python, JavaScript, HTML, CSS
Керівник проєкту (роботи) Євген ПАВЛОВСЬКИЙ, ст. викл. каф. ПМІ
(ім'я та прізвище, науковий ступінь, вчене звання)
затверджені наказом від "06" травня 2022 року № 180
2. Строк подання студентом проєкту (роботи) 8 червня 2022 року
3. Вихідні дані до проєкту (роботи) результати науково-дослідної роботи, матеріали переддипломної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) аналіз предметної області і постановка завдання, 2) проєктування програмної системи, 3) розробка програмної системи для автоматизації торгівлі комп'ютерної техніки з можливістю конфігурування апаратної частини 4) тестування програмної системи.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
1) діаграма варіантів використання; 2) діаграма класів; 3)екранні форми; 4) діаграма послідовності; 5) модель бази даних; 6) структура додатку; 7) слайди презентації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 18 квітня 2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Об’єктний аналіз предметної області і постановка завдання	18.04.2022 – 22.04.2022	
2.	Розробка моделі взаємодії даних	23.04.2022 – 25.04.2022	
3.	Проєктування бази даних	26.04.2022 – 30.04.2022	
4.	Проєктування додатку	01.05.2022 – 05.05.2022	
5	Розробка програмного засобу	06.05.2022 – 18.05.2022	
6	Тестування додатку й аналіз	19.05.2022 – 22.05.2022	
7	Оформлення пояснювальної записки	23.05.2022 - 01.06.2022	
8	Підготовка слайдів презентації	05.06.2022	
9.	Рецензування роботи	06.06. 2022 – 08.06.2022	
10.	Захист кваліфікаційної роботи	23.06. 2020	

Студент _____ Ігор УСАТИЙ
(підпис) (ім’я та прізвище)

Керівник роботи _____ Євген ПАВЛОВСЬКИЙ
(підпис) (ім’я та прізвище)

АНОТАЦІЯ

Ігор УСАТИЙ. Розробка програмної системи для автоматизації торгівлі комп'ютерної техніки з можливістю конфігурування апаратної частини. / Випускна кваліфікаційна робота на здобуття освітньо-кваліфікаційного рівня «бакалавр» за спеціальністю 122 Комп'ютерні науки – ДВНЗ ДонНТУ, Луцьк, 2022.

Випускну бакалаврську роботу присвячено розробці програмної системи для полегшення процесу замовлення комп'ютерної техніки, або комплектуючих до неї, серед наявних за багатьма критеріями, а також автоматизації процесу підбору обладнання до вже існуючої конфігурації комп'ютера.

Метою роботи є аналіз предметної області та її аналогів, визначення методів і алгоритмів реалізації програмного засобу, проєктування моделей задля візуалізації зовнішніх і внутрішніх процесів програмного продукту «Техніка».

Об'єктом роботи є принципи розробки веборієнтованих додатків для автоматизації торгівлі комп'ютерної техніки.

Предметом роботи є мови програмування та сучасні підходи до реалізації розробки автоматизованої системи торгівлі комп'ютерної техніки.

Практичне значення роботи полягає у розробці додатку для автоматизації торгівлі комп'ютерної техніки, спрямоване на оволодіння базовими відомостями щодо розробки програмних модулів, інтерфейсу користувача та бази даних засобами Python, JavaScript, HTML, CSS.

Ключові слова: CSS, JavaScript, HTML, мова Python, вебдодаток, інтернет-магазин, конфігуратор, проєктування.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

ОЗП – оперативний запам’ятовуючий пристрій

ООП – об'єктно-орієнтоване програмування

ОС – оперативна система

ПЗ – програмне забезпечення

СКБД – система керування базами даних

CMS – Content Management System

CSS – Cascading Style Sheets

MVC – Model-view-controller

RAM – Random Access Memory

UML – Unified Modeling Language

ВСТУП

У сучасній світовій економіці на основі використання інформації та інформаційно-комунікаційних технологій набула розвитку електронна комерція. Електронна комерція визначається як будь-яка транзакція, що здійснюється через мережу пов’язаних між собою комп’ютерів, після завершення якої відбувається відторгнення та передача права власності або права користування речовим товаром або послугою.

Зручності та переваги торгівлі через Інтернет швидко стали очевидними для підприємців і електронний бізнес став швидко розвиватися. Зокрема популярності набули послуги з продажу комп’ютерної техніки. Але середньостатистичний покупець не завжди має потрібну кваліфікацію, щоб обрати необхідні компоненти комп’ютерної техніки, які відповідають його особистим вимогам. Тому з’являється необхідність у сервісах, що допомагають зорієнтуватися у цьому великої кількості комплектуючих, та обрати необхідну конфігурацію, що задовольнить сформовані потреби.

Основною задачею проєкту є розробка інтернет-системи, яка дозволить швидко і просто замовляти комп’ютерну техніку, або комплектуючі до неї, допоможе адміністратору системи обробляти замовлення та автоматизує процес підбору обладнання до вже існуючої конфігурації комп’ютера.

Потенційними користувачами програмної системи є фізичні особи, які бажають придбати комп’ютерну техніку, в залежності від особистих потреб.

Об’єктом дослідження є автоматизація процесу пошуку та замовлення комп’ютерної техніки.

Методи розробки базуються на використанні основних положень теорії алгоритмів, теорії прийняття рішень, структур і баз даних, математичних методів дослідження операцій, проєктування програмного забезпечення, теорії обчислювальних систем.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ З ПРОЄКТУВАННЯ ТА РОЗРОБКИ WEB-ІНТЕРФЕЙСУ КОРИСТУВАЧА СИСТЕМИ

1.1 Особливості сучасних систем електронної комерції

В сучасному всесвіті вебсистем можна виділити три найпопулярніші типи інтернет-магазинів: статичні, динамічні односторінкові сайти, динамічні багатосторінкові з тізерами-даних [1].

Статичний сайт – представляє собою набір вебсторінок, пов’язаних між собою гіперпосиланнями. Перевагами цього сайту можна виділити простоту створення та дешевий хостинг, відсутність бази даних та програмного коду. Недоліками є те, що оновлення товару ускладняється необхідністю постійного контактування з сервером. Також контент-менеджеру або оператору потрібно володіти базовими навичками html. Все це плавно призводить до такої проблеми, як додатковий найм працівників та збільшення витрат на заробітну платню.

Динамічний односторінковий сайт має тільки одну фізичну сторінку, але інформація, що виводиться на ньому, залежить від того, з якого посилання виводиться сторінка.

Найчастіше такі види сайтів використовуються для продажу одного акційного товару, мають якірну навігацію впродовж сторінки, та контактну форму для зв’язку з продавцем. База даних тут необов’язкова, але бувають і такі, що оснащені панеллю адміністратора, для зручного оновлення інформації власником сайту безпосередньо. Недоліком таких сайтів можна назвати більш високі витрати на хостинг, обмежену кількість товарів, що продаються, окремий облік взаємодії з покупцями та обмежений функціонал для самого клієнта.

Динамічний багатосторінковий сайт – найбільш популярне і оптимальне рішення для інтернет-магазину. Найчастіше оснащений особистим кабінетом покупця та адміністратора, формами для додавання

товару, формою зворотного зв’язку, обліком продажу, фільтрами вибору товару [2]. Все це робить зручним роботу адміністратора або власника магазину, а також спрощує і прискорює процес купівлі для покупця. До недоліків належать витрати на хостинг, та недешеву роботу розробника.

Проаналізувавши існуючі види сайтів, можна зробити висновок, що динамічний багатосторінковий сайт відповідає усім вимогам та поставленим задачам для створення інтернет-магазину.

1.2 Огляд готових рішень для створення інтернет-магазину

Часто веброзробники пропонують створити сайт на основі готової платформи або системи керування контентом (CMS). Тобто пропонують шаблонні рішення, і не є оптимізовані для того чи іншого бізнесу. При цьому готовий інтернет-магазин не може мати ексклюзивний дизайн, в повному обсязі задовольняти умови замовника. Також функціонал магазину часто теж буває обмеженим, або його доповнення може бути ускладненим. У готових інтернет-магазинах важко змінити атрибути товарів, функціонал, платіжні системи, алгоритми обробки замовлень.

Готові інтернет-платформи, такі як PROM або ROZETKA є недорогим рішенням, але можуть мати обмеження на кількість товарів. Є платформи, що не дозволяють використовувати власні доменні імена безкоштовно, натомість пропонують використовувати домен-платформи.

Перевага готових інтернет-магазинів полягає в тому, що такий магазин можна дуже швидко реалізувати. Однак основні витрати припадуть не на відкриття, а на експлуатацію інтернет-магазину. Насамперед, великі витрати може бути формування каталогу товарів, якщо він великий. Як правило, фотографію кожного товару доводиться обробляти вручну. Крім того, це витрати на рекламу сайту інтернет-магазину [2].

Фільтри – це центральний аспект перегляду продукту електронної комерції користувачами. Коли покупець обирає та порівнює товари, він найчастіше спирається на інформацію у категоріях зі списком товарів –

лістингах. Функціонал фільтрації покликаний спрощувати вибір, але зміст та логіка роботи фільтрів не завжди відповідають очікуванням користувачів. Так за результатами дослідження Baymard Institut [4], було встановлено, що користувачам було надзвичайно важко знайти потрібний сумісний товар, лише 35% справлялися із завданням. Практично на всіх сайтах був фільтр по брендах, але його не вистачало для визначення сумісності – не всі блоки живлення одного бренду підходять до всіх ноутбуків того ж виробника. Не завжди один виробник випускає продукти і аксесуари до них. Фільтр сумісності знижує ризик вибрати несумісний товар, отже, і показник повернення товарів. У таблиці 1.1 наведені порівняльні характеристики існуючих торгівельних систем.

Таблиця 1.1 – Порівняльні характеристики існуючих систем

	COMPX	ROZETKA	TELEMART
Пошук товарів	так	так	так
Робота з кошиком	так	так	так
Фільтрація за сумісністю	ні	ні	так
Обробка замовлень	так	так	так
Адміністрування каталогу товарів	так	так	ні
Можливість використання системи зовнішніми продавцями	ні	так	ні

Таким чином можна зробити висновок, що не усі торгові системи задовольняють вимогам продавця або покупця. Тому виникла необхідність розробити систему з урахуванням актуальних потреб користувача.

1.3 Загальна характеристика вебдодатку «Техніка»

Головна задача магазину – продаж комп’ютерної техніки та комплектуючих. Покупець може обрати готову запропоновану зборку, окремі

частини чи аксесуари, або скористуватися конфігуратором, для онлайн підбору комплектуючих до персонального комп’ютера.

Модуль дозволяє почати вибір компонента з будь-якої категорії комплектуючих, наприклад це може бути процесор, ОЗП або корпус. У процесі підбору модуль повинен пропонувати тільки сумісні комплектуючі, які доступні до покупки [8].

Готова збірка додається до «Кошику» та після перевірки замовлення відправляється продавцю. Продавець додає, корегує атрибути товарів, через особистий кабінет, до якого має доступ лише після авторизації, через форму.

1.4 Постановка задачі на розробку ПЗ

Система для автоматизації торгівлі – це реалізоване в мережі Інтернет представництво, шляхом створення вебсерверу для продажу товарів та надання послуг іншим користувачам мережі Інтернет [2]. Електронний магазин – це спільнота, яка уявляє собою територіально-роз’єднаних співробітників магазину, таких як адміністраторів, продавців та покупців, які можуть спілкуватися та обмінюватися інформацією через електронні засоби зв’язку при повній або мінімальній відсутності особистого прямого контакту.

1.4.1 Функціональні вимоги

– користувач системи повинен мати можливість фільтрувати сумісні товари за запитом;

– користувач системи повинен мати можливість додати обрані товари до кошику;

– користувач системи повинен мати можливість переглянути поточний стан кошику а загальну вартість обраних товарів;

– адміністратор системи повинен мати можливість зберігати та вести облік товарів;

– адміністратор системи повинен мати можливість вести облік замовлень користувачів.

1.4.2 Нефункціональні вимоги

Виходячи із сучасних вимог успішного та конкурентоспроможного існування вебсистем, можна сформувати ряд нефункціональних вимог до вебдодатку «Техніка»:

− належний вигляд в усіх сучасних браузерах;

− можливість локалізації клієнтського інтерфейсу: українська;

− інтуїтивно-зрозумілий інтерфейс як для «упевненого» користувача, так і для новачка;

− система повинна бути реалізована на мові програмування Python, засобами БД SQLite;

− система повинна бути швидкодійною і стабільною у роботі;

− система повинна реалізовувати створення власного облікового запису та зберігання цих даних у БД;

− система повинна містити навігаційне меню із вкладками додатку;

− система повинна мати можливість звернення користувачів до розробника через електронну адресу або за номером мобільного телефону тощо.

1.4.3 Структура вебдодатку

Клієнтська частина вебдодатку – це графічний інтерфейс. Графічний інтерфейс відображається у браузері. Користувач взаємодіє з вебдодатком за допомогою інтерфейсу у вигляді кнопок та посилань [2].

Серверна частина – це програма або скрипт на сервері, що обробляє запити браузера. При переході користувача по посиланню, браузер відсилає запит до сервера. Коли сервер обробляє запит, він викликає скрипт, що формує вебсторінку та відправляє клієнту по мережі. Браузер відображає результат у вигляді вебсторінки. База даних – програмне забезпечення на сервері, спеціалізується на зберіганні даних. Серверна частина вебдодатку

звертається до бази даних, викликає дані, необхідні для формування сторінки, за запитом користувача [19].

1.4.4 Динамічні елементи

Види користувацького пошуку, що потрібно реалізувати:

− пошук по статтям та продукції;

− пошук по каталогу за обраними значеннями;

− пошук товарів за сумісництвом.

Форма зворотного зв’язку є зручним комунікаційним сервісом, що дозволяє відвідувачам ресурсу швидко відправляти повідомлення. У більшості випадків контактні форми використовуються для відправлення замовлень, питань по асортименту, заявок на зворотний дзвінок, запитів та відгуків. Використання форм зворотного зв’язку інтуїтивно зрозуміло для користувачів різного віку – для надсилання повідомлення потрібно лише заповнити кілька полів і клацнути на кнопку «Надіслати».

Навігація по сайту – система, за допомогою якої відвідувач орієнтується на сайті, переходить на сторінки другого рівня вкладеності, шукає потрібну інформацію. Важливо, щоб навігація сайту була зручною та зрозумілою відвідувачу [9].

1.5 Висновки за розділом

У цьому розділі були визначені мета розробки, основна предметна область та її призначення. На основі порівняння існуючих видів онлайн додатків, визначилися, чому було обрано принцип побудови багатосторінкової вебсистеми. Були описані основні принципи роботи існуючих інтернет-магазинів та додатків, орієнтованих на продаж комп’ютерної техніки. Були проаналізовані існуючі додатки, їх недоліки та переваги. Виявлені основні проблеми, що мають бути вирішенні при розробці даної системи.

Описані функціональні та нефункціональні вимоги до вебзастосунку. Сформовані основні вимоги до дизайну та комунікацій. Виділено дві основні ролі користувачів та їх функції.

2 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ ТА ВИБІР ПРОГРАМНИХ ЗАСОБІВ ЇЇ РЕАЛІЗАЦІЇ

2.1 Системний аналіз розроблюваного програмного додатку

Для детального опису процесів додатку використано UML-діаграми. UML – уніфіцирована мова моделювання, що підходить для широкого класу програмних систем, що проєктується, різноманітних областей додатків, типів організацій, розмірів проєктів [10].

2.1.1 Діаграма варіантів використання

На наступній UML-діаграмі (рис. 2.1) зображені варіанти використання користувачів програмного додатка.

Першорядними користувачем є адміністратор (продавець) та користувач (покупець) – кожен взаємодіє з програмним додатком задля виконання операційних задач.

Нижче додано описи варіантів використання додатка.

Додати/скорегувати категорію товару – товар може належати до одної чи деяких категорій. Адміністратор додає та корегує інформацію по категоріям.

Додати/скорегувати товар – адміністратор може додати новий товар, або скорегувати існуючий, а також додати фото існуючого товару.

Переглянути/скорегувати свій профіль – адміністратор може переглянути особисту інформацію для авторизації та змінити свій пароль за потреби.

Переглянути замовлення – адміністратор може переглянути замовлення, що надіслані покупцем, а також контакти самого покупця.

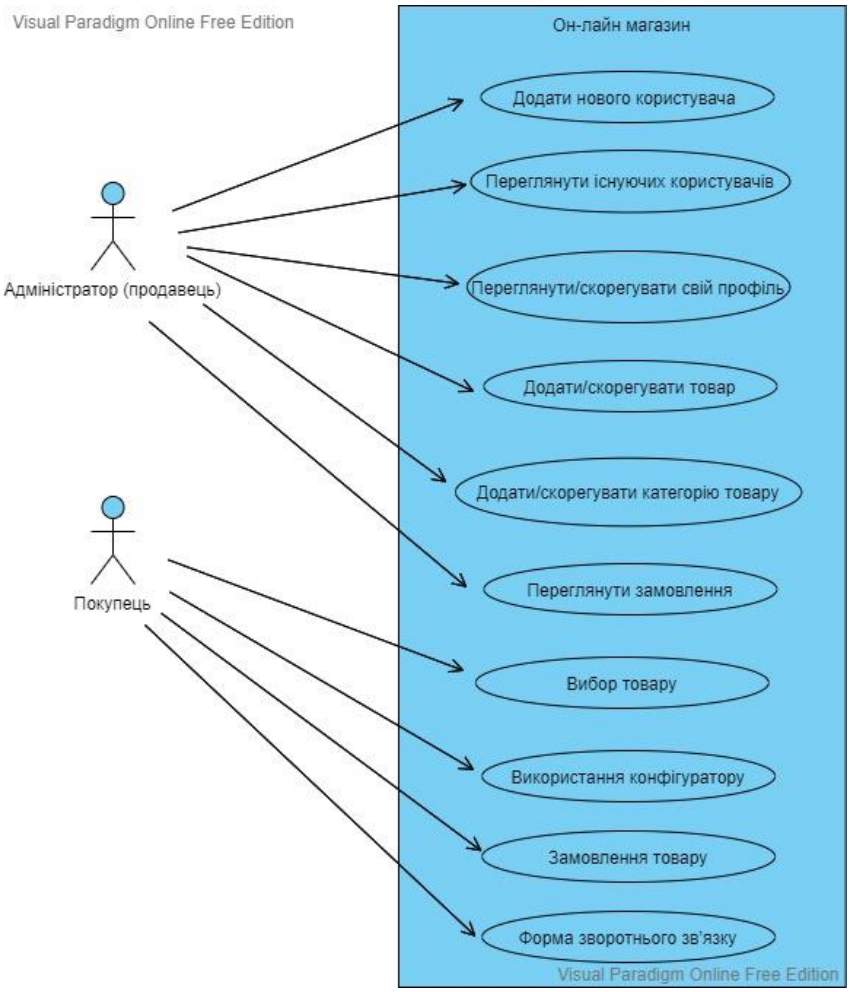


Рисунок 2.1 – Діаграма варіантів використання

Додати нового користувача – адміністратор може додавати нового продавця до системи.

Обрати товар – покупець може вибрати існуючий товар та додати його до кошику.

Використання конфігуратору – покупець може скористуватися конфігуратором для комплектації товару з запропонованих.

Замовлення товару – покупець може замовити товар через кошик.

Зв’язок з продавцем – покупець може задати питання продавцю за допомогою форми зворотнього зв’язку.

2.1.2 Діаграма класів

Розробка логічної моделі системи у вигляді діаграми класів займає центральне місце в об’єктно-орієнтованому програмуванні. Діаграма класів відображає зв’язки між сутностями предметної області, а також описувати їх структуру та типи відносин [10]. На рисунку 2.2 наведена діаграма класів системи. Клас – це абстракція сутностей з однаковими властивостями (атрибутами та полями) та поведінкою (функціями та методами). Класи зв’язані між собою наслідками та асоціаціями. При успадкуванні клас-нащадки мають (успадковують) атрибути батьківського класу, а також свої власні [10]. Поля в класі пов’язані з полями таблиці бази даних, що видно в таблицях 2.1-2.5.

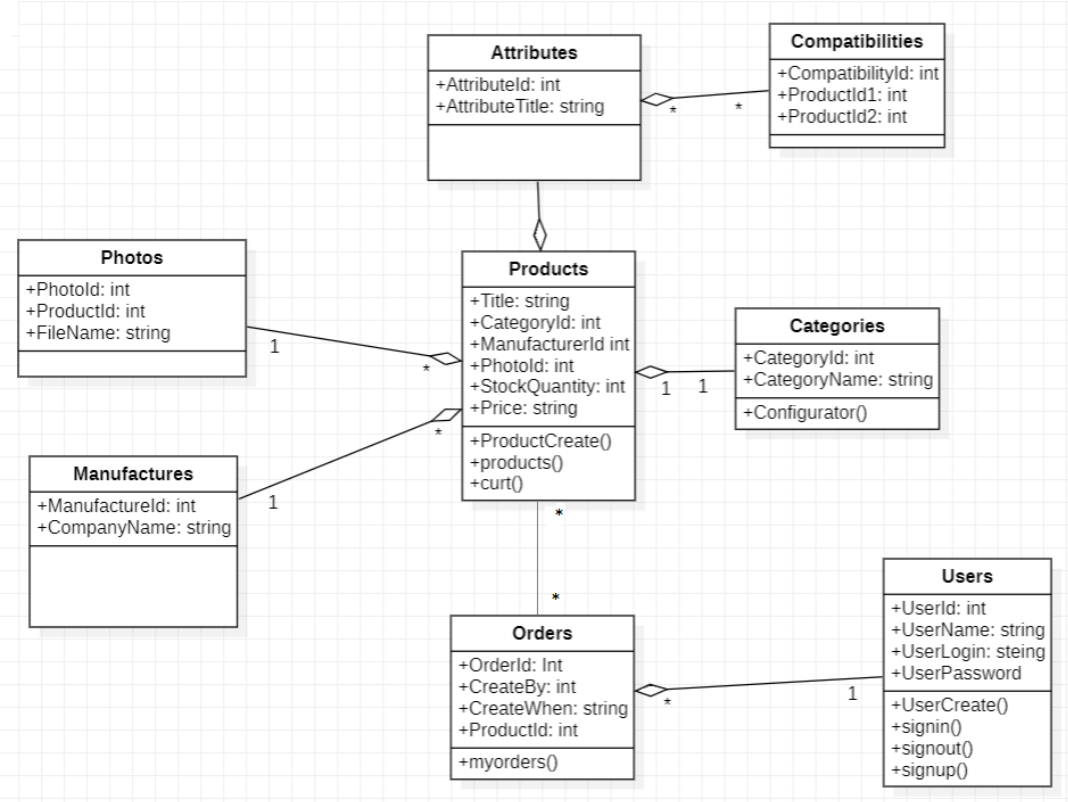


Рисунок 2.2 – Діаграма класів системи автоматизації продажу комп’ютерної техніки

У таблицях 2.1-2.5 представлено описи специфікацій діаграм класів.

Таблиця 2.1 – Опис структури класу «User»

Назва	Назва в БД	Тип поля
Ідентифікатор користувача	UserId	INTEGER
Ім’я користувача	UserName	TEXT
Пошта користувача	Email	TEXT
Пароль, хеш-строка утворена з паролю користувача	Password	TEXT

Таблиця 2.2 – Опис структури класу «Products»

Назва	Назва в БД	Тип поля
Ідентифікатор товару	ProductId	INTEGER
Найменування товару	Title	TEXT
Ідентифікатор категорії товару	CategoryId	INTEGER
Ідентифікатор виробника товару	ManufacturerId	INTEGER
Кількість на складі	PhotoId	INTEGER
Вартість товару	StockQuantity	REAL

Таблиця 2.3 – Опис структури класу «Categories »

Назва	Назва в БД	Тип поля
Ідентифікатор категорії товару	CategoryId	INTEGER
Назва категорії	CategoryName	TEXT

Таблиця 2.4 – Опис структури класу «Attributes»

Назва	Назва в БД	Тип поля
Унікальний номер атрибуту	AttributeId	INTEGER
Назва атрибуту	AttributeTitle	TEXT

Таблиця 2.5 – Опис структури класу «Compatibilities»

Назва	Назва в БД	Тип поля
Ідентифікатор сумісництва товарів	CompatibilityId	INTEGER
Ідентифікатор першого товару	ProductId1	INTEGER
Ідентифікатор другого товару	ProductId2	INTEGER

Таблиця 2.6 – Опис структури класу «Manufactures»

Назва	Назва в БД	Тип поля
Ідентифікатор виробника	ManufacturerId	INTEGER
Назва компанії виробника	CompanyName	TEXT

Таблиця 2.7 – Опис структури класу «Orders»

Назва	Назва в БД	Тип поля
Ідентифікатор замовлення	OrderId	INTEGER
Ідентифікатор користувача, який прийняв замовлення	CreatedBy	INTEGER
Дата/час створення замовлення	CreatedWhen	TEXT

Таблиця 2.8 – Опис структури класу «Photos»

Назва	Назва в БД	Тип поля
Ідентифікатор фото товару	PhotoId	INTEGER
Шлях до фото	Path	TEXT

2.1.3 Діаграма послідовності

На діаграмі послідовностей для адміністратора (рис. 2.3) відображено взаємодія сутностей серед зазначених прецедентів для адміністратора.

На діаграмах послідовностей наведені наступні об’єкти:

− адміністратор, викликає потрібні функції;

− програмний додаток, оброблює запит користувача;

− система зберігання даних, видає або записує дані за запитом.

Елементи діаграми зображують можливі варіанти використання вебдодатку: додавання/корегування категорії товару, додавання/корегування опису товару, перегляд/корегування профілю, перегляд замовлень, перегляд/корегування існуючих користувачів, додавання нового користувача [10].

При виклику функції адміністратором, вона оброблюється додатком, за необхідністю, проходить звернення додатком до бази даних та отримується відповідний запис.

Наступна UML-діаграма послідовності (рис. 2.4) зображує життєві цикли та взаємодію сутностей у рамках зазначених прецедентів для покупця. Вона відображає ті ж самі об’єкти взаємодії, але перелік можливих функцій для кадрового спеціаліста відрізняється. До функції доступних для покупця входять: вибори товару, використання конфігуратору, замовлення товару через кошик, зв’язок з продавцем через форму зворотного зв’язку.

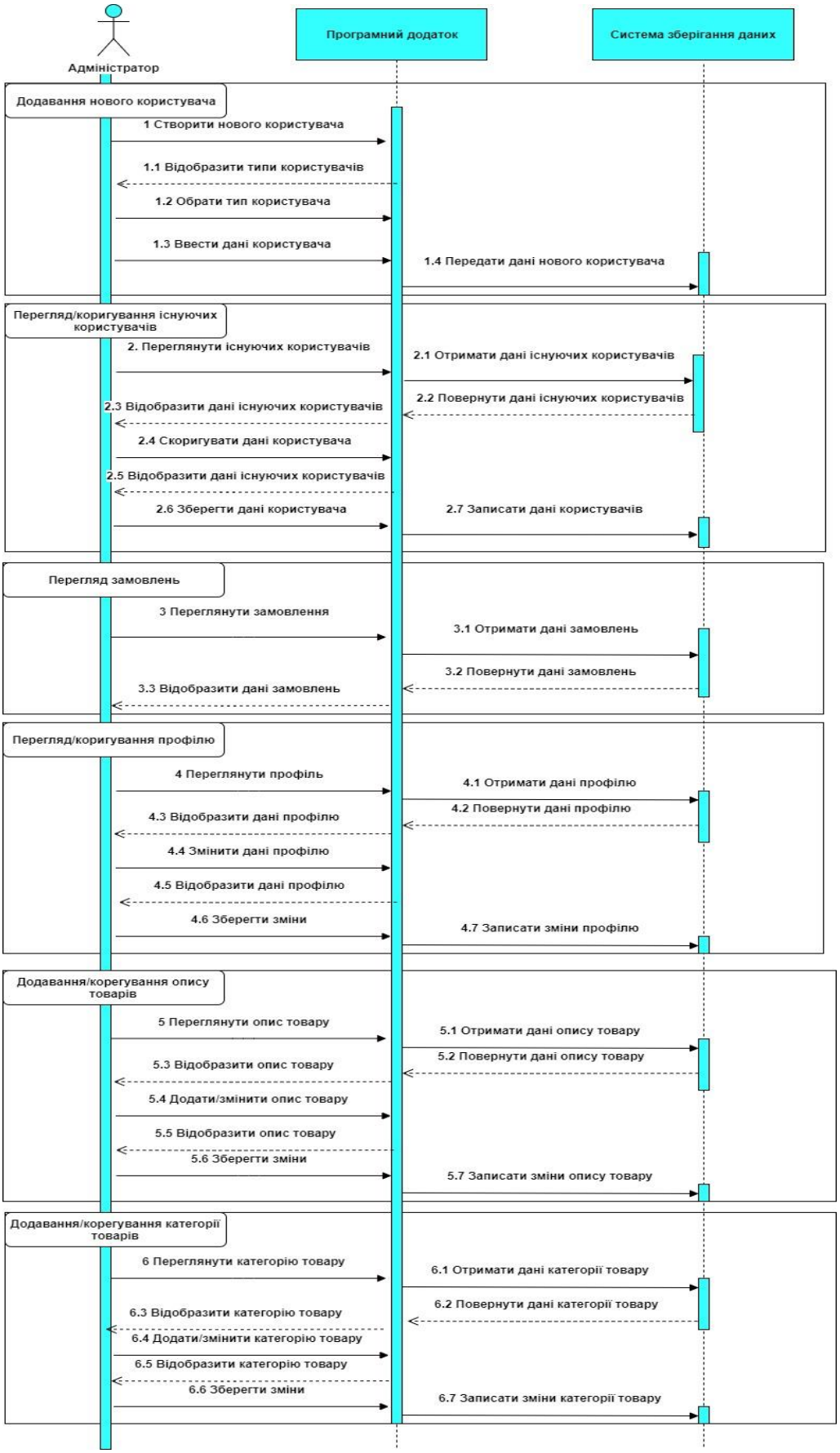


Рисунок 2.3 – Діаграма послідовностей для адміністратора

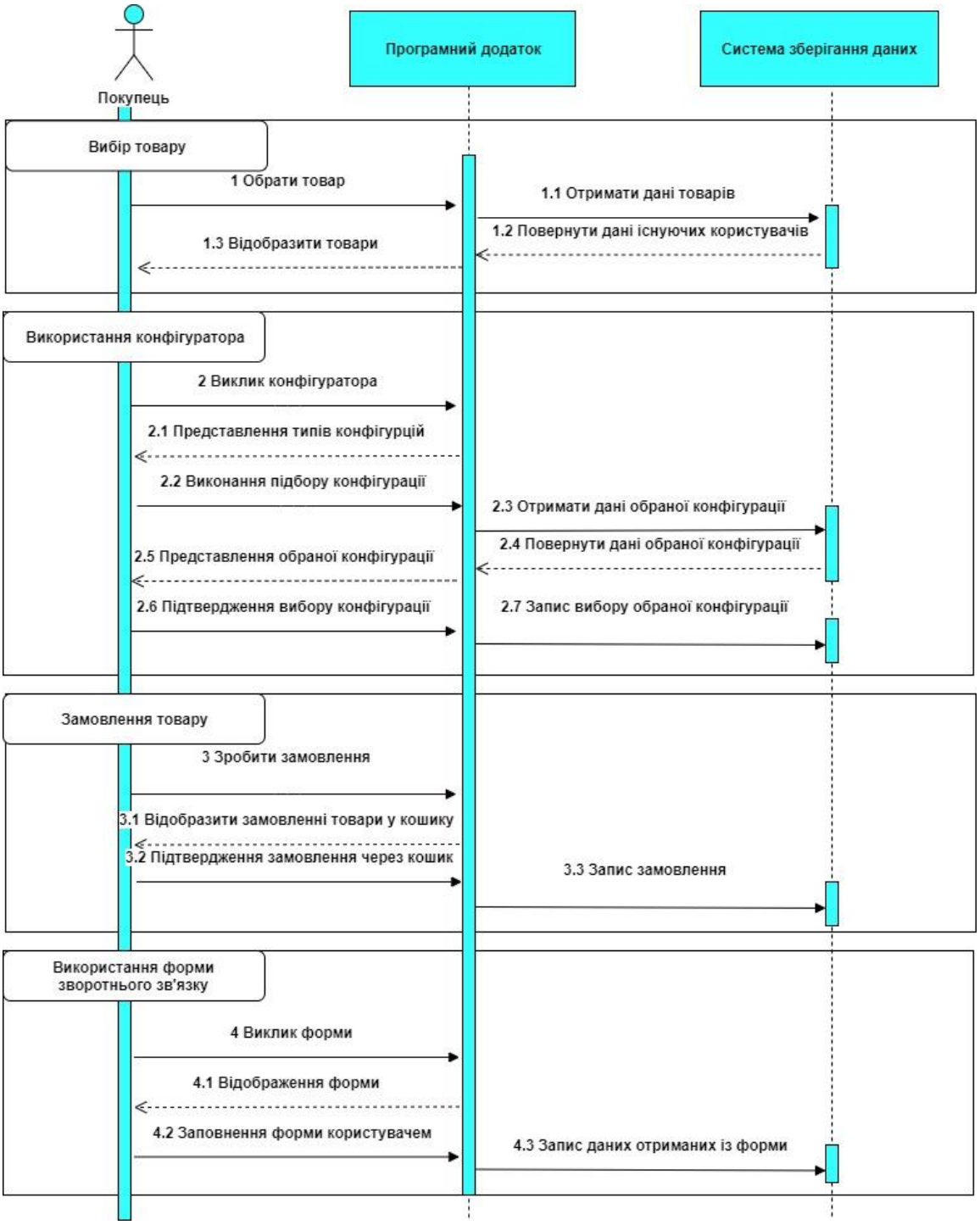


Рисунок 2.4 – Діаграма послідовностей для клієнта (покупця)

2.1.4 Діаграми компонентів

На рисунку 2.5 представлена діаграма компонентів системи для автоматизації торгівлі комп’ютерною технікою.

Під компонентами розуміють фізичні модулі програмного коду, що показують, яким образом одні компоненти взаємодіють з іншими [10].

Компоненти діаграми повинні бути зв’язані між собою, оскільки кожний модуль залежить від іншого.

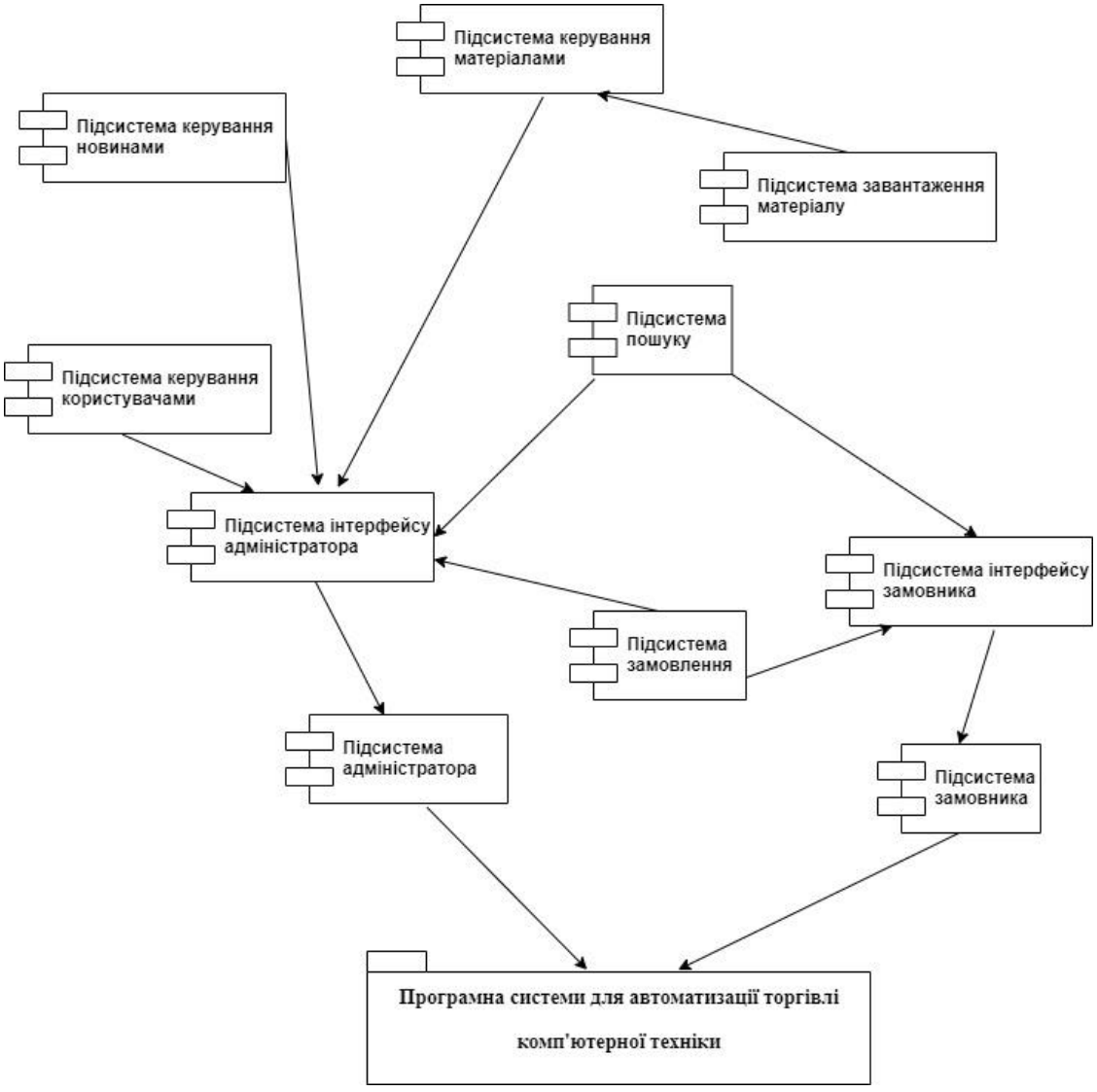


Рисунок 2.5 – Діаграма компонентів системи для автоматизації торгівлі комп’ютерною технікою

2.2 Вибір архітектури програмного додатку

Підтримка різного типу користувачів, які використовують різні типи пристроїв, є найголовнішою задачею при створенні вебдодатку. Інтерфейс, що розроблюється, повинен розрізнятися, якщо запит надходить з персонального комп'ютера або мобільного телефону. Система повертає однакові дані, але за допомогою окремих інтерфейсів [5].

З вищеописаного можна зробити висновок, що безсумнівною перевагою буде використання шаблону «модель-вид-контролер» (Model-View-Controller).

Додаток розділяють на три основних компоненти, кожна з них відповідає за окремій задачі.

Контролер керує запитами користувача, що отримуються у вигляді запитів HTTP GET або POST, коли користувач натискає на елементи інтерфейсу при виконувані різноманітних дій. Його головна функція – викликати та координувати дії необхідних ресурсів та об'єктів, необхідних для виконання дій, що задаються користувачем. Контролер викликає відповідну модель для задачі та обирає необхідний вид.

Модель – це дані та правила, що використовуються для роботи з даними, що представляють концепцію керування додатком. У будь-якому додатку уся структура моделюється, як дані, що обробляються відповідно правилам [16].

Контролер містить організаційну логіку для самого додатку.

Вид забезпечує різноманітні способи представлення даних, які отримані із моделі. Він може бути шаблоном, який заповнюється даними. Може бути кілька різноманітних видів, і контролер обирає, який підходить найбільше для поточної ситуації [16].

Рисунок 2.6 – Принцип роботи концепції Model-View-Controller

Найочевидніша перевага, яку ми отримуємо від використання концепції MVC – це чіткий поділ логіки уявлення (інтерфейсу користувача) та логіки програми.

Крім того, завдяки MVC суттєво зменшує складність великих додатків, код більш структуровано, а це спрощує підтримку, тестування та повторне використання рішення [17].

2.3 Проєктування реляційної бази даних

У базі даних «Техніка» використовуються наступні вхідні дані:

− інформація про користувачів;

− інформація про товари;

− інформація про категорії товарів;

− інформація про сумісність;

− інформація про покупки, що були здійснені.

Вихідною інформацією є результати роботи запитів, на екран інформація виводиться у вигляді таблиць [8].

У базі даних необхідно створити такі обмеження:

− заборонено вводити негативні числові значення;

− деякі поля не можуть бути порожніми;

− пароль користувача потрібно зберігати за зашифрованому вигляді;

− значення деяких полів повинно бути унікальним.

На першому етапі очевидна потреба о трьох таблицях-сутностей. Перша – це таблиця, що зберігає інформацію про товар (назву, ціну, кількість, продавця, покупця, категорію товару, а також інформацію про сумісність з іншими). Друга – це таблиця-сутність, що зберігає інформацію про користувача (логін, пароль, email). Третя – це таблиця сутність, що зберігає інформацію, про замовлення (код товару, код покупця, дату).

Проаналізувавши таблицю, що зберігає дані про товар, можна зробити висновки, що вона переповнена даними. До однієї категорії може належати декілька товарів, тому в рамках нормалізації бази даних можна відокремити ці дані в окремі таблиці, що будуть зберігати лише код і назву категорії, або атрибуту. Та пов’язати їх з таблицею товарів логічним зв’язком «один-до багатьох». Атрибут же, навпаки, може належати до багатьох категорій, а категорія до кількох атрибутів. Ці таблиці можна зв’язати логічним зв’язком «багато-до-багатьох». При цьому типі зв’язку доцільно створити проміжну таблицю, наприклад «категорія-товар».

2.4 Вибір інструментальних засобів розробки програмного додатку

Вибір засобу розробки залежить від багатьох факторів таких як: підтримка перспективних платформ, вартість ліцензії, наявність активної спільноти розробників, захищеність коду та інше. Опираючись на це, був проведений порівняльний аналіз щодо існуючих мов та серед програмування та СКБД.

2.4.1 Вибір мови програмування

Серед мов, які в основному призначені для роботи з вебтехнологіями, можна зазначені такі, як: JavaScript, PHP, Python, Ruby, C#, Perl, Java [6]

Розглянемо недоліки та переваги кожного з них.

JavaScript – одна з найпоширеніших мов. Область його застосування велика і майже безмежна. На JavaScript пишуть серверні, мобільні та

комп’ютерні програми. Будь-який браузер та будь-яка операційна система добре знайома з JavaScript [17]. Всі сценарії виконуються безпосередньо в браузері пристрою, користувачеві не потрібно робити будь-які дії. У більшості випадків він використовується для створення простих анімацій, скриптів та об’єктів інтерфейсу користувача.

Головна перевага PHP – код мови не конфліктує з HTML версткою і може використовуватись одночасно для розмітки зовнішнього вигляду сторінки за допомогою HTML-тегів та функціоналу сторінки php-частиною. Він легкий у освоєнні на всіх етапах вивчення. Відрізняється розвиненою підтримкою даних, підходить під апаратні платформи та відомі ОС. Ця мова програмування призначена спеціально для роботи на стороні сервера. Бібліотека мови підходить для завдань, які багаторазово під час розробки сайту.

Python вважають ідеальною в DataScience (методика аналізу даних з використанням машинного навчання та штучного інтелекту). Одним із головних плюсів Python вважається його простота. За наявності бажання, з його особливостями та тонкощами програмування зможе розібратися кожен охочий. До того ж Python сприяє економії часу програміста, тому що пропонує велику кількість спеціальних бібліотек з готовими програмними конструкціями.

Також не можна не зауважити ряд недоліків. По-перше, у програміста з’являється звичка простоти. Працюючи з Python, фахівець починає шукати таку саму лаконічність в інших мовах, але не знаходить. По-друге, низька швидкість. Втім, у багатьох проєктах така особливість Python не приносить дискомфорту і не є критичною, оскільки різниця не помітна для ока користувача. Але вже це не можна сказати щодо великих проєктів із величезною базою даних. Різниця буде відчутною. По-третє, динамічна типізація. Програміст може писати коротко, не оголошувати тип змінної. Час заощаджується, але це часто призводить до появи помилок. Тому часто доводиться робити додаткові перевірки.

Основне призначення Ruby – формувати та програмувати сайти, а також мобільні програми. Навколо мови Ruby склалася думка про його повільність і неможливість масштабувати великі проєкти. На самому початку існування у плані продуктивності Ruby дійсно поступався PHP та Python. Однак численні оновлення мови докорінно виправили ситуацію, прийдешні апгрейди мають принести й інші зміни – можливість роботи з паралельними потоками. Повільність роботи сучасного додатку на Ruby повністю залежить від можливостей програміста та правильності побудови архітектури. З переваг можна відзначити легкість вивчення мови початківцем, часто його використовують завдяки простим методам запису [17].

Мова програмування C# перейняла багато від Java та C++. Більше половини його синтаксичних можливостей ідентичні з Java. Спочатку використовувався як створення вебсайтів. Зазначимо, що сьогодні C# активно розвивається, виходять оновлення та доповнення, з'явилися асинхронні методи, динамічні зв'язування. Якщо порівнювати його з іншими популярними мовами, можна відзначити відносну молодість C#: його перша версія з'явилася 2002 року.

Мови програмування для веброзробки важко уявити без Perl. У самих витоках виникнення, Perl призначався для позбавлення необхідності написання різних програм і сценаріїв різними мовами, поєднуючи можливості системного адміністрування та обробки документів в єдине мовне середовище. На даний момент Perl активно використовується при написанні інтерактивних додатків, адмініструванні серверів і адаптований до всіх популярних платформ – Windows, Mac та інші.

Легко виділити основні переваги Perl:

− наявність безлічі готових бібліотек;

− простота обробки великого обсягу даних;

− підтримка роботи з регулярними виразами;

− вільний синтаксис.

Але з переваг утворюються невеликі вади. Наявність великої кількості бібліотек ускладнюють пошук одного конкретного модуля, необхідного програмісту, що може гальмувати процес роботи.

Мова Java часто використовується з метою створення мобільних додатків, мережевих програм. Вважається основною мовою розробки для Android. Мова йде в ногу з часом і сьогодні актуальна як ніколи. Він включає об'єктно-орієнтоване програмування (ООП) – методику спрощення складного коду, при якому ділянка коду з функціями, що конфліктують один з одним, ділиться на незалежні об'єкти, кожен з яких містить у собі ті ж функції і дані, які активуються при безпосередньому зверненні до них, а чи не одночасно, створюючи конфлікт (як у процедурному програмуванні). До інших переваг Java варто віднести безпеку, надійність і простий синтаксис [17].

Найоптимальнішою мовою програмування для розробки додатку є Python, через його гнучкість. Він легко працює на серверах під керуванням Linux та Windows. Ідеально підходить для розрахунків, створення запитів, та форм вебдодатку. Щоб знизити навантаження на сервер при перевірці даних, що вводяться, задіяно JavaScript. Також цю мову планується використовувати, для оформлення випадаючого меню, а також для зміни зовнішнього вигляду окремих елементів сторінки (стилі CSS).

2.4.2 Вибір системи керування базами даних

Діяльність системи направлено на вирішення цілого ряду задач:

− зберегти дані за запитом;

− змінити дані за запитом;

− знайти дані за запитом;

− обмежити доступ до інформації залежно від статусу користувача;

− не дати даним загубитися;

− масштабування.

Усю необхідну інформацію додатку треба зберігати у базах даних. База даних – це набор організованих даних. А вирішити перелічені задачі допоможе СКБД. Системи керування базами даних (СКБД) це такий засіб маніпуляцій даними в базі [7].

Оглянемо найпоширеніші сучасні реляційні СКБД: PostgreSQL, MySQL та SQLite.

MySQL – найпопулярніша серед серверних БД. Не намагається повністю реалізувати SQL-стандарти, але пропонує широкий функціонал.

Переваги:

− легко налаштувати, багато сторонніх інструментів, в т.ч. і візуальні, що спрощують роботу з БД;

− підтримує більшу частину функціоналу SQL;

− має багато функцій безпеки;

− може працювати з великим обсягом даних;

− працює швидко.

Обмеженість функціоналу та певні питання до застарілих принципів безпеки можна назвати основними недоліками цієї СКБД.

PostgreSQL – відрізняється від інших СКБД тим, що має об’єктно-орієнтований функціонал. Відмінно справляється з одночасною обробкою декількох завдань, так як заснована на потужній технології Postgres.

Переваги:

− повна sql-сумісність;

− підтримка опитної спільноти;

− використовується в багатьох інструментах, пов’язаних з СКБД;

− можна програмно розширити за рахунок збережених процедур;

− об’єктно-орієнтована СКБД.

Головний недолік цього виду СКБД – складність використання, через це інструмент не дуже популярний, і не всі хостинг-провайдери забезпечують його підтримку.

SQLLite – бібліотека, що вбудовується в додаток. Файлова БД, вона надає відмінний набор інструментів для простої обробки будь-яких видів даних.

Коли додаток використовує SQLLite, їх зв’язок здійснюється за допомогою функціональних та прямих викликів файлів, що утримують дані, а не через інтерфейс. Що збільшує швидкість та продуктивність операцій [35].

Кожна з цих СКБД може бути використана створення вебдодатку магазину. Але після оцінювання недоліків та переваг, можна сказати, що SQLite – найоптимальніший вибір. Підключення просте, за рахунок невикористання моделі «клієнт-сервер», а сама бібліотека для роботи з SQLite входить до стандартної бібліотеки мови Python.

2.5 Аналіз системних вимог

В процесі роботи були сформовані системні вимоги, яким повинно відповідати програмне забезпечення для розробки системи[11].

Операційні системи: Microsoft Windows 10 64-біт Windows 8 64-біт; macOS 10.13 чи вище; середа GNOME або KDE.

RAM не менш ніж 2ГБ, але рекомендовано 8ГБ.

Місце на диску: інсталяція потребує 2,5ГБ, рекомендоване використання SSD.

Роздільна здатність екрану: не менш ніж 1024x768 пікселів.

Python 2.7, Python 3.5 або більш пізніша версія.

Робота клієнтського модуля повинна підтримуватися усіма сучасними браузерами.

Передача даних повинна відбуватись з використанням шифрування SSL/TLS.

2.6 Опис методів розв’язання

Поставлене завдання може бути представлене як сукупність задач, кожна з яких має свій алгоритм реалізації.

2.6.1 Створення облікового запису користувача

Створення облікового запису користувача складається з наступних етапів:

1) користувач переходить на сторінку реєстрації;

2) користувач заповнює форму основними даними (логін – ідентифікатор користувача, пароль, ім’я, електронну пошту);

3) програма перевіряє наявність усіх обов’язкових даних та їх формату:

4) якщо є помилки, то виводиться повідомлення про помилку, та користувач знову заповнює некоректні поля;

5) інакше виконується підтвердження паролю;

6) якщо паролі збігаються, то стає доступною кнопка реєстрації, інакше користувач знову підтверджує пароль;

7) користувач натискає кнопку реєстрації та чекає на відповідь від сервера;

8) сервер перевіряє унікальність логіну (виконує пошук у базі даних);

9) якщо логін існує, то повертається відповідне повідомлення, та користувач вводить нові дані;

10) інакше сервер повторно перевіряє наявність та формат даних;

11) якщо є помилки, то повертається відповідне повідомлення, та користувач вводить нові дані;

12) інакше сервер створює обліковий запис та відправляє його до бази даних; повертається повідомлення про успішну реєстрацію.

2.6.2 Вхід користувача до облікового запису

1) користувач переходить на сторінку входу в обліковий запис;

2) користувач заповнює форму входу (логін та пароль);

3) програма перевіряє наявність та формат даних;

4) якщо дані некоректні, то підсвічуються помилки та користувач знову вводить дані для входу;

5) інакше, стає доступною кнопка входу;

6) користувач натискає кнопку входу та очікує відповіді від сервера;

7) сервер отримує логін та пароль, та виконує їх перевірку (перевіряє існування користувача й відповідність даних);

8) якщо знайдені помилки, то повертається відповідне повідомлення про помилку;

9) інакше сервер дістає дані користувача з бази даних;

10) сервер створює тимчасовий токен;

11) сервер повертає токен та загальні дані про користувача (ім’я, логін, роль);

12) клієнтська частина отримує дані та записує їх у локальне сховище браузера;

13) надається доступ до дій, що дозволені ролі даного користувача; − виводиться повідомлення про успішний вхід в обліковий запис.

2.6.3 Фільтрування техніки за обраними категоріями

Фільтри та сортування спрощують пошук потрібних товарів. Хоча існують параметри фільтрації, універсальні для всіх категорій (ціна, рейтинг тощо), користувачів більше цікавить фільтрація за унікальними параметрами Найважливіша характеристика товару виводиться при попередньому перегляді картки товару, тому має бути можливість відфільтрувати весь перелік товарів за цією характеристикою. У попередньому перегляді картки процесора виведені такі параметри: виробник, сокет, базова частота процесора, кількість продуктивних ядер, об’єм кешу. Саме по цім параметрам необхідно організувати фільтрацію в категорії «Процесор». Вибір деяких товарів визначається сумісністю з іншими товарами, якими

користувач має, цей параметр теж необхідно врахувати. Таким чином було сформовано алгоритм вибору товарів за обраними критеріями:

1) користувач переходить на сторінку конфігуратора;

2) робиться запит на отримання з сервера переліка категорій, наявних тегів та фільтрів, що доступні користувачеві;

3) сервер дістає відповідні дані з бази даних та повертає їх;

4) користувач натискає на іконку додавання процесору;

5) сервер отримує запит на пошук наявних процесорів;

6) сервер дістає відповідні дані, що стосуються процесорів;

7) критерії пошуку з’являються у секції прев’ю (ціна, виробник);

8) користувач перевіряє дані (за необхідності змінює їх) та натискає кнопку пошуку за критеріями;

9) сервер отримує запит на пошук комплектуючих у вигляді фільтру;

10) сервер дістає з бази даних усі товари, що відповідають критеріям;

11) клієнтська частина отримує перелік процесорів та показує їх на екрані;

12) користувач додає до кошика процесор, що відповідає вимогам;

13) користувач натискає на іконку вибору материнської плати;

14) сервер отримує запит на пошук наявних материнських плат, що є сумісними з обраним процесором;

15) сервер дістає відповідні дані;

16) клієнтська частина отримує перелік процесорів, та показує їх на екрані;

17) клієнт обирає усі комплектуючі, що відповідають його вимогам, та додає до кошику;

18) сервер отримує запит на відображення сторінки кошика;

19) клієнтська частина отримує перелік обраних товарів та відображає їх на екрані разом із кнопкою створення замовлення.

2.6.4 Розробка модулю створення карточки товару

Створення карточки товару відбувається за наступним алгоритмом:

1) користувач входить в обліковий запис;

2) користувач переходить до сторінки додавання товару;

3) користувач заповнює основну інформацію про товар (назва, опис, фотографія, ціна);

4) користувач додає фільтри: програма робить запит до сервера; сервер отримує перелік категорій з бази даних та відсилає його до клієнтської частини; користувач обирає категорії зі списку та додає їх до товару);

5) програма перевіряє чи коректна заповнено інформацію та формати;

6) якщо знайдені помилки, то підсвічуються відповідні поля, та користувач вводить нові дані;

7) користувач натискає кнопку додавання товару;

8) програма відправляє запит на збереження інформації до сервера;

9) сервер отримує інформацію та додає ідентифікатор поточного користувача у якості автора запису;

10) сервер відправляє товар до бази даних та отримує його ідентифікатор;

11) сервер повертає ідентифікатор нового товару;

12) повідомлення про успішне створення виводиться на екран клієнтською частиною.

2.7 Висновки за розділом

У даному розділі було проаналізовано бізнес-процеси даного програмного забезпечення, був проведений порівняльний аналіз засобів розробки програмного забезпечення. Спроєктовані UML-діаграми варіантів використання, діаграма класів, діаграма послідовностей та діаграма компонентів. А також обґрунтовано вибір інструментів розробки: Python, JavaScript, CSS, HTML.

3 РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ ДЛЯ АВТОМАТИЗАЦІЇ ТОРГІВЛІ КОМП'ЮТЕРНОЇ ТЕХНІКИ

3.1 Опис структури програмного додатку

На рисунку 3.1 представлена структура сторінок додатку, що представлена в ієрархічній моделі. Схема допомагає оцінити обсяг сторінок, що формують додаток, а також зрозуміти логіку взаємозв'язку сторінок.

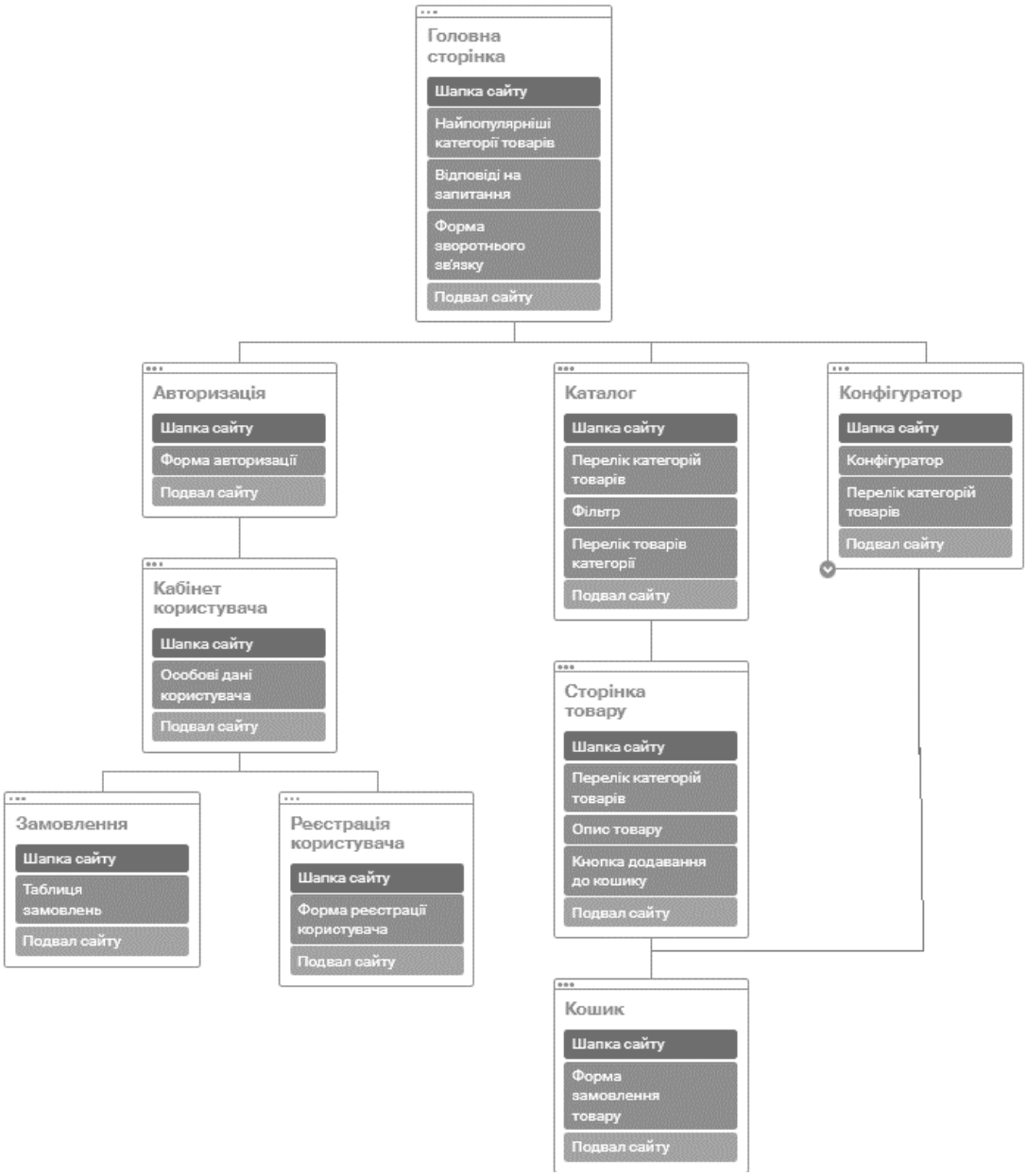


Рисунок 3.1 – Ієрархічна модель сторінок додатку

3.2 Розробка моделі бази даних

На рисунку 3.2 представлена інфологічна модель бази даних, на якій відображені всі сутності БД, відношення між ними і атрибути.

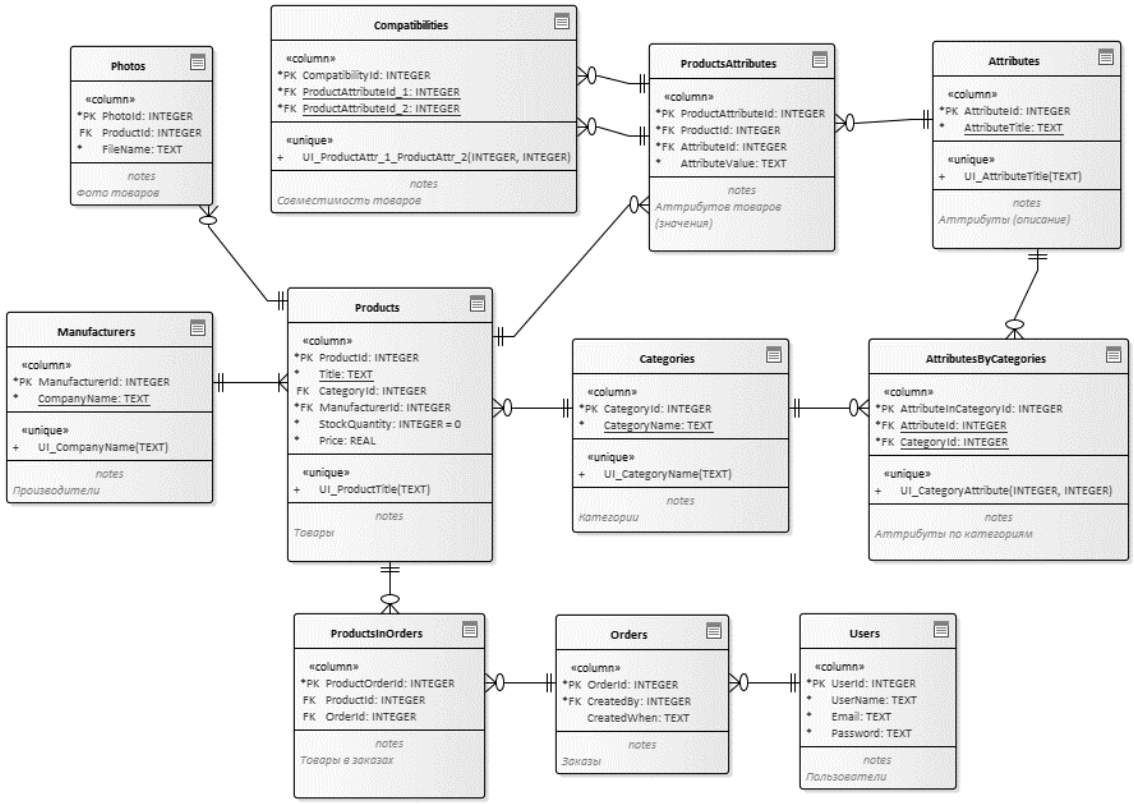


Рисунок 3.2 – ER-діаграма бази даних

Розглянемо призначення, окремих таблиць та зв’язки між ними.

«Products» – таблиця-сутність, призначена для зберігання даних про товари. Поле «PhotoId» пов’язує відношенням «багато-до-одного» з таблицею «Photos». Дана таблиця-сутність наведена у таблиці 3.1.

У полі «Title» зберігається назва товару, не може бути двох товарів з однаковою назвою. У полі «StockQuantity» зберігається кількість товару на складі. Значення «StockQuantity» не може бути NULL, за замовченням значення дорівнює 0.

Таблиця 3.1 – Таблиця «Products»

Найменування	Тип поля	NOT NULL	Коментар
ProductId	INTEGER	True	Первинний ключ AutoNum = True
Title	TEXT	True	Найменування товару Унікальне значення
CategoryId	INTEGER	False	ID категорії
ManufacturerId	INTEGER	True	ID виробника
PhotoId	INTEGER	False	ID фото
StockQuantity	INTEGER	True	Значення по замовченню: 0 Кількість на складі
Price	REAL	True	Вартість

Таблиця має три зовнішні ключі:

– «ManufacturerId» пов’язує відношенням «багато-до-одного» з таблицею «Manufacturers»;

– «CategoryId» пов’язує відношенням «багато-до-одного» з таблицею «Categories»;

– «PhotoId» пов’язує відношенням «багато-до-одного» з таблицею «Photos»;

«ProductsAttrubutes» – таблиця, що містить значення даних атрибутів, яка наведена у таблиці 3.2.

Таблиця має два зовнішні ключі:

– «ProductId» пов’язує відношенням «багато-до-одного» з таблицею «Product»;

– «AttributeId» пов’язує відношенням «багато-до-одного» з таблицею «Attribute».

Таблиця 3.2 – Таблиця «ProductsAttrubutes»

Найменування	Тип поля	NOT NULL	Коментар
ProductAttributeId	INTEGER	True	Первинний ключ AutoNum = True
ProductId	INTEGER	True	ID товару
AttributeId	INTEGER	True	ID атрибуту
AttributeValue	REAL	True	Значення атрибуту

Таблиця «Attributes» – таблиця-сутність містить в собі перелік назв атрибутів (здатності процесорів, пам’яті чи об’єм кешу). Дана таблиця-сутність наведена у таблиці 3.3.

Таблиця 3.3 – Таблиця «Attributes»

Найменування	Тип поля	NOT NULL	Коментар
AttributeId	INTEGER	True	Первинний ключ Унікальний номер атрибуту
AttributeTitle	TEXT	True	Назва атрибуту

Таблиця «AttributesByCategories» – допоміжна таблиця, що пов’язує таблиці «Categories» та «Attributes» відношенням «багато-до-багатьох» репрезентуючи співвідношення атрибутів та категорій. Дана таблиця наведена у таблиці 3.4.

Таблиця 3.4 – Таблиця «AttributesByCategories»

Найменування	Тип поля	NOT NULL	Коментар
Attributeincategoryid	Integer	True	Первинний ключ Autonum = true
Attributeid	Integer	True	Id атрибута
Categoryid	Integer	True	Id категорії

Має два зовнішні ключі:

– «Attributeid» – зовнішній ключ таблиці, пов’язує її з таблицею Attributes відношенням «багато-до-одного»;

– «Categoryid» – зовнішній ключ таблиці, пов’язує її з таблицею «Categories» відношенням «багато-до-одного».

Таблиця 3.5 – Таблиця «Categories»

Найменування	Тип поля	NOT NULL	Коментар
Categoryid	Integer	True	Первинний ключ Autonum = true
Categoryname	Text	True	Категорія

У таблиці «Compatibilities» зберігається інформація про сумісництво товарів. Зберігає ідентифікатори сумісних товарів.

Поле «ProductAttribyteId» пов’язує її відношенням «багато-до-одного» з таблицею «ProductsAttrubutes» по ідентифікатору першого товару. А поле «ProductAttribyteId_2» пов’язує її відношенням «багато-до-одного» з таблицею «ProductsAttrubutes» по ідентифікатору другого товару. У таблиці 3.6 відображена структура таблиці «Compatibilities».

Таблиця 3.6 – структура таблиці «Compatibilities»

Найменування	Тип поля	NOT NULL	Коментар
CompatibilityId	INTEGER	True	Первинний ключ AutoNum = True
ProductAttribyteId_1	INTEGER	True	Товар №1
ProductIdAttribyteId_2	INTEGER	True	Товар №2

«Manufacturer» – таблиця-сутність, призначена для зберігання даних про виробника товару. Структура даних зазначено у таблиці 3.7.

Таблиця 3.7 – «Manufacturer»

Найменування	Тип поля	NOT NULL	Коментар
ManufacturerId	INTEGER	True	Первинний ключ AutoNum = True
CompanyName	TEXT	True	Назва компанії виробника

«Orders» – таблиця-сутність, призначена для зберігання даних про замовлення. Поле «CreatedBy» пов’язує її відношенням «багато-до-одного» з таблицею «Users». Структура даних зазначено у таблиці 3.8.

Таблиця 3.8 – Orders

Найменування	Тип поля	NOT NULL	Коментар
OrderId	INTEGER	True	Первинний ключ AutoNum = True
CreatedBy	INTEGER	True	Id користувача, що прийняв замовлення
CreatedWhen	TEXT	False	Дата/час створення замовлення

У полі «Path» таблиці «Photos» зберігається відносний шлях до файлу фото. У таблиці 3.9 наведена структура таблиці «Photos».

Таблиця 3.9 – Таблиця «Photos»

Найменування	Тип поля	NOT NULL	Коментар
PhotoId	INTEGER	True	Первинний ключ AutoNum = True
Path	TEXT	True	Шлях до фото

«Users» – таблиця-сутність, призначена для зберігання даних користувачів додатку. Дана таблиця-сутність наведена у таблиці 3.10.

Таблиця 3.10 – Структура таблиці «Users»

Найменування	Тип поля	NOT NULL	Коментар
UserId	INTEGER	True	Первинний ключ AutoNum = True
UserName	TEXT	True	Ім’я користувача
Email	TEXT	True	Пошта користувача
Password	TEXT	True	Пароль, хеш-рядок утворена з паролю користувача, використовується для перевірки правильності вводу паролю

3.3 Розробка інтерфейсу користувача

При розробці стандартного сайту немає тривалого етапу проєктування, вигадування оригінальної графічної ідеї та нестандартної подачі матеріалу. Додаток повинен бути інтуїтивно зрозумілими всім категоріям користувачів.

При проєктуванні макету конфігуратора окрему увагу потрібно приділити розташуванню фільтрів. Краще відразу звузити поле варіантів, ніж отримати список товарів з величезною кількістю товарів із нерелевантних категорій. З цією метою список категорій розташовано на видному місці – зліва, а у мобільному варіанті – зверху.

На рисунках 3.3-3.4 зображені спроєктовані макети головної сторінки додатку та сторінки конфігуратора.

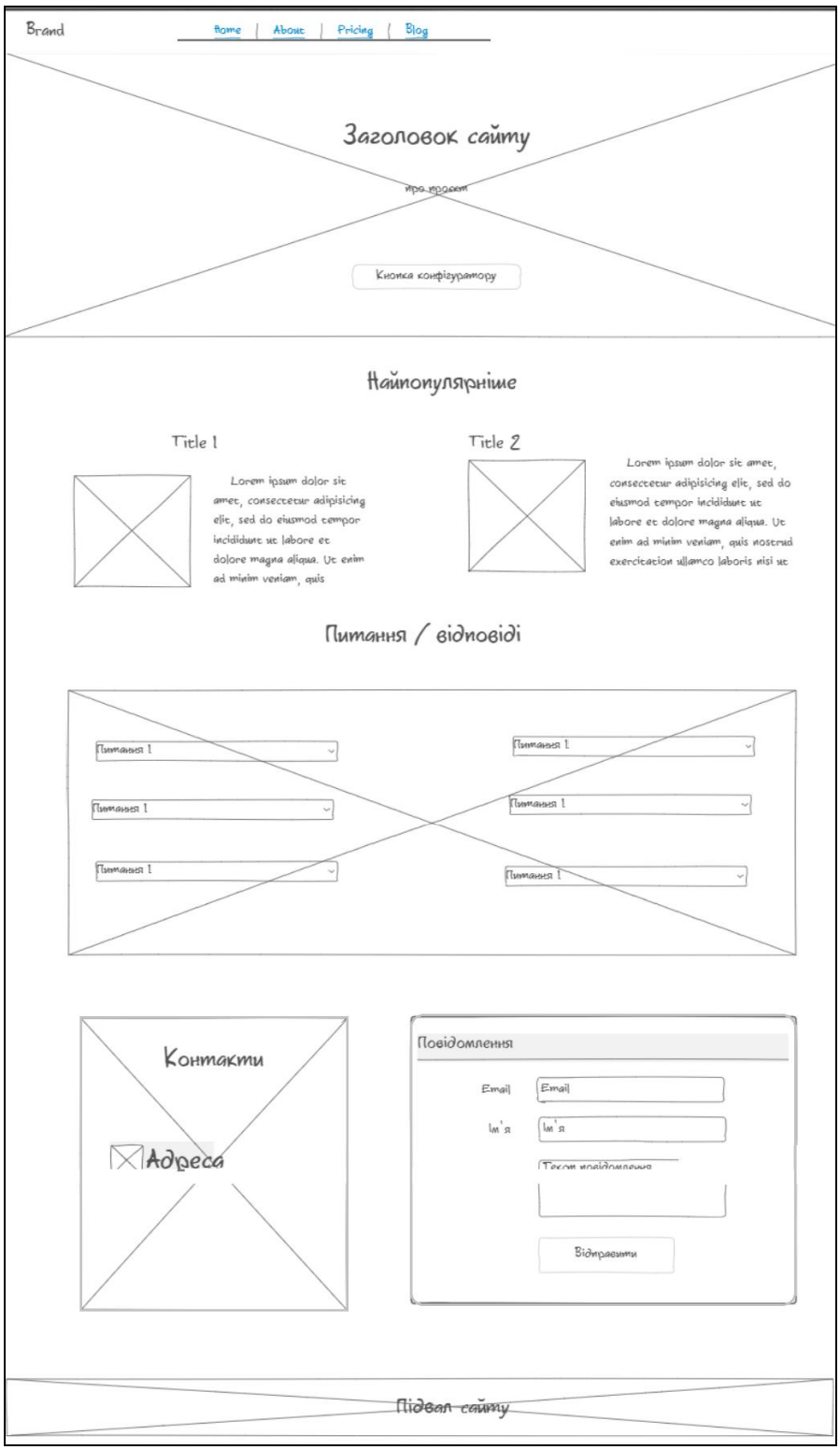


Рисунок 3.3 – Зовнішній макет структури головної сторінки

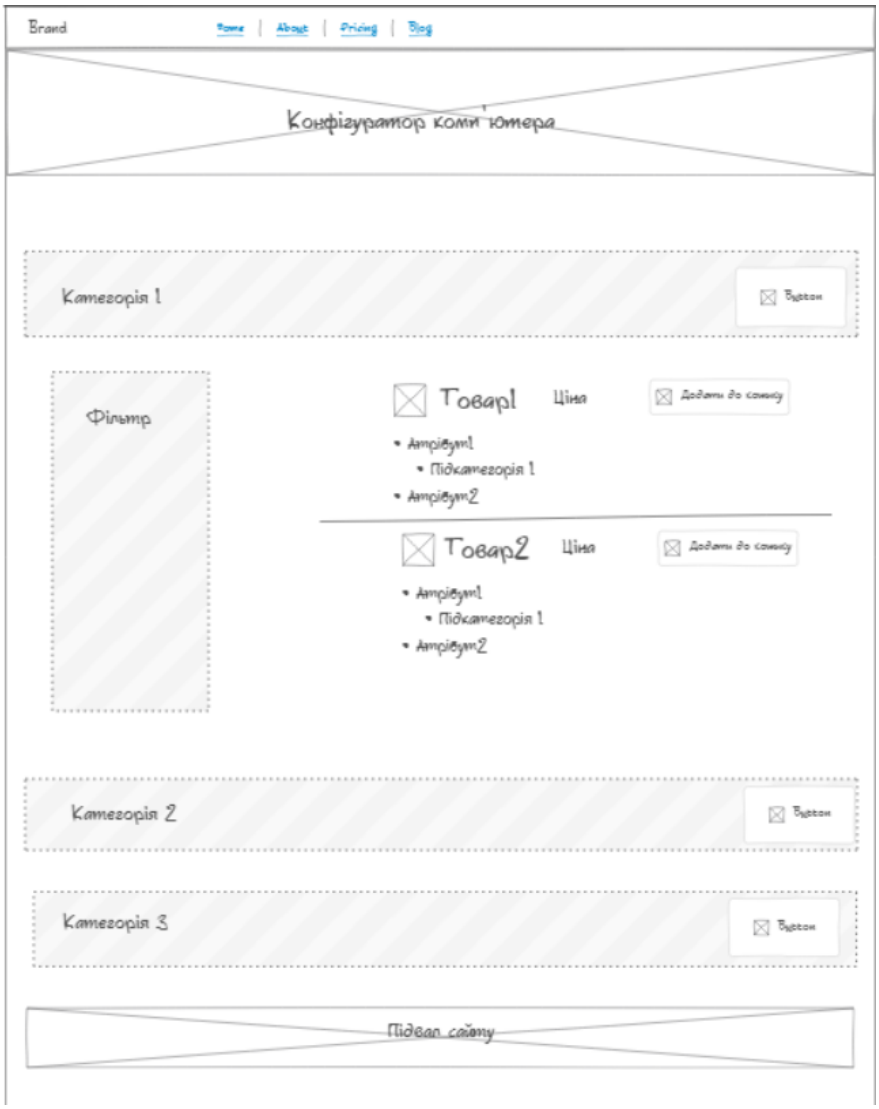


Рисунок 3.4 – Макет сторінки конфігуратора

За допомогою мови розмітки HTML5 створюємо базовий шаблон base.html та розширення шаблонів для зменшення повторюваної розмітки, яка зазвичай з'являється на кожній сторінці додатку, наприклад навігація, блок шапки сайту та підвал. Також сюди включені DOCTYPE, тег html, тег head, метатег, тег title та тег body.

Для визначення місця куди буде динамічно вставлятися розмітка з інших шаблонів під час виконання, використовуємо спеціальний синтаксис. Зовнішній вигляд додатку розробляється за допомогою бібліотеки Bootstrap та каскадних таблиць стилів CSS3.

3.4 Розробка вебдодатку

Для тестування та розробки вебдодатку використовується інтегрована середа розробки для мови програмування Python PyCharm.

Цей програмний продукт легко налаштовується, завдяки системі підказок були обрані та інстальовані необхідні плагіни (рис. 3.5) та версія інтерпретатору Python.

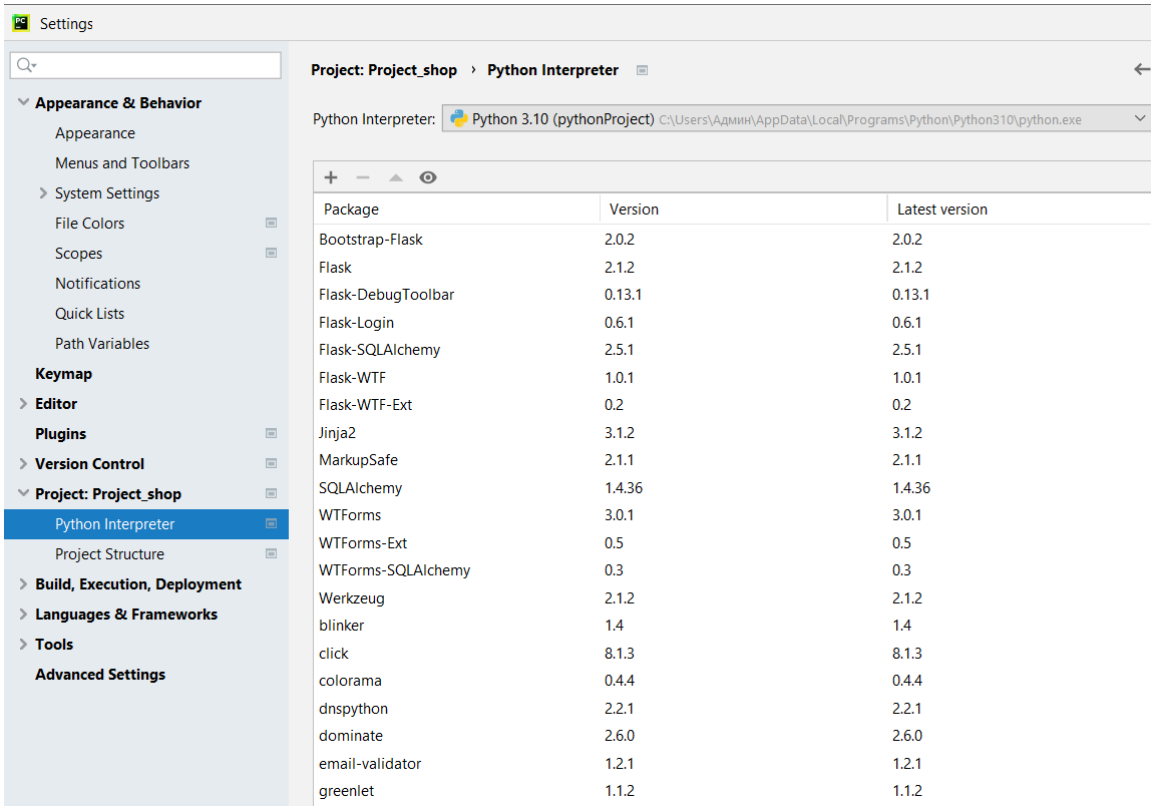


Рисунок 3.5 – Налаштування середовища розробки

Для функціонування вебдодатку потрібно становити наступні бібліотеки:

– мікрофреймворк Flask, використовується для обробки HTTP-запитів та формування інтерфейсу користувача;

– SQLAlchemy – програмна бібліотека, що синхронізує об’єкти Python та реляційної бази;

– WTForms – бібліотека перевірки та візуалізації форм для веброзробки на Python;

– Flask-DebugToolbar – плагін, який допомагає перевірити та налагодити файли, що виконуються.

PyCharm автоматично підсвічує синтаксис та забезпечує підтримку HTML, CSS, JavaScript. Всі маніпуляції будуть проводитися в файлах з розширенням «.py», а також «.css» та «.js».

На початку розробки потрібно створити директорію проєкту та розмістити в неї файли проєкту. На рисунку 3.6 зображена структура проєкту.

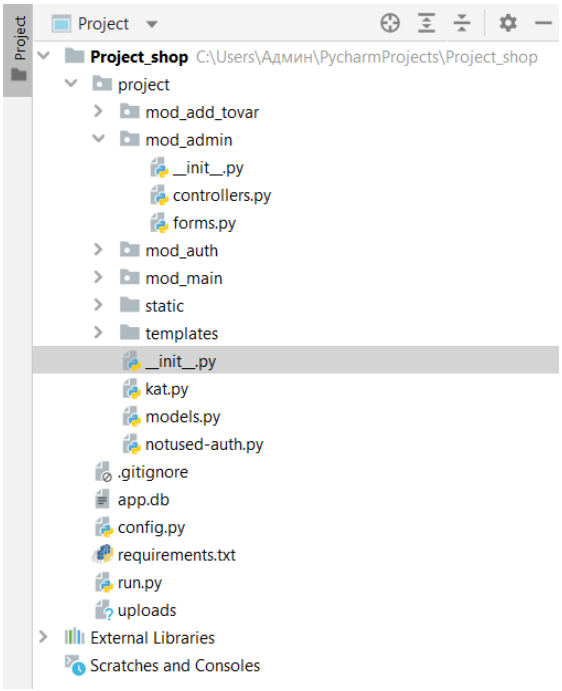


Рисунок 3.6 – Структура проєкту

Папка «static» є місцем зберігання статичних ресурсів проєкту, таких як css, js, images.

Папка «templates» зберігає HTML-шаблони проєкту.

Файл «run.py» – це вхід до проєкту.

Файл «config.py» – зберігає налаштування баз даних та параметри конфігурацій, необхідних для роботи та тестування системи. У таблиці 3.11 наведений опис основних параметрів конфігурації.

Таблиця 3.11 – Параметри конфігурації додатку

Параметр	Опис
BASE_DIR	Змінна, в яку міститься виконувана директорія скрипта.
DEBUG	Визначає появу повідомлень про помилки у тестовому оточенні.
SECRET_KEY	Використовується для підпису cookie, при його зміні користувачам потрібно логінитися вдруге.
SQLALCHEMY_DATABASE_URI DATABASE_CONNECT_OPTIONS	Налаштування підключення SQL-Alchemy.
CSRF_ENABLED WTF_CSRF_SECRET_KEY	Захищають від заміни POST-повідомлень.
BOOTSTRAP_BTN_STYLE	Містить налаштування кнопок форм.

У файлі «models.py» треба створити моделі, які будуть представляти таблиці бази даних у Python-коді. Було визначено дев’ять моделей у вигляді класів по кількості сутностей, а поля класів визначають назву та тип даних стовпців відповідних таблиць.

Далі створюються модулі у такому порядку: визначаємо моделі, пов’язані з моделями константи, форму, уявлення та шаблони.

Так були створені три основні модулі:

– модуль авторизації «mod_auth»;

– модуль адміністрування «mod_admin»;

– головний модуль «mod_main».

Усі модулі мають однакову структуру, що складається із трьох файлів. На рисунку 3.7 зображено структуру модулю «mod_auth».

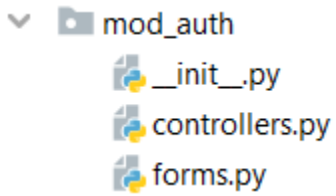


Рисунок 3.7 – Структура модулю «mod_auth»

У файлі «__init__.py» створюється екземпляр класу.

```
ACCESS = {
    'guest': 0,
    'user': 1,
    'admin': 77
}

AccessLevelForAdmin = 77
```

Рисунок 3.8 – Константи модулю

Константи модулю авторизації зберігають в собі рівні доступу користувачів. Функціонал додатку буде розподілено в залежності від рівня доступу користувача.

Файл «controllers.py» містить основні методи модулю, а файл «forms.py» – форми модулю. Шаблони кожного модулю містяться у папці «templates».

Наразі усі користувачі розроблюваного додатку поділяються на дві групи: неавторизовані користувачі та адміністратори. Але згодом планується включити ще третю групу авторизованих користувачів.

Якщо користувач авторизувався, він має доступ до усіх сторінок додатку, таких як «index.html», «/login», «/catalog», особистого кабінету, форми реєстрації користувача, та додавання нового товару. На сторінці перегляду профілю можлива зміна паролю. Передбачено перехід на будь-яку сторінку та вихід з облікового запису за допомогою пунктів меню з кожної сторінки.

Якщо користувач не авторизований, то він має доступ до усіх сторінок, окрім реєстрації, та форми додавання товару.

Форма реєстрації буде запитувати ім’я користувача, адресу електронної пошти та пароль, буде використано валідатори для перевірки коректності введених користувачем даних. Форма входу забезпечена полями «email» та «password» з аналогічними валідаторами. На випадок, якщо потрібно запам’ятати користувача, також додано поле «BooleanField» з ім’ям «remember». Це поле позначено, як «required», тобто користувач буде зобов’язаний відзначити чекбокс, що генерується формою. Перший параметр для кожного поля – його мітка, наприклад, для поля «email» у формі задана мітка «Ім’я або Email».

Створені форми використовують макроси для автоматизації створення html-полів. Оскільки цей макрос використовуватиметься в різних модулях, він поміщений в окремий файл «macros.html».

Модуль «mod_auth» реалізує методи реєстрації та авторизації користувача за допомогою бібліотек Flask-Login. Для реєстрації користувача було створено метод «signup()». За допомогою методу «SignUpForm(request.form)» генерується форма реєстрації користувача (рисунок 3.9). Через метод POST отримуються дані з форми, порівнюється поле «email» через виклик запиту до таблиці користувачів «User» «User.query.filter_by(Email=signup_form.email.data)». Якщо пошту не знайдено у базі, створюється новий екземпляр моделі користувача, інакше виконання методу буде припинено, а користувач отримує повідомлення про помилку.

Нового користувача буде записано до таблиці «User» через методи Flask-Sqlalchemy «db.session.add(new_user)» та «db.session.commit()».

Перелік існуючих користувачів сформовано за допомогою методу «User.query.all()». Дані у вигляді таблиці адміністратор переглядає на сторінці реєстрації.(рисунок 3.9).

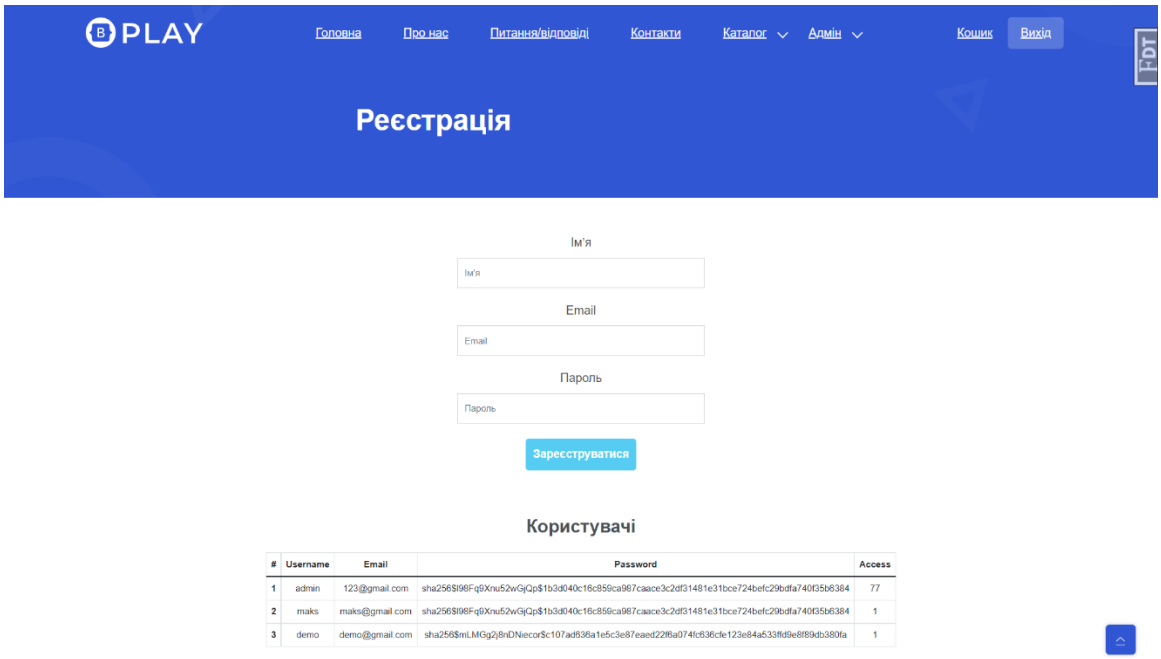


Рисунок 3.9 – Сторінка реєстрації користувача

Авторизацію користувачів системи реалізовано за допомогою методу «Signin()». Якщо авторизація вдала, користувач отримує доступ до свого профілю, де може додавати нові товари, а також переглядати існуючі замовлення. Якщо авторизація невдала, система формує повідомлення про помилку, та користувачеві пропонується ще раз ввести дані. За допомогою методу «LoginForm(request.form)», що успадковується від класу «Flask-Login», будується відповідна форма, що містить поля вводу логіну та паролю. Для механізму захисту від підробки на боці клієнта, додане приховане поле з ідентифікатором «csrf_token». При ініціюванні запиту в зовнішньому інтерфейсі, «csrf_token» передається серверній частині на перевірку у вигляді значення прихованого поля. Якщо перевірка відповідає csrf_token – запит не підроблено.

Модуль «mod_admin» дозволяє створювати нові картки товарів, редагувати та видаляти записи із бази даних. Метод «Products_create()» викликає метод «ProductCreateForm()» для створення відповідної форми. Метод «form_products_create.validate_on_submit()» виконує стандартні перевірки, щодо коректного вводу інформації, якщо дані введено без

помилок, інформація про новий товар зберігається відповідно у таблицях «Products», «Photos» та «ProductsAttributes»

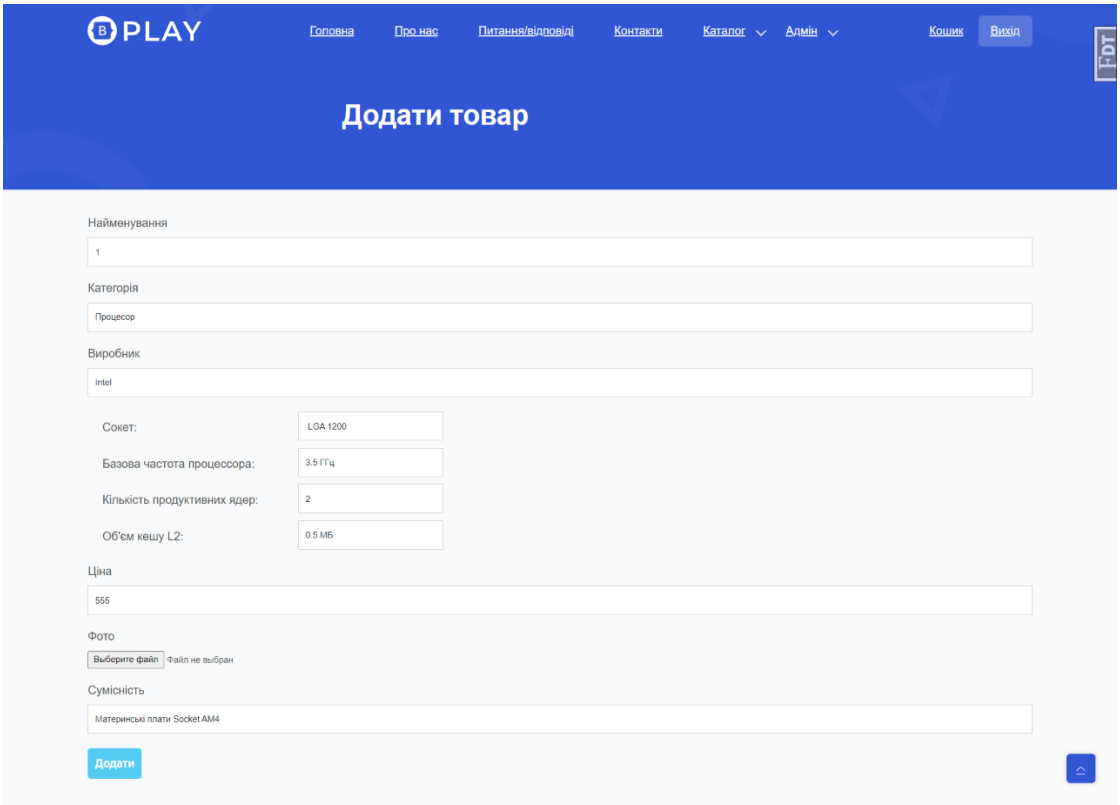


Рисунок 3.10 – Форма додавання товару

За допомогою методу «orders()» адміністратор може переглянути сформований список актуальних замовлень.

Модуль «mod_main» містить класи та методи генерації основних сторінок системи, таких як «Конфігуратор» та «Кошик». Для реалізації функцій пошуку та фільтрації за різноманітними критеріями було розроблено методи «filter_by_category()», «filter_by_manufacturer()», «filter_by_compatibility()». Усі методи «filter_by» виконують ідентичні запити до бази даних, що відрізняються лише таблицею, до якої створено запит. Наприклад, фільтр за сумісністю виконує фільтрацію за атрибутами товарів, використовуючи таблицю «ProductsAttrubutes».

Додавання товарів до кошику виконується за допомогою методу «add_product_to_cart» класу «Cart». При натисканні кнопки «Замовити» здійснюється відправка замовлення методом «submit_cart».

Навігаційна панель розроблена за допомогою бібліотеки «Bootstrap» та JavaScript. Панель навігації (рисунок 3.11) включає в себе логотип, випадаюче меню, та кнопки авторизації.

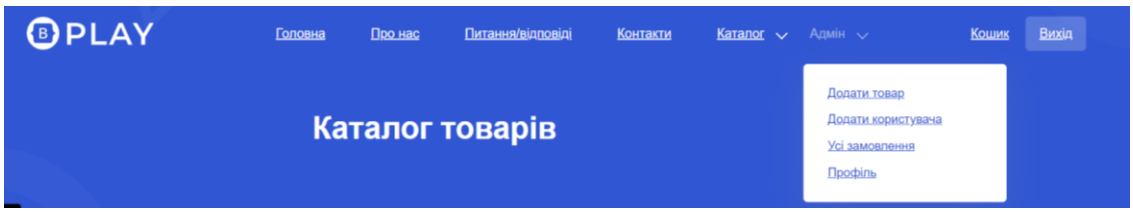


Рисунок 3.11 – Панель навігації додатку

Пункти «Про нас», «Питання/відповіді», «Контакти» це посилання-якоря на відповідні місця головної сторінки додатку. Пункт меню «Адмін» доступний лише для авторизованих користувачів. Для цього було розроблено метод «current_user.is_admin()», що визначає, чи авторизований користувач має права доступу адміністратора. Таким чином було сформовано кнопки «Вихід» для авторизованого користувача та «Авторизація» для неавторизованого відповідно.

Для зручності користувача було прийнято рішення зробити меню фіксованим у верхній частині додатку. Для цього була розроблена функція JavaScript «ud_header.classList.contains("sticky")», яка при скролінгу сторінки змінює css-клас навігаційної панелі на «sticky», який викликає властивість «position: fixid», а також змінює гаму панелі та логотипу.

Для того, щоб навігація прийнятно відображалася на пристроях з роздільною здатністю екрану менш ніж 700 пікселів, було прийнято рішення розгортати зміст панелі за допомогою спеціальної кнопки, або тоглеру. На рисунку 3.12 можна побачити скріншот екрану мобільного пристрою, меню сховано за допомогою спеціальної кнопки.

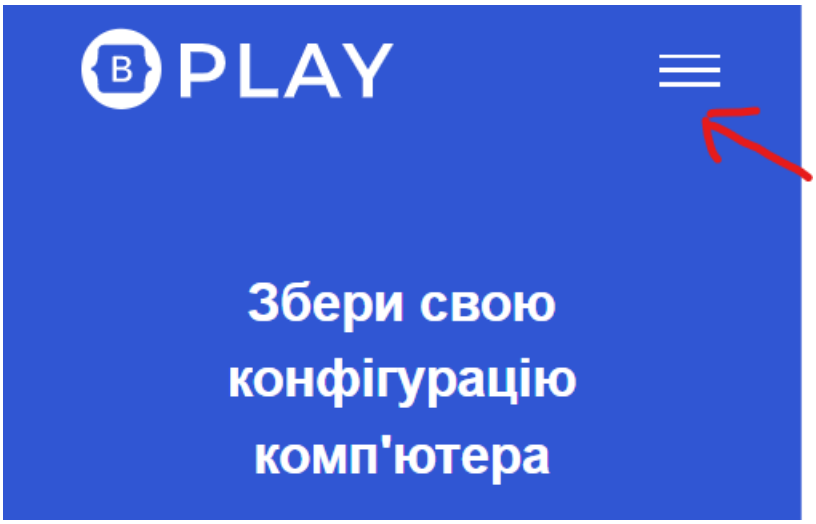


Рисунок 3.12 – Кнопка розгортання меню

3.5 Висновки за розділом

У даному розділі було розроблено програмний додаток автоматизації торгівлі комп’ютерної техніки, з можливістю конфігурування апаратної частини персональних комп’ютерів. По-кроково було спроєктовано та реалізовано модель бази даних для відображення даних предметної області, було розроблено програмні модулі обробки запитів користувача та користувацький інтерфейс. Розроблений додаток є вебдодатком та реалізує шаблон проєктування «модель-вид-контролер».

Були розроблені модулі авторизації, пошуку та додавання товарів за принципами розведення логіки та інтерфейсу. Використані можливості Python та фреймворку Flask для відокремлення видів від контролерів. Використано JavaScript для побудови інтерактивних інтерфейсів вебдодатку.

Для розробки інтерфейсу користувача були використані інструменти HTML5 та CSS3. А коректне відображення додатку на мобільних пристроях було досягнуто засобами бібліотеки Bootstrap 5 та медіа-запитами CSS.

4 ОПИС ТА ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

В даній кваліфікаційній роботі створений додаток був протестований з метою перевірки роботи основного функціоналу програмного додатку.

Були проведені функціональні типи тестів: тестування функціоналу збереження інформації до бази даних, можливість створення нового товару, та користувача, і протестована робота інтерфейсу користувача. В таблиці 4.1 зазначені загальні кроки тестування форми авторизації.

Таблиця 4.1 – Опис контрольного прикладу введення даних до форми авторизації

Назва тесту	Введення даних
Опис	Авторизація неіснуючого користувача.
Кроки тестування	– ввести в поле «Логін» неіснуючий логін; – ввести в поле пароль дані; – натиснути кнопку вхід; – ввести в поле «Логін» існуючий логін, залишити поле вводу пароля пустим; – натиснути кнопку вхід; – ввести в поле «Пароль» існуючий пароль, залишити поле вводу логіна пустим; – натиснути кнопку вхід.
Очікуваний результат	У поле записались введені данні, при натисненні кнопки «Вхід» виводиться інформація про помилку.

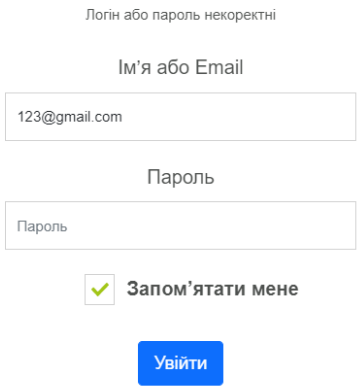


Рисунок 4.1 – Результат некоректного вводу імені користувача або паролю

На рисунку 4.1 ми бачимо результат некоректного вводу імені користувача або паролю. У поле записались введені данні, при натисненні кнопки «Увійти» виводиться інформація про помилку «Логін або пароль некоректні».

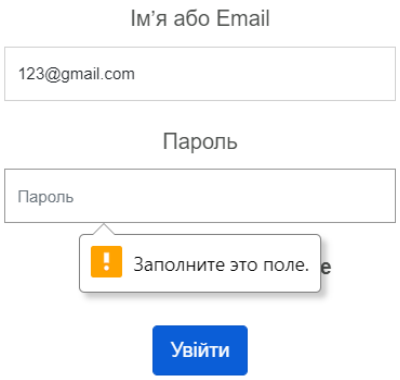


Рисунок 4.2 – Результат некоректного вводу імені користувача або паролю

В таблиці 4.2 зазначені загальні кроки тестування форми реєстрації.

Таблиця 4.2 – Тестування форми авторизації.

Назва тесту	Додавання даних
Опис	Реєстрація нового користувача.
Кроки тестування	– ввести в поле «Ім’я» існуючого користувача 123; – ввести в поле «Ім’я» нового користувача NoobikTV; – ввести в поле «Email» некоректну електронну адресу; – ввести в поле «Email» коректну електронну адресу; – ввести в поле «Пароль» дані; – натиснути кнопку реєстрації;
Очікуваний результат	– повідомлення, що користувач вже існує; – повідомлення, що електронна адреса несправжня; – повідомлення, що користувача додано; – У таблиці «Users» буде додано запис з даними нового користувача.

На рис 4.2-4.3 ми можемо по-кроково переглянути етапи тестування. Спочатку була спроба зареєструвати користувача з існуючим email «123@gmail.com», було видане повідомлення, про існуючий email.

Другим кроком була спроба створити користувача з некоректним e-email, в результаті, отримали повідомлення про помилку.

На третьому кроці, були введені коректні дані, в результаті отримали повідомлення про успішне створення користувача.

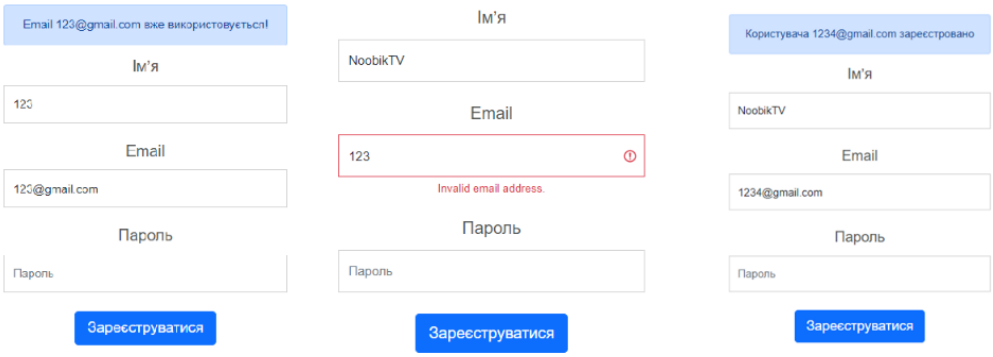
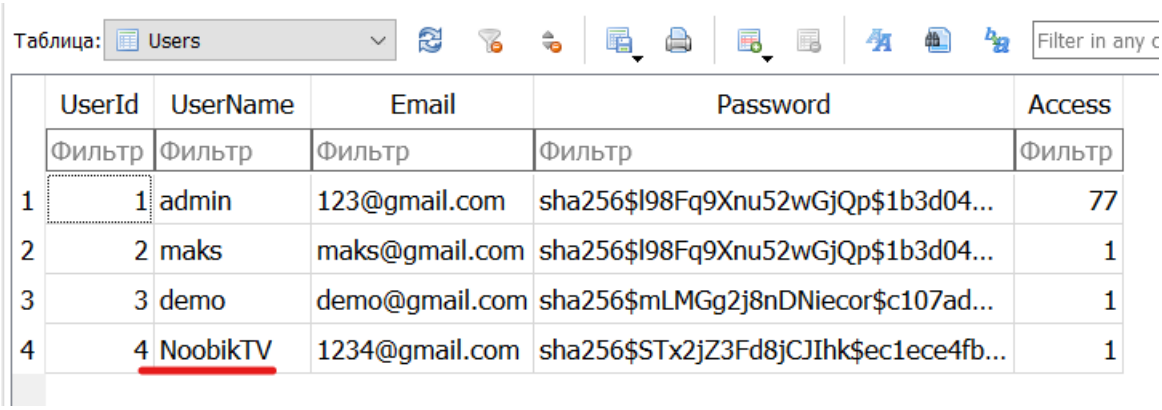


Рисунок 4.3 – Створення нового користувача



	UserId	UserName	Email	Password	Access
	Фільтр	Фільтр	Фільтр	Фільтр	Фільтр
1	1	admin	123@gmail.com	sha256$l98Fq9Xnu52wGjQp$1b3d04...	77
2	2	maks	maks@gmail.com	sha256$l98Fq9Xnu52wGjQp$1b3d04...	1
3	3	demo	demo@gmail.com	sha256$mLMGg2j8nDNiecor$c107ad...	1
4	4	NoobikTV	1234@gmail.com	sha256$STx2jZ3Fd8jCJIhk$ec1ece4fb...	1

Рисунок 4.4 – Додавання запису про нового користувача у базу даних

Для тестування роботи конфігуратора було використано розділ «SQLAlchemy queries» плагіну «Flask-DebugToolbar». На рисунку 4.5 зображено результати виконання та час обробки запитів до бази даних.

Рисунок 4.5 – Запити до бази даних при запуску конфігуратора

Тестування кросбраузерної сумісності – важлива складова частина розробки системи. Було вирішено перевірити, як виглядають всі вебсторінки додатку при перегляді в найпопулярніших браузерах, таких як Chrome, Firefox та Opera.

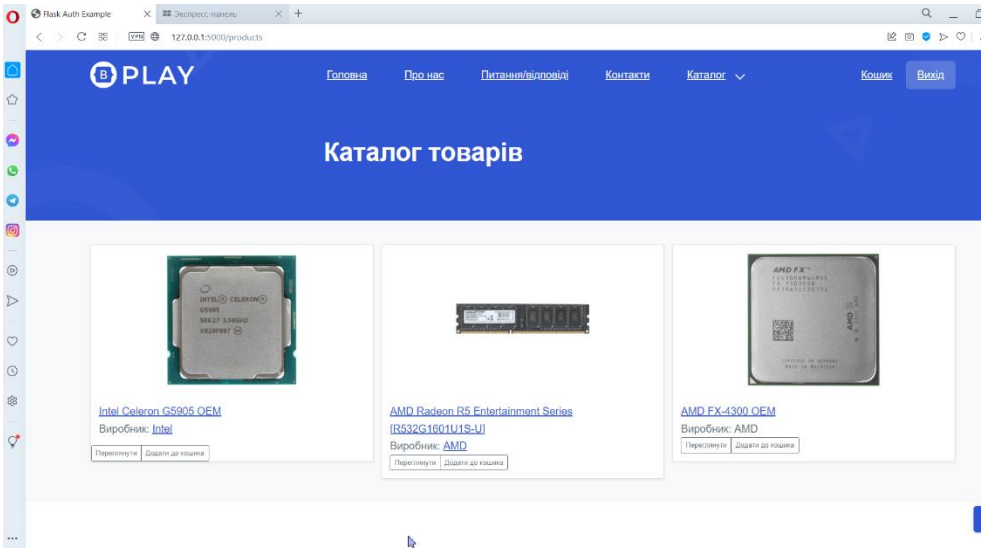


Рисунок 4.6 – Відображення сторінки каталогу в браузері Opera

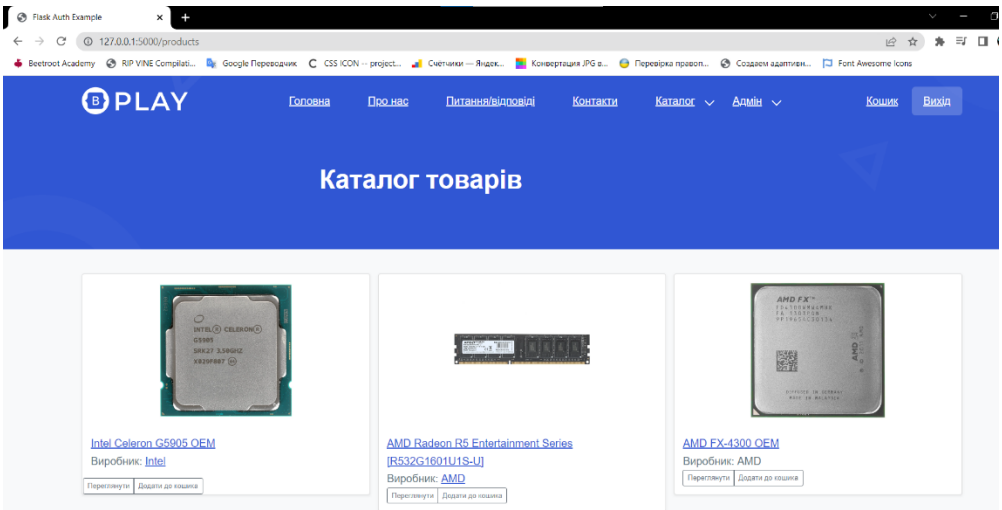


Рисунок 4.7 – Відображення сторінки каталогу в браузері Chrome

Із рисунків можна зробити висновки, що система однаково відображається на пристроях з роздрібною якістю більш ніж 1200 пікселів.

На рисунку 4.8 можна побачити, що на екранах мобільних пристроїв шапка сайту займає багато місця, а табличний вигляд фільтрів не вміщується, що не є зручним для користування користувачами.

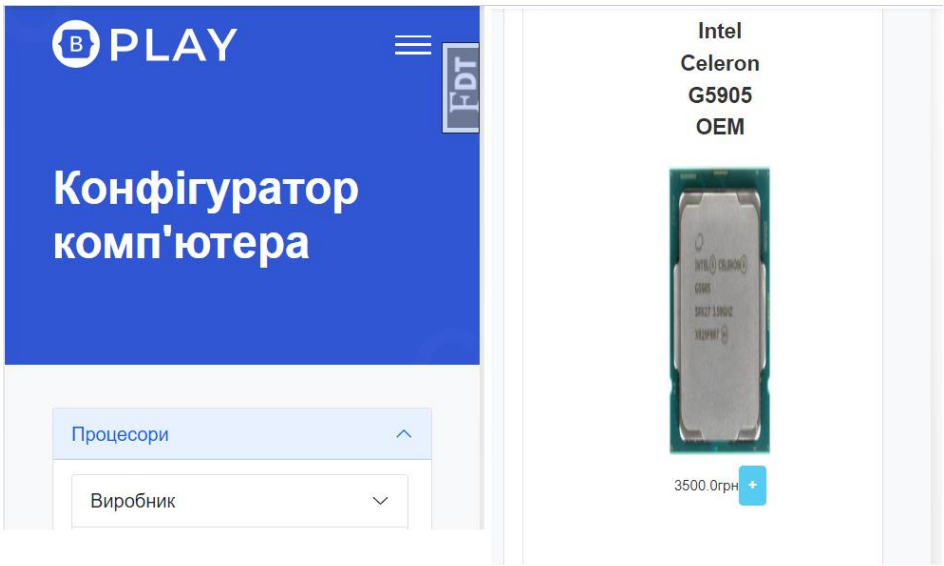


Рисунок 4.8 – Тестування додатку на мобільних пристроях

Щоб це виправити, були створені нові css-стилі для екранів з роздільною здатністю менш за 700 пікселів. Для цього було впроваджено систему медіа-запитів, яка призначена для створення адаптивних дизайнів.

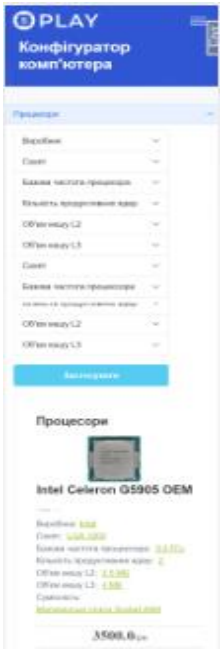


Рисунок 4.9 – Вигляд додатку на мобільному екрані після адаптації

В таблиці 4.3 зазначені загальні кроки тестування конфігуратору.

Таблиця 4.3 – Опис контрольного прикладу підбору комплектуючих, використовуючи наявний конфігуратор

Назва тесту	Робота конфігуратора
Опис	Підбір комплектуючих відповідно до їх взаємної сумісності.
Кроки тестування	– викликати конфігуратор для відображення всіх доступних комплектуючих; – обрати процесор із запропонованих та додати його у кошик; – обрати ОЗП з тих варіантів, що пропонує конфігуратор, спираючи на вибір процесора; – обрати материнську плату з тих, що пропонує конфігуратор, враховуючи обрані раніше комплектуючі; – обрати відеокарту з тих, що пропонує конфігуратор, враховуючи обрані раніше комплектуючі; – обрати дисковий накопичувач з тих, що пропонує конфігуратор, враховуючи обрані раніше комплектуючі.
Очікуваний результат	Під час використання конфігуратора, отримати коректний підбір комплектуючих, спираючи на їх сумісність та можливість працювати разом. Не виводити для користувача комплектуючі, які не сумісні із тими, що були обрані ним раніше.

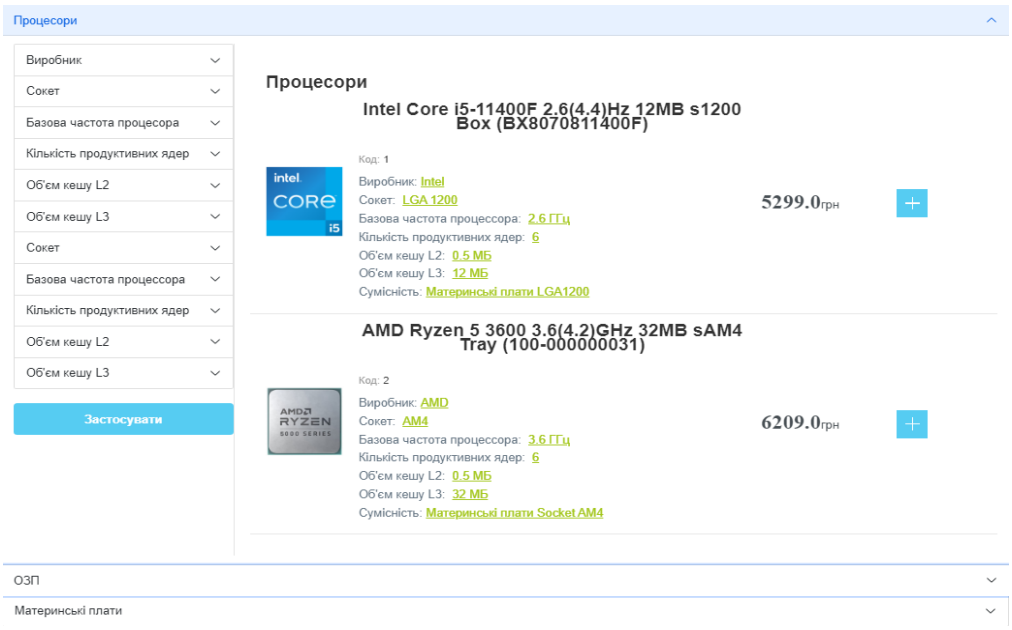


Рисунок 4.10 – Результат роботи конфігуратора при відображенні всіх доступних комплектуючих без урахування їх сумісності

На рисунку 4.10 ми бачимо результат представлення користувачу усіх доступних комплектуючих. На даному етапі не виконано вибір хоча б одного типу комплектуючого, тому відбір по сумісності не проводився. На рисунках 4.11-4.14 зображені наступні кроки вибору сумісних товарів.

Рисунок 4.11– Результат роботи конфігуратора при відображенні всіх доступних модулів пам’яті ОЗП після обрання процесора

На даному етапі користувач обрав процесор (рис.4.11), і перейшов до підбору ОЗП. Тому конфігуратор виводить лише ті позиції, які мають параметри, сумісні з обраним раніше процесором, наприклад, частота роботи модулів пам’яті, яка необхідна для роботи процесора, а також тип самої пам’яті, яку підтримує процесор.

Рисунок 4.12 – Результат роботи конфігуратора при відображенні всіх доступних материнських плат після обрання процесора та ОЗП

Після обрання декількох позицій, при переході користувача до вибору материнської плати, відображаються лише ті, які мають сумісність з процесором по параметру «Socket», а також з ОЗП по параметру «Тип пам’яті».

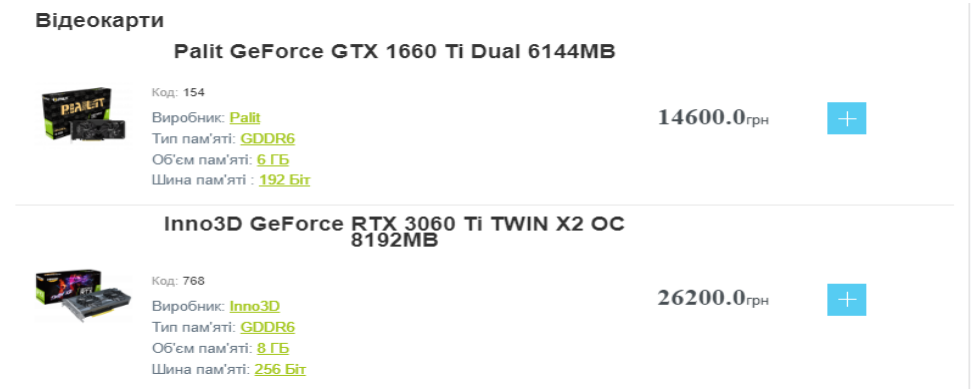


Рисунок 4.13 – Результат роботи конфігуратора при відображенні всіх доступних відеокарт після обрання процесора, ОЗП та материнської плати

Під час виведення доступних відеокарт, конфігуратор враховує лише можливість з’єднання відеокарти до материнської плати за своїм роз’ємом, всі інші параметри комплектуючого є сумісними з іншими.

Рисунок 4.14 – Результат роботи конфігуратора при відображенні всіх доступних дискових накопичувачів після обрання процесора, ОЗП, материнської плати та відеокарти

Після формування переліку дискових накопичувачів, які сумісні з обраними раніше комплектуючими, конфігуратор перевіряє можливість встановлення цього накопичувача на обраній материнській платі

ВИСНОВКИ

Під час виконання даного дипломного проєкту була розроблена програмна система для автоматизації торгівлі комп'ютерної техніки, яка дозволяє: шукати товари за різноманітними критеріями (включаючи, пошук за сумісництвом деталей конфігурації та категоріями); представляти товари у вигляді зручному для перегляду та специфічного пошуку; адміністратору керувати товарами.

У розділі «Аналіз існуючих рішень з проєктування та розробки вебінтерфейсу користувача системи» були визначені мета розробки, основна предметна область та її призначення. На основі порівняння існуючих видів онлайн додатків, визначилися, чому було обрано принцип побудови багатосторінкової вебсистеми. Описані функціональні та нефункціональні вимоги до вебзастосунку. Сформовані основні вимоги до дизайну та комунікацій. Виділено дві основні ролі користувачів та їх функції.

У розділі «Проєктування програмної системи та вибір програмних засобів її реалізації» було проведено системний аналіз розробленого додатку, проаналізовано та обрано інструменти розробки, створено структуру бази даних. Сформовані системні вимоги до програмного забезпечення. Описані покрокові алгоритми фільтрації товарів за категоріями та їх сумісництвом.

У розділі «Розробка програмного додатку для автоматизації торгівлі комп'ютерної техніки» була реалізована структура вебзастосунку, розроблена модель та архітектура додатку, створено інтерфейс користувача та основні модулі системи.

У розділі «Аналіз та тестування програмного додатку» було проаналізовано якість програмного продукту, розроблено основні тест кейси та описано процес розгортання вебзастосування на локальній машині. Розроблене вебзастосування відповідає усім вказаним вимогам, має належну якість, забезпечує безпеку та цілісність даних.

В процесі розробки програмного додатку були отримані наступні результати:

– проведений аналіз існуючих програмних систем, які вирішують питання в даній предметній області;

– сформована постановка задачі на роботу;

– здійснене проєктування процесів програмного продукту у вигляді діаграм, моделі баз даних і макетів користувальницького інтерфейсу;

– виконана програмна реалізація програмного продукту засобами Python, JavaScript, HTML, CSS;

– виконане тестування програмного додатку та усунені всі недоліки, що були виявлені під час тестування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вебсайт [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Вебсайт – Дата доступу: квітень 2022. – Назва з титул. екрана.

2. Що таке веб додаток? Різниця між сайтом, вебдодатком, SPA і PWA [Електронний ресурс]. – Режим доступу: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/ – Дата доступу: квітень 2022. – Назва з титул. екрана.

3. Лутц М. Программирование на Python / М. Лутц.— Символ-Плюс, 2011. – 992 с.

4. The Current State Of E-Commerce Filtering [Електронний ресурс]. – Режим доступу: https://www.smashingmagazine.com/2015/04/the-current-state-of-e-commerce-filtering/ – Дата доступу: квітень 2017. – Назва з титул. екрана.

5. Тлумачний словник української мови [Електронний ресурс]. – Режим доступу: https://slovnyk.ua/index.php – Дата доступу: квітень 2022. – Назва з титул. екрана

6. Гома, Хассан UML. Проєктування систем реального часу, паралельних і розподілених додатків / Хассан Гома. – М .: ДМК Пресс, 2016. – 700 с.

7. Програмування числових методів мовою Python / А.В. Анісімов, А.Ю. Дорошенко, С.Д. Погорілий, Я.Ю. Дорогий. – Київ: Видавничо-поліграфічний центр "Київський університет", 2014. – 640 с.

8. Шаймарданов Р.Б. Моделювання та автоматизація проєктування структур баз даних / Р.Б. Шаймарданов. – М.: Радио и связь, 2017. – 120 с.

9. Основи будування сайтів / В. Манако, О. Войченко, Д. Манако, О. Данилова, 2015. – 120 с.

10. Гультяев А.К. Проєктування і дизайн користувальницького інтерфейсу / А.К. Гультяев, В.А. Машин. – М .: Корона-Принт, 2015. – 350 с.

11. UML для бизнес-моделирования: зачем нужны диаграммы процессов [Електронний ресурс]. – Режим доступу: https://evergreens.com.ua/ru/articles/uml-diagrams.html – Дата доступу: квітень 2022. – Назва з титул. екрана.

12. Дрігалкін В.В. HTML в прикладах. Як створити свій Web-сайт: самоучитель В.В.. Дрігалкін. – М.: Діалектика, 2013. – 167 с.

13. HTTPS [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTTPS.

14. SSL [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SSL.

15. Загрози інформаційній безпеці [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: http://infosafery.blogspot.nl.

16. Фрейн Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств / Бэн Фрейн. – Санкт-Петербург: Издательский дом «Питер», 2016. – 634 с. – (2-е издание).

17. MVC [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Glossary/MVC – Дата доступу: квітень 2022. – Назва з титул. екрана.

18. Рейтинг мов програмування 2021: частка Python зменшується, а TypeScript обійшов C++ [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/language-rating-jan-2021/ – Дата доступу: квітень 2022. – Назва з титул. екрана.

19. Громов Г.Ю., Кириллов В.В. Введение в реляционные базы данных. - СПб.: БХВ – Петербург, 2009.

20. Appropriate Uses For SQLite [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: http://infosafery.blogspot.nl – Дата доступу: квітень 2022. – Назва з екрану.

21. Что такое PostgreSQL? [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://azure.microsoft.com/ru-ru/overview/what-is-postgresql – Дата доступу: квітень 2022. – Назва з екрану.

22. Рейтинг мов програмування 2021: частка Python зменшується, а TypeScript обійшов C++ [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/language-rating-jan-2021/ – Дата доступу: квітень 2022. – Назва з титул. екрана.

23. Дейт К. Дж. Введение в системы баз данных. – М.: Вильямс, 2006. – 1328 с. – ISBN 0-321-19784-4.

24. Редько В. Н., Бассараб И.А. Базы данных и информационные системы. – М.: Знание, 2011. – 602 с.

Додаток А

Зауваження нормоконтролера

Таблиця А.1 – Зауваження нормоконтролера

Сторінка	Позначення	Зміст зауваження
С.12, ЗЗ		Невірний перелік
ПЗ С. 14, 26…		Посилання на джерела не завжди у порядку використання
ПЗ		Нещільне заповнення сторінок ПЗ

Додаток Б

Лістинг програми

Б.1 Код методів модулів адміністратора mod_admin/controllers.

```
import os
from flask import Blueprint, render_template, flash, current_app
from flask_login import login_required, current_user
from werkzeug.utils import secure_filename
from project.mod_admin.forms import ProductCreateForm
from project.models import Order
admin_blueprint = Blueprint('admin', __name__, url_prefix='/admin')
@admin_blueprint.route('orders')
@login_required
def orders():
    all_orders = Order.query.all()
    return render_template('admin/orders.html', all_orders=all_orders,
current_user=current_user)
@admin_blueprint.route('/products/create', methods=['GET', 'POST'])
@login_required
def products_create():
    form_products_create = ProductCreateForm()
    if form_products_create.validate_on_submit():
        filename = secure_filename(form_products_create.Photo.data.filename)
        path_to_uploads = os.path.join(current_app.config['BASE_DIR'], 'project',
'uploads', filename)
        form_products_create.Photo.data.save(path_to_uploads)
        flash('Товар додано. %s !' % form_products_create.Title)
    return render_template('admin/products-create.html',
form_products_create=form_products_create)
mod_admin/forms.py
from flask_wtf import FlaskForm
from flask_wtf.file import FileField
from wtforms import StringField, SubmitField
from wtforms.validators import InputRequired
from wtforms_sqlalchemy.fields import QuerySelectField
from project.models import Manufacturer
def manufacturers_choices():
    return Manufacturer.query.all()
class ProductCreateForm(FlaskForm):
    Title = StringField('Найменування', [InputRequired(message='Название товара
обязательно')])
    Manufacturer = QuerySelectField(
        'Виробник',
        query_factory=manufacturers_choices,
        get_label='CompanyName'    )
    Price = StringField('Ціна', [InputRequired(message='Цена товара обязательно')])
    Photo = FileField('Фото')
    Submit = SubmitField('Додати')
```

Б.2 Створення бази даних, ініціалізація користувача, .

```
from flask import Flask, render_template
from flask_login import LoginManager
from flask_sqlalchemy import SQLAlchemy
from flask_bootstrap import Bootstrap4
from flask_debugtoolbar import DebugToolbarExtension

app = Flask(__name__)
app.config.from_object('config')
db = SQLAlchemy(app)
db.init_app(app)
Bootstrap4(app)
DebugToolbarExtension(app)

login_manager = LoginManager()
login_manager.login_view = 'auth.signin'
login_manager.init_app(app)

from project.models import User

@login_manager.user_loader
def load_user(user_id):
    # since the user_id is just the primary key of our user table, use it in the query for the
user
    return User.query.get(int(user_id))

# Sample HTTP error handling
@app.errorhandler(404)
def not_found(error):
    return render_template('404.html'), 404

# Import a module / component using its blueprint handler variable (mod_auth)
from project.mod_auth.controllers import auth_blueprint
from project.mod_main.controllers import main_blueprint
from project.mod_admin.controllers import admin_blueprint

# Register blueprint(s)
app.register_blueprint(auth_blueprint)
app.register_blueprint(main_blueprint)
app.register_blueprint(admin_blueprint)

# Build the database:
# This will create the database file using SQLAlchemy
db.create_all()
```

Б.3 Код методів модулів користувача mod_auth/controllers.

```
from flask import Blueprint, request, render_template, flash, session, redirect, url_for
from flask_login import login_user, logout_user, login_required
from sqlalchemy import or_
from werkzeug.security import check_password_hash, generate_password_hash
```

```
from project import db
from project.mod_auth.forms import SignUpForm, LoginForm
from project.models import User
# Define the blueprint: ‘auth’, set its url prefix: app.url/auth
# auth_blueprint = Blueprint(‘auth’, __name__, url_prefix=‘/auth’)
auth_blueprint = Blueprint(‘auth’, __name__)
@auth_blueprint.route(‘/signin’, methods=[‘GET’, ‘POST’])
def signin():
    users_demo = User.query.all()
    login_form = LoginForm(request.form)
    if login_form.validate_on_submit():
        name_or_email = or_(User.UserName == login_form.email.data, User.Email ==
login_form.email.data)
        user = User.query.filter(name_or_email).first()
        if user and check_password_hash(user.Password, login_form.password.data):
            # session[‘user_id’] = user.id
            login_user(user, remember=login_form.remember)
            flash(‘Ласкаво просимо, %s !’ % user.UserName)
            return redirect(url_for(‘main.profile’))
        flash(‘Помилковий логін або пароль’, ‘error-message’)
    return render_template(‘auth/signin.html’, all_users=users_demo,
login_form=login_form)
@auth_blueprint.route(‘/signout’)
@login_required
def signout():
    session.pop(‘_flashes’, None)
    logout_user()
    return redirect(url_for(‘auth.signin’))
@auth_blueprint.route(‘/signup’, methods=[‘GET’, ‘POST’])
def signup():
    signup_form = SignUpForm(request.form)
    if signup_form.validate_on_submit():
        existing_user = User.query.filter_by(Email=signup_form.email.data).first()
        if existing_user:
            flash(f’Email {existing_user.Email} вже використовується!’)
            return redirect(url_for(‘auth.signup’))
        # create new user with the form data. Hash the password so plaintext version isn’t
saved.
        new_user = User(
            UserName=signup_form.username.data,
            Email=signup_form.email.data,
            Password=generate_password_hash(signup_form.password.data,
method=‘sha256’),
            Access=1)
        db.session.add(new_user)
        db.session.commit()
        flash(f’Користувача {new_user.Email} зареєстровано’)
        return redirect(url_for(‘auth.signin’))
    return render_template(‘auth/signup.html’, signup_form=signup_form)
```

Б.4 Формування форми користувача

```
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, BooleanField, SubmitField
from wtforms.validators import InputRequired, Email
# Define the login form (WTForms)
class LoginForm(FlaskForm):
   email = StringField('Ім'я або Email',
                  [InputRequired(message='Forgot your email address?')],
                  render_kw={'placeholder': 'Ім'я або Email'}
                  )
   password = PasswordField('Пароль',
                     [InputRequired(message='Must provide a password. ;-)')],
                     render_kw={'placeholder': 'Пароль'}
                     )
   remember = BooleanField('Запом'ятати мене')
   submit = SubmitField('Увійти')
class SignUpForm(FlaskForm):
   username = StringField('Ім'я',
                    [InputRequired(message='Forgot your email address?')],
                    render_kw={'placeholder': 'Ім'я'}
                    )
   email = StringField('Email',
                  [Email(), InputRequired(message='Forgot your email address?')],
                  render_kw={'placeholder': 'Email'}
                  )
   password = PasswordField('Пароль',
                     [InputRequired(message='Must provide a password. ;-)')],
                     render_kw={'placeholder': 'Пароль'}
                     )
   submit = SubmitField('Зареєструватися')
```

Б.5 Обробка методів каталогу та фільтрів

```
from flask import Blueprint, render_template, abort
from flask_login import login_required, current_user
from project.mod_main.forms import FilterProductsForm
from project.models import Order, Category, User, AttributesByCategory
from markupsafe import escape
main_blueprint = Blueprint('main', __name__)
@main_blueprint.route('/')
def index():
   return render_template('index.html')
@main_blueprint.route('/configurator')
@login_required
def configurator():
   filter_form = FilterProductsForm()
   # .outerjoin(Category.attributes_by_categories) \
   filters = Category.query \
      .outerjoin(Category.attributes_by_categories) \
      .outerjoin(AttributesByCategory.Attribute) \
      .all()
   #print(filters)
```

```
            all_categories = Category.query.all()
            return render_template('main/configurator.html',
                                   allCategories=all_categories,
                                   filter_form=filter_form
                                   )
        @main_blueprint.route('/katalog')
        @login_required
        def katalog():
            return render_template('main.html', products=PRODUCTS)
        @main_blueprint.route('/profile')
        @login_required
        def profile():
            return render_template('main/profile.html', current_user=current_user)
        @main_blueprint.route('/myorders')
        @login_required
        def myorders():
            ###
            my_orders = Order.query. \
                join(User, User.id == Order.CreatedBy). \
                filter(User.id == current_user.id). \
                all()
            # current_user=current_user
            ### add check if there is no orders
            return render_template('main/myorders.html', my_orders=my_orders,
current_user=current_user)
        # @main_blueprint.route('/<productName>')
        # def product(productName):
        #     return productName
        # @main_blueprint.route('/products/<products:product>')
        # def product(products):
        #     # выведет subpath после /path/
        #     return render_template('main/product.html', products=product)
        @main_blueprint.route('/products')
        def products():
            return render_template('main/products.html', products=PRODUCTS)
        @main_blueprint.route('/product/<key>')
        def product(key):
            product = PRODUCTS.get(key)
            if not product:
                abort(404)
            return render_template('main/product.html', product=product)
        @main_blueprint.route('/curt')
        def curt():
            all_orders = Order.query.all()
            return render_template('main/curt.html', all_orders=all_orders,
current_user=current_user)
```

Б.6 Формування форм додатку

```
from flask_wtf import FlaskForm
from wtforms import SelectMultipleField, widgets
```

```
class MultiCheckboxField(SelectMultipleField):
    """
    A multiple-select, except displays a list of checkboxes.
    Iterating the field will produce subfields, allowing custom rendering of
    the enclosed checkbox fields.
    """
    widget = widgets.ListWidget(prefix_label=False)
    option_widget = widgets.CheckboxInput()
class FilterProductsForm(FlaskForm):
    Manufacturer = MultiCheckboxField('Виробник',
                                choices=[('cpp', 'Intel'), ('py', 'AMD')])
    Soket = MultiCheckboxField('Сокет',
                                choices=[('cpp', 'LGA 1200'), ('py', 'AM3+')])
    BaseProc = MultiCheckboxField('Базова частота процессора',
                        choices=[('cpp', '3.5 ГГц'), ('py', '3.8 ГГц')])
    Jadra = MultiCheckboxField('Кількість продуктивних ядер',
                        choices=[('cpp', '2'), ('py', '4')])
    KeshL2 = MultiCheckboxField('Об\'єм кешу L2',
                        choices=[('cpp', '0.5 МБ'), ('py', '4 МБ')])
    KeshL3 = MultiCheckboxField('Об\'єм кешу L3',
                        choices=[('cpp', '4 МБ')])
```

Б.7 Форми замовлень

```
{% from 'bootstrap4/table.html' import render_table %}
<!-- templates/profile.html -->
{% extends "base.html" %}
{% block content %}
<!-- ====== Banner Start ====== -->
<section class="ud-page-banner">
    <div class="container">
        <div class="row">
            <div class="col-lg-12">
                <div class="ud-banner-content">
                    <h1>Усі замовлення</h1>
                </div>
            </div>
        </div>
    </div>
</section>
<!-- ====== Banner End ====== -->
<main>
    <div class="album py-5 bg-light">
        <div class="container">
            <h1 class="title">Усі замовлення</h1>
            {% for ord in all_orders %}
            <h2 class="title">заказ номер {{ ord.OrderId }}  от {{ ord.CreatedWhen }}
                заказчик:{{ ord.CreatedByUser.UserName }}
                на сумму {{ ord.totalamount() }}
            </h2>
            <table class="table">
                <thead>
```

```
                <tr>
                    <td>Найменування</td>
                    <td>Виробник</td>
                    <td>Цінв</td>
                    <td>Замовник</td>
                    <td>Створено</td>
                </tr>
            </thead>
            {% for ProdInOrd in ord.products_in_orders %}
            <tr>
                <td>{{ ProdInOrd.Product.Title }}</td>
                <td>{{ ProdInOrd.Product.Manufacturer.CompanyName }}</td>
                <td>{{ ProdInOrd.Product.Price }}</td>
                <td></td>
                <td>{{ ord.CreatedWhen }}</td>
            </tr>
            {% endfor %}
        </table>
        {% endfor %}
    </div>
  </div>
</main>
{% endblock %}
```

Б.8 Форма створення продукту

```
{% from ‘bootstrap4/form.html’ import render_form %}
{% from ‘bootstrap4/utils.html’ import render_messages %}
{% from ‘bootstrap4/table.html’ import render_table %}
{% extends ‘base.html’ %}
{% block title %}
<title>shop</title>
{% endblock %}
{% block content %}
<!-- ====== Banner Start ====== -->
<section class="ud-page-banner">
  <div class="container">
    <div class="row">
      <div class="col-lg-12">
        <div class="ud-banner-content">
          <h1>Додати товар</h1>
        </div>
      </div>
    </div>
  </div>
</section>
<!-- ====== Banner End ====== -->
<main>
  <div class="album py-5 bg-light">
    <div class="container">
      {{ render_messages() }}
```

```
            {{ render_form(form_products_create, button_style='w-100 btn btn-lg btn-
primary') }}
          </div>
        </div>
      </main>
      {% endblock %}
```

Б.9 Форма реєстрації

```
{% from 'bootstrap4/form.html' import render_form %}
{% from 'bootstrap4/utils.html' import render_messages %}
<!-- templates/signup.html -->
{% extends "base.html" %}
{% block content %}
<!-- ====== Banner Start ====== -->
<section class="ud-page-banner">
  <div class="container">
    <div class="row">
      <div class="col-lg-12">
        <div class="ud-banner-content">
          <h1>Реєстрація</h1>
        </div>
      </div>
    </div>
  </div>
</section>
<!-- ====== Banner End ====== -->
<main class="form-signin text-center ">
  {{ render_messages() }}
  {{ render_form(signup_form, button_style='w-100 btn btn-lg btn-primary') }}
</main>
{% endblock %}
```

Б.10 Форма Авторизації

```
{% from 'bootstrap4/form.html' import render_form %}
{% from 'bootstrap4/utils.html' import render_messages %}
{% from 'bootstrap4/table.html' import render_table %}
<!-- templates/signin.html -->
{% extends "base.html" %}
{% block content %}
<!-- ====== Banner Start ====== -->
<section class="ud-page-banner">
  <div class="container">
    <div class="row">
      <div class="col-lg-12">
        <div class="ud-banner-content">
          <h1>Авторизація</h1>
        </div>
      </div>
    </div>
```

```
    </div>
  </section>
  <!-- ====== Banner End ====== -->
  {{ render_table(all_users) }}
  <main class="form-signin text-center ">
    {{ render_messages() }}
    {{ render_form(login_form, button_style='w-100 btn btn-lg btn-primary') }}
  </main>
  {% endblock %}
```

Б.11 Форма конфігуратора

```
  {% from 'bootstrap4/form.html' import render_form, render_form_row %}
  {% from 'macros.html' import render_accordion_item %}
  {% macro render_filter_row(attribute) -%}
  {% call render_accordion_item('filter-by-attribute-' ~ attribute.AttributeId,
attribute.AttributeTitle) %}
  <div class="form-check">
    <input class="form-check-input" type="checkbox" value="">
    <label class="form-check-label" for="flexCheckDefault">
  Checked checkbox
    </label>
  </div>
  <div class="form-check">
    <input class="form-check-input" type="checkbox" value="" checked>
    <label class="form-check-label" for="flexCheckChecked">
      Checked checkbox
    </label>
  </div>
  {% endcall %}
  {%- endmacro %}
  {% macro render_filter_block(category) -%}
  <div class="b-cwp-bi-side">
    <div class="accordion">
      <form method="post">
        {{ filter_form.csrf_token() }}
        {% call render_accordion_item('filter-by-manufacturer', 'Виробник') %}
        {{ render_form_row(filter_form.Manufacturer) }}
        {% endcall %}
        {% call render_accordion_item('filter-by-soket', 'Сокет') %}
        {{ render_form_row(filter_form.Soket) }}
        {% endcall %}
        {% call render_accordion_item('filter-by-BaseProc', 'Базова частота
процесора') %}
        {{ render_form_row(filter_form.BaseProc) }}
        {% endcall %}
        {% call render_accordion_item('filter-by-Jadra', 'Кількість продуктивних
ядер') %}
        {{ render_form_row(filter_form.Jadra) }}
        {% endcall %}
        {% call render_accordion_item('filter-by-KeshL2', 'Об\'єм кешу L2') %}
        {{ render_form_row(filter_form.KeshL2) }}
```

```
                {% endcall %}
                {% call render_accordion_item('filter-by-KeshL3', 'Об\'єм кешу L3') %}
                {{ render_form_row(filter_form.KeshL3) }}
                {% endcall %}
                {% for ac in category.attributes_by_categories %}
                {{render_filter_row(ac.Attribute)}}
                {% endfor %}
                <button data-category="{{category.CategoryId}}" type="button" class="filter btn
btn-primary">Застосувати</button>
              </form>
            </div>
          </div>
          {%- endmacro %}
          {% macro render_attribute_row(productAttribute) -%}
          {{productAttribute.Attribute.AttributeTitle}}: 
          <a href="/processor/filter/socket-am4/"
target="_blank">{{productAttribute.AttributeValue}}</a><br>
          {%- endmacro %}
          {% macro render_product_row(product) -%}
          <li class="b-item" data-id_product="{{product.ProductId}}" data-
price="{{product.Price}}">
            <div class="b-cng-item-prod product_wrapper" data-
id_product="{{product.ProductId}}"
                data-prod-id="{{product.ProductId}}" data-prod-articul="{{product.ProductId}}"
                data-prod-name="{{product.Title}}"
                data-prod-brand="{{product.Manufacturer.CompanyName}}" data-prod-variant=""
                data-prod-type="{{product.Category.CategoryName}}"
                data-prod-
category="{{product.Category.CategoryName}}/{{product.Manufacturer.CompanyName}}"
                data-prod-price="{{product.Price}}" data-prod-list="Конфігуратор"
                data-prod-position="">
                <div class="b-cngip-pic">
                  <a href="/products/amd-ryzen-5-3600-3642ghz-32mb-sam4-tray-100-
000000031/"
                    target="_blank">
                    <img src="{{ product.photos[0].file_path() }}"
                      width="100" height="100"
                      alt="Фото {{product.Title}}">
                  </a>
                </div>
                <div class="b-cngip-mid-col">
                  <div class="meta hidden"></div>
                  <div class="b-cngip-head">
                    <a target="_blank" class="shop-item-title"
href="/products/{{product.Title}}">{{product.Title}}</a>
                  </div>
                  <div class="b-cngip-mc-meta">
                    <div class="b-i-product-articul">Код: <i>{{product.ProductId}}</i></div>
                    <div class="b-cngip-short-chars">
                      <p>Виробник: <a href="/processor/filter/amd-ryzen-r5/"
target="_blank">{{product.Manufacturer.CompanyName}}</a><br>
                        {% for pa in product.products_attributes %}
```

```
                        {{ render_attribute_row(pa) }}
                        {% endfor %}
                        Сумісність: <a href="/motherboard/filter/socket-am4/"
target="_blank">Материнські плати Socket
                            AM4</a><br>
                    </p>
                  </div>
                </div>
              </div>
              <div class="b-cngip-meta-col">
                <div class="b-cngip-price">{{product.Price}}<i>грн</i></div>
              </div>
              <div class="b-cngip-meta-btn">
                <!-- <button  class="b-cngip-btn-add add_assembly_product" data-modal-
target="#modal">X</button>-->
                 <button  class="b-cngip-btn-add add_assembly_product btn btn-primary shop-
item-button"  type="button" data-modal-target="#modal">X</button>
                <ul class="b-htconf-btns-list">
                  <li>
        <!--                <a class="b-cngip-btn-add add_assembly_product"-->
        <!--                  href="javascript: AlertMsg();">Додати</a>-->
                  </li>
                </ul>
              </div>
            </div>
        </li>
        {%- endmacro %}
        {% macro renderCategoryCard(cat) -%}
        {% call render_accordion_item('cat-' ~ cat.CategoryId, cat.CategoryName) %}
        <div class="b-cwp-body-inner-pad" style="display: table;">
          {{ render_filter_block(cat) }}
          <div class="b-cwp-bi-main">
            <div class="b-catalog-top-info b-no-border">
              <div class="b-pagehead"><span class="b-head">{{ cat.CategoryName
}}</span></div>
              <ul class="b-cwp-bi-result-list">
                {% for product in cat.products %}
                {{ render_product_row(product) }}
                {% endfor %}
                <section class="container content-section">
              <{% macro render_product_row(product) -%}
        <li class="b-item" data-id_product="{{product.ProductId}}" data-
price="{{product.Price}}">
          <div class="b-cng-item-prod product_wrapper" data-
id_product="{{product.ProductId}}"
            data-prod-id="{{product.ProductId}}" data-prod-articul="{{product.ProductId}}"
            data-prod-name="{{product.Title}}"
            data-prod-brand="{{product.Manufacturer.CompanyName}}" data-prod-variant=""
            data-prod-type="{{product.Category.CategoryName}}"
            data-prod-
category="{{product.Category.CategoryName}}/{{product.Manufacturer.CompanyName}}"
            data-prod-price="{{product.Price}}" data-prod-list="Конфігуратор"
```

```
                data-prod-position="">
                <div class="b-cngip-pic">
                  <a href="/products/amd-ryzen-5-3600-3642ghz-32mb-sam4-tray-100-
000000031/"
                    target="_blank">
                     <img src="{{ product.photos[0].file_path() }}"
                        width="100" height="100"
                        alt="Фото {{product.Title}}">
                  </a>
                </div>
                <div class="b-cngip-mid-col">
                  <div class="meta hidden"></div>
                  <div class="b-cngip-head">
                    <a target="_blank" class="shop-item-title"
href="/products/{{product.Title}}">{{product.Title}}</a>
                  </div>
                  <div class="b-cngip-mc-meta">
                    <div class="b-i-product-articul">Код: <i>{{product.ProductId}}</i></div>
                    <div class="b-cngip-short-chars">
                      <p>Виробник: <a href="/processor/filter/amd-ryzen-r5/"
target="_blank">{{product.Manufacturer.CompanyName}}</a><br>
                        {% for pa in product.products_attributes %}
                        {{ render_attribute_row(pa) }}
                        {% endfor %}
                        Сумісність: <a href="/motherboard/filter/socket-am4/"
target="_blank">Материнські плати Socket
                          AM4</a><br>
                      </p>
                    </div>
                  </div>
                </div>
                <div class="b-cngip-meta-col">
                  <div class="b-cngip-price">{{product.Price}}<i>грн</i></div>
                </div>
                <div class="b-cngip-meta-btn">
                  <!-- <button  class="b-cngip-btn-add add_assembly_product" data-modal-
target="#modal">X</button>-->
                   <button  class="b-cngip-btn-add add_assembly_product btn btn-primary shop-
item-button"  type="button" data-modal-target="#modal">X</button>
                  <ul class="b-htconf-btns-list">
                    <li>
        <!--              <a class="b-cngip-btn-add add_assembly_product"-->
        <!--                href="javascript: AlertMsg();">Додати</a>-->
                    </li>
                  </ul>
                </div>
              </div>
        </li>
        {%- endmacro %}
              </section>
                  </ul>
                  <div class="clear"></div>
```

```
            </div>
          </div>
        </div>
        {% endcall %}
        {%- endmacro %}
        {% extends 'base.html' %}
        {% block title %}
        <title>shop</title>
        {% endblock %}
        {% block content %}
        <!-- ====== Banner Start ====== -->
        <section class="ud-page-banner">
          <div class="container">
            <div class="row">
              <div class="col-lg-12">
                <div class="ud-banner-content">
                  <h1>Конфігуратор комп'ютера</h1>
                </div>
              </div>
            </div>
          </div>
        </section>
        <!-- ====== Banner End ====== -->
        <main>
         <section class="container content-section">
                <div class="cart-row">
                  <span class="cart-item cart-header cart-column">ITEM</span>
                  <span class="cart-price cart-header cart-column">PRICE</span>
                  <span class="cart-quantity cart-header cart-column">QUANTITY</span>
                </div>
                <div class="cart-items">
                </div>
                <div class="cart-total">
                  <strong class="cart-total-title">Total</strong>
                  <span class="cart-total-price">$0</span>
                </div>
                <button class="btn btn-primary btn-purchase"
type="button">PURCHASE</button>
              </section>
          <div class="album py-5 bg-light">
            <div class="container">
              <div class="b-config-head-fscr-mob">
                <div class="b-head">Збірка конфігурації ПК з нуля</div>
              </div>
              <div class="accordion" id="accordionExample">
                {% for cat in allCategories %}
                {{ renderCategoryCard(cat) }}
                {% endfor %}
              </div>
               <div class="modal" id="modal">
          <div class="modal-header">
           <div class="title"> Додано до конфігурації</div>
```

```
    <button data-close-button class="close-button">&times;</button>
   </div>
  </div>
  <div id="overlay"></div>
      </div>
   </div>
</main>
{% endblock %}
{% block scripts %}
<script src="{{ url_for('static', filename='js/configurator.js') }}"></script>
{% endblock %}
{% block js_block %}
<script src="{{ url_for('static', filename='js/curt.js') }}"></script>
{% endblock %}

     const pageLink = document.querySelectorAll(".ud-menu-scroll");
     pageLink.forEach((elem) => {
       elem.addEventListener("click", (e) => {
         e.preventDefault();
         document.querySelector(elem.getAttribute("href")).scrollIntoView({
           behavior: "smooth",
           offsetTop: 1 - 60,

    function onScroll(event) {
       const sections = document.querySelectorAll(".ud-menu-scroll");
       const scrollPos =
         window.pageYOffset ||
         document.documentElement.scrollTop ||
         document.body.scrollTop;
       for (let i = 0; i < sections.length; i++) {
         const currLink = sections[i];
         const val = currLink.getAttribute("href");
         const refElement = document.querySelector(val);
         const scrollTopMinus = scrollPos + 73;
         if (
           refElement.offsetTop <= scrollTopMinus &&
           refElement.offsetTop + refElement.offsetHeight > scrollTopMinus
         ) {
           document
             .querySelector(".ud-menu-scroll")
             .classList.remove("active");
           currLink.classList.add("active");
         } else {
           currLink.classList.remove("active");
         }
       }
     }
     window.document.addEventListener("scroll", onScroll);
   </script>
```

Б.12 Макрос виводу товарів у формі «акордеону»

```
{% macro render_accordion_item(key, header) -%}
<div class="accordion-item">
   <h2 class="accordion-header">
      <button class="accordion-button collapsed" type="button"
            data-bs-toggle="collapse" data-bs-target="#panelsStayOpen-{{key}}"
            aria-expanded="false" aria-controls="panelsStayOpen-{{key}}">
         {{ header }}
      </button>
   </h2>
   <div class="accordion-collapse collapse"
       id="panelsStayOpen-{{key}}"
       aria-labelledby="panelsStayOpen-{{key}}">
      <div class="accordion-body">
         {{ caller() }}
      </div>
   </div>
</div>{%- endmacro %}
```

Б.13 Створення моделей

```
# # coding: utf-8
from flask_login import UserMixin
from sqlalchemy import func

from project import db

ACCESS = {
   'guest': 0,
   'user': 1,
   'admin': 77
}
AccessLevelForAdmin = 77
db.metadata.clear()

class Attribute(db.Model):
   __tablename__ = 'Attributes'
   AttributeId = db.Column(db.Integer, primary_key=True)
   AttributeTitle = db.Column(db.Text, nullable=False, unique=True)
class AttributesByCategory(db.Model):
   __tablename__ = 'AttributesByCategories'
   __table_args__ = (
      db.UniqueConstraint('CategoryId', 'AttributeId'),
   )

   AttributeInCategoryId = db.Column(db.Integer, primary_key=True)
   AttributeId = db.Column(db.ForeignKey('Attributes.AttributeId'), nullable=False,
index=True)
   CategoryId = db.Column(db.ForeignKey('Categories.CategoryId'), nullable=False,
index=True)
   Attribute = db.relationship('Attribute', primaryjoin='AttributesByCategory.AttributeId
== Attribute.AttributeId',          backref='attributes_by_categories')
```

```
            Category = db.relationship('Category', primaryjoin='AttributesByCategory.CategoryId
== Category.CategoryId',                    backref='attributes_by_categories')
        class Category(db.Model):
            __tablename__ = 'Categories'

            CategoryId = db.Column(db.Integer, primary_key=True)
            CategoryName = db.Column(db.Text, nullable=False, unique=True)
            def __repr__(self):
                return f'{self.CategoryId}. {self.CategoryName}'
         class Compatibility(db.Model):
            __tablename__ = 'Compatibilities'
            __table_args__ = (
                db.UniqueConstraint('ProductId1', 'ProductId2'),
            )

            CompatibilityId = db.Column(db.Integer, primary_key=True)
            ProductId1 = db.Column(db.ForeignKey('Products.ProductId'), nullable=False,
index=True)
            ProductId2 = db.Column(db.ForeignKey('Products.ProductId'), nullable=False,
index=True)

            Product = db.relationship('Product', primaryjoin='Compatibility.ProductId1 ==
Product.ProductId',
                                      backref='product_compatibilities')
            Product1 = db.relationship('Product', primaryjoin='Compatibility.ProductId2 ==
Product.ProductId',
                                       backref='product_compatibilities_0')
        class Manufacturer(db.Model):
            __tablename__ = 'Manufacturers'

            ManufacturerId = db.Column(db.Integer, primary_key=True)
            CompanyName = db.Column(db.Text, nullable=False, unique=True)
        class Order(db.Model):
            __tablename__ = 'Orders'

            OrderId = db.Column(db.Integer, primary_key=True)
            CreatedBy = db.Column(db.ForeignKey('Users.UserId'), nullable=False, index=True)
            CreatedWhen = db.Column(db.Text)

            CreatedByUser = db.relationship('User', primaryjoin='Order.CreatedBy == User.id',
backref='orders')

             select sum(products.price)
              FROM
              Products
             inner JOIN
             (SELECT ProductsInOrders.ProductId from ProductsInOrders WHERE
ProductsInOrders.orderid= @CurrentOrderId) t2
             on Products.ProductId = t2.ProductId

            def totalamount(self):
```

```
                qw=ProductsInOrder.query.filter(ProductsInOrder.OrderId ==
self.OrderId).subquery()
                qw1= Product.query.join(qw, Product.ProductId == qw.c.ProductId).subquery()
                sum=Product.query.with_entities(func.sum(qw1.c.Price).label('total')).first().total

                return sum

        class Photo(db.Model):
            __tablename__ = 'Photos'

            PhotoId = db.Column(db.Integer, primary_key=True)
            ProductId = db.Column(db.ForeignKey('Products.ProductId'), nullable=True,
index=True)
            FileName = db.Column(db.Text, nullable=False)
            Product = db.relationship('Product', primaryjoin='Photo.ProductId ==
Product.ProductId', backref='photos')

            def file_path(self):
                return f'static/photos/{self.ProductId}/{self.FileName}'

        class Product(db.Model):
            __tablename__ = 'Products'
            ProductId = db.Column(db.Integer, primary_key=True)
            Title = db.Column(db.Text, nullable=False, unique=True)
            CategoryId = db.Column(db.ForeignKey('Categories.CategoryId'), index=True)
            ManufacturerId = db.Column(db.ForeignKey('Manufacturers.ManufacturerId'),
nullable=False, index=True)
            StockQuantity = db.Column(db.Integer, nullable=False,
server_default=db.FetchedValue())
            Price = db.Column(db.Float, nullable=False)
            Category = db.relationship('Category', primaryjoin='Product.CategoryId ==
Category.CategoryId', backref='products')
            Manufacturer = db.relationship('Manufacturer', primaryjoin='Product.ManufacturerId
== Manufacturer.ManufacturerId',  backref='products')
        class ProductsAttribute(db.Model):
            __tablename__ = 'ProductsAttributes'

            ProductAttributeId = db.Column(db.Integer, primary_key=True)
            ProductId = db.Column(db.ForeignKey('Products.ProductId'), nullable=False,
index=True)
            AttributeId = db.Column(db.ForeignKey('Attributes.AttributeId'), nullable=False,
index=True)
            AttributeValue = db.Column(db.Float, nullable=False)
            Attribute = db.relationship('Attribute', primaryjoin='ProductsAttribute.AttributeId ==
Attribute.AttributeId', backref='products_attributes')
            Product = db.relationship('Product', primaryjoin='ProductsAttribute.ProductId ==
Product.ProductId',
                                      backref='products_attributes')
        class ProductsInOrder(db.Model):
            __tablename__ = 'ProductsInOrders'
            ProductOrderId = db.Column(db.Integer, primary_key=True)
            ProductId = db.Column(db.ForeignKey('Products.ProductId'), index=True)
```

```
    OrderId = db.Column(db.ForeignKey('Orders.OrderId'), index=True)
    Order = db.relationship('Order', primaryjoin='ProductsInOrder.OrderId ==
Order.OrderId',
                            backref='products_in_orders')
    Product = db.relationship('Product', primaryjoin='ProductsInOrder.ProductId ==
Product.ProductId',
                              backref='products_in_orders')

 Define a User model
class User(db.Model, UserMixin):
    __tablename__ = 'Users'

    id = db.Column('UserId', db.Integer, primary_key=True)
    UserName = db.Column(db.Text, nullable=False)
    Email = db.Column(db.Text, nullable=False)
    Password = db.Column(db.Text, nullable=False)
    Access = db.Column(db.Integer, nullable=False)

    def __repr__(self):
        return f'User {self.id} {self.UserName} {self.Email}'

    def is_admin(self):
        return self.Access >= AccessLevelForAdmin

from flask import Blueprint, render_template
from project.models import Product
```

Б.14 Метод створення каталогу товарів та генерація переліку товарів

```
kat = Blueprint("kat", __name__)

@kat.route('/katalog')
def katalog():
    items = Product.query.order_by(Product.Price).all()
    return render_template('pricing.html', data=items)
```

Додаток В

Роздатковий матеріал

Тема: Розробка програмної системи для автоматизації торгівлі комп'ютерної техніки з можливістю конфігурування апаратної частини

Спецчастина: Розробка програмних модулів, інтерфейсу користувача та бази даних засобами Python, JavaScript, HTML, CSS

Автор: ст. гр. КН-18 Усатий І.О.
Керівник: ст. викл. каф. ПМІ Павловський Є.В.

Рисунок В.1 – Титульний аркуш

Об’єкт та предмет дослідження

Об’єкт дослідження - принцип розробки веб орієнтованих додатків для автоматизації торгівлі комп’ютерної техніки.

Предмет дослідження - мови програмування та сучасні підходи до реалізації розробки автоматизованої системи торгівлі комп’ютерної техніки.

Рисунок В.2 – Об’єкт та предмет дослідження

Мета роботи

Мета бакалаврської роботи – аналіз предметної області ті її аналогів, визначення методів і алгоритмів реалізації програмного засобу, проектування моделей задля візуалізації зовнішніх і внутрішніх процесів розробленого додатку

Рисунок В.3 – Мета роботи

Задачі роботи

- вивчити об’єкт дослідження;
- провести аналіз та виявити проблеми автоматизації торгівлі комп’ютерною технікою;
- сформувати вимоги до програмної системи та описати процес її проектування;
- розробити структуру бази даних, програмний засіб для автоматизації торгівлі технікою, який допоможе користувачу самостійно підібрати повністю сумісні компоненти ПК.

Рисунок В.4 – Задачі роботи

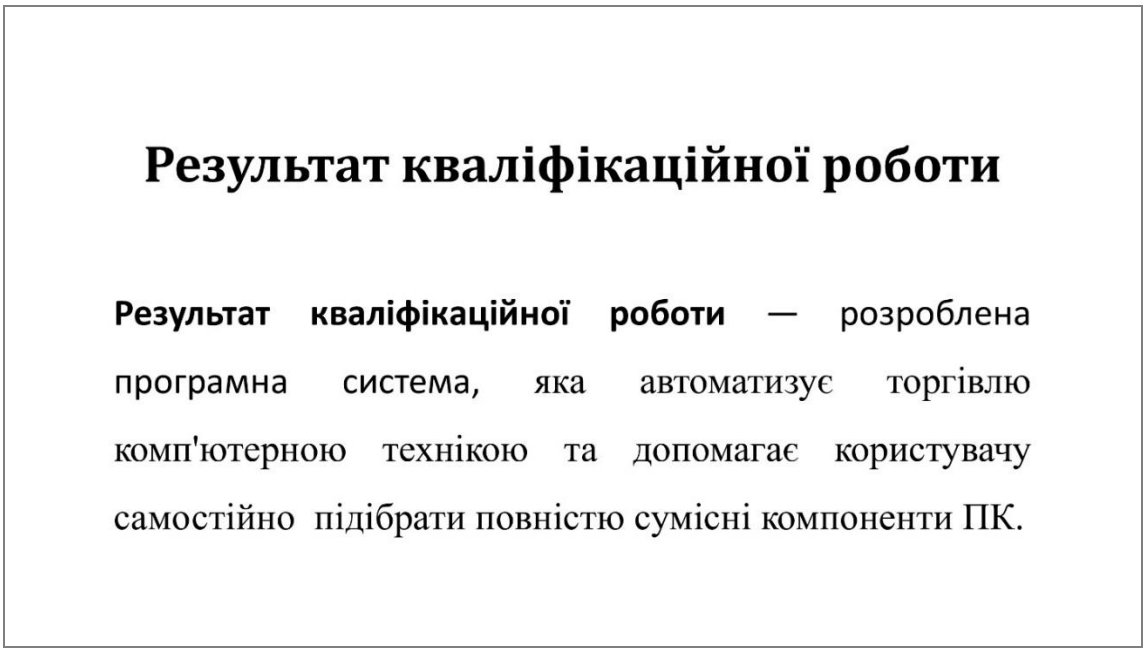


Рисунок В.5 – Результат кваліфікаційної роботи

Рисунок В.6 – Аналіз готових рішень серед CMS

Аналіз готових рішень для створення веб-системи

1) Готові **інтернет-платформи**

prom ROZETKA AliExpress

Недоліки:

- витрати на рекламу в мережі платформи
- обмежений дизайн
- обмежена кількість товарів
- не кожна платформа дозволяє використовувати власні домени

Рисунок В.7 – Аналіз готових рішень серед інтернет-платформ

Діаграма варіантів використання

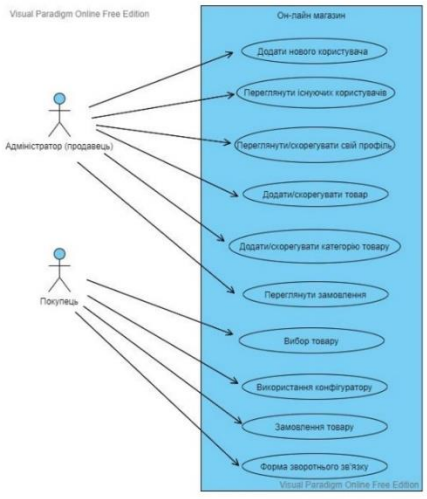


Рисунок В.8 – Діаграма варіантів використання

Рисунок В.9 – Діаграма компонентів

Рисунок В.10 – Діаграми послідовностей для адміністратора та покупця

Рисунок В.11 – Схема бази даних

Рисунок В.12 – Структура додатку

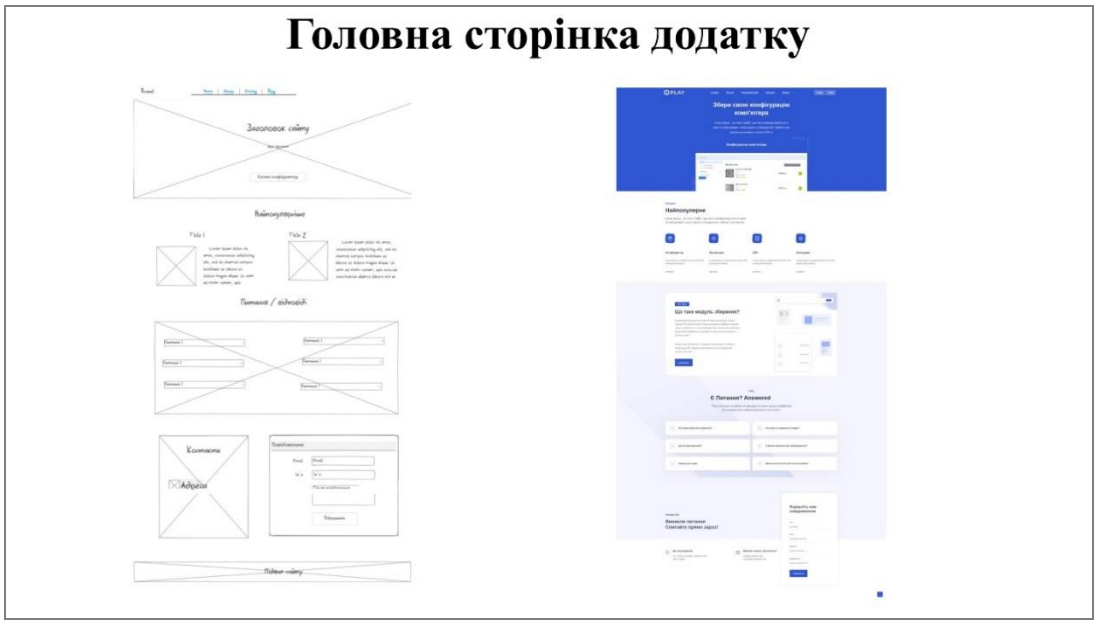


Рисунок В.13 – Головна сторінка додатку

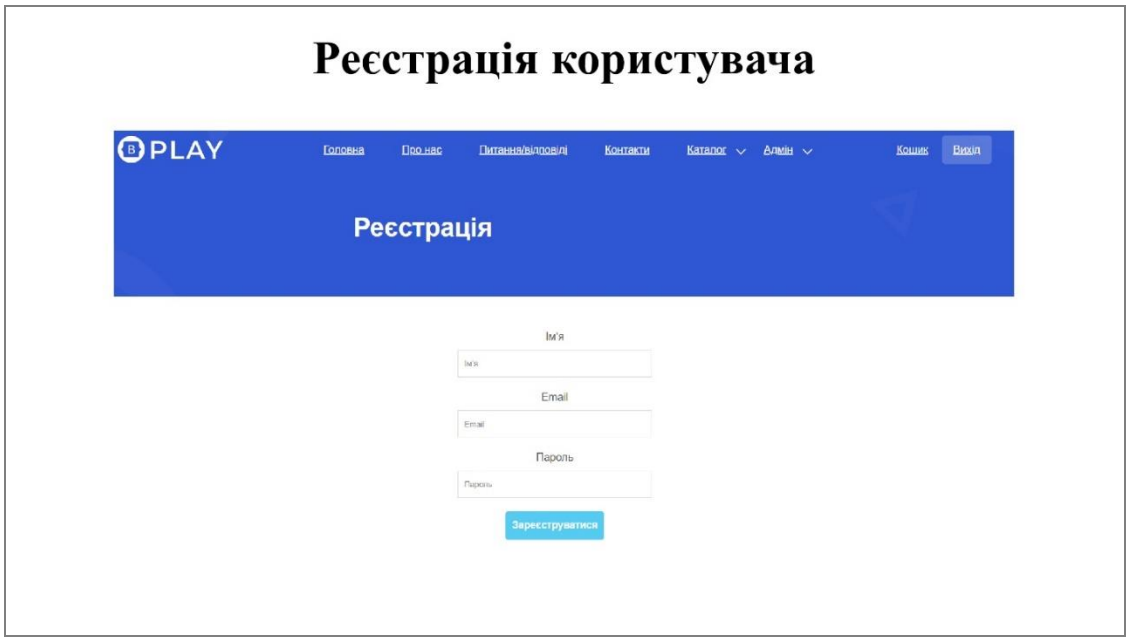


Рисунок В.14 – Реєстрація користувача

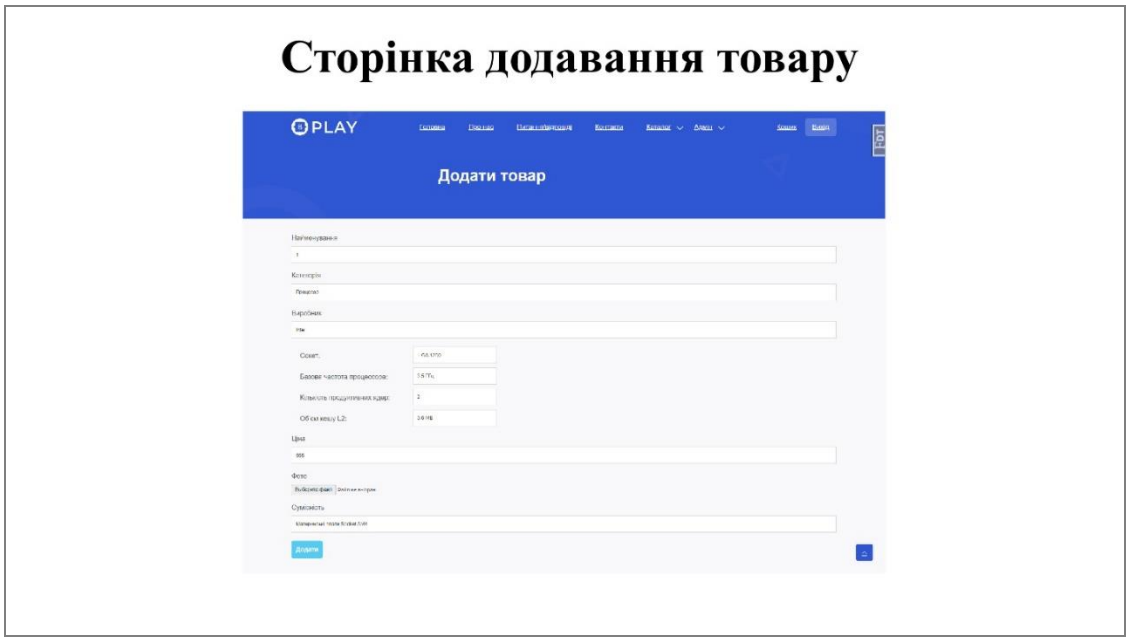


Рисунок В.15 – Сторінка додавання товару

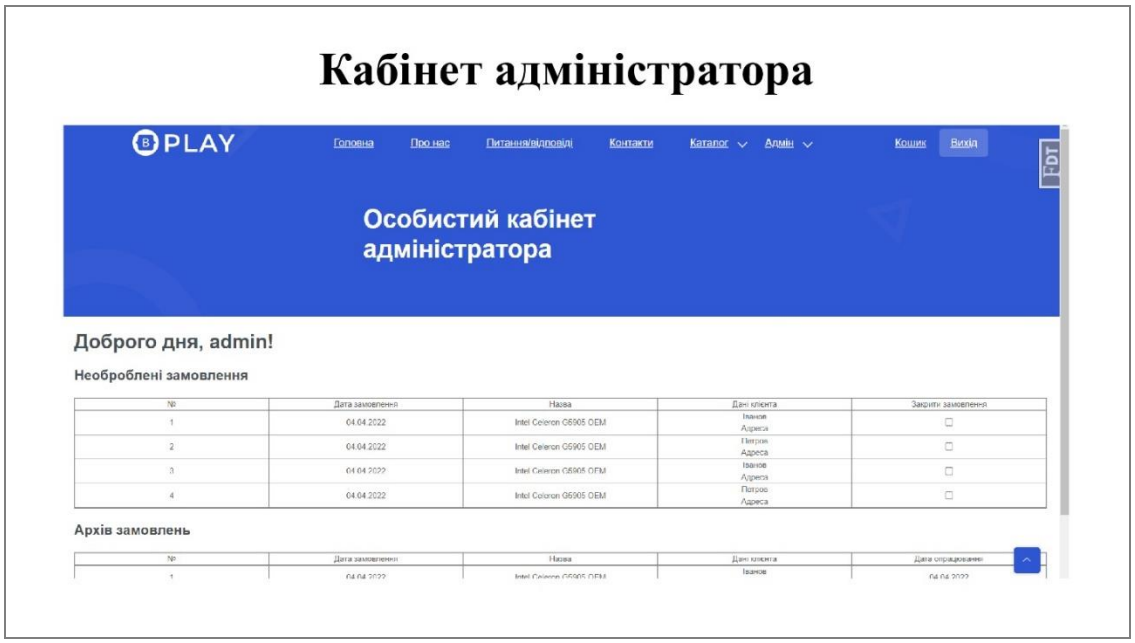


Рисунок В.16 – Кабінет адміністратора

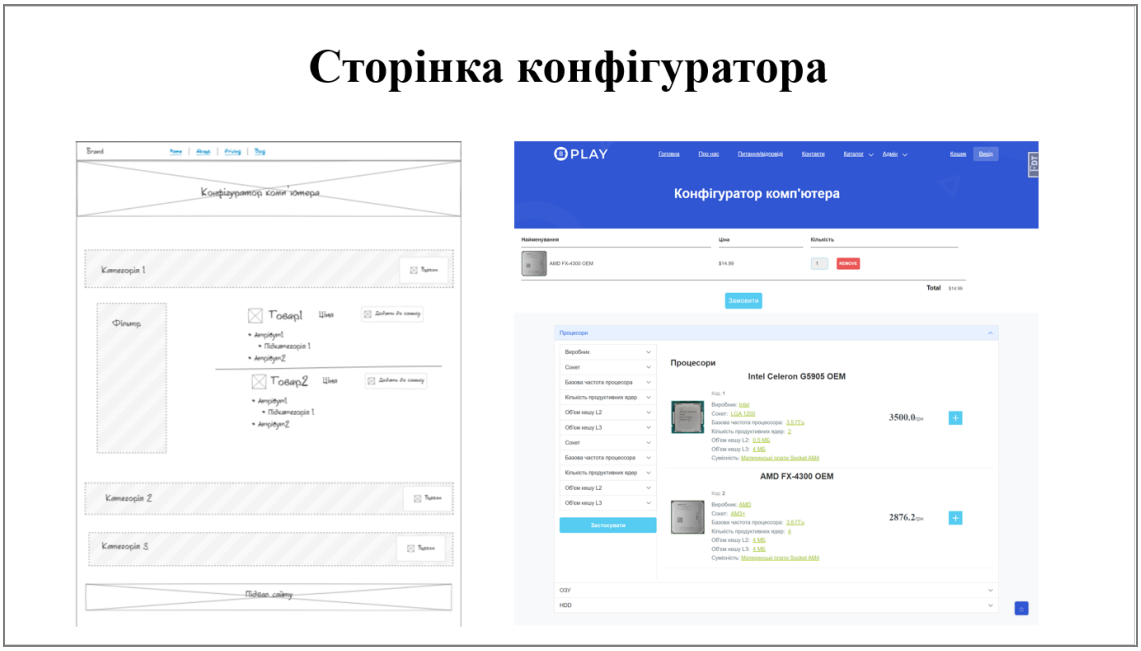


Рисунок В.17 – Сторінка конфігуратора

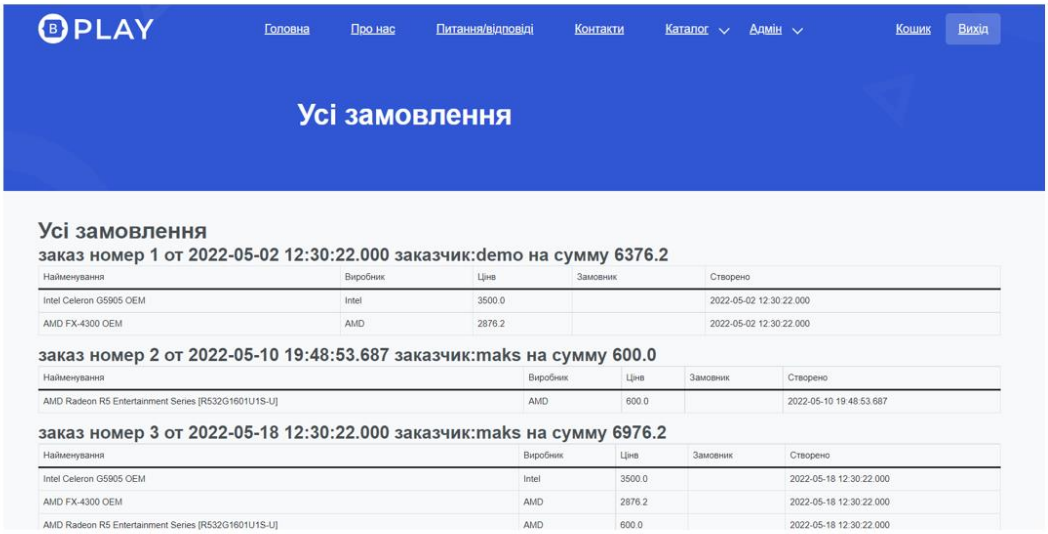


Рисунок В.18 – Сторінка товарів в обробці

Отримані результати

Для реалізації програмного додатку були в повній мірі виконані поставлені задачі:

- проведений аналіз існуючих програмних систем, які вирішують питання в даній предметній області;
- сформована постановка задачі на роботу;
- здійснене проєктування процесів програмного продукту у вигляді діаграм, моделі баз даних і макетів користувальницького інтерфейсу;
- виконана програмна реалізація програмного продукту засобами Python, JavaScript, HTML, CSS;
- виконане тестування програмного додатку та усунені всі недоліки, що були виявлені під час тестування

Рисунок В.19 – Отримані результати