ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка web-застосунку для підбору комплектуючих персонального комп'ютера»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Юлія САВЧЕНКО
(підпис)

Виконала: здобувачка вищої освіти групи ПД-41

Юлія САВЧЕНКО

Керівник: Оксана ЗОЛОТУХІНА
к.т.н.,доцент

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Савченко Юлії Вячеславівні

1. Тема кваліфікаційної роботи: «Розробка web-застосунку для підбору комплектуючих персонального комп'ютера»
керівник кваліфікаційної роботи к.т.н., доцент Оксана ЗОЛОТУХІНА,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: опис сучасних підходів до створення web-застосунків для підбору комп'ютерних комплектуючих, сайти існуючих інтернет-магазинів і систем перевірки сумісності, технічна документація фреймворку Laravel, специфікації API для роботи з базами даних і імпорту товарів, а також вимоги замовника щодо функціоналу web-магазину.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області

2. Проєктування та моделювання системи для підбору комплектуючих ПК
3. Проєктування web-застосунку для підбору сумісних комплектуючих для ПК
4. Тестування розробленого web-застосунку.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до web-застосунку.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання користувачем.
5. Діаграма варіантів використання гостем.
6. Діаграма варіантів використання адміном.
7. Діаграма класів систем.
8. Діаграма процесу підбору комплектуючих.
9. Екранні форми.
10. Екранні форми.
11. Апробація результатів дослідження

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2025	
3	Проєктування та моделювання системи для підбору комплектуючих ПК	14.03 - 31.03.2025	
4	Проєктування web-застосунку для підбору сумісних комплектуючих для ПК	01.04 - 19.04.2025	
5	Тестування web-додатку	20.04 - 28.04.2025	
7	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
8	Розробка демонстраційних матеріалів	12.05-18.05.2025	
9	Попередній захист роботи	19.05-30.05.2025	

Здобувачка вищої освіти _____ Юлія САВЧЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Оксана ЗОЛОТУХІНА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 63 стор., 1 табл., 44 рис., 24 джерел.

Мета роботи — спрощення процесу підбору комплектуючих персонального комп'ютера.

Об'єкт дослідження — процес підбору комплектуючих персонального комп'ютера з урахуванням вимог користувача.

Предмет дослідження — методи та програмні засоби web-розробки для підбору комплектуючих персонального комп'ютера з урахуванням вимог користувача.

Короткий зміст роботи: У роботі проаналізовано предметну галузь та методи розробки web-застосунку для підбору комплектуючих персонального комп'ютера. Проведено огляд існуючих платформ та інструментів, що підтримують автоматизований вибір та перевірку сумісності комп'ютерних компонентів. Розроблено вебсайт із функціоналом авторизації, каталогу товарів з фільтрами, системою пошуку, особистим кабінетом користувача, кошиком та механізмами автоматичної перевірки сумісності вибраних комплектуючих. Програмно реалізовані основні функціональні можливості: робота з базою даних через Eloquent ORM, застосування middleware, система валідації, захист від типових web-загроз (XSS, CSRF, SQL-ін'єкції) та інтеграція інтерфейсу з використанням Blade і Vue.js, що забезпечує швидкий і зручний процес підбору конфігурацій ПК.

В роботі використано фреймворк Laravel для бекенду та сучасні технології фронтенду. Проведено тестування окремих модулів застосунку.

Сферою використання застосунку є організація автоматизованого підбору сумісних комплектуючих персонального комп'ютера для користувачів, що прагнуть зібрати оптимальну конфігурацію ПК.

КЛЮЧОВІ СЛОВА: WEB-ЗАСТОСУНОК, ПІДБІР КОМПЛЕКТУЮЧИХ, СУМІСНІСТЬ КОМПОНЕНТІВ, LARAVEL, MIDDLEWARE, ВАЛІДАЦІЯ, АВТОМАТИЗАЦІЯ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – Програмне забезпечення

WEB –World Wide Web

IDE – Integrated Development Environment

API – Application Programming Interface

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

SQL – Structured Query Language

ORM – Object-Relational Mapping

JSON – JavaScript Object Notation

PHP – Hypertext Preprocessor

MVC – Model-View-Controller

CSRF – Cross-Site Request Forgery

XSS – Cross-Site Scripting

БЖ – Блок живлення

CRUD – Create, Read, Update, Delete

UI – User Interface

Laravel – PHP Framework для web-розробки

Vue.js – Прогресивний JavaScript-фреймворк

Blade – Шаблонізатор Laravel

NPM – Node Package Manage

ВСТУП

Актуальність: у сучасному світі все більше користувачів прагнуть мати комп’ютер, який повністю відповідає їхнім індивідуальним потребам. Через це зростає попит на персоналізовані комп’ютерні збірки, де кожен компонент обирається з урахуванням бажаної продуктивності, бюджету та сумісності. Проте ручний підбір комплектуючих вимагає від користувача глибоких технічних знань, а помилки на цьому етапі можуть призвести до несумісності компонентів, збоїв у роботі системи або навіть пошкодження обладнання.

У таких умовах особливо важливо мати інструмент, який допомагає автоматизувати процес підбору сумісних деталей. Розробка web-застосунку, який здійснює аналіз сумісності на основі бази даних характеристик компонентів, дозволить значно спростити цей процес для звичайного користувача. Це не лише зменшить кількість помилок при виборі, а й підвищить ефективність та стабільність зібраних систем. Такий підхід сприятиме розвитку ринку комп’ютерних комплектуючих і водночас зробить процес складання ПК більш доступним, зручним і безпечним навіть для новачків.

Об’єктом дослідження: процес підбору комплектуючих персонального комп’ютера з урахуванням вимог користувача.

Предметом дослідження: розробка логіки сумісності компонентів ПК та її інтеграція у функціонал web-застосунку.

Метою роботи є спрощення процесу підбору комплектуючих персонального комп’ютера..

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1. Провести порівняльний аналіз існуючих онлайн-сервісів для підбору ПК-компонентів з метою виявлення переваг і недоліків.
2. Розробити структуру бази даних для збереження інформації про компоненти ПК

3. Реалізувати механізм перевірки сумісності компонентів на основі введених користувачем даних.

4. Забезпечити імпорт компонентів та готових збірок у систему.

5. Провести тестування розробленого web-застосунку.

Методи дослідження: включають застосування підходів моделювання та проєктування web-застосунків, методів об’єктно-орієнтованого програмування, а також технік розробки клієнт-серверної архітектури.

Практична значущість результатів полягає у створенні web-застосунку, який спрощує підбір сумісних комплектуючих, знижує кількість помилок і економить час користувачів. Застосунок може використовуватися в онлайн-магазинах та слугувати базою для подальшого розвитку автоматизованих систем підбору.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Web-застосунок для конфігурації ПК

Web-застосунки для конфігурації персонального комп’ютера — це сучасні онлайн-платформи, які допомагають користувачам легко і зручно зібрати власний ПК, підбираючи між собою сумісні комплектуючі. Вони стають справжніми помічниками для широкого кола людей — від тих, хто тільки починає свій шлях у світі IT, до досвідчених користувачів, які прагнуть отримати максимально продуктивну систему, адаптовану під конкретні завдання: навчання, офісна робота, ігри або навіть ресурсоємні професійні задачі, такі як відеомонтаж чи 3D-моделювання.

Головна перевага таких платформ — це значне спрощення вибору комплектуючих. Завдяки автоматизованій перевірці сумісності між процесором, материнською платою, оперативною пам’яттю, відеокартою, блоком живлення та іншими деталями, користувачі можуть бути впевнені, що їх вибір правильний, а ризик помилок зведений до мінімуму. Це особливо важливо для новачків, які ще не мають глибоких технічних знань, але хочуть отримати надійний та ефективний комп’ютер.

Зазвичай такі web-застосунки пропонують зручний поетапний вибір комплектуючих із широким каталогом, де можна знайти докладні технічні характеристики та застосувати різноманітні фільтри для швидкого пошуку. Платформи візуалізують конфігурацію, показують загальну вартість і навіть дають змогу оформити замовлення безпосередньо на сайті.

Відповідно до дослідження Huang T.E. та Jones M., проєктування масштабованих систем для перевірки сумісності компонентів ПК вимагає не лише врахування технічних характеристик окремих пристроїв, а й побудови гнучкої архітектури, здатної до динамічного оновлення баз знань[1]. Автори акцентують увагу на необхідності стандартизації формату даних та

впровадження адаптивних алгоритмів, які можуть ефективно обробляти велику кількість взаємозв’язків між комплектуючими. Такий підхід дозволяє підвищити точність перевірки сумісності, зменшити ризики конфліктів між компонентами та забезпечити масштабованість системи у випадку розширення асортименту.

Важливо пам’ятати, що ринок комп’ютерних комплектуючих постійно змінюється, тому такі платформи потребують регулярного оновлення даних, щоб забезпечувати актуальність інформації. Крім практичної користі, вони мають і великий освітній потенціал — допомагають краще розуміти, як працюють комп’ютерні компоненти, що особливо цінно для молоді та всіх, хто хоче розвиватися у сфері ІТ.

Отже, web-застосунки з функцією конфігурації ПК вже стали невід’ємною частиною сучасної електронної комерції у галузі комп’ютерної техніки. Вони роблять процес створення персоналізованого комп’ютера простим, безпечним і приємним. Щороку їх популярність зростає, а функціонал стає дедалі ширшим, що свідчить про великий потенціал і актуальність цього напрямку у світі ІТ.

1.2 Функціональні елементи web-застосунків для збирання ПК

Ключовими складовими web-конструкторів персональних комп’ютерів є функціональні модулі, які відповідальні за вибір, перевірку та візуалізацію комплектуючих. Саме вони створюють основу для зручного, швидкого і водночас надійного процесу підбору компонентів, що дозволяє кожному користувачу, незалежно від рівня технічної підготовки, зібрати свій ідеальний ПК.

Одним із найважливіших елементів такого конструктора є зручний інтерфейс. Він організовує процес поетапного вибору основних компонентів — починаючи від центрального процесора і материнської плати, через оперативну пам’ять, відеокарту, накопичувачі, блок живлення, корпус, і завершуючи системою охолодження. Кожен крок підкреслено логікою і підказками, що

допомагають не загубитися в різноманітті моделей і параметрів. Адже всі ці компоненти тісно взаємопов'язані, і саме правильний підбір забезпечує ефективну, стабільну та довговічну роботу комп'ютера.

Розглянемо детальніше ключові компоненти:

– процесор (CPU) : це "мозок" системи, який виконує всі основні обчислення. Від його тактової частоти, кількості ядер і потоків залежить продуктивність у повсякденних задачах, іграх, професійних програмах.

– Материнська плата: це фундамент, який фізично і логічно з'єднує всі частини ПК. Вона визначає сумісність із процесором, типом оперативної пам'яті, кількістю портів і слотів, а також підтримує певні технології.

– Оперативна пам'ять (RAM): забезпечує швидкий доступ до даних, необхідних для роботи програм, що значно впливає на швидкодію системи.

– Відеокарта (GPU): відповідає за виведення графіки на екран і є особливо важливою у задачах, що потребують високої продуктивності візуалізації — іграх, 3D-моделюванні, відеомонтажі.

– Накопичувачі (SSD, HDD): це сховище даних, де зберігаються операційна система, програми, файли. Вибір між твердотілим накопичувачем і жорстким диском визначає швидкість роботи системи і обсяг зберігання.

– Блок живлення: відповідає за стабільне і надійне електроживлення усіх компонентів, що є запорукою стабільної роботи системи.

– Корпус і система охолодження: не лише забезпечують ергономіку і естетику, але й підтримують оптимальний температурний режим, що подовжує термін служби комплектуючих.

Важливою складовою web-конструктора є система автоматичної перевірки сумісності. Вона аналізує обрані користувачем компоненти і миттєво повідомляє про можливі конфлікти або несумісності. Така функція є справжнім рятувальником — вона знижує ризик помилок і допомагає зробити вибір безпечним і виваженим, навіть якщо користувач не є експертом у технічних деталях.

Суттєвим напрямком розвитку є використання інтелектуальних алгоритмів автоматичного вибору. Gonzalez J.E., Schmidt A. та Lee K. у своєму дослідженні підкреслюють, що застосування таких алгоритмів дозволяє ефективно обробляти великі обсяги інформації про технічні характеристики, сумісність та продуктивність комплектуючих[2]. Це значно підвищує точність і швидкість процесу підбору та знижує ризик помилок, що виникають при ручному виборі.

Ще один ключовий модуль — компонент каталогу, який містить велику базу даних з актуальними технічними характеристиками комплектуючих. Завдяки зручним фільтрам (за сокетом, обсягом пам’яті, частотою, типом інтерфейсів та іншими параметрами) користувачі легко знаходять потрібні деталі, адаптуючи конфігурацію під свої потреби.

Не менш важливою є візуальна складова — інтерактивний модуль, який показує обрану конфігурацію у вигляді зрозумілого списку або навіть візуальної збірки. Він відображає всі вибрані компоненти, підраховує загальну вартість та дає змогу швидко вносити зміни. Такий підхід робить процес збирання комп’ютера наочним і захопливим, що особливо цінують користувачі будь-якого рівня.

Всі ці елементи разом формують єдину інтерактивну систему, яка поєднує в собі гнучкість, точність і зручність, роблячи збірку персонального комп’ютера простою, зрозумілою і приємною справою. Web-конструктори ПК — це не просто інструмент, це справжній помічник, який відкриває двері у світ персоналізованих технологій і дає змогу кожному реалізувати свої найсміливіші технічні ідеї.

1.3 Цільова аудиторія

Цільова аудиторія web-застосунків для збирання персонального комп’ютера є досить широкою та різноманітною, охоплюючи користувачів з різним рівнем технічної підготовки, знань і практичного досвіду. Такий

застосунок стає не лише зручним інструментом, а й потужним помічником у виборі компонентів як для побутових потреб, так і для професійної діяльності.

У першу чергу, це новачки — користувачі, які тільки починають своє знайомство зі світом комп'ютерних комплектуючих. Вони можуть прагнути зібрати свій перший ПК для навчання, домашнього використання або комп'ютерних ігор. Для цієї категорії особливо важливо, щоб інтерфейс був зрозумілим, кроки — логічними, а процес підбору — максимально автоматизованим. Наявність підказок, системи автоматичної перевірки сумісності та попередження про можливі конфлікти між компонентами допомагає уникнути найпоширеніших помилок та підвищує впевненість у правильності вибору. Такий підхід також сприяє навчанню користувачів — вони поступово починають розуміти, які характеристики є ключовими та на що варто звертати увагу.

Іншою важливою групою є ентузіасти та досвідчені користувачі, які вже мають певний рівень технічних знань і зазвичай самостійно займаються збиранням, апгрейдом або оптимізацією ПК. Вони зацікавлені в гнучкому налаштуванні, наявності детальних технічних характеристик, можливості глибокого порівняння компонентів, а також у швидкій візуалізації створюваної конфігурації. Для них важливо, щоб система дозволяла здійснювати вибір у будь-якому порядку — незалежно від того, з процесора вони починають чи, наприклад, з корпусу. Це забезпечує більше свободи і відповідає їхньому стилю роботи, коли деякі компоненти вже є в наявності, а інші — потрібно лише підібрати.

Окрему, але не менш значущу нішу становлять спеціалісти IT-сфери — зокрема, працівники сервісних центрів, менеджери з продажу, консультанти комп'ютерної техніки та збирачі на замовлення. Для них web-застосунок є професійним інструментом, що дозволяє швидко формувати індивідуальні конфігурації для клієнтів, демонструвати варіанти підбору у режимі реального часу, а також оперативно оцінювати сумісність і вартість системи. Це суттєво економить час і підвищує якість обслуговування клієнтів.

Отже, подібні платформи орієнтовані не лише на побутового користувача, а й на фахівців, що робить їх універсальними та затребуваними у різних сегментах ринку. У поєднанні з продуманим дизайном, гнучкістю взаємодії, інтерактивністю та актуальністю представлених даних, web-застосунок для збирання ПК стає важливим інструментом, що спрощує складні процеси та відкриває широкі можливості для кожного користувача — від новачка до професіонала.

1.4 Дослідження існуючих аналогів

Сьогодні на ринку представлено кілька популярних web-застосунків для конфігурації персональних комп'ютерів, які допомагають користувачам легко підібрати сумісні комплектуючі. Для порівняння розглянемо такі сервіси, як Can, Compx, Chipchip та Click, кожен з яких має свої особливості та переваги.

Інтернет-магазин Can зроблений з акцентом на простоту та доступність, тому його інтерфейс сучасний і зрозумілий навіть для новачків[3]. Для пошуку товарів доступні базові фільтри за основними характеристиками, що дозволяє швидко знаходити потрібні комплектуючі.

Основні функції Can включають:

– Простий інтерфейс: логічне розташування елементів, зрозуміле навіть новачкам.

– Базова система фільтрації: дозволяє відсортувати товари за основними характеристиками — типом, брендом, обсягом пам'яті тощо.

– Класичний пошук: можливість знаходити товари за ключовими словами.

– Легка доступність: сайт працює швидко та коректно на більшості пристроїв.

Переваги:

- Сучасний і простий інтерфейс, зручний для новачків
- Базові фільтри для швидкого пошуку товарів за характеристиками

– Доступність і легкість у користуванні

Однак платформа не має деяких важливих функцій, які сьогодні очікують користувачі сучасних web-конструкторів ПК.

Недоліки:

– Відсутність автоматичної перевірки залишкової потужності блока живлення, що може призводити до помилок у підборі комплектуючих

– Відсутність динамічного оновлення списку сумісних компонентів під час вибору, що зменшує гнучкість конфігурації

– Відсутність темного режиму, що ускладнює роботу у вечірній час

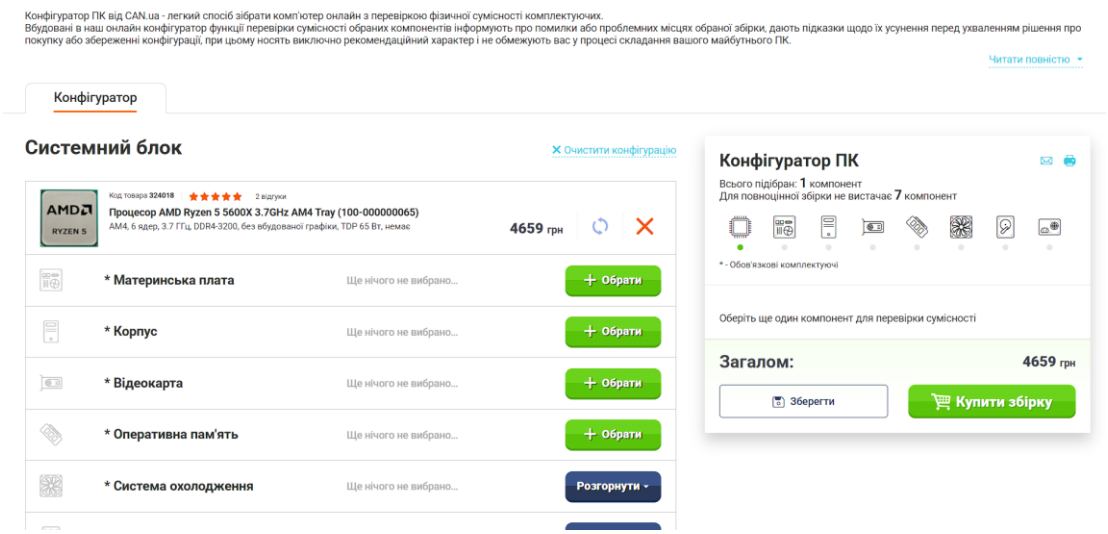


Рис.1.1 Приклад конфігуратору Can

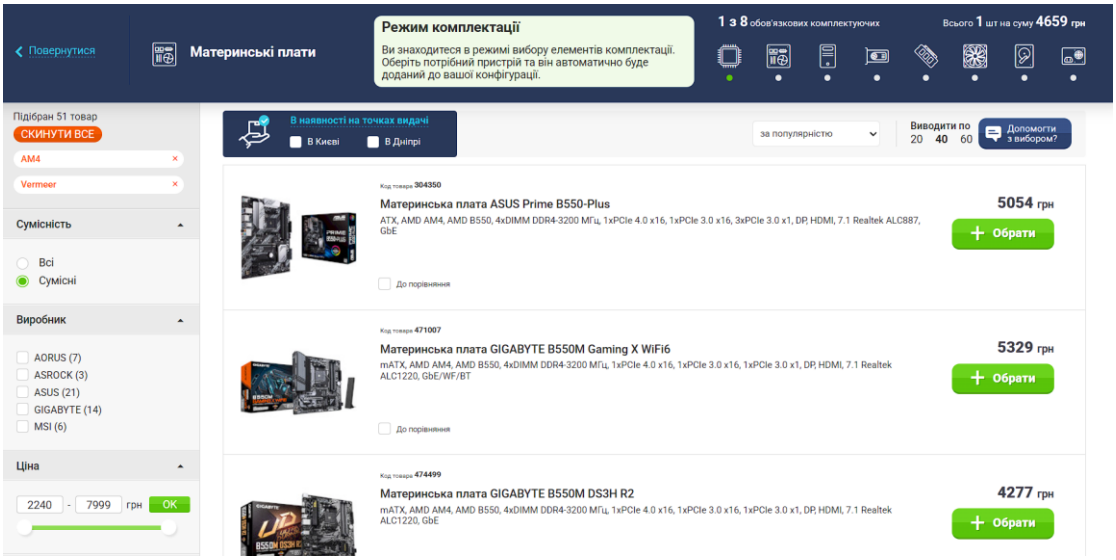


Рис.1.2 Приклад вибору компонету Can

Chipchip — це сучасний інтернет-магазин, який спеціалізується на продажу комп’ютерних комплектуючих, аксесуарів та електроніки[4]. Chipchip надає базовий, але водночас функціональний набір можливостей, які забезпечують зручний користувацький досвід для повсякденного підбору комп’ютерних комплектуючих.

Основні функції ChipChip включають:

– Каталог з фільтрами: зручно структурований каталог товарів із можливістю фільтрації за ключовими характеристиками (тип, бренд, обсяг пам’яті, інтерфейси тощо). Це дозволяє швидко орієнтуватися серед великого асортименту.

– Пошук за назвою та параметрами: система пошуку дозволяє легко знаходити конкретні моделі або товари з бажаними параметрами.

– Збереження історії переглядів: особливо корисна функція, яка дозволяє швидко повертатися до раніше переглянутих позицій, не втрачаючи цінні варіанти в процесі порівняння.

– Кошик та оформлення замовлення: стандартний набір інструментів для додавання товарів, обліку вартості та формування замовлення з можливістю обрати спосіб доставки й оплати.

– Особистий кабінет: користувачі можуть переглядати свої замовлення, керувати обраними товарами та редагувати особисту інформацію.

– Оцінки та відгуки користувачів: можливість ознайомитись з реальними враженнями інших покупців щодо якості товарів або обслуговування.

Переваги:

– Широкий вибір товарів: у каталозі представлено безліч категорій — від процесорів до периферії, що дозволяє користувачам підібрати всі необхідні компоненти в одному місці.

– Збереження історії переглядів: зручна функція, яка дає змогу повертатися до раніше переглянутих позицій без необхідності шукати їх повторно, що особливо корисно при тривалому виборі.

– Збалансований зрозумілий інтерфейс: навіть ті, хто не має досвіду користування подібними платформами, швидко адаптуються завдяки простому розташуванню елементів і логічній структурі.

Недоліки:

– Відсутність перевірки енергоспоживання: система не аналізує загальне споживання енергії компонентами, тому користувачам доводиться самостійно розраховувати, чи вистачить потужності блока живлення.

– Відсутність функції порівняння: покупці не мають можливості зручно зіставити технічні характеристики кількох товарів одночасно без зовнішніх інструментів чи ручного перегляду.

– Немає темного режиму: тривала робота з сайтом у вечірній час або в умовах слабкого освітлення може викликати втому очей, що впливає на комфорт користування.

– Відсутність інтегрованого механізму підбору сумісних компонентів: після вибору певного елементу (наприклад, процесора) користувач змушений переходити на іншу сторінку для вибору материнської плати або пам’яті

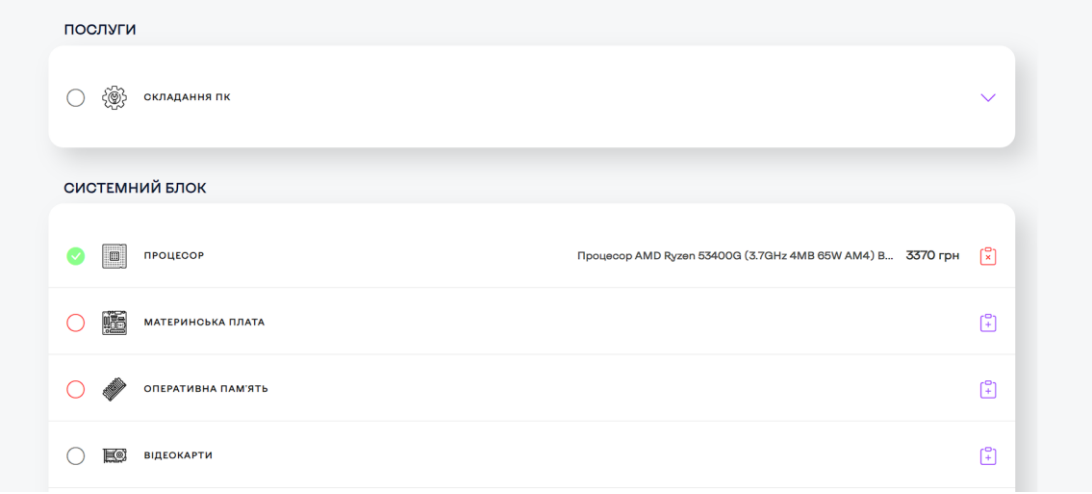


Рис.1.3 Приклад конфігуратору ChipChip

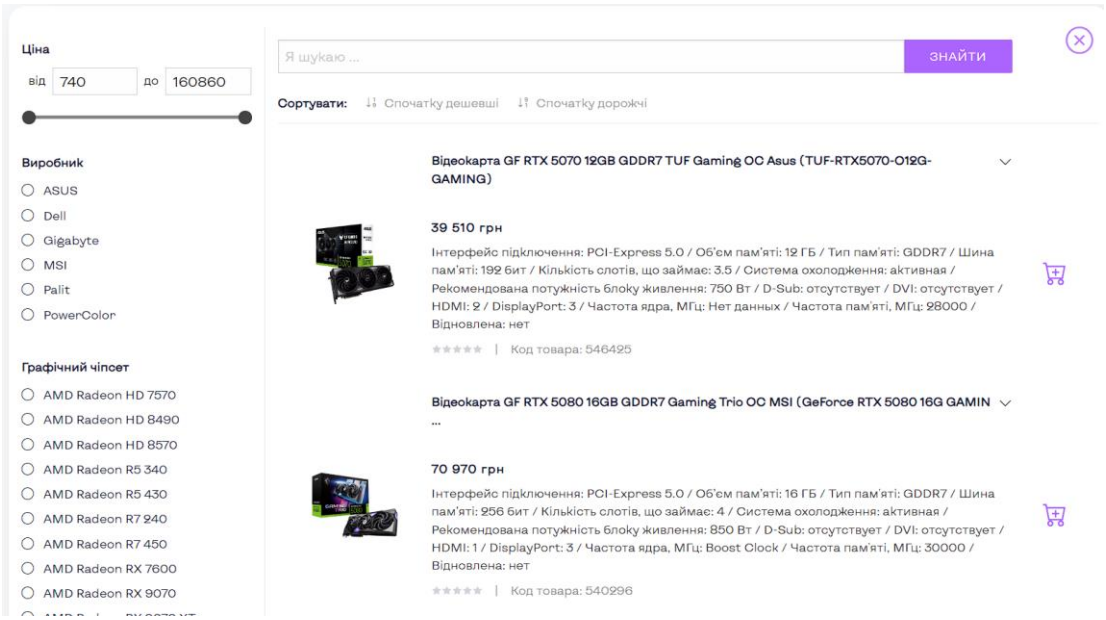


Рис. 1.4 Приклад вибору компоненту ChipChip

Платформа Click є популярним інтернет-магазином комплектуючих, орієнтованим на широку аудиторію користувачів[5]. Вона приваблює завдяки зручному інтерфейсу, великому асортименту та функціям, які забезпечують комфорт під час пошуку й перегляду товарів. Однак, незважаючи на всі переваги, сайт не має спеціалізованого інструментарію для підбору сумісних компонентів, що є критичним для web-конструкторів ПК.

Основні функції платформи Click:

- розширений каталог комплектуючих: велика кількість товарів у різних категоріях, включно з актуальними цінами, наявністю та зображеннями.
- потужні фільтри: доступна фільтрація за параметрами, такими як об’єм пам’яті, частота, інтерфейси, бренд, ціна, тип охолодження тощо.
- історія переглядів: користувачі можуть легко повернутись до раніше переглянутих товарів.
- обліковий запис користувача: функція обраного, історія замовлень, відстеження статусу доставки.
- пошук за ключовими словами: швидкий пошук потрібного товару за найменуванням або характеристиками.

Переваги платформи:

- зручний та сучасний інтерфейс: зрозумілий навіть новачкам.
- великий вибір товарів: охоплено широкий спектр комплектуючих і аксесуарів.
- розширені фільтри: користувачі можуть точно підібрати необхідний продукт.
- історія переглядів і обране — покращують користувацький досвід при повторному відвідуванні.

Недоліки платформи:

- відсутність конструктора ПК — немає інтерактивного підбору сумісних компонентів у форматі поетапної збірки.
- немає перевірки залишкової потужності блоку живлення — що може призвести до вибору компонента з недостатніми параметрами.
- відсутня система порівняння компонентів — не можна порівняти декілька товарів за характеристиками в зручному форматі.
- відсутність інтегрованого механізму підбору сумісних компонентів: після вибору певного елементу (наприклад, процесора) користувач змушений переходити на іншу сторінку для вибору материнської плати або пам’яті
- відсутність темного режиму — може бути незручно при користуванні у темний час доби.

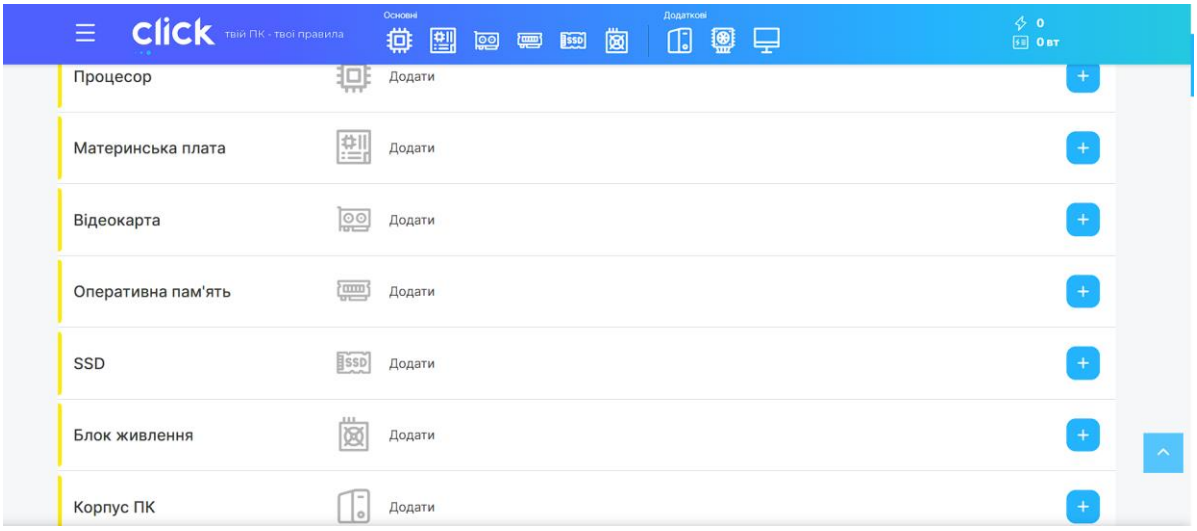


Рис.1.5 Приклад кофігуратору Click

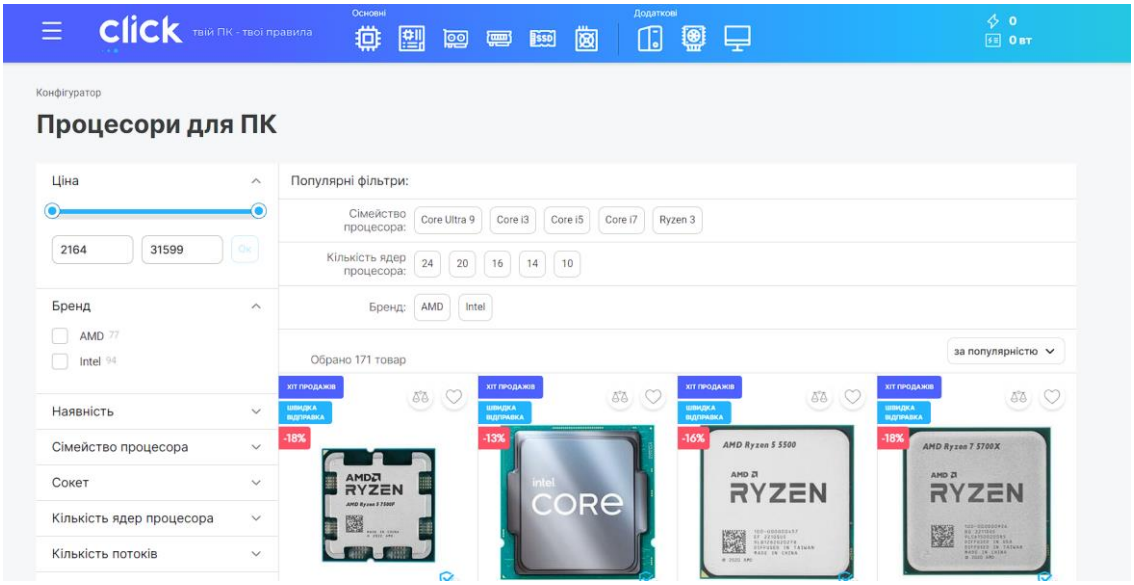


Рис.1.6 Приклад вибору компоненту Click

Платформа Compx демонструє сучасний підхід до підбору комплектуючих, поєднуючи ефективність автоматизації та зручність інтерфейсу[6]. Завдяки підтримці основних перевірок сумісності та динамічного оновлення списків компонентів, вона значно спрощує процес конфігурації ПК для користувачів з різним рівнем досвіду.

Основні функції:

- автоматичне оновлення сумісних компонентів: після вибору одного з елементів інтерфейс оновлюється без переходу на іншу сторінку: відкривається панель з доступними варіантами, у якій автоматично відсіюються несумісні компоненти.
- перевірка залишкової потужності блока живлення: система аналізує енергоспоживання вибраних компонентів і пропонує відповідні блоки живлення.
- розширена система фільтрів: дозволяє швидко звузити вибір компонентів за ключовими характеристиками.
- історія переглядів: дає змогу повернутися до раніше переглянутих товарів без потреби здійснювати повторний пошук.

Переваги:

- реалізована система перевірки живлення: підвищує надійність вибору.
- зручна навігація та пошук за допомогою фільтрів.
- збереження історії переглядів: полегшує роботу з великою кількістю товарів.

Недоліки:

- відсутність візуального конструктора ПК: користувач не має змоги бачити наочно, як виглядає зібрана конфігурація.
- немає функції порівняння товарів: ускладнює вибір між кількома подібними компонентами.
- відсутній темний режим: інтерфейс не оптимізований для роботи у темний час доби, що знижує комфорт використання.

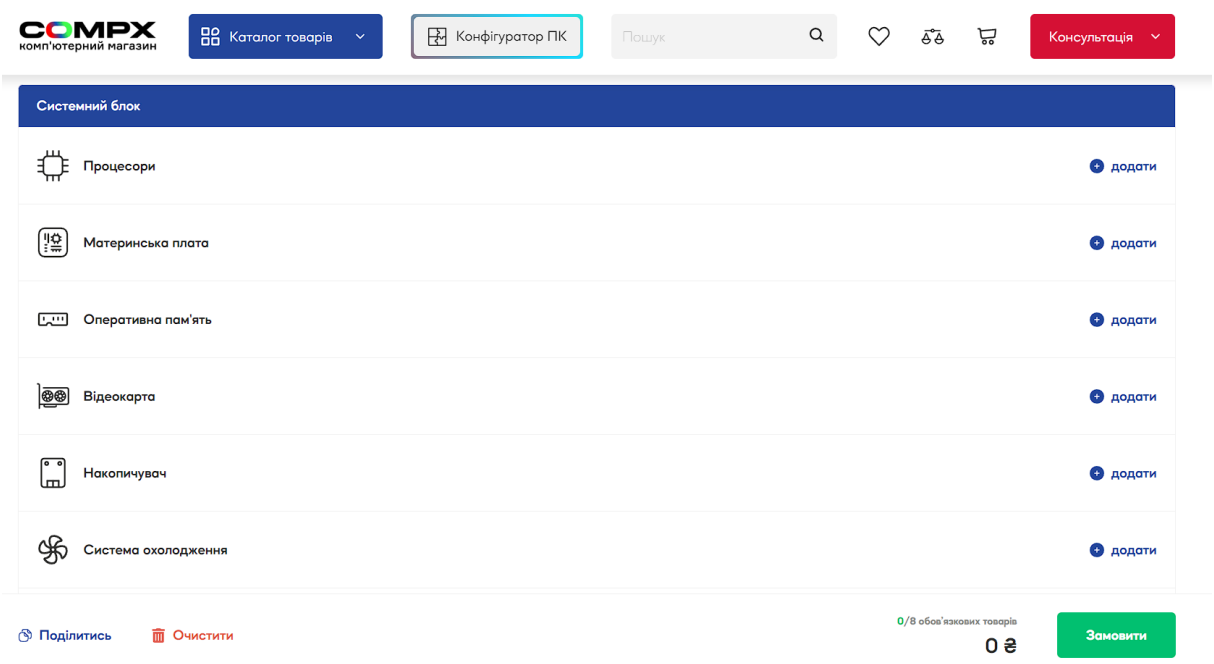


Рис.1.7 Приклад конфігуратора Compx

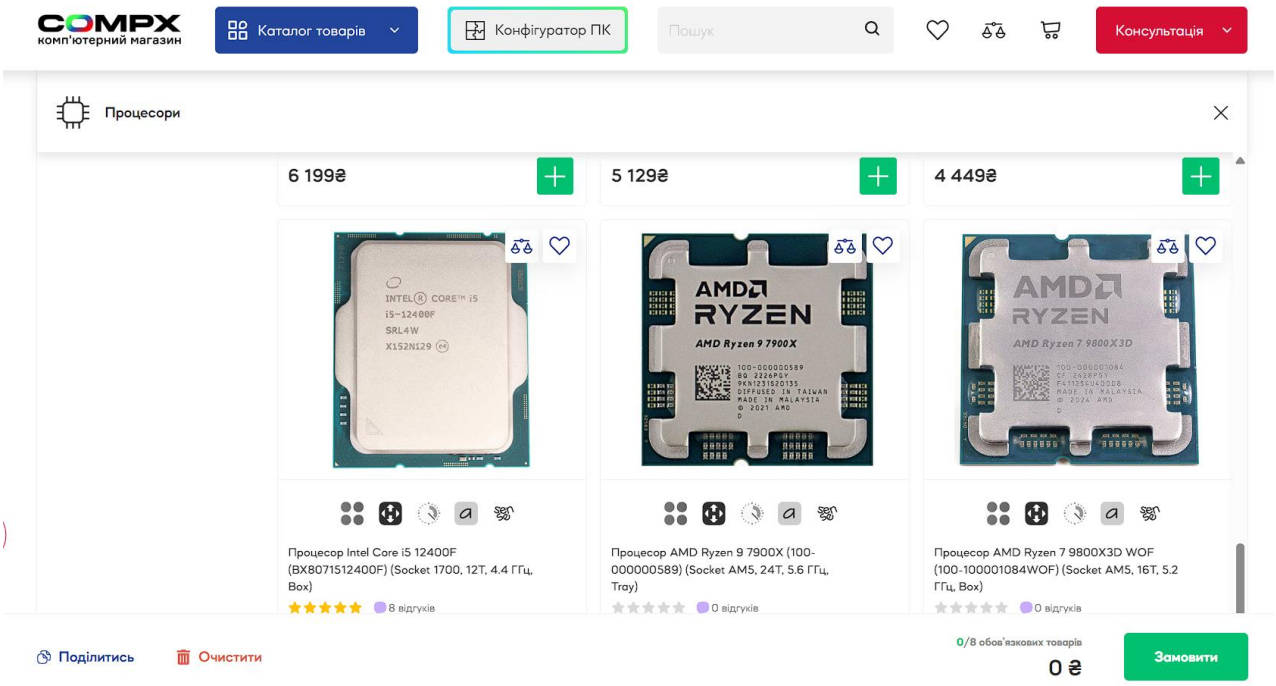


Рис.1.8 Приклад вибору компоненту Compx

У таблиці 1.1 зведено результати аналізу існуючих web-застосунків для підбору комп’ютерних комплектуючих з метою їх порівняння.

Таблиця 1.1

Зведена таблиця аналізу web-застосунків для підбору комп’ютерних комплектуючих та їх порівняння

Показник	Can	Click	ChipChip	Compx
Інтерфейс	Сучасний, простий та доступний навіть для новачків	Зручний, зрозумілий, орієнтований на швидкий пошук	Сучасний, з добре реалізованими фільтрами	Збалансований, з добре реалізованими фільтрами
Фільтри	Базові, за основними характеристиками	Розширені, дозволяють швидко знаходити товари	Потужні, зручні для досвідчених користувачів	Потужні, зручні для досвідчених користувачів
Порівняння компонентів	Відсутнє	Відсутнє	Відсутнє	Відсутнє

Продовження таблиці 1.1

Зведена таблиця аналізу web-застосунків для підбору комп’ютерних комплектуючих та їх порівняння

Перевірка залишкової потужності БЖ	Відсутня	Відсутня	Реалізована	Реалізована
Автоматичне оновлення сумісності	Потрібно переходити на інші сторінки для кожного компонента	Потрібно переходити на інші сторінки для кожного компонента	Реалізоване: оновлення відбувається	Реалізоване: оновлення відбувається
Темний режим	Відсутній	Відсутній	Відсутній	Відсутній
Цільова аудиторія	Початківці, що шукають простий спосіб знайти деталі	Користувачі, що хочуть швидко підібрати товари, не заглиблюючись у технічні аспекти	Досвідчені користувачі та ентузіасти, що цінують автоматизацію і точність підбору компонентів	Досвідчені користувачі та ентузіасти, що цінують автоматизацію і точність підбору компонентів

На основі проведеного аналізу видно, що всі розглянуті онлайн-сервіси мають певні сильні сторони. Зокрема, вони вирізняються простотою використання, наявністю зручних фільтрів, функцією перевірки потужності блока живлення, а також зрозумілим інтерфейсом, що робить їх доступними як для новачків, так і для досвідчених користувачів. Деякі платформи пропонують додаткові можливості, як-от історія переглядів чи детальні фільтри підбору.

Водночас усі сервіси мають низку спільних недоліків. Зокрема, в жодному з них немає функції порівняння компонентів безпосередньо в конструкторі, відсутнє динамічне оновлення сумісних варіантів після вибору, не реалізовано темний режим, а також, за винятком окремих випадків, немає автоматичної перевірки відповідності потужності блока живлення. Ці обмеження стали орієнтиром для визначення функціональних покращень у власному web-застосунку.

1.5 Визначення вимог до застосунку

Проаналізувавши специфіку застосунку, його цільову аудиторію та функціональні можливості аналогів, було визначено наступні вимоги до web-застосунку для підбору комп’ютерних комплектуючих:

– авторизація: реєстрація нових користувачів, вхід та вихід із системи, розподіл прав доступу для адміністратора, продавця та покупця.

– каталог товарів: перегляд комплектуючих за категоріями (процесори, материнські плати, відеокарти тощо), відображення характеристик і цін.

– фільтрація: пошук та фільтрація товарів за технічними характеристиками, виробником, ціною.

– конструктор ПК: покроковий підбір комплектуючих із динамічною перевіркою сумісності (сокет, тип оперативної пам’яті, інтерфейси тощо).

– перевірка потужності БЖ: автоматичне обчислення необхідної потужності блока живлення на основі вибраних компонентів.

– кошик та оформлення замовлення: додавання товарів до кошика, редагування замовлення, підстановка даних авторизованого користувача.

– порівняння товарів: можливість додати товари до списку порівняння та перегляд їхніх характеристик у таблиці.

– адмін-панель: управління товарами, категоріями, користувачами, замовленнями; імпорт CSV-файлів з характеристиками.

– історія замовлень та профіль користувача: доступ до списку попередніх покупок і особистих даних.

– темний режим: можливість перемикання інтерфейсу в темну кольорову схему для зручної роботи у вечірній час.

1.6 Висновки по розділу 1

Web-застосунки для конфігурації персонального комп’ютера — це сучасні, інтерактивні платформи, що спрощують процес підбору сумісних комплектуючих та відкривають доступ до створення персоналізованої системи для кожного користувача. Їх основними перевагами є автоматична перевірка сумісності, зручний інтерфейс поетапного вибору компонентів, велика база даних із технічними характеристиками та візуалізація зібраної конфігурації.

Такі платформи мають освітній і практичний потенціал: допомагають уникнути помилок новачкам, оптимізувати вибір досвідченим користувачам та заощаджують час. Вони охоплюють широку цільову аудиторію — від звичайних користувачів до фахівців, що збирають системи під специфічні задачі.

Розвиток таких рішень базується на впровадженні інтелектуальних алгоритмів, стандартизації даних і динамічному оновленні баз знань, що забезпечує точність, масштабованість і актуальність. Усе це робить web-конструктори ПК важливою складовою сучасної IT-інфраструктури та електронної комерції.

2.ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ СИСТЕМИ ДЛЯ ПІДБОРУ КОМПЛЕКТУЮЧИХ ПК

2.1 Середовище розробки

Інтегроване середовище розробки (IDE) — це комплекс програмних інструментів, який об’єднує в одному додатку всі необхідні засоби для створення, редагування, тестування та налагодження програмного коду. IDE значно підвищує продуктивність розробника, автоматизуючи багато рутинних завдань, таких як автодоповнення коду, перевірка синтаксису, управління версіями, запуск відлагоджувача та інтеграція з системами контролю версій. Завдяки цьому програміст може зосередитися на логіці застосунку, не витрачаючи час на налаштування окремих інструментів.

PhpStorm — це потужне інтегроване середовище розробки (IDE), створене компанією JetBrains, спеціально для розробки на мові програмування PHP [7]. Це середовище забезпечує високий рівень підтримки як серверної, так і клієнтської частин вебзастосунків, завдяки широкій інтеграції з популярними технологіями та фреймворками.

PhpStorm надає повноцінний набір інструментів, які дозволяють ефективно створювати, тестувати та налагоджувати програмний код. Середовище має вбудовану підтримку HTML, CSS, JavaScript, SQL, а також фреймворків на кшталт Laravel, Symfony, Vue.js, React тощо. Завдяки цьому воно є універсальним інструментом для розробників повного циклу (full-stack).

Однією з ключових переваг PhpStorm є інтелектуальна система підказок та автозаповнення, яка значно пришвидшує написання коду та знижує ймовірність помилок. Крім того, середовище містить:

– потужний візуальний налагоджувач (debugger) з підтримкою Xdebug;

– інтеграцію з системами контролю версій, такими як Git, що дозволяє відслідковувати зміни у коді безпосередньо в IDE;

– інструменти для тестування, зокрема підтримку PHPUnit;

– вбудований інтерфейс для роботи з базами даних;

– глибоку інтеграцію з Composer для керування залежностями PHP-проєкту.

У межах даного проєкту PhpStorm використовувалося як основне середовище розробки, в якому реалізовувалась логіка вебзастосунку на PHP з використанням фреймворку Laravel, компоненти на JavaScript і Vue.js, а також шаблонізація інтерфейсу за допомогою Blade.

OpenServer — це універсальне локальне серверне середовище для операційної системи Windows, що дозволяє розробникам запускати web-застосунки та тестувати їх безпосередньо на власному комп'ютері [8]. OpenServer поєднує в собі повний набір необхідних компонентів для розробки, серед яких web-сервери Apache і Nginx, інтерпретатор PHP різних версій, системи управління базами даних MySQL і MariaDB, а також інструменти для роботи з поштовими серверами і FTP.

Однією з ключових переваг OpenServer є його простота у використанні — користувач може швидко запускати або зупиняти сервери через зручний графічний інтерфейс без необхідності налаштовувати конфігураційні файли вручну. Це значно економить час і зусилля, особливо на початкових етапах розробки. Крім того, OpenServer підтримує одночасне використання декількох версій PHP, що дає змогу тестувати застосунки у різних середовищах без додаткових налаштувань.

OpenServer також забезпечує безпеку локальної розробки, імітуючи поведінку реального web-сервера, що дозволяє виявляти помилки ще до розгортання застосунку на реальному хостингу. У проекті він використовувався для стабільного запуску серверної частини застосунку на базі Laravel, що дозволило ефективно тестувати функціональність, налагоджувати код і перевіряти роботу бази даних у режимі реального часу.

2.2 Мови програмування

Мова програмування — це формалізована система запису інструкцій, яку використовують програмісти для створення програмного забезпечення. Вона дозволяє описувати алгоритми, обробку даних і взаємодію з апаратною частиною комп’ютера або іншими системами. Мови програмування мають власний синтаксис і правила, що забезпечують точне та однозначне виконання команд комп’ютером.

PHP (Hypertext Preprocessor) — це популярна високорівнева мова програмування з відкритим кодом, спеціально створена для розробки web-застосунків і генерації динамічного web-контенту [9]. Вона є однією з найдавніших і водночас найбільш поширених мов у сфері web-розробки, що забезпечує її широку підтримку, стабільність та багатий набір інструментів і бібліотек.

В межах даної дипломної роботи PHP виконує ключову роль як основна мова програмування, що забезпечує логіку функціонування серверної частини web-застосунку. Зокрема, PHP відповідає за обробку HTTP-запитів користувачів, виконання бізнес-логіки проєкту, динамічну генерацію HTML-вмісту, роботу з сесіями, куками, авторизацією, валідацією форм, обробкою даних та їх зберіганням у базі даних. Завдяки можливостям мови було реалізовано інтелектуальну систему підбору сумісних комплектуючих ПК, що є центральною функцією створеного web-магазину.

Однією з переваг PHP є його простий, зрозумілий синтаксис, що робить мову доступною для швидкого засвоєння та ефективної роботи над великими проєктами. Підтримка об'єктно-орієнтованого програмування (ООП) дозволяє будувати модульну архітектуру, зменшувати дублювання коду та покращувати масштабованість системи. PHP також чудово інтегрується з HTML і JavaScript, що дозволяє створювати динамічні й зрозумілі інтерфейси користувача.

Для реалізації проєкту PHP був використаний у поєднанні з сучасним web-фреймворком Laravel, який додатково забезпечив високий рівень організації

коду, використання шаблону MVC (Model-View-Controller), зручну маршрутизацію, роботу з базами даних через ORM Eloquent, використання middleware, системи аутентифікації та авторизації, а також гнучке керування залежностями через Composer. Це дало змогу значно спростити й пришвидшити процес розробки, дотримуючись сучасних стандартів у web-програмуванні.

JavaScript — це високорівнева, інтерпретована мова програмування, яка стала невід'ємною частиною сучасної web-розробки [10]. Вона надає можливість створювати динамічний, інтерактивний та адаптивний інтерфейс користувача, що значно покращує взаємодію людини з web-застосунком. JavaScript працює безпосередньо в браузері, дозволяючи змінювати вміст сторінки, реагувати на дії користувача та комунікувати з сервером у фоновому режимі без перезавантаження сторінки.

У контексті даної дипломної роботи JavaScript є ключовим інструментом для реалізації зручного інтерфейсу конструктора ПК, а також динамічного фільтрування товарів і порівняння їх характеристик у реальному часі. Завдяки цьому користувач отримує швидкий відгук системи, що суттєво підвищує комфорт і ефективність підбору сумісних компонентів.

Однією з найважливіших особливостей JavaScript є підтримка різних парадигм програмування — імперативного, об'єктно-орієнтованого та функціонального, що дає змогу будувати гнучкий і масштабований код. Мова також забезпечує широкі можливості роботи з подіями, асинхронними операціями, маніпуляціями з DOM (Document Object Model), що є основою для зміни структури web-сторінки [11].

Для підвищення продуктивності та організації коду у цьому проєкті JavaScript використовується разом із сучасними фронтенд-технологіями, зокрема з Vue.js, який спрощує створення компонентів, керування станом застосунку та забезпечує реактивність інтерфейсу. Це дозволяє швидко реалізовувати складні функції, такі як динамічний вибір комплектуючих із перевіркою сумісності, інтерактивні таблиці порівнянь та інші інтерактивні елементи.

2.3 Web-фрейворки

Web-фреймворки — це програмні засоби, призначені для спрощення та структурування процесу розробки web-застосунків. Вони надають набір готових компонентів і шаблонів для обробки запитів, керування маршрутизацією, роботи з базами даних, забезпечення безпеки та формування інтерфейсу користувача. Завдяки web-фреймворкам розробники можуть зосередитися на логіці застосунку, а не на реалізації рутинних технічних аспектів.

Laravel — це PHP-фреймворк із відкритим кодом, створений для ефективної та сучасної розробки web-застосунків[12]. Він базується на архітектурі MVC (Model-View-Controller), що допомагає розділити відповідальність між різними шарами додатка, підвищуючи читабельність і підтримку коду [13]. Принцип роботи MVC полягає у розділенні додатку на три основні компоненти:

– Model (модель) відповідає за роботу з даними та логіку бізнес-процесів

– View (представлення) відповідає за відображення інформації користувачу

– Controller (контролер) обробляє запити користувача, взаємодіє з моделлю та передає дані до представлення для виводу.

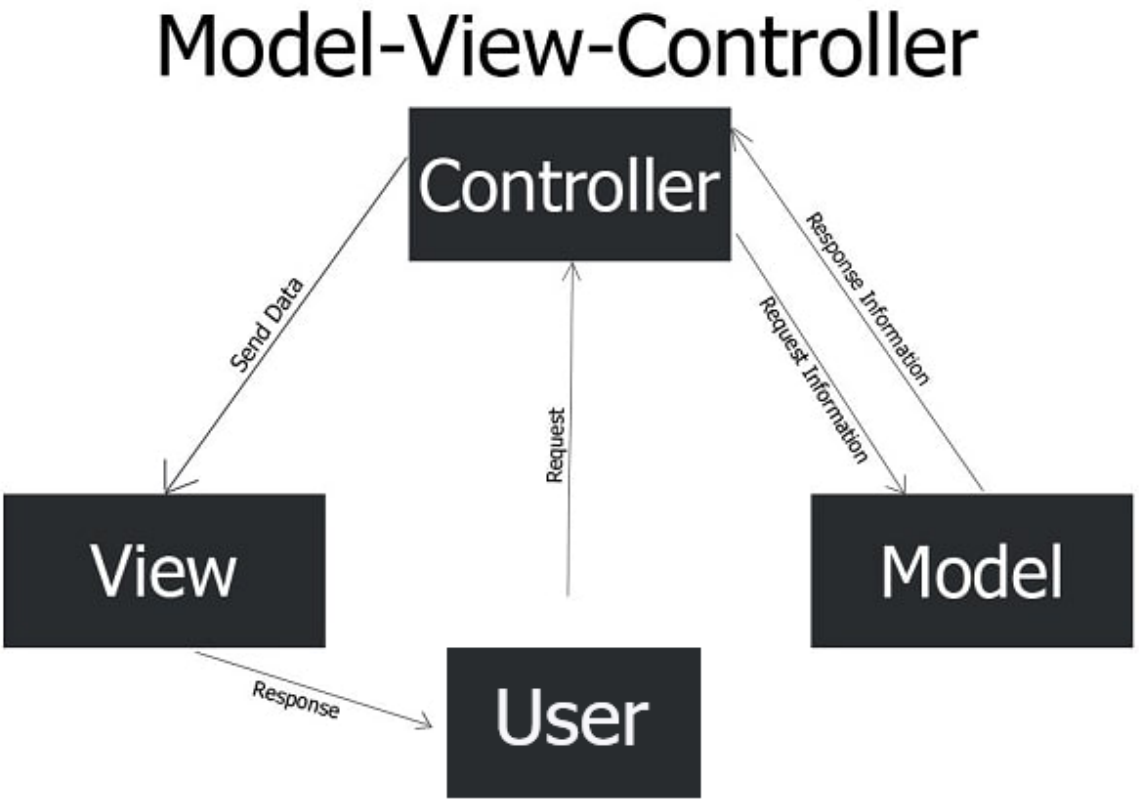


Рис.2.1 Принцип роботи MVC

Фреймворк пропонує багатий набір інструментів і бібліотек, які автоматизують багато рутинних завдань розробника. Зокрема, Eloquent ORM забезпечує зручну роботу з базою даних через об’єктно-реляційне відображення, дозволяючи легко взаємодіяти з таблицями як з об’єктами, підтримуючи зв’язки, фільтрацію, сортування та інші операції без необхідності писати складні SQL-запити [14].

Для організації web-маршрутів Laravel пропонує гнучку систему маршрутизації, де можна визначати URL, пов’язувати їх з контролерами або анонімними функціями, а також групувати маршрути з застосуванням спільних налаштувань. Важливою складовою є механізм middleware — проміжний програмний код, який виконується перед тим, як HTTP-запит досягне контролера. Це дозволяє реалізувати ключові функції, такі як аутентифікація користувачів, перевірка ролей і прав доступу, захист від CSRF-атак, логування, обробка CORS тощо. Middleware відіграє роль “посередника” між клієнтом і

сервером, що допомагає стандартизувати обробку запитів, забезпечити безпеку, повторне використання логіки та масштабованість [15]. Як і загалом у сфері програмного забезпечення, middleware у Laravel дозволяє ефективно ізолювати загальні завдання в окремі компоненти, зменшуючи дублювання коду та спрощуючи супровід застосунку. Завдяки цьому додаток стає не лише безпечнішим, а й більш структурованим і гнучким у подальшому розвитку.

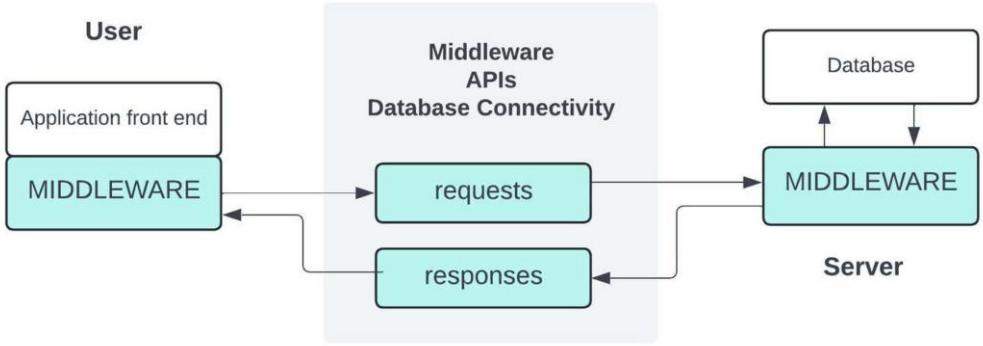


Рис.2.2 Принцип роботи Middleware у фреймворку Laravel

Для роботи з базою даних Laravel має систему міграцій, що дозволяє версіонувати структуру БД, легко вносити зміни і підтримувати синхронізацію з різними середовищами розробки. Крім того, сидери допомагають наповнювати базу тестовими або початковими даними, що значно прискорює тестування і розробку.

Однією з переваг Laravel є вбудований шаблонізатор Blade, який спрощує створення інтерфейсів. Він дозволяє розробникам використовувати шаблонні змінні, директиви, компоненти та наслідування шаблонів для підтримки DRY-принципу (Don't Repeat Yourself).

Laravel також містить потужний командний інтерфейс Artisan, який автоматизує безліч завдань: генерація коду, запуск тестів, керування міграціями, робота з чергами, кешування тощо. Artisan дозволяє значно підвищити продуктивність розробки.

У плані безпеки Laravel підтримує захист від основних web-загроз — XSS, CSRF, SQL-ін’єкцій — через автоматичну обробку вхідних даних і застосування відповідних фільтрів[16]. Окрім того, він має вбудовані засоби для роботи з аутентифікацією та авторизацією, включно з підтримкою OAuth, соціального входу, двофакторної аутентифікації.

Для побудови сучасних API Laravel пропонує засоби для створення RESTful сервісів, включаючи підтримку маршрутизації API, аутентифікації за токенами, обробки запитів у форматах JSON тощо.

Велика та активна спільнота розробників, широкий вибір пакетів (через Composer), офіційна підтримка та регулярні оновлення роблять Laravel одним із найкращих фреймворків для web-розробки на сьогодні. Він підходить як для невеликих проектів, так і для масштабних корпоративних систем, що потребують надійності, швидкості розробки і легкості підтримки.

Vue.js — це прогресивний JavaScript-фреймворк для створення інтерфейсів користувача, який відзначається простотою, гнучкістю та високою продуктивністю [17]. Він розроблений так, щоб бути максимально зрозумілим для розробників будь-якого рівня, дозволяючи легко інтегруватися як у невеликі проєкти, так і у великі складні застосунки. Vue.js використовує декларативний підхід до опису інтерфейсів, що робить код чистим, структурованим і легко підтримуваним.

У межах даної дипломної роботи Vue.js відіграє ключову роль у створенні динамічного, інтерактивного інтерфейсу web-застосунку для підбору комплектуючих ПК. З його допомогою реалізовано реактивне оновлення контенту без перезавантаження сторінки, що забезпечує швидкий та плавний користувацький досвід. Завдяки компонентній архітектурі Vue.js, складні UI-елементи, такі як форми вибору, таблиці порівнянь і фільтри товарів, були розбиті на невеликі, самостійні блоки, що значно полегшило їх розробку, тестування та масштабування.

Однією з найсильніших сторін Vue.js є його реактивна система, яка автоматично відслідковує зміни в даних і відповідно оновлює інтерфейс

користувача, зводячи до мінімуму ручне маніпулювання DOM. Це дозволяє зосередитись на логіці бізнес-процесів і дизайні, не втрачаючи час на складні оптимізації.

Крім того, Vue.js легко інтегрується з іншими бібліотеками та інструментами, що робить його універсальним вибором для проєктів різної складності. У поєднанні з Laravel у бекенді, Vue.js забезпечує повноцінний стек для розробки сучасних web-застосунків із чітким розподілом відповідальності між серверною логікою та фронтендом.

Blade — це потужний та зручний шаблонізатор, вбудований у web-фреймворк Laravel, який забезпечує ефективне створення динамічного HTML-коду [18]. Він спрощує роботу з виводом даних, керуванням логікою відображення та структурою web-сторінок, дозволяючи розробникам легко комбінувати PHP-код із HTML без втрати читабельності.

У межах даної дипломної роботи Blade виконує важливу роль у формуванні фронтенд-частини web-застосунку. Завдяки йому реалізовано чітку структуру шаблонів, що дозволяє повторно використовувати загальні елементи інтерфейсу, такі як шапка сайту, футер, меню та картки товарів. Це значно спрощує підтримку та масштабування коду, а також пришвидшує розробку нових сторінок.

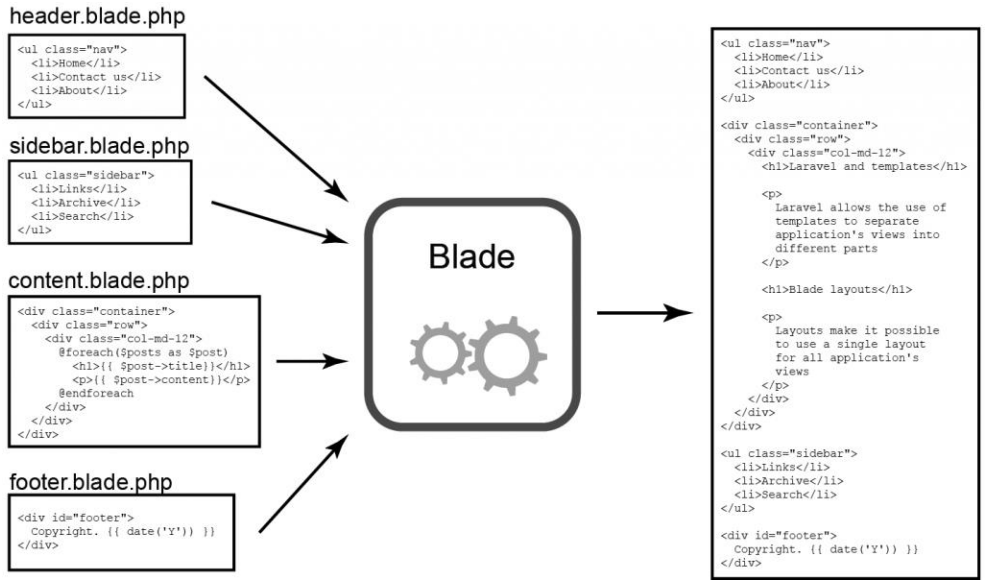


Рис.2.3 Принцип роботи Blade

Blade підтримує механізми наслідування шаблонів і компоненти, що допомагає розробляти складні інтерфейси у модульному стилі. Він також дозволяє використовувати умови, цикли та інші конструкції без необхідності писати громіздкий PHP-код у представленнях. Крім того, Blade автоматично екранує вихідні дані, що підвищує безпеку застосунку, запобігаючи атакам типу XSS.

Інтеграція Blade з Laravel забезпечує безшовну взаємодію між серверною логікою та фронтендом, дозволяючи динамічно формувати сторінки залежно від даних, які надходять з контролерів. Це створює стабільну та ефективну основу для розробки web-магазину з функціоналом підбору сумісних комплектуючих ПК, підвищуючи зручність користувачів і продуктивність команди розробників.

2.4 Система управління базою даних

Система управління базою даних (СУБД) — це програмне забезпечення, яке дозволяє створювати, зберігати, організовувати і керувати даними у базі. Вона забезпечує зручний і безпечний доступ до інформації, підтримує виконання запитів, обробку даних і контроль цілісності, що необхідно для ефективної роботи будь-якого web-застосунку чи інформаційної системи.

MySQL — це система керування реляційними базами даних (СКБД), яка широко застосовується у web-розробці для надійного зберігання, обробки та доступу до структурованих даних[19]. Вона підтримує мову SQL (Structured Query Language), що дозволяє ефективно працювати з таблицями, запитами, фільтрацією, сортуванням і агрегацією даних.

У рамках дипломного проєкту MySQL використовується як основна база даних для збереження інформації про комплектуючі ПК, категорії товарів, користувачів, замовлення, а також дані для порівняння характеристик. Її роль — забезпечити централізоване сховище, до якого звертається PHP-код через ORM Eloquent (Laravel), для отримання, створення, оновлення або видалення даних.

Серед основних переваг MySQL: висока продуктивність, масштабованість, підтримка транзакцій, індексів і зовнішніх ключів, що гарантує цілісність і узгодженість даних. Важливу роль відіграє також підтримка складних зв’язків між таблицями, що особливо актуально при реалізації логіки сумісності комплектуючих у конструкторі ПК.

Інтеграція MySQL з Laravel дозволяє легко мігрувати структуру бази даних, створювати таблиці через миграції, наповнювати її тестовими даними та використовувати зручний механізм запитів, що значно пришвидшує розробку та супровід web-застосунку.

PhpMyAdmin — це зручний web-застосунок, що забезпечує графічний інтерфейс для взаємодії з базою даних MySQL [20]. Він значно спрощує адміністрування та керування базами, дозволяючи розробнику виконувати більшість операцій без необхідності вручну писати SQL-запити. Такий підхід особливо корисний під час активної розробки, коли важливо мати швидкий доступ до структури та вмісту таблиць, а також можливість оперативно вносити зміни.

У процесі створення дипломного проєкту phpMyAdmin використовувався як основний інструмент для управління базою даних. За його допомогою було реалізовано створення структури таблиць, перевірку логіки зв’язків між даними, а також додавання й редагування вмісту вручну. Це стало у пригоді на початкових етапах, коли тестувалась логіка сумісності компонентів ПК, а також під час імпорту CSV-файлів з характеристиками товарів, де була потреба швидко переглянути або скоригувати завантажені дані.

Також, phpMyAdmin надає візуальне представлення структури бази, що дозволяє легко орієнтуватися в іменах таблиць, типах полів, зв’язках і ключах. Вбудований редактор SQL-запитів дає можливість перевіряти більш складні умови вибірки, фільтрації або обчислення даних, не виходячи з браузера. У підсумку, використання phpMyAdmin значно пришвидшило роботу з базою даних, зробивши процес розробки більш контрольованим і прозорим.

2.5 Менеджери пакетів

Менеджер пакетів — це інструмент, який автоматично завантажує, встановлює, оновлює та керує бібліотеками та залежностями, необхідними для роботи програмного проєкту. Він спрощує розробку, дозволяючи швидко підключати сторонні модулі та слідкувати за їхніми оновленнями.

Composer — це потужний інструмент керування залежностями у світі PHP, який автоматизує процес підключення зовнішніх бібліотек до проєкту[21]. Завдяки Composer розробник може легко встановлювати сторонні пакети, контролювати їх версії, а також ефективно оновлювати компоненти, не турбуючись про ручне налаштування структури файлів. У дипломному проєкті Composer відіграє важливу роль, забезпечуючи зручне підключення та керування всіма необхідними залежностями, включно з фреймворком Laravel, бібліотеками для обробки CSV-файлів, роботою з API та інструментами безпеки.

Коли створюється новий проєкт на Laravel, Composer автоматично завантажує всі потрібні бібліотеки, описані у файлі composer.json, і розміщує їх у директорії vendor. Це не лише пришвидшує початкове розгортання застосунку, а й дає змогу підтримувати проєкт у актуальному стані протягом усього циклу розробки. Наприклад, при потребі підключити пакет для обробки зображень чи побудови PDF-файлів достатньо лише однієї команди в терміналі — Composer сам завантажить і зареєструє залежність у проєкті.

Ще однією перевагою Composer є можливість автозавантаження класів через PSR-4, що дозволяє уникнути великої кількості require-інструкцій у коді. Це спрощує архітектуру проєкту, підвищує читабельність і забезпечує чисту модульну структуру. У контексті дипломної роботи Composer дозволив інтегрувати всі ключові інструменти, зберігаючи чистоту коду та дотримуючись сучасних стандартів web-розробки.

NPM (Node Package Manager) — це потужний менеджер пакетів для JavaScript, який є невід’ємною частиною сучасної фронтенд-розробки[22]. Він надає розробникам доступ до величезного репозиторію відкритих бібліотек і

інструментів, які значно полегшують і прискорюють процес створення web-застосунків. NPM дозволяє швидко встановлювати, оновлювати та керувати залежностями проєкту, автоматизуючи багато рутинних процесів і забезпечуючи високу гнучкість у налаштуванні середовища розробки.

У рамках дипломної роботи NPM відіграє важливу роль як на етапі ініціалізації проєкту, так і протягом усього циклу розробки клієнтської частини web-застосунку. З його допомогою було встановлено та налаштовано Vue.js — сучасний JavaScript-фреймворк, який використовується для створення інтерактивних компонентів інтерфейсу.

Використання NPM дозволяє ефективно розділити логіку клієнтської частини від серверної. Це не лише покращує структуру коду, але й сприяє масштабованості проєкту, роблячи його більш зручним для командної роботи та подальшого розширення. Наприклад, зміни в інтерфейсі можна вносити незалежно від змін на бекенді, що забезпечує гнучкість у розробці та тестуванні.

Важливо зазначити, що завдяки системі керування залежностями NPM, забезпечується узгодженість версій бібліотек у різних середовищах. Це дозволяє уникнути конфліктів і помилок, які можуть виникнути при різних конфігураціях. Усі залежності описуються у файлі package.json, що спрощує відтворення проєкту на іншій машині або при спільній роботі кількох розробників.

Загалом, використання NPM стало критично важливим етапом у побудові клієнтської частини дипломного проєкту. Він забезпечив надійність, масштабованість та зручність у роботі з сучасними фронтенд-технологіями, зробивши процес розробки більш систематизованим, а результат — більш стабільним і функціональним.

3 ПРОЄКТУВАННЯ WEB-ЗАСТОСУНКУ ДЛЯ ПІДБОРУ СУМІСНИХ КОМПЛЕКТУЮЧИХ ПК

3.1 Моделювання вимог

У процесі моделювання вимог до системи було визначено як функціональні, так і нефункціональні вимоги, які є основою для подальшої розробки та впровадження web-застосунку. Вони забезпечують чітке розуміння ключових функцій, які система повинна виконувати, а також визначають критерії якості, безпеки та зручності використання.

Функціональні вимоги включають:

– реалізацію інтерактивного конструктора ПК для підбору сумісних комплектуючих;

– забезпечення можливості порівняння характеристик комплектуючих;

– реєстрацію та авторизацію користувачів для доступу до персоналізованих даних;

– реалізацію перегляду історії замовлень у профілі користувача для зручності повторних покупок;

– можливість пошуку та фільтрації комплектуючих за різними параметрами.

Нефункціональні вимоги передбачають:

– захист персональних даних користувачів за допомогою CSRF-токенів, хешування паролів і middleware контролю доступу;

– оформлення сайту у темних тонах для зниження навантаження на очі та забезпечення комфортної роботи користувачів;

– забезпечення імпорту та оновлення комплектуючих і збірок без значних затримок для адміністратора;

– адаптацію системи для коректного відображення у популярних

браузерах, таких як FireFox, Edge, Opera та Chrome.

В результаті, чітке формулювання цих вимог дозволяє створити надійну та зручну у використанні систему, що максимально відповідає потребам користувачів та адміністрації.

3.2 Опис функціонування системи

Функціонування системи базується на тісній взаємодії її основних модулів, що забезпечує ефективне виконання всіх необхідних операцій. Кожен модуль виконує свою спеціалізовану роль, при цьому взаємодія між ними організована таким чином, щоб підтримувати узгодженість даних і безперервність бізнес-процесів. Завдяки цьому система працює як єдиний цілісний механізм, здатний динамічно адаптуватися до вимог користувачів і адміністрації.

Щоб чітко відобразити основні сценарії взаємодії користувача з web-застосунком, було створено UML-діаграму варіантів використання. Вона демонструє ключові дії, які може виконувати користувач: від вибору та порівняння комплектуючих у конструкторі ПК до оформлення замовлень і керування особистим кабінетом. Ця діаграма допомагає краще зрозуміти функціональні можливості системи з точки зору кінцевого користувача та забезпечує основу для подальшої розробки інтерфейсів і бізнес-логіки.

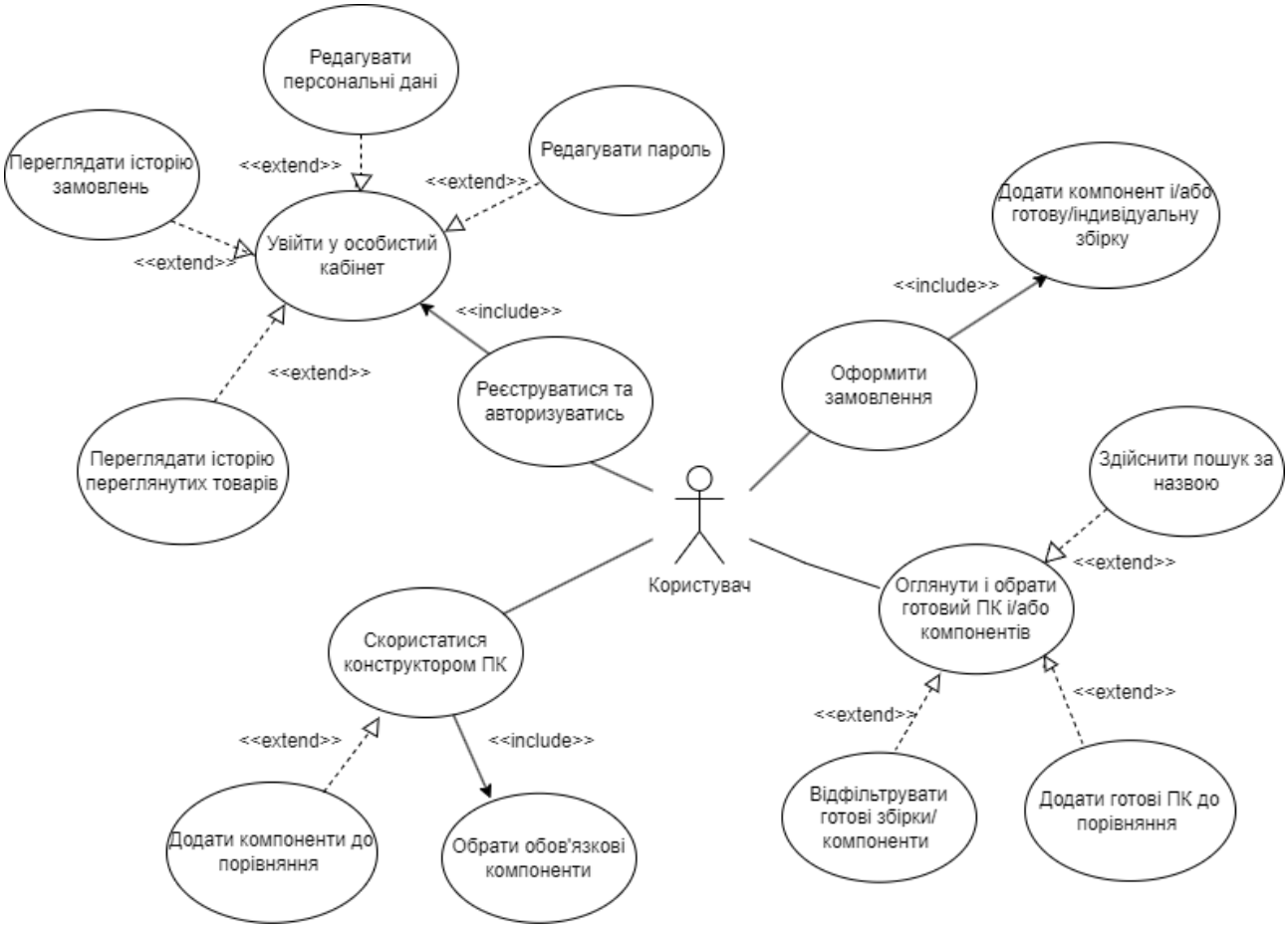


Рис 3.1 Діаграма варіантів використання користувачем

Діаграма варіантів використання ілюструє основні дії, доступні гостю системи, який не має можливості користуватися персональним кабінетом. Гість може переглядати товари, користуватися конструктором ПК, додавати компоненти до кошика та оформлювати замовлення, але для доступу до розширених функцій, таких як збереження налаштувань або перегляд історії замовлень, необхідна реєстрація та авторизація в системі. Такий підхід забезпечує зручність ознайомлення з ресурсом без обов’язкової реєстрації, водночас стимулюючи користувачів до створення облікового запису.

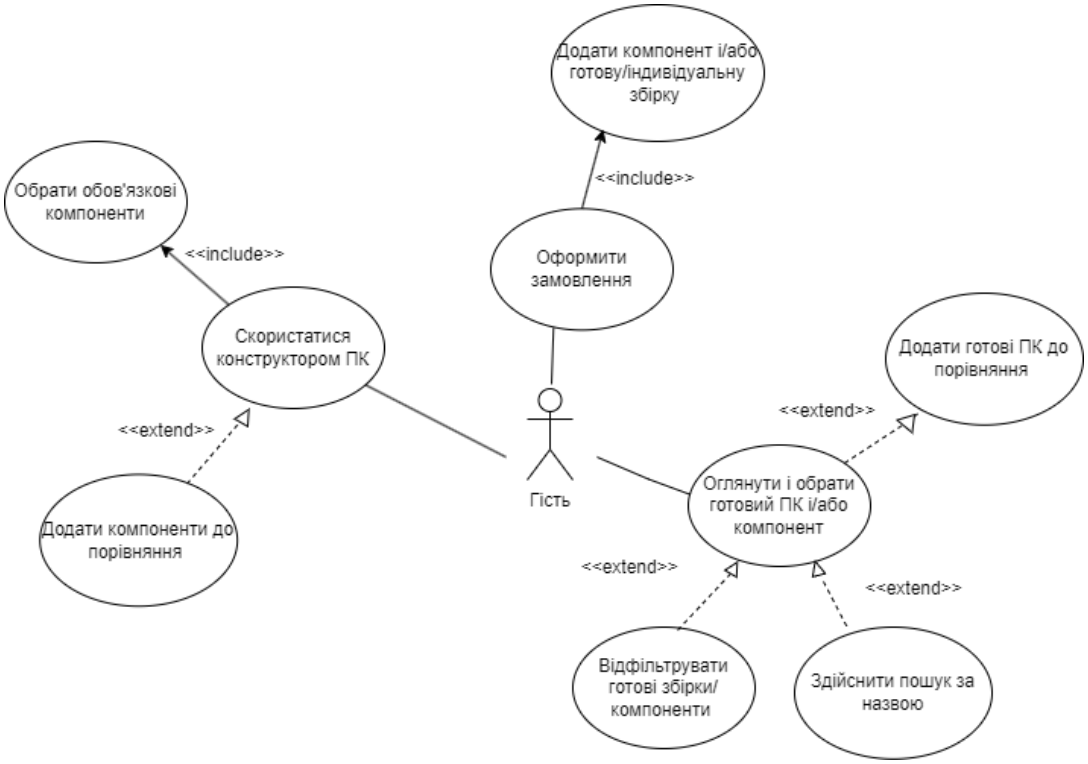


Рис 3.4 Діаграма варіантів використання гостем

Адміністратори, зі свого боку, отримують зручний інтерфейс для керування всіма складовими системи — товарами, замовленнями та користувачами. Вони можуть додавати нові компоненти, редагувати існуючі чи імпортувати готові збірки, що безпосередньо впливає на інформацію, доступну у конструкторі та інших користувацьких інтерфейсах. Керування замовленнями в адмін-панелі забезпечує контроль над процесом покупки і дозволяє оперативно обробляти запити, що гарантує якісний сервіс.

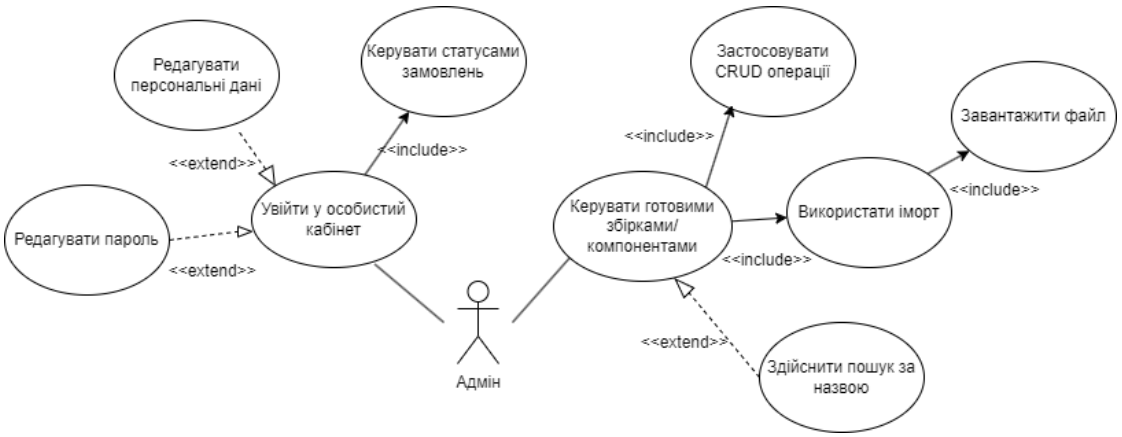


Рис.3.4 Діаграма варіантів використання адміном

Основні функціональні модулі web-застосунку тісно взаємопов’язані і разом створюють цілісний досвід для користувачів і адміністраторів. Серцем системи є інтерактивний конструктор ПК, реалізований як Vue-компонент PcConfigurator.vue. Цей компонент не просто дозволяє вибирати комплектуючі, а й активно взаємодіє з бекендом, щоб миттєво оновлювати список сумісних деталей, порівнювати їх характеристики і підраховувати загальну вартість збірки в режимі реального часу. Така динамічна взаємодія допомагає користувачу уникнути помилок і максимально спростити процес збору персонального комп’ютера.

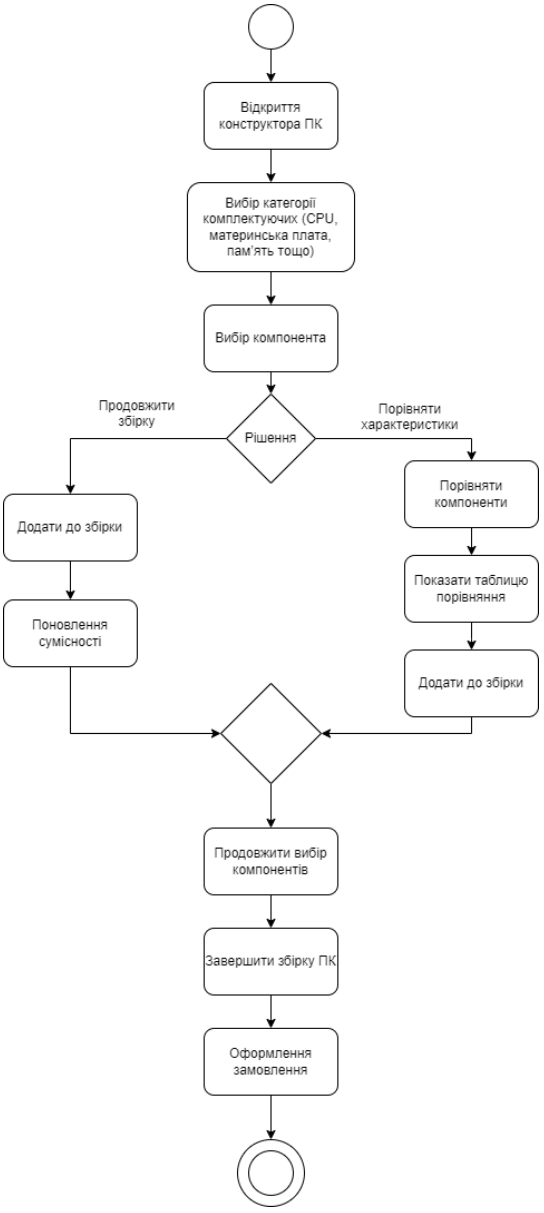


Рис.3.2 Діаграма діяльності процесу підбору комплектуючих

Щоб наповнювати систему актуальними даними, існує механізм імпорту комплектуючих через клас ImportPcParts. Він автоматизує завантаження інформації із CSV-файлів у базу даних, що дозволяє швидко оновлювати асортимент товарів і їх характеристики, включно з автоматичним додаванням зображень. Це забезпечує безперервний зв’язок між базою даних і конструктором, адже нові або оновлені компоненти відразу стають доступними для вибору користувачем.

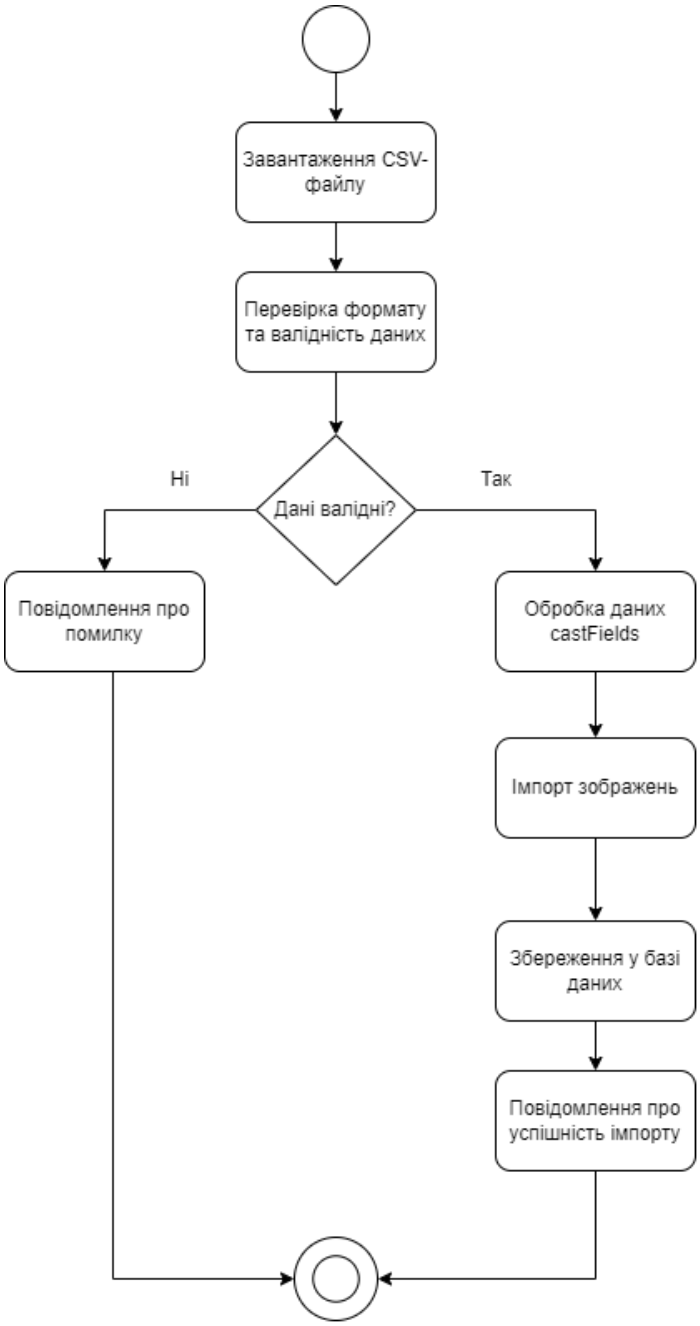


Рис.3.3 Діаграма діяльності процесу імпорту даних

Для кінцевого користувача розроблено набір зручних та логічно структурованих інтерфейсів, що забезпечують комфортну навігацію та ефективну роботу з web-застосунком.

Головна сторінка: містить навігаційне меню, пошук та кнопки швидкого доступу до категорій. Дизайн адаптовано для зручного перегляду та взаємодії користувача.

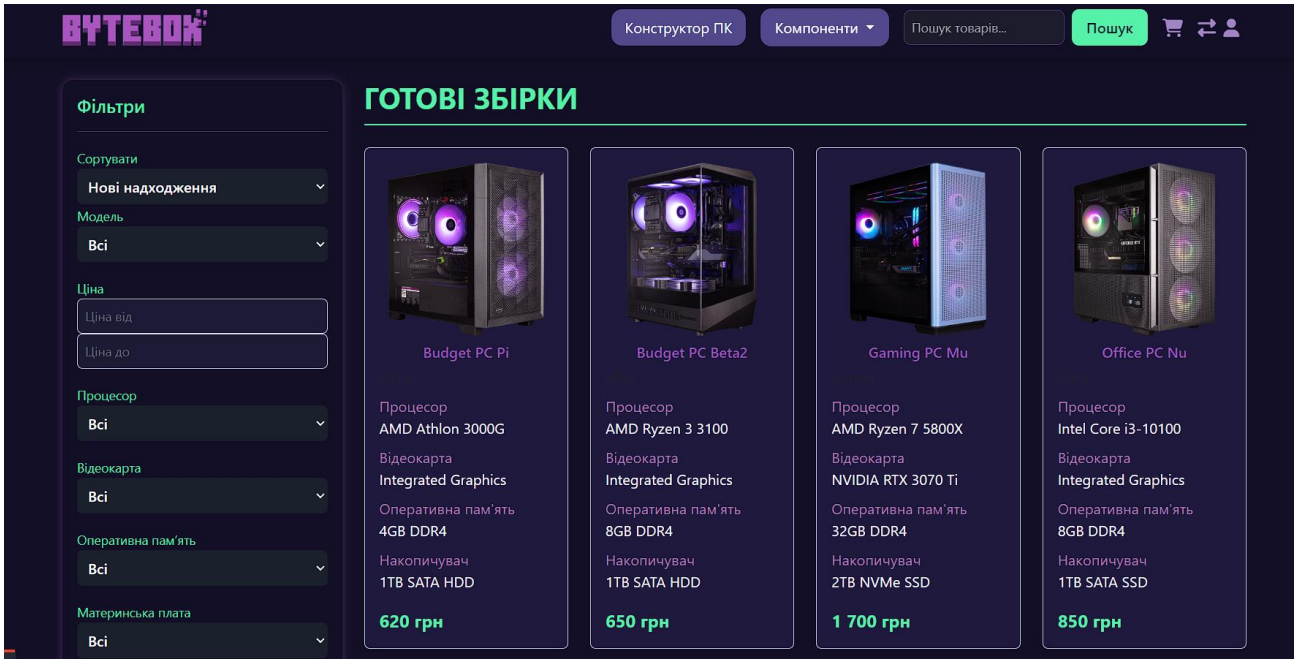


Рис.3.5 Приклад головної сторінки ByteBox

Сторінка вибору компонента: детальна інформація про окремий товар з усіма характеристиками і можливістю додати його до збірки або кошика.

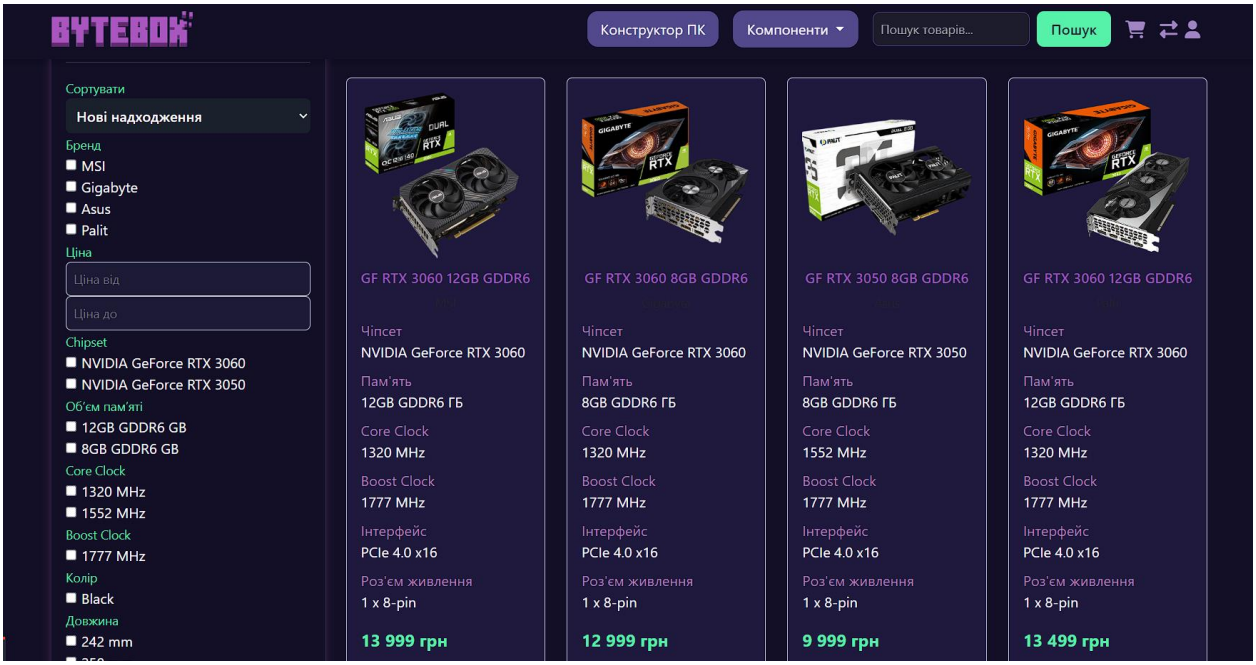


Рис.3.6 Приклад вибору компоненту

Сторінка порівняння готових збірок: дає можливість переглянути і вибрати найбільш підходящий варіант комп’ютера.

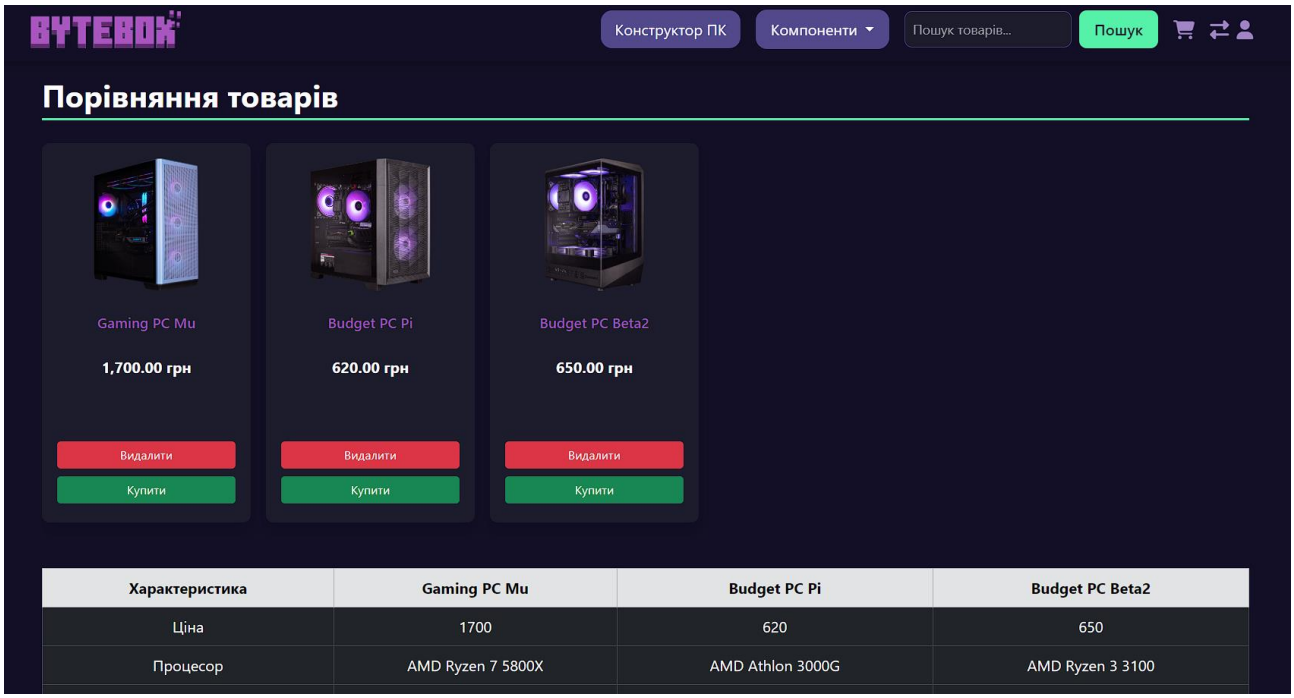


Рис.3.7 Приклад порівняння готових збірок ByteBox

Сторінка конструктора ПК: інтерфейс для поетапного підбору та перевірки сумісності комплектуючих із інтерактивними функціями вибору і порівняння компонентів.

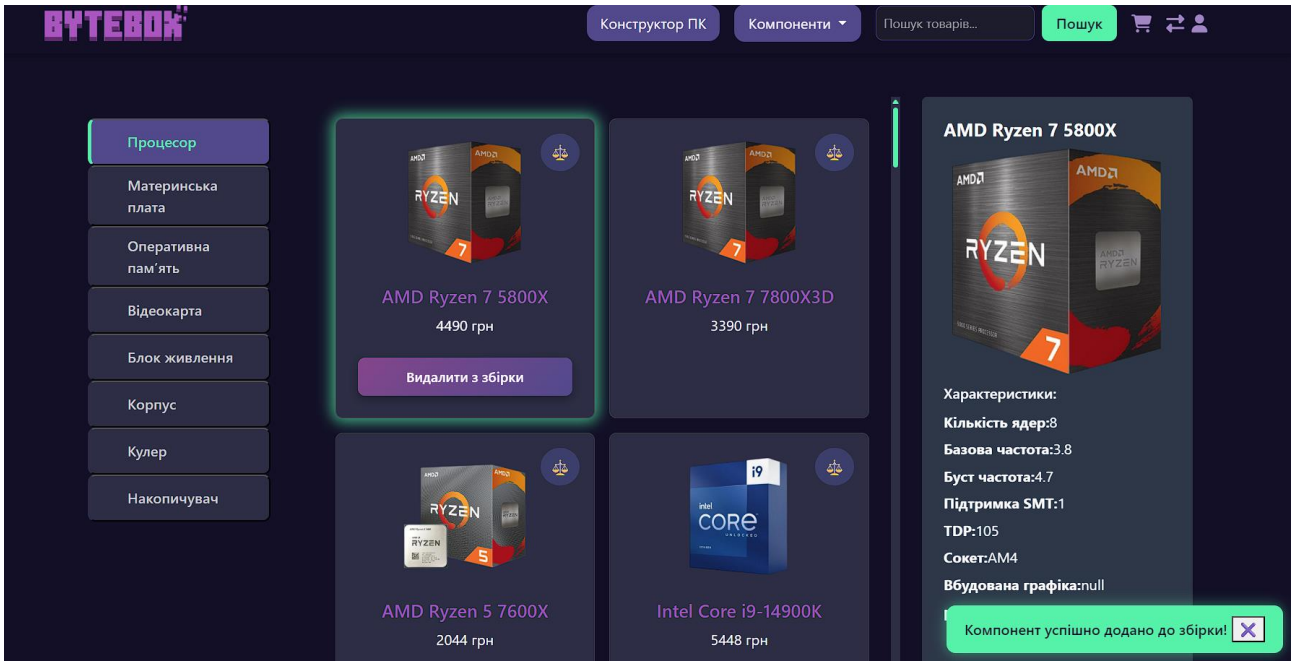


Рис.3.8 Приклад конструктору ПК ByteBox

Сторінка порівняння компонентів: дозволяє аналізувати характеристики декількох товарів для прийняття обґрунтованого рішення.

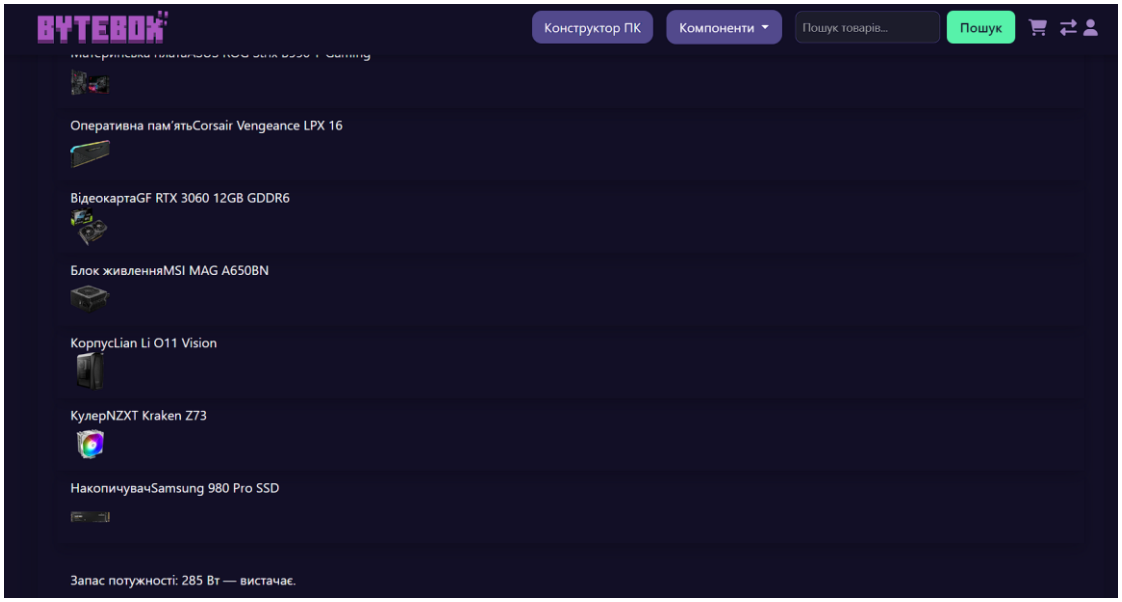


Рис.3.9 Приклад вибору компонентів ByteBox

Кошик: відображає вибрані товари з можливістю редагування кількості або видалення.

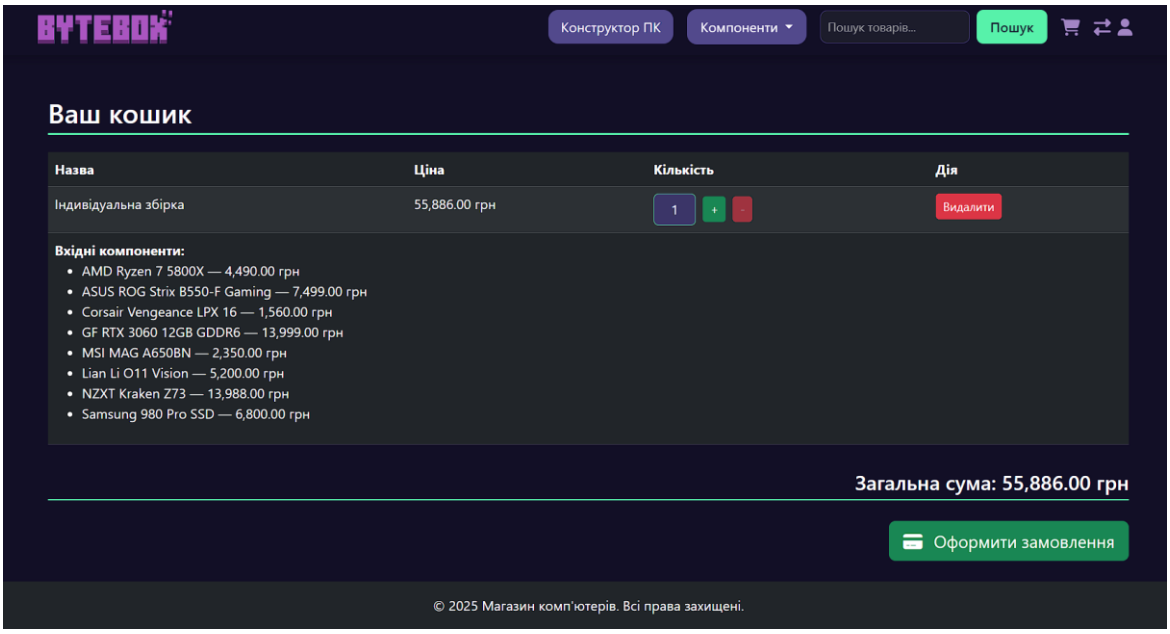


Рис.3.10 Приклад кошику ByteBox

Оформлення замовлення: форма для введення контактних даних та підтвердження покупки.

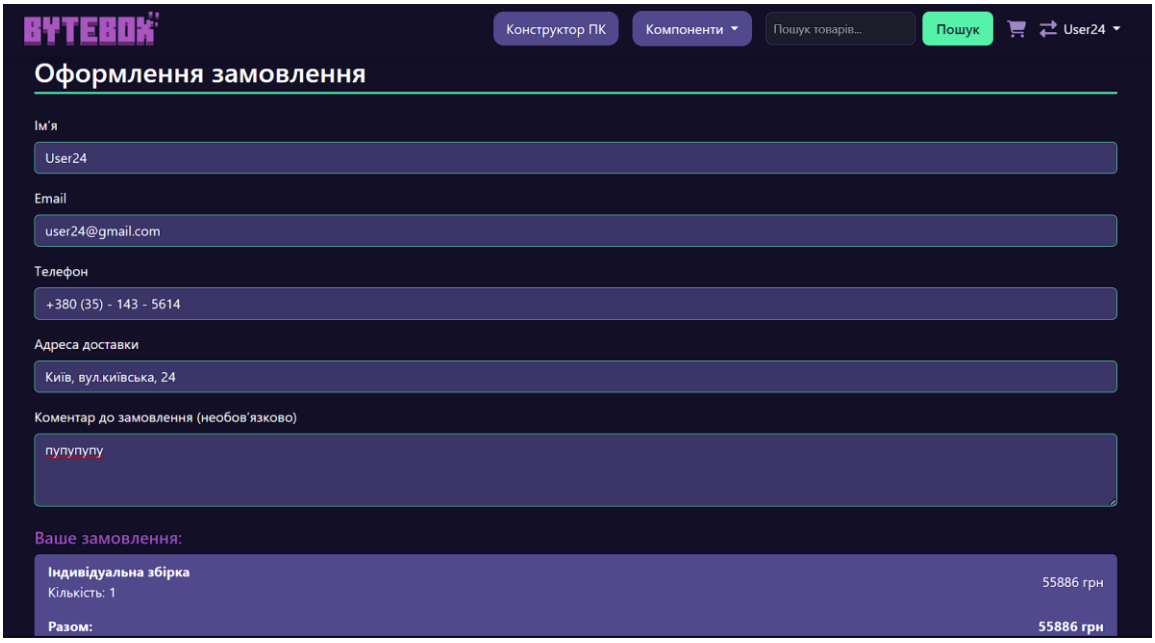


Рис.3.13 Приклад оформлення замовлення ByteBox

Реєстрація та логін: стандартні інтерфейси для створення облікового запису та входу в систему.

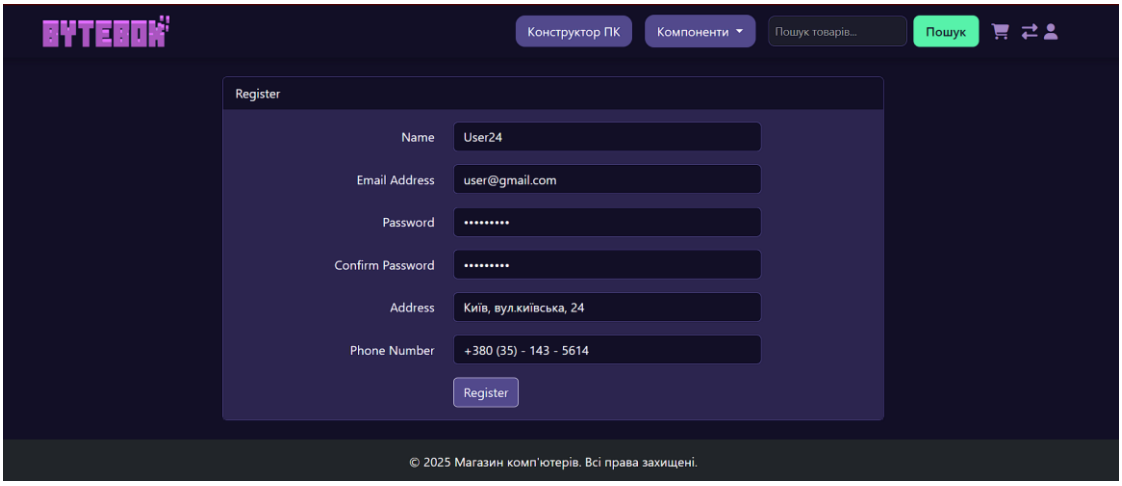


Рис.3.11 Приклад реєстрації ByteBox

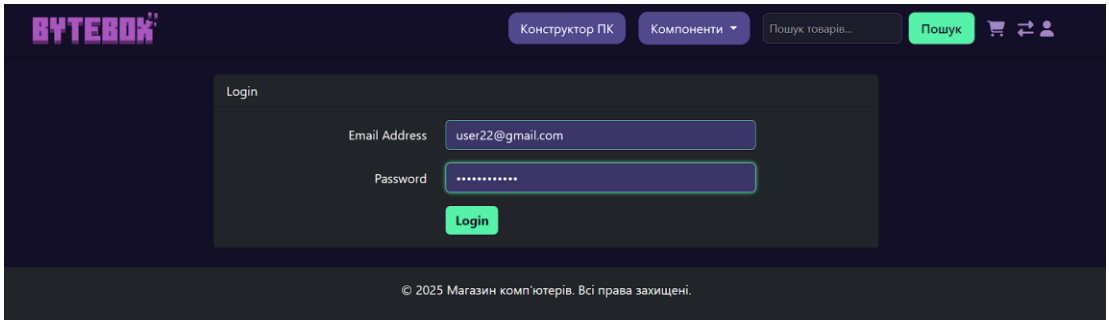


Рис. 3.12 Приклад логіну ByteBox

Особистий кабінет: дає доступ до профілю користувача, історії замовлень, порівнянь, переглянутих товарів і налаштувань акаунту.

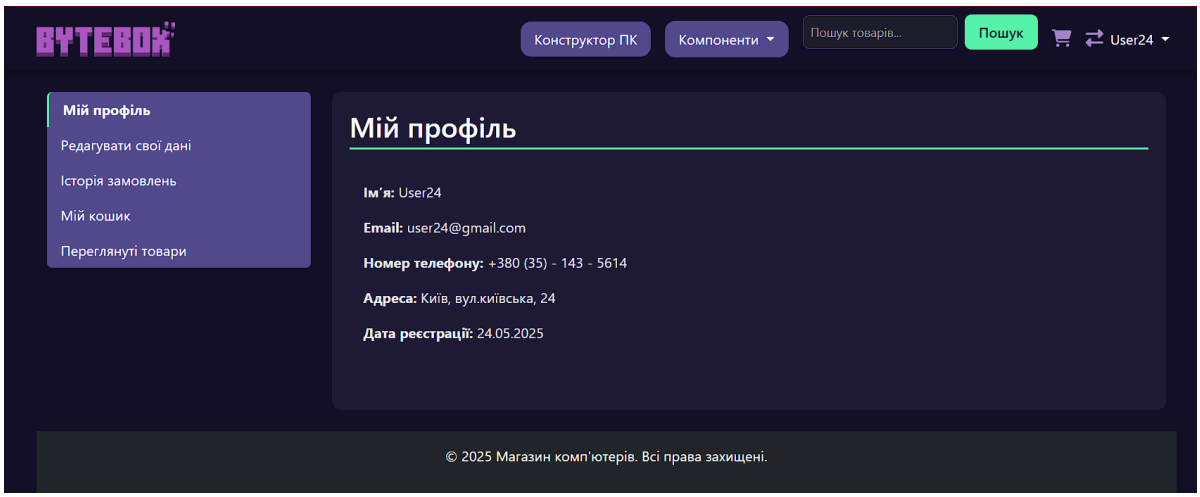


Рис.3.14 Приклад профілю користувача ByteBox

Користувач має змогу переглядати та редагувати власну особисту інформацію через інтерфейс кабінету. Це дозволяє підтримувати актуальні контактні дані та забезпечує безпеку облікового запису.

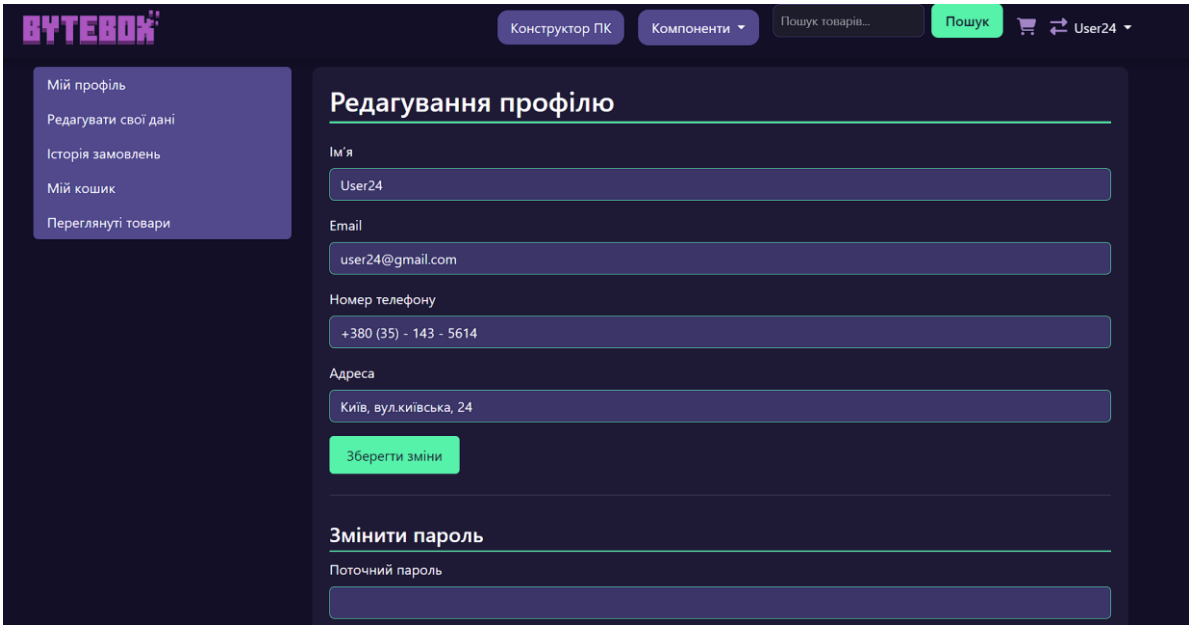


Рис.3.15 Приклад редагування профілю користувача ByteBox

Сторінка переглянутих товарів: дозволяє користувачам швидко переглянути перелік товарів, які вони вже відвідували, що значно підвищує зручність повторного вибору або покупки.

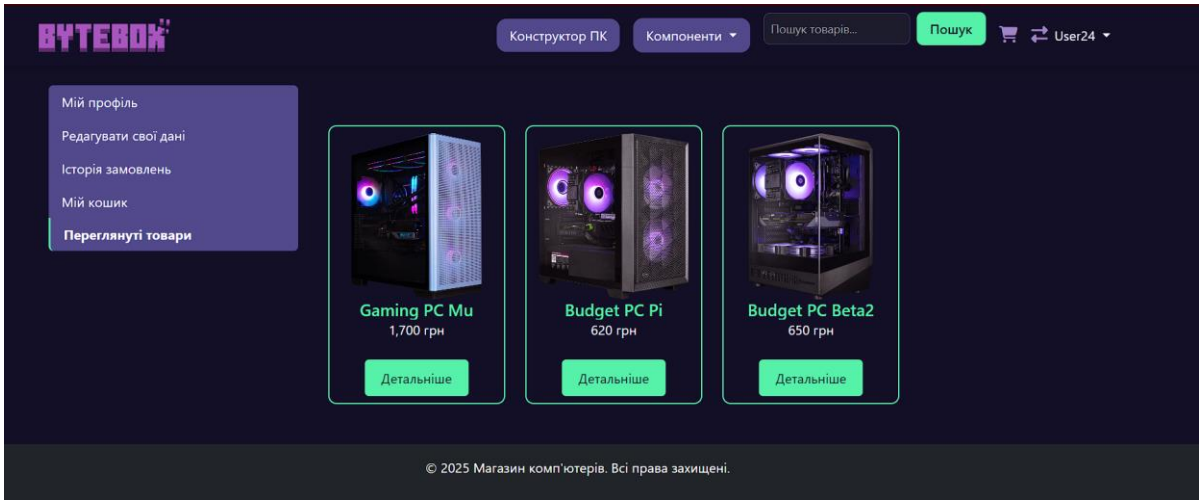


Рис.3.16 Приклад переглянутих товарів користувачем ByteBox

Сторінка історія замовлень: надає користувачеві можливість переглядати всі свої попередні замовлення, слідкувати за їх статусами та деталями, що робить процес покупки прозорим, зручним та контрольованим.

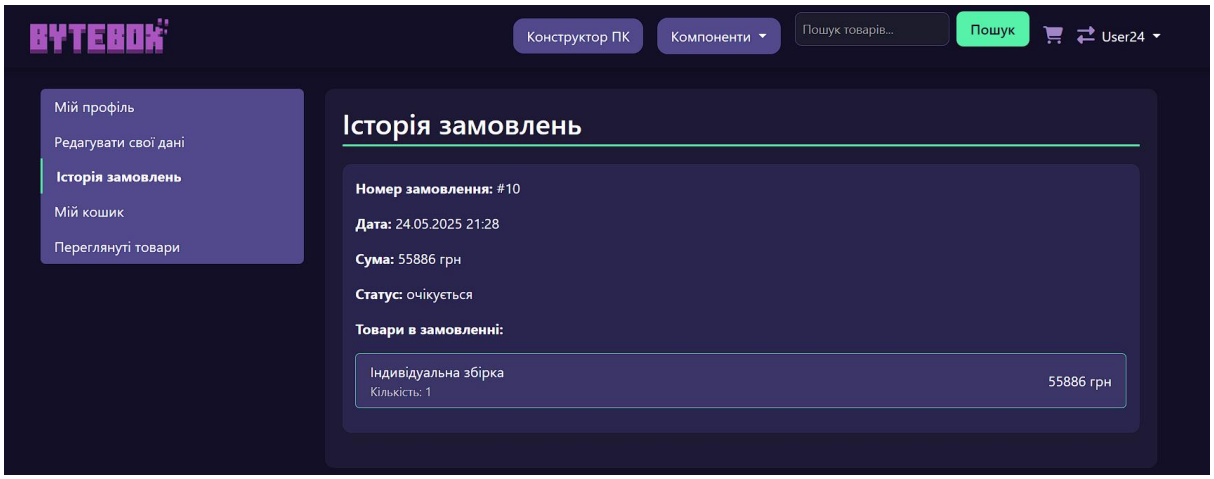


Рис.3.17 Приклад історії замовлень користувачем

Головна сторінка в адмін-частині: містить список усіх товарів із візуальним поданням у вигляді карток. Під кожною карткою розміщені кнопки "Редагувати" та "Видалити" для зручного керування. Зліва розташована бічна панель навігації з функціональними кнопками: "Додати товар", "Імпортувати ПК-збірку", "Імпортувати компонент"

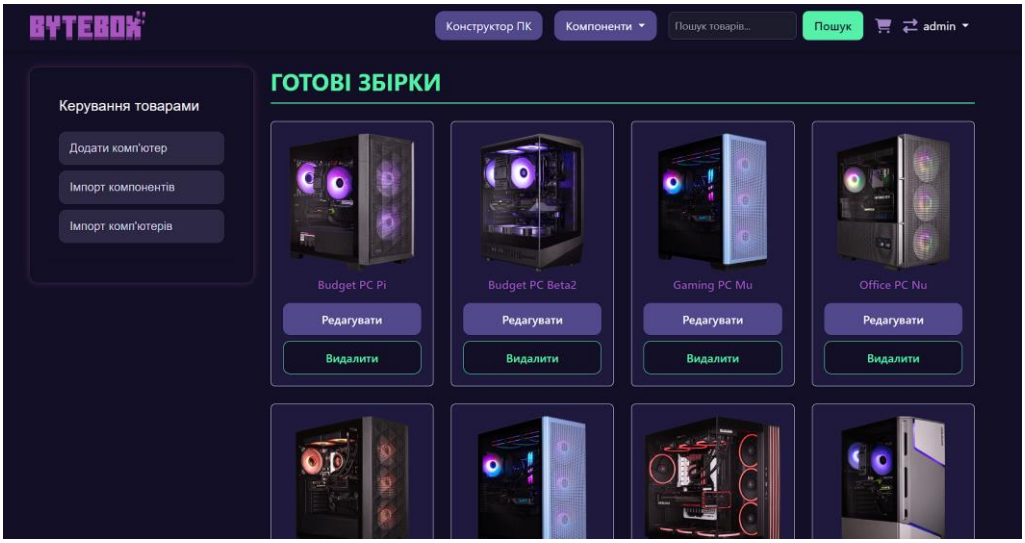


Рис.3.18 Приклад вигляду головної сторінки в адмін-частині ByteBox

Імпорт готових збірок: реалізовано механізм завантаження наперед сформованих конфігурацій ПК у систему для подальшого перегляду, порівняння та редагування.

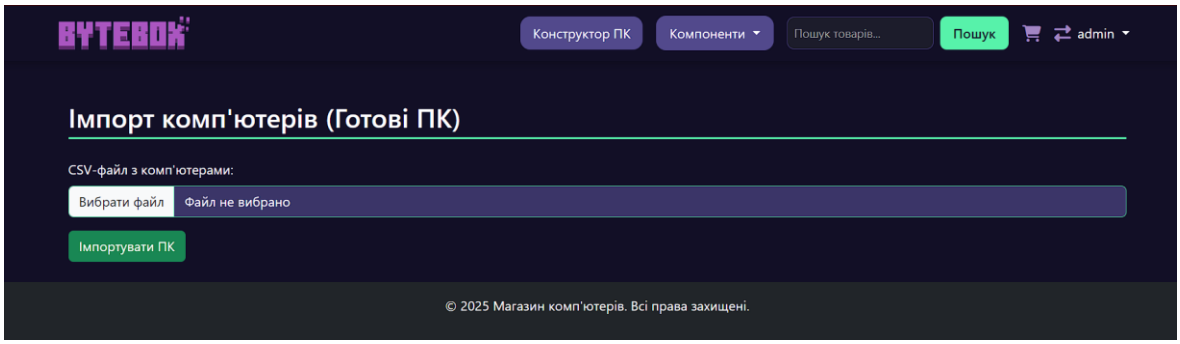
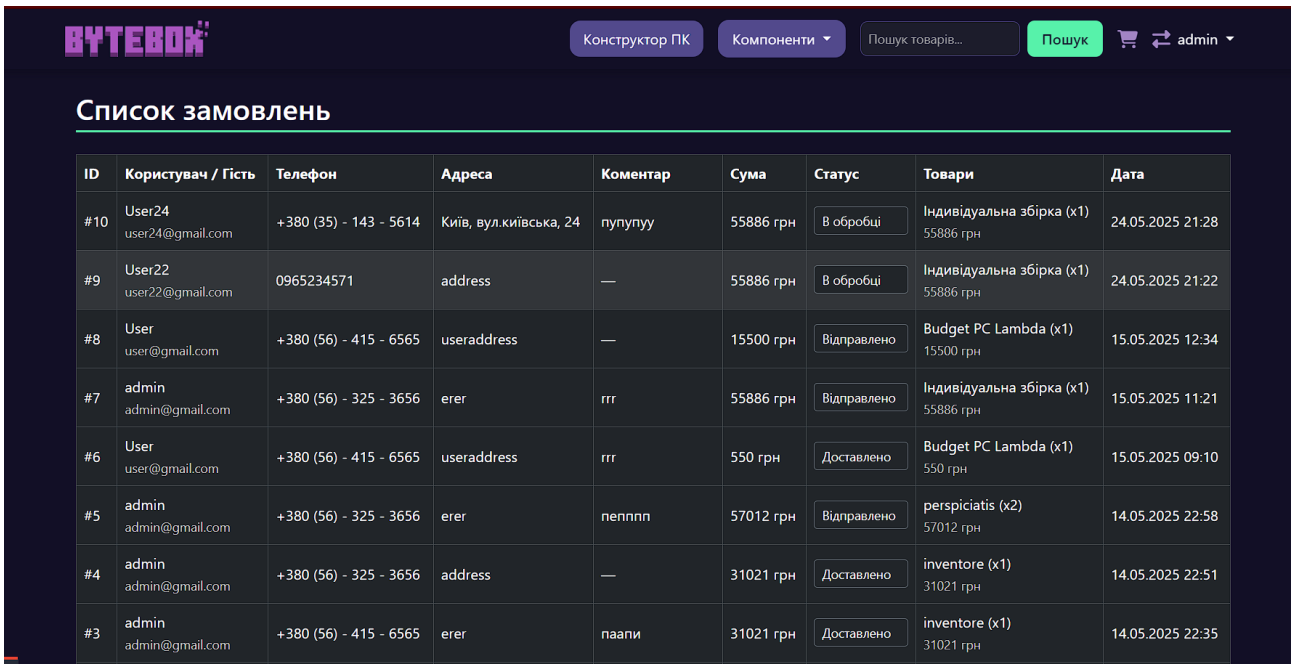


Рис.3.19 Приклад сторінки імпорту готових збірок ByteBox

Керування замовленнями: адміністратор має доступ до списку замовлень користувачів із можливістю перегляду деталей, зміни статусу та обробки кожного замовлення.



ID	Користувач / Гість	Телефон	Адреса	Коментар	Сума	Статус	Товари	Дата
#10	User24 user24@gmail.com	+380 (35) - 143 - 5614	Київ, вул.київська, 24	пупупуу	55886 грн	В обробці	Індивідуальна збірка (x1) 55886 грн	24.05.2025 21:28
#9	User22 user22@gmail.com	0965234571	address	—	55886 грн	В обробці	Індивідуальна збірка (x1) 55886 грн	24.05.2025 21:22
#8	User user@gmail.com	+380 (56) - 415 - 6565	useraddress	—	15500 грн	Відправлено	Budget PC Lambda (x1) 15500 грн	15.05.2025 12:34
#7	admin admin@gmail.com	+380 (56) - 325 - 3656	erer	rrr	55886 грн	Відправлено	Індивідуальна збірка (x1) 55886 грн	15.05.2025 11:21
#6	User user@gmail.com	+380 (56) - 415 - 6565	useraddress	rrr	550 грн	Доставлено	Budget PC Lambda (x1) 550 грн	15.05.2025 09:10
#5	admin admin@gmail.com	+380 (56) - 325 - 3656	erer	пеппп	57012 грн	Відправлено	perspiciatis (x2) 57012 грн	14.05.2025 22:58
#4	admin admin@gmail.com	+380 (56) - 325 - 3656	address	—	31021 грн	Доставлено	inventore (x1) 31021 грн	14.05.2025 22:51
#3	admin admin@gmail.com	+380 (56) - 415 - 6565	erer	паапи	31021 грн	Доставлено	inventore (x1) 31021 грн	14.05.2025 22:35

Рис.3.20 Приклад керування замовленнями ByteBox

Цей набір інтерфейсів створює повноцінне середовище для комфортної взаємодії користувача з web-застосунком, забезпечуючи легкість навігації та функціональність на кожному етапі.

3.3 Діаграма класів

З метою візуалізації архітектури програмної системи було розроблено діаграму класів, яка демонструє основні сутності web-застосунку, їхні атрибути та відношення між ними. Ця діаграма слугує логічною моделлю предметної області й дозволяє зрозуміти, як елементи системи взаємодіють між собою.

У центрі структури знаходиться модель User, яка уособлює зареєстрованого користувача. Кожен користувач може здійснювати багато замовлень, що реалізовано через зв’язок «один до багатьох» із моделлю Order. У свою чергу, замовлення складається з кількох позицій, представлених моделлю OrderItem. Кожен OrderItem пов'язаний з певним товаром (Item), що дозволяє зберігати інформацію про куплені продукти. За потреби, OrderItem може містити посилання на CustomBuild, якщо використовується функціонал конструктора ПК.

Модель Item є центральною для управління асортиментом товарів. Кожен товар може мати кілька зображень, які реалізовані через поліморфний зв’язок із моделлю Image (morphMany). Це дає змогу уніфіковано зберігати зображення як для товарів, так і потенційно для інших моделей, наприклад користувачів.

Також передбачено зв’язок між Item та User через поле user_id, що дозволяє визначати продавця або автора товару. Це відкриває можливість реалізації багаторольової системи, навіть без явного поділу в моделі користувача.

Окремі контролери — CabinetController, OrderController, ItemController, CompareController — взаємодіють із вказаними моделями через логіку та доступ до сесійних даних, зокрема при роботі з кошиком або списком порівняння. Вони

не мають прямих Eloquent-зв’язків, але використовуються для обробки бізнес-логіки, відображення інформації та взаємодії з користувачем.

Класи ImportItems та ImportPcParts відповідають за імпорт даних із зовнішніх джерел у модель Item і створення відповідних зображень (Image). Вони не є частиною безпосередньої взаємодії між моделями, але відіграють важливу роль у підтримці та оновленні бази даних.

3.4 Специфікація ключових класів та їх методів

У межах web-застосунку реалізовано низку класів-контролерів, які відповідають за основну бізнес-логіку проєкту. Особливу увагу зосереджено на реалізації функціоналу замовлень, управління користувацьким кабінетом, імпорту даних та перевірки сумісності компонентів ПК. Нижче подано характеристику ключових класів і методів, що реалізують відповідні функції.

OrderController — цей клас відповідає за логіку, пов’язану з формуванням замовлень, управлінням вмістом кошика та завершенням покупки:

– addToCart($request): додає обраний товар до кошика, враховуючи параметри, передані з форми (наприклад, кількість, конфігурації). Цей метод забезпечує швидке і безпомилкове додавання, що підвищує зручність користування

– removeFromCart($id): видаляє товар із кошика за унікальним ідентифікатором, дозволяючи користувачу коригувати свій вибір у будь-який момент

– updateCart($request): оновлює параметри товарів у кошику, включно зі зміною кількості або інших опцій, що підвищує гнучкість управління замовленням

– checkout():ініціює процедуру оформлення замовлення, перевіряючи коректність введених даних доставки, способу оплати і наявність підтвердження від користувача. Цей етап є ключовим для забезпечення надійності та зручності завершення покупки

– confirmOrder($request): після проходження всіх перевірок створює запис у таблиці замовлень, зберігає всі необхідні дані та очищує кошик, забезпечуючи цілісність та актуальність інформації

– statusIndex():виводить список замовлень користувача з відображенням актуальних статусів (обробляється, відправлено, доставлено тощо), що дозволяє контролювати процес виконання замовлень

– addSingleComponentToCart($componentId): спеціалізований метод для додавання одного окремого комплектуючого, що особливо корисно у випадках швидкого вибору деталей

– addCustomConfigToCart($config): реалізує можливість додати до кошика повністю кастомну збірку ПК, створену користувачем у конструкторі, що значно розширює функціональні можливості застосунку

CabinetController — цей клас забезпечує функціональність особистого кабінету користувача, що включає управління профілем, замовленнями, кошиком, порівняннями тощо:

– profile():відображає повну особисту інформацію користувача, забезпечуючи прозорість і комфорт у взаємодії з платформою

– edit():відкриває форму редагування профілю, дозволяючи користувачу швидко оновити свої дані

– updateProfile($request): приймає оновлені дані і зберігає їх у базу, підтримуючи актуальність інформації

– orders():виводить історію замовлень користувача з докладною інформацією про кожне з них, допомагаючи відстежувати покупки

– comparison():надає функціонал для перегляду і редагування списку товарів для порівняння, що особливо важливо для користувачів, які прагнуть зробити найкращий вибір

– cart():відображає поточний стан кошика, даючи змогу швидко побачити і змінити вибрані товари

– updateCart($request): дозволяє оновлювати вміст кошика безпосередньо з особистого кабінету, покращуючи інтерактивність

– editPassword():відкриває форму для зміни пароля, підвищуючи безпеку користувача

– updatePassword($request): зберігає новий пароль після проведення всебічної валідації

– showViewedProducts():демонструє історію переглянутих товарів, що допомагає користувачу швидко повернутися до раніше зацікавлених позицій

ImportItems — окремий клас-сервіс, який відповідає за імпорт даних про готові збірки з CSV-файлів до бази даних, а також обробку зображень:

– importCsv($path): зчитує CSV-файл, парсить отримані дані і створює відповідні записи у базі даних. Цей метод є основою для регулярного оновлення асортименту без зайвих витрат часу

– handle():головний метод, який координує весь процес імпорту: проводить перевірку вхідних даних, керує обробкою зображень, а також ініціалізує збереження в базу, забезпечуючи цілісність і коректність інформації

– castFields($data): перетворює типи даних у потрібні формати (наприклад, рядки у числа чи логічні значення), що мінімізує помилки при імпорті

– importImages($directory): виконує імпорт зображень комплектуючих із вказаної директорії, що підвищує візуальну привабливість каталогу і покращує користувацький досвід

– description():повертає короткий опис поточного імпорту, який можна використовувати для логування чи інформаційних повідомлень

– signature():генерує унікальний хеш або підпис для ідентифікації конкретного імпорту, що допомагає контролювати версії та відстежувати зміни.

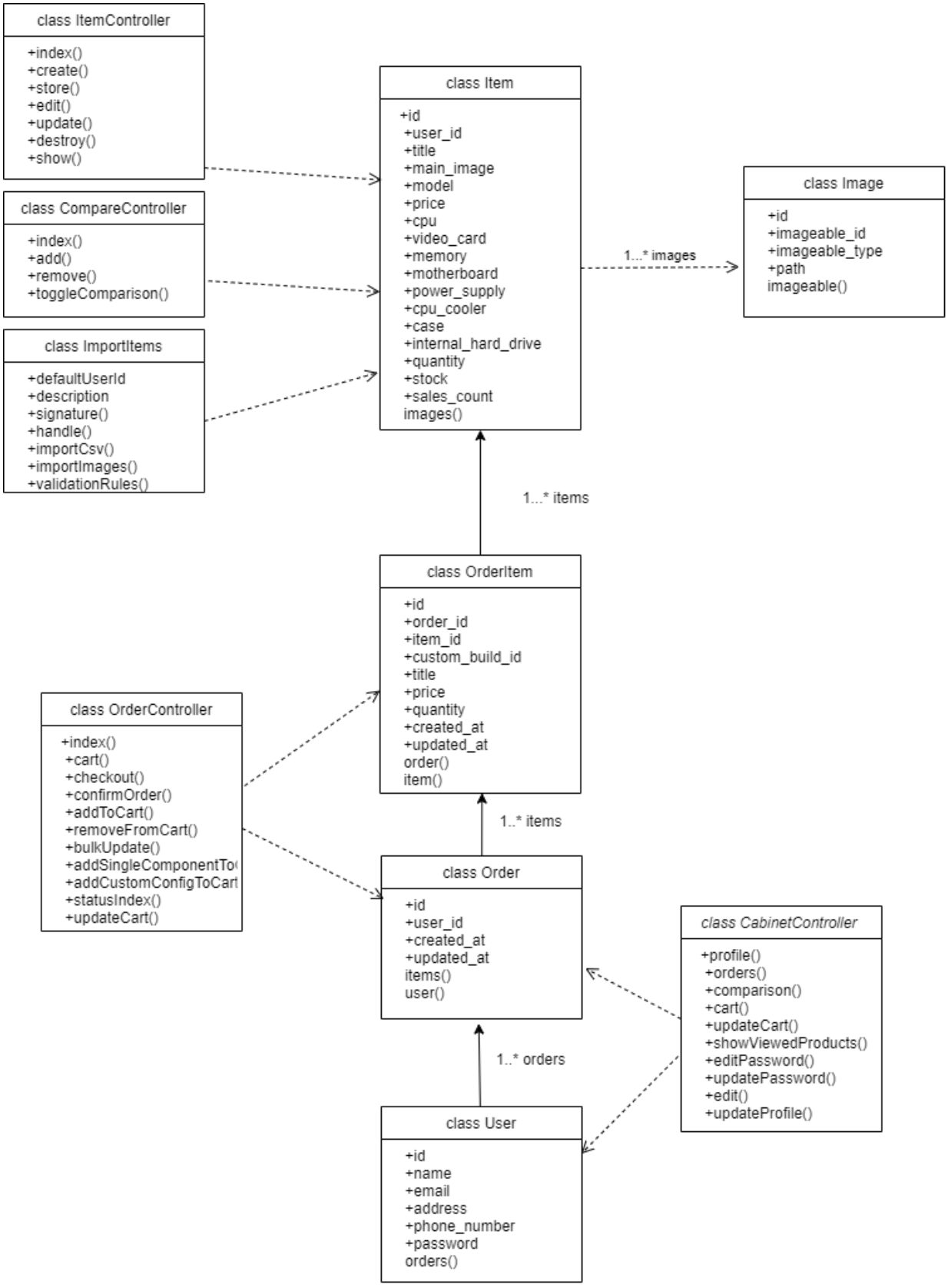


Рис.3.21 Діаграма класів

Завдяки чітко структурованим класам-контролерам та їх методам, web-застосунок забезпечує гнучке керування даними, зручну взаємодію користувача з системою, а також масштабованість і підтримку основної бізнес-логіки проєкту. Такий підхід дозволяє ефективно обробляти замовлення, налаштовувати персональний простір користувача та автоматизувати імпорт даних для підтримки актуального стану каталогу комплектуючих.

3.4 Реалізація перевірки сумісності комплектуючих ПК

Однією з ключових функціональних особливостей розробленого web-застосунку є повноцінна реалізація логіки перевірки сумісності комп’ютерних комплектуючих під час створення індивідуальної конфігурації системного блоку. Ця функція покликана забезпечити коректний підбір компонентів, попередити помилки, які можуть призвести до технічної несумісності, і значно спростити процес складання комп’ютера як для досвідчених користувачів, так і для новачків.

На відміну від традиційних конфігураторів, у цьому web-застосунку реалізовано можливість вибору компонентів у будь-якому порядку. Користувач може розпочати збірку з будь-якого елементу — процесора, відеокарти, оперативної пам’яті або навіть корпусу — і система автоматично аналізуватиме зроблений вибір, динамічно фільтруючи всі інші компоненти відповідно до поточних параметрів сумісності. Такий підхід забезпечує гнучкість і зручність, дозволяючи адаптуватися до індивідуальних пріоритетів користувача (наприклад, якщо він вже має певний компонент або орієнтується на відеокарту конкретного виробника).

Уся логіка сумісності побудована на детальному аналізі технічних характеристик кожного елементу. Наприклад, при виборі процесора система враховує тип сокета, що повинен відповідати сокету материнської плати. Також система фільтрує лише ті плати, які підтримують обраний процесор. У разі

вибору материнської плати першою — перелік доступних процесорів буде обмежено лише тими, що сумісні з її сокетом.

Кулер для процесора фільтрується за параметрами сокета, теплового пакета (TDP) і габаритів, щоб він точно підійшов до корпусу. Аналогічно, корпус фільтрується відповідно до форм-фактора материнської плати та розмірів відеокарти і кулера. Оперативна пам’ять підбирається за типом (наприклад, DDR4, DDR5), кількістю модулів і загальним обсягом, який не повинен перевищувати можливості плати. Відеокарта перевіряється на сумісність із інтерфейсом (PCIe), габаритами та енергоспоживанням, що, у свою чергу, впливає на вибір блоку живлення.

Жорсткі диски та SSD пристрої підбираються відповідно до наявних інтерфейсів (SATA, NVMe) та роз’ємів на материнській платі, а також до посадкових місць у корпусі. Блок живлення обирається залежно від загального енергоспоживання всієї системи, з урахуванням потужності відеокарти, процесора та інших компонентів, із необхідним запасом потужності. Також перевіряється форм-фактор блоку живлення щодо сумісності з корпусом.

Незалежно від послідовності вибору, система автоматично коригує перелік доступних опцій і не дозволяє зібрати несумісну конфігурацію. Такий підхід дозволяє користувачеві зберігати гнучкість при створенні системи, а також мінімізує ймовірність помилок. Уся логіка працює як на клієнтському рівні для зручності інтерфейсу, так і на серверному — для гарантії цілісності даних і захисту від некоректних запитів.

Завдяки такій архітектурі користувач отримує зрозумілий та надійний інструмент для створення персонального ПК з максимальною впевненістю в сумісності кожного обраного компонента.

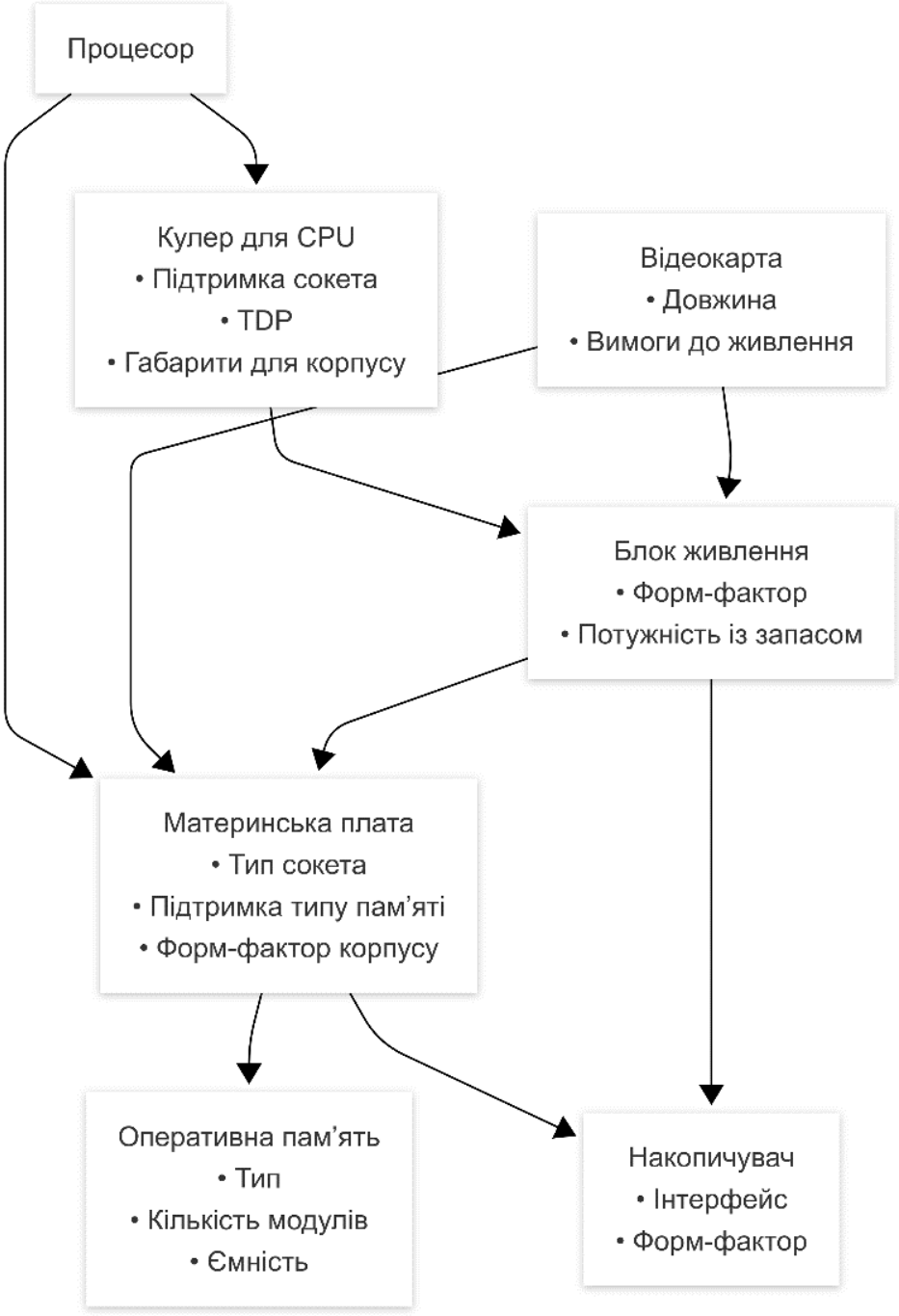


Рис.3.22 Логіка сумісності компонентів

Реалізація перевірки сумісності здійснюється на рівні фронтенду за допомогою компонента PcConfigurator.vue. Цей компонент відповідає за взаємодію користувача з інтерфейсом конструктора ПК та містить низку методів, які забезпечують гнучке керування конфігурацією:

– addToBuild(): додає вибраний компонент до поточної збірки з урахуванням його відповідності іншим компонентам;

– selectComponent(): обробляє вибір компонента з переліку доступних для певної категорії;

– filteredComponents(): повертає лише ті компоненти, які є сумісними з уже обраними

– showDependencyWarning(): активується у разі несумісності вибраного елементу з існуючими у збірці, виводячи відповідне попередження користувачу

– addCustomConfig(): формує власну конфігурацію на основі обраних компонентів

– clearBuild(): очищає поточну збірку

– toggleCompare(): дозволяє додавати або прибирати елементи зі списку порівняння

– totalPrice(): обчислює загальну вартість збірки в реальному часі.

Інтеграція цих методів дозволила створити зручну, інтерактивну систему підбору комп’ютера, що поєднує технічну коректність вибору з високим рівнем користувацького досвіду. Такий підхід забезпечує впевненість користувача у правильності конфігурації та сприяє підвищенню загальної якості взаємодії з web-застосунком.

4. ТЕСТУВАННЯ WEB-ЗАСТОСУНКУ

4.1 Unit-тестування

Unit-тестування є невід'ємною частиною процесу розробки web-застосунку, особливо коли йдеться про перевірку коректності роботи окремих модулів системи. У рамках дипломного проєкту unit-тестування було проведено з метою перевірки окремих компонентів застосунку, таких як сервіси, контролери, моделі та окремі логічні блоки.

Для реалізації unit-тестів використовувався вбудований інструментарій Laravel на базі PHPUnit [23]. Цей підхід дозволяє:

- автоматизувати перевірку основних функціональних елементів;
- швидко виявляти помилки в логіці додатку;
- гарантувати правильну поведінку компонентів навіть після внесення змін у код;
- підвищити стабільність і передбачуваність роботи web-застосунку загалом.

У тестах моделювалися типові та крайні сценарії використання функцій — наприклад, створення товару з коректними та некоректними даними, робота сервісів фільтрації або підбору сумісності комплектуючих, доступ користувачів до сторінок згідно з їхніми ролями тощо.

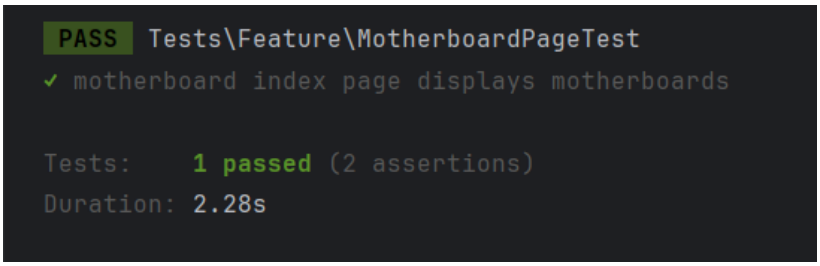


Рис.4.1 Результат Unit-тесту на коректний вивід даних

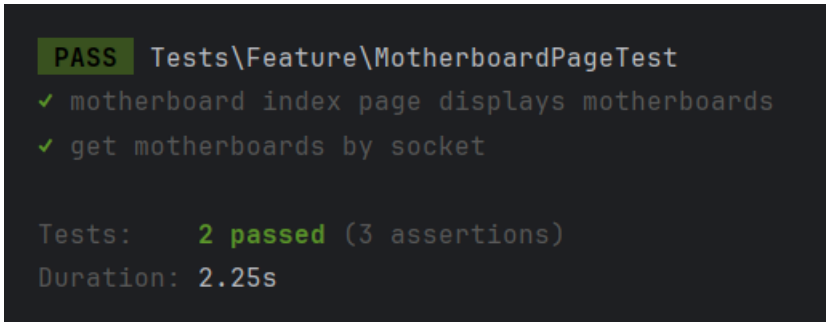


Рис.4.2 Результат Unit-тесту фільтрації

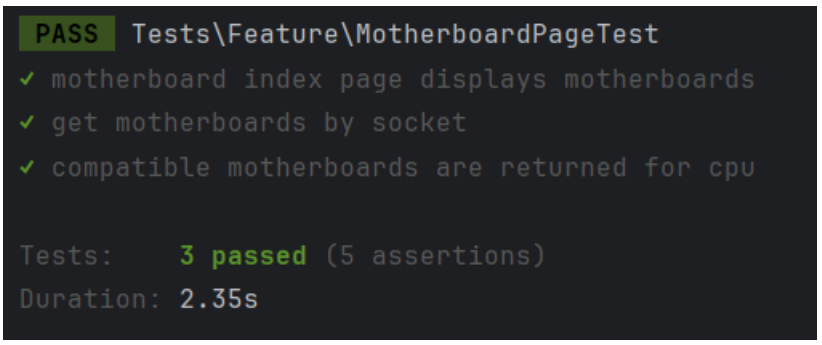


Рис.4.3 Результат Unit-тесту на сумісність компонентів

4.2 Тестування валідації даних

Важливою частиною перевірки є тестування валідації даних, яке здійснюється як у формах на стороні клієнта, так і на сервері. Це запобігає некоректному введенню та зберігає цілісність бази даних.

Важливою частиною перевірки є тестування валідації даних, яке здійснюється як на стороні клієнта, так і на стороні сервера. Такий підхід дозволяє забезпечити багаторівневий захист системи від введення некоректних або потенційно небезпечних даних.

На клієнтському рівні валідація реалізована засобами HTML5, а також через перевірки у JavaScript/ Vue.js. Вона забезпечує зручний користувацький досвід і попереджає помилки ще до надсилання форми на сервер. Наприклад, у формі реєстрації перевіряється, чи заповнені всі обов’язкові поля, чи правильний формат електронної адреси, чи збігаються паролі тощо.

На серверному боці валідація реалізована за допомогою стандартного механізму Laravel ($request->validate() або Form Request класів) [24]. Тут перевіряються такі аспекти, як:

- наявність обов’язкових полів (required);
- правильність формату даних (наприклад, email, url, numeric);
- унікальність записів (наприклад, перевірка, що електронна адреса вже не зареєстрована);
- відповідність даних наявним у базі (наприклад, існування category_id у таблиці категорій);
- мінімальні та максимальні значення, кількість символів тощо.

Це критично важливо для захисту web-застосунку, оскільки запити можна змінити вручну або з допомогою спеціальних інструментів, і лише серверна перевірка гарантує, що система не буде скомпрометована.

```php
protected function validator(array $data)
{
    return Validator::make($data, [
        'name' => ['required', 'string', 'max:255'],
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'password' => ['required', 'string', 'min:8', 'confirmed'],
        'address' => ['required', 'string', 'max:255'],
        'phone_number' => [
            'required',
            'regex:/^\+380 \(\d{2}\) - \d{3} - \d{4}$/'
        ],
    ], [
        'phone_number.regex' => 'Номер телефону має бути у форматі +380 (XX) - XXX - XXXX',
    ]);
}
```

Рис.4.3 Приклад коду валідації даних

У результаті тестування було переконливо доведено, що система коректно реагує на всі основні типи помилок введення. У разі виявлення некоректних даних користувачу відображається відповідне повідомлення, яке допомагає швидко виправити помилку.

Це дозволяє не лише підвищити зручність користування сайтом, але й забезпечити надійність, безпеку і стабільну роботу web-застосунку загалом.

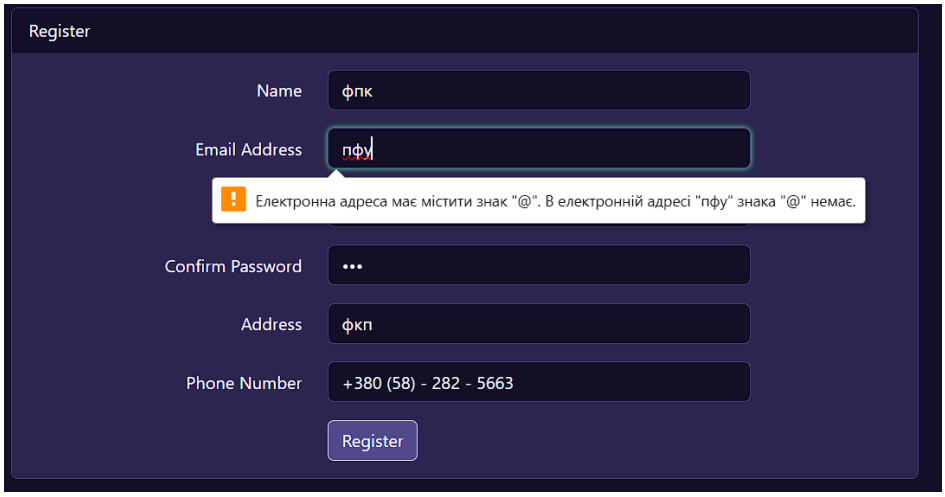


Рис.4.4 Результат валідації даних

4.3 Перевірка кросбраузерної сумісності

Для забезпечення доступності web-застосунку на різних платформах було проведено перевірку відображення інтерфейсу та роботи функціоналу в сучасних браузерах:

– Google Chrome (v.124);
– Mozilla Firefox (v.125);
– Microsoft Edge (v.124);
– Opera (v.109).

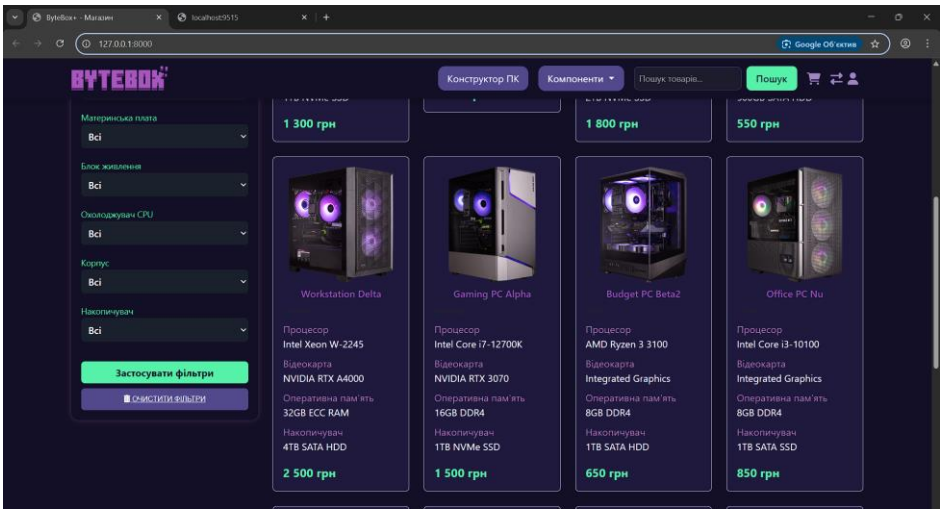


Рис.4.5 Приклад роботи web-додатку в Google Chrome

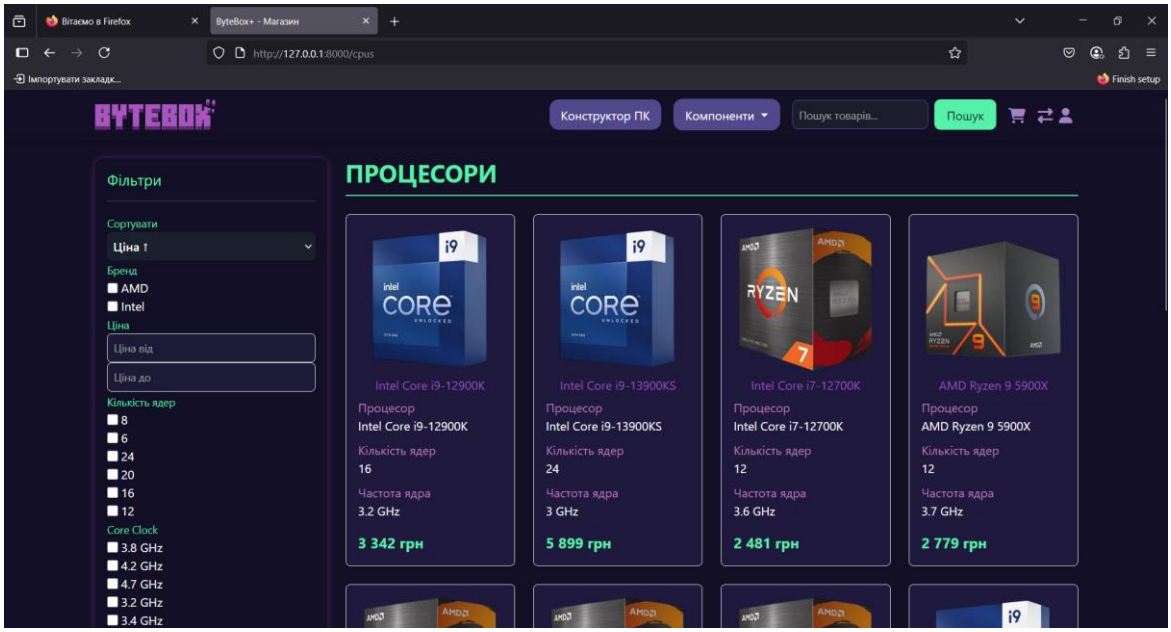


Рис.4.5 Приклад роботи web-додатку в Mozilla Firefox

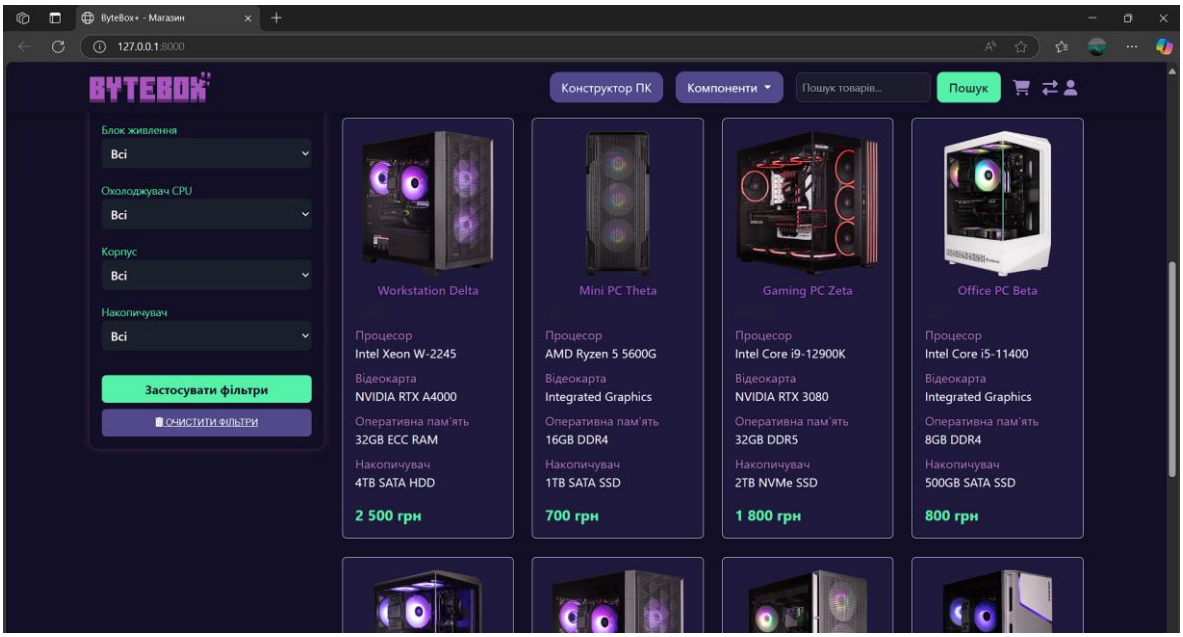


Рис.4.6 Приклад роботи web-додатку в Microsoft Edge

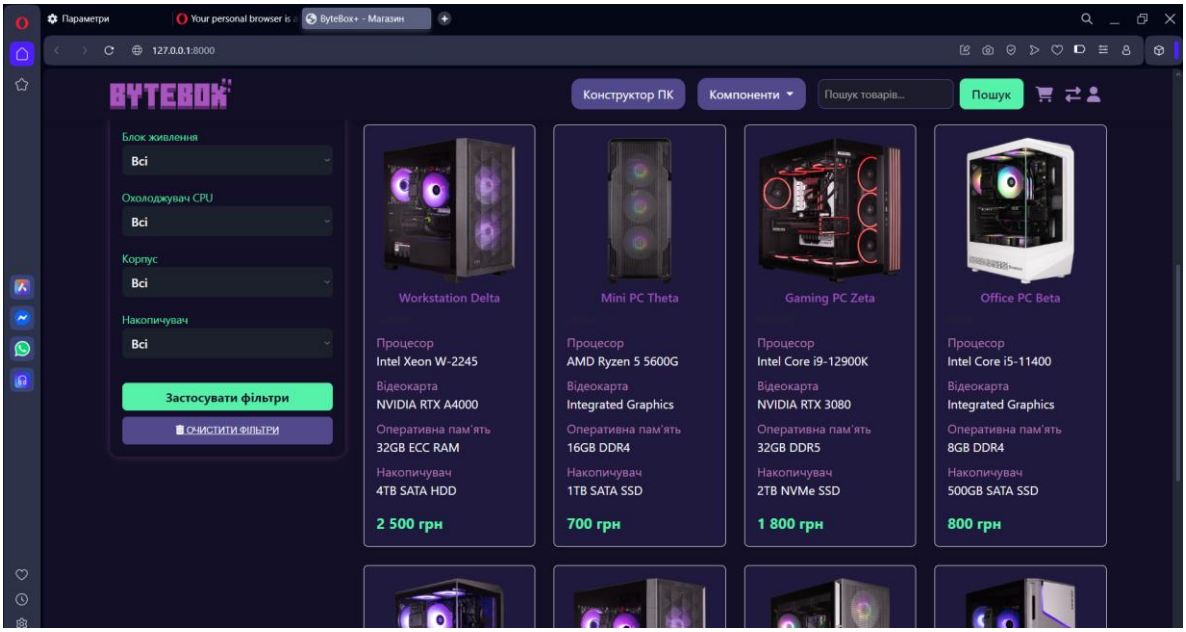


Рис.4.7 Приклад роботи web-додатку в Opera

ВИСНОВКИ

У ході виконання дипломної роботи було створено сучасний web-застосунок для підбору сумісних комплектуючих персонального комп’ютера та оформлення замовлень. Для реалізації було використано мову програмування PHP і фреймворк Laravel, що дозволило досягти високого рівня безпеки, масштабованості та гнучкості системи. Фронтенд реалізовано за допомогою Blade і Vue.js, а для збереження даних застосовано MySQL із використанням Eloquent ORM.

У результаті були виконані всі поставлені задачі:

1. Проведено аналіз предметної області — вивчено принципи сумісності комплектуючих, потреби цільової аудиторії та функціональні можливості існуючих сервісів. На основі аналізу сформульовано вимоги до майбутньої системи, окреслено її сильні сторони та конкурентні переваги.

2. Розроблено структуру бази даних — спроєктовано моделі сутностей, враховано зв’язки типу "один-до-багатьох", "багато-до-багатьох" та поліморфні відношення для зображень. Особливу увагу приділено модульності та безпечності архітектури.

3. Реалізовано механізм перевірки сумісності компонентів — користувач має змогу підбирати сумісні комплектуючі за заданими параметрами (наприклад, сокет процесора, тип пам’яті, роз’єм живлення тощо), що значно полегшує процес складання ПК.

4. Забезпечено імпорт товарів і збірок у систему — реалізовано обробку CSV-файлів, автоматичне створення відповідних моделей, прив’язку зображень та оновлення бази даних комплектуючих.

5. Проведено тестування застосунку — перевірено працездатність основних модулів, зокрема імпорту, перевірки сумісності, механізму порівняння, фільтрації, а також процесу оформлення замовлень.

У висновку, поставлені цілі досягнуто — створено зручну, функціональну й захищену web-систему, яка забезпечує повноцінний цикл взаємодії користувача з сайтом: від підбору сумісних компонентів до оформлення замовлення. Отриманий

результат може бути успішно використаний як основа для розширення сервісу або запуску реального онлайн-магазину.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Савченко Ю.В., Золотухіна О.А. Розробка веб-застосунку для підбору комплектуючих персонального копм’ютера // Матеріали VI Міжнародної науково-технічної конференції «Сучасний стан та перспективи розвитку IoT». Збірник тез. 15.04.2025, ДУІКТ, Київ, України (подано до друку).

2. Савченко Ю.В., Золотухіна О.А. Забезпечення безпеки даних у веб-даних засобами фреймворку Laravel // Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». Збірник тез. 24.04.2025, ДУІКТ, Київ, Україна, С. 419-420.

ПЕРЕЛІК ПОСИЛАНЬ

1. Huang T.E., Jones M. Designing scalable systems for PC component compatibility // Journal of Computer Systems. – 2020. – Т. 45, № 3. – С. 123–135.
2. Gonzalez J.E., Schmidt A., Lee K. Intelligent component selection algorithms for PC configuration systems // International Journal of Intelligent Systems. – 2021. – Т. 36, №. 4. – С. 412–427.
3. Can – Онлайн магазин комп’ютерної техніки. URL: https://can.ua (date of access: 20.05.2025).
4. ChipChip – Інтернет-магазин електроніки та комп’ютерної техніки. URL: https://chipchip.ua (date of access: 20.05.2025).
5. Click – Онлайн магазин електроніки та побутової техніки. URL: https://click.ua (date of access: 20.05.2025).
6. Compx – Інтернет-магазин комп’ютерної техніки. URL: https://compx.ua (date of access: 20.05.2025).
7. PhpStorm – IDE for PHP Developers by JetBrains. URL: https://www.jetbrains.com/phpstorm/ (date of access: 20.05.2025).
8. OpenServer – Локальний сервер для розробки. URL: https://ospanel.io/ (date of access: 20.05.2024).
9. Mann E. PHP Cookbook: Modern Code Solutions for Professional Developers. – Sebastopol: O'Reilly Media, 2014. – 600 p.
10. Freeman E., Robson E. Head First JavaScript Programming. – Sebastopol: O’Reilly Media, 2014. – 704 с.
11. Marini J. Document Object Model: Processing Structured Documents. – 1st ed. – McGraw-Hill/Osborne Media, 2022. – 400 с.
12. Stauffer M. Laravel: Up & Running: A Framework for Building Modern PHP Apps. – 2-е видання. – Sebastopol: O'Reilly, 2019. – 320 с.
13. Freeman A., Sanderson J. Professional ASP.NET MVC 5. – Birmingham: Wrox, 2021. – 624 с.

14. Malatesta F. Learning Laravel's Eloquent. – Birmingham: Packt Pub Ltd, 2023. – 202 c.

15. Introduction to Middleware: Web Services, Object Components, and Cloud Computing. – New York: Wiley, 2024. – 320 p.

16. Laravel Security Features – Laravel Documentation. URL: https://laravel.com/docs/11.x/security (date of access: 20.05.2025).

17. Hanchett E., Listwon B. Vue.js in Action. – Shelter Island: Manning Publications, 2018. – 320 c.

18. Blade templates – Laravel 11 documentation, URL: https://laravel.com/docs/11.x/blade (date of access: 20.05.2024).

19. Dyer R. Learning MySQL and MariaDB. – Birmingham: Packt Publishing, 2022. – 480 c.

20. phpMyAdmin – Official website. URL: https://www.phpmyadmin.net/ (date of access: 20.05.2025).

21. Composer – Dependency Manager for PHP. URL: https://getcomposer.org/ (date of access: 20.05.2025).

22. Bruno A. New Public Management (NPM) and the Introduction of an Accrual Accounting System: A Case Study of an Italian Regional Government Authority. – Rome: Academic Press, 2020. – 250 c.

23. Lively M. Instant Hands-on Testing with PHPUnit How-to. – Birmingham: Packt Publishing, 2018. – 150 p.

24. Laravel Validation – Official Documentation. URL: https://laravel.com/docs/11.x/validation (date of access: 20.05.2025).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка web-застосунку для підбору комплектуючих персонального комп'ютера

Виконала студентка 4 курсу
Групи ПД-41
Савченко Юлія Вячеславівна
Керівник роботи
к.т.н., доц., доцент кафедри ІПЗ Золотухіна Оксана Анатоліївна

Київ – 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – спрощення процесу підбору комплектуючих персонального комп'ютера.

Об'єкт дослідження – процес підбору комплектуючих персонального комп'ютера з урахуванням вимог користувача.

Предмет дослідження – розробка логіки сумісності компонентів персонального комп'ютера та її інтеграція у функціонал веб-застосунку.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести порівняльний аналіз існуючих онлайн-сервісів для підбору ПК-компонентів з метою виявлення переваг і недоліків.
2. Розробити структуру бази даних для збереження інформації про компоненти ПК.
3. Реалізувати механізм перевірки сумісності компонентів на основі введених користувачем даних.
4. Забезпечити імпорт компонентів та готових збірок у систему.
5. Провести тестування розробленого веб-застосунку.

3

АНАЛІЗ АНАЛОГІВ

Показник	can.ua	chipchip.ua	click.ua	compx.ua	**ByteBox**
Перевірка залишкової потужності БЖ	-	-	+	+	+
Динамічне оновлення доступних компонентів	-	-	-	+	+
Порівняння компонентів у конструкторі	-	-	-	-	+
Особистий кабінет користувача з історією переглянутих товарів	-	+	+	+	+
Фільтрація товарів за характеристиками	+	+	+	+	+
Темний інтерфейс, що знижує навантаження на очі	-	-	-	-	+

4

ВИМОГИ ДО WEB-ЗАСТОСУНКУ

Функціональні вимоги:

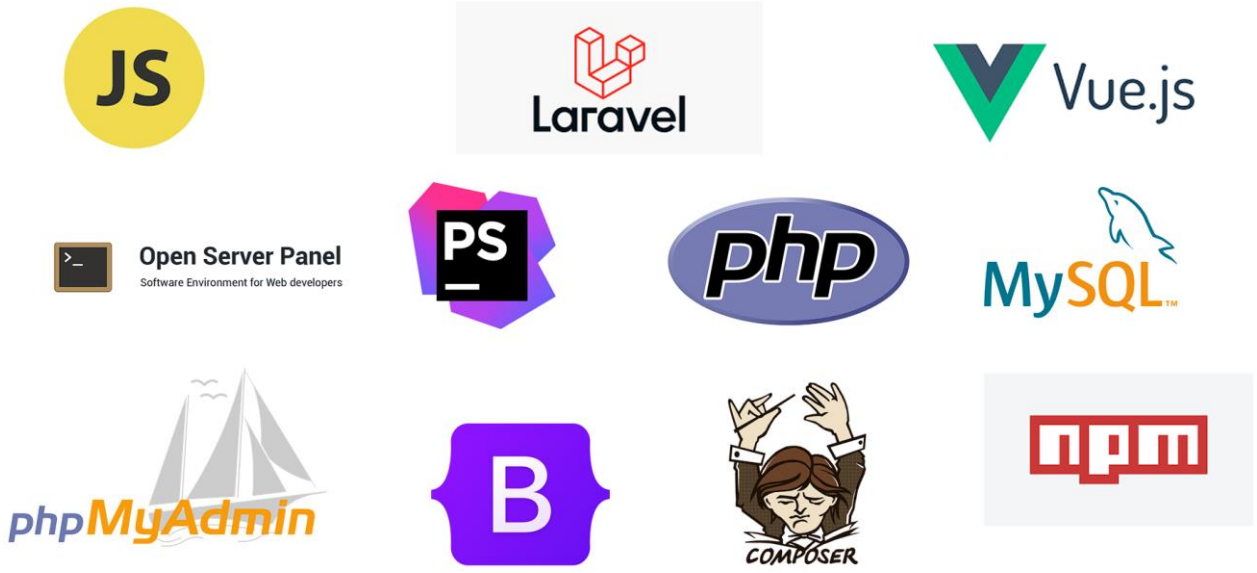
- Реалізація конструктора ПК для підбору сумісних комплектуючих.
- Забезпечення можливості порівняння характеристик комплектуючих.
- Реєстрація та авторизація користувачів для доступу до персоналізованих даних.
- Реалізація перегляду історії замовлень у профілі користувача для зручності повторних покупок.
- Можливість пошуку та фільтрації комплектуючих за різними параметрами.

Нефункціональні вимоги:

- Персональні дані користувачів повинні бути захищені за допомогою CSRF-токенів, хешування паролів і middleware контролю доступу.
- Виконати сайт у темних тонах для зниження навантаження на очі та забезпечення комфортної роботи користувачів.
- Забезпечити імпорт та оновлення комплектуючих і збірок без значних затримок для адміністратора.
- Система має бути адаптована для коректного відображення для таких браузерів як FireFox, Edge, Opera, Chrome.

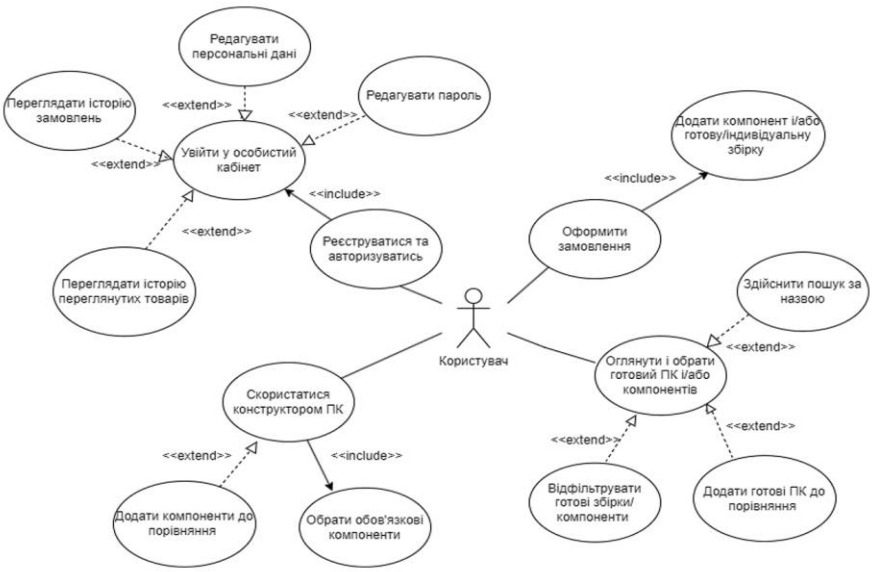
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



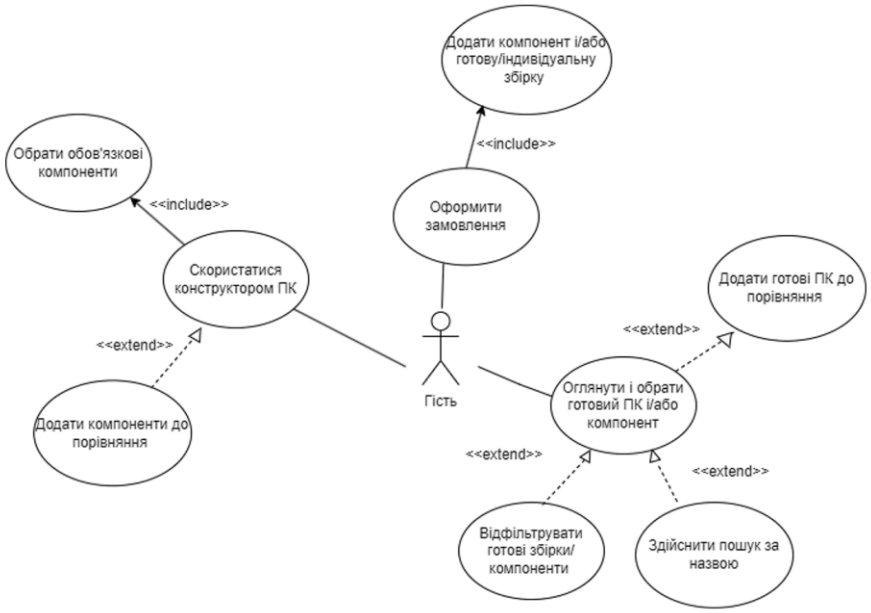
6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ КОРИСТУВАЧЕМ



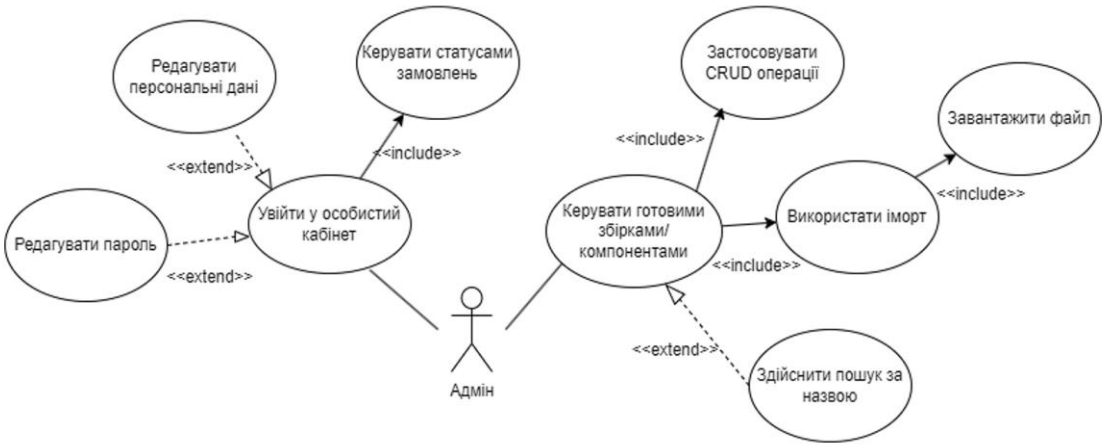
7

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ ГОСТЕМ



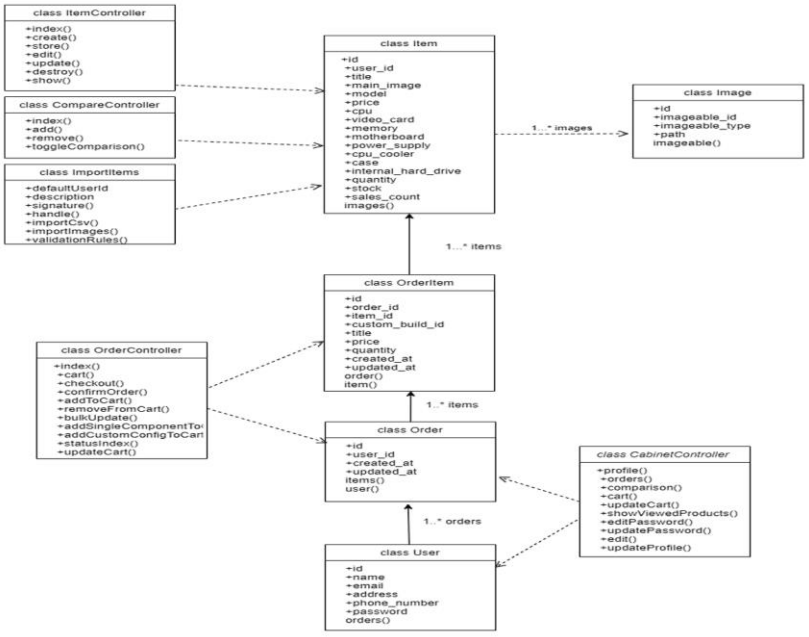
8

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ АДМІНОМ



9

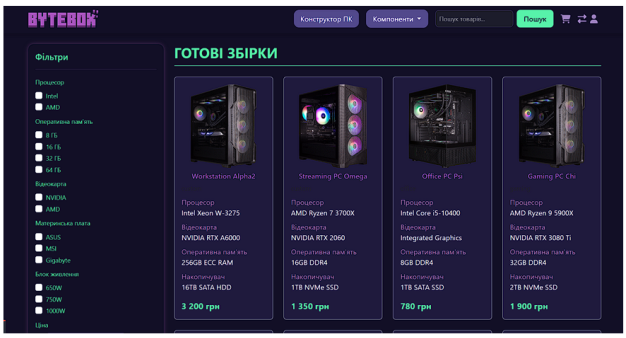
Діаграма класів системи



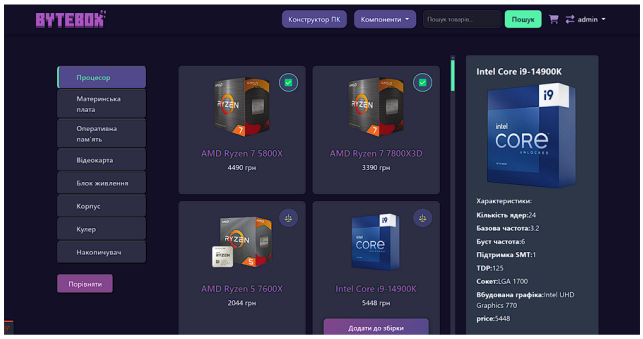
10

Діаграма діяльності процесу підбору комплектуючих

ЕКРАННІ ФОРМИ

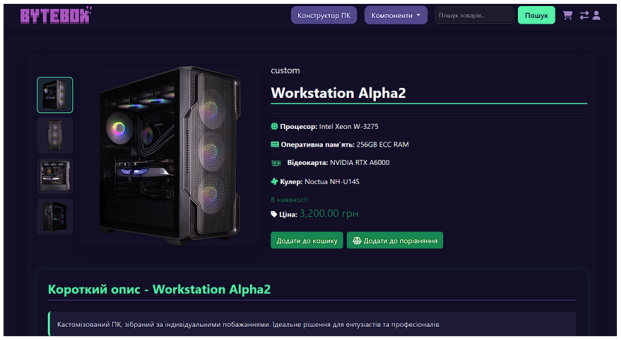


Головна сторінка

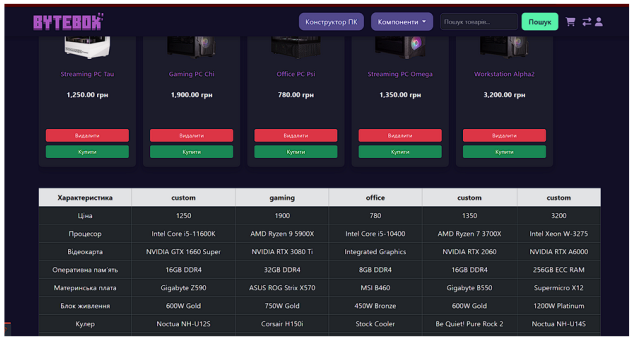


Сторінка конструктора ПК

ЕКРАННІ ФОРМИ



Сторінка детальніше про готову збірку



Сторінка порівняння готових збірок

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Савченко Ю.В., Золотухіна О.А. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ ПІДБОРУ КОМПЛЕКТУЮЧИХ ПЕРСОНАЛЬНОГО КОМП'ЮТЕРА. // Матеріали VI Міжнародної науково-технічної конференції «Сучасний стан та перспективи розвитку IoT». 15.04.2025, ДУІКТ, Київ, України, (подано до друку).

Савченко Ю.В., Золотухіна О.А. ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ДАНИХ У ВЕБ-МАГАЗИНІ ЗАСОБАМИ ФРЕЙМВОРКУ LARAVEL. // Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 24.04.2025, ДУІКТ, Київ, Україна, С. 419-420.

ВИСНОВКИ

1. Проведено порівняльний аналіз існуючих онлайн-сервісів для підбору ПК-компонентів, що допомогло виявити їхні переваги та недоліки.
2. Розроблено структуру бази даних для збереження інформації про компоненти ПК, що забезпечує ефективну організацію даних.
3. Реалізовано механізм перевірки сумісності компонентів на основі введених користувачем даних, що підвищує точність підбору комплектуючих.
4. Забезпечено імпорт компонентів та готових збірок у систему для автоматичного оновлення інформації.
5. Проведено тестування розробленого веб-застосунку.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Вихідний код модуля PcConfigurator

```
import { ref, reactive, computed } from 'vue'
import {
  isGpuCompatibleWithCase,
  isGpuCompatibleWithPsu,
  isPsuCompatibleWithCase,
  isCpuCompatibleWithMotherboard,
  isMotherboardCompatibleWithCase,
  isCoolerCompatibleWithCase,
  isDriveCompatibleWithMotherboard,
  isDriveCompatibleWithCase,
  isCoolerCompatibleWithCpu,
  isPsuWattageSufficient,
  isMemoryCompatibleWithMotherboard
} from '@/compatibilityRules'
const props = window.configuratorData
console.log('props', props)
const categories = [
  'cpus', 'motherboards', 'memory',
  'video_cards', 'power_supplies', 'cases',
'cpu_coolers','internal_hard_drives'
]
const psuPowerInfo = computed(() => {
  if (!selected.power_supplies) return null;
  const psu = selected.power_supplies;
  const components = {
    cpu: selected.cpus,
    gpu: selected.video_cards,
    memory: selected.memory,
    internal_hard_drives: selected.internal_hard_drives || []
  };
  console.log('CPU:', selected.cpus);
  console.log('GPU:', selected.video_cards);
  console.log('CPU TDP:', selected.cpu?.tdp);
  console.log('GPU Wattage:',
selected.video_cards?.wattage);
  console.log('drive wattage:',
selected.internal_hard_drives?.wattage);
  return isPsuWattageSufficient(psu, components);
});
const categoryLabels = {
  cpus: 'Процесор',
  motherboards: 'Материнська плата',
  memory: 'Оперативна пам’ять',
  video_cards: 'Відеокарта',
  power_supplies: 'Блок живлення',
  internal_hard_drives: 'Накопичувач',
  cases: 'Корпус',
  wireless_network_cards: 'Мережевий адаптер',
  internal_hard_drive: 'Накопичувач',
  cpu_coolers: 'Кулер'
}
const activeCategory = ref('cpus')
const activeComponentId = ref(null)
const selected = reactive({})
const showNotification = ref(false)
const notificationMessage = ref('')
const notificationType = ref('')
const prevSelected = ref(null)
const prevActiveComponentId = ref(null)
const notificationTypeClass = computed(() => ({
  success: 'notification-success',
  error: 'notification-error',
  warning: 'notification-warning'
}[notificationType.value] || 'notification-info'))
const notify = (message, type = 'info') => {
  notificationMessage.value = message
  notificationType.value = type
  showNotification.value = true
  setTimeout(() => (showNotification.value = false), 3000)
}
const getStaticImage = (category) => {
  const staticImages = {
    cpus: 'storage/icons/gpu.png',
    motherboards: 'storage/icons/motherboard.png',
    memory: 'storage/icons/ram.png',
    video_cards: 'storage/icons/video_card.png',
    power_supplies: 'storage/icons/psu.png',
    cases: 'storage/icons/computer-tower.png',
    cpu_coolers: 'storage/icons/fan.png',
  };
  return staticImages[category] || 'storage/icons/gpu.png';
};
const setActiveComponent = (item) => {
  console.log('Компонент вибрано:', item.name);
  activeComponentId.value = item.id;
  updateComponentDetails();
};
const undoAction = ref(null)

const notifyWithUndo = (message, type = 'info') => {
  showNotification.value = true
  notificationMessage.value = message
  notificationType.value = type
  undoAction.value = () => {
    Object.assign(selected, prevSelected.value)
    activeComponentId.value =
prevActiveComponentId.value
  }
  setTimeout(() => {
    showNotification.value = false
    undoAction.value = null
  }, 5000)
}
const selectComponent = (category, item) => {
  selected[category] = item
  activeCategory.value = category
  activeComponentId.value = item.id
}
const updateComponentDetails = () => {
  console.log('updateComponentDetails викликано');
};
const activeComponent = computed(() =>
  filteredComponents.value.find(c => c.id ===
activeComponentId.value) || null
);
const addToBuild = (item) => {
```

```
  console.log('Клік по компоненту:', item.name)
  console.log('Компонент вибрано:', item.name);
  const category = activeCategory.value
  prevSelected.value = JSON.parse(JSON.stringify(selected))
  prevActiveComponentId.value = activeComponentId.value

  const incompatibilities = {
    cpu: () => selected.motherboard && !isCpuCompatibleWithMotherboard(item, selected.motherboard),
    motherboard: () => selected.cpu && !isCpuCompatibleWithMotherboard(selected.cpu, item),
    cpu_cooler: () => selected.cpu && !isCoolerCompatibleWithCpu(selected.cpu, item),
    case: () => (
      (selected.video_card && !isGpuCompatibleWithCase(selected.video_card, item)) ||
      (selected.power_supply && !isPsuCompatibleWithCase(selected.power_supply, item))
    ),
    power_supply: () => (
      (selected.video_card && !isGpuCompatibleWithPsu(selected.video_card, item)) ||
      (selected.case && !isPsuCompatibleWithCase(item, selected.case)) ||
      !isPsuWattageSufficient(item, {
        cpu: selected.cpu,
        gpu: selected.video_card,
        memory: selected.memory,
        internal_hard_drives: selected.internal_hard_drives ? [selected.internal_hard_drives] : []
      })
    ),
    memory: () => selected.motherboard && !isMemoryCompatibleWithMotherboard(selected.motherboard, item),
  }
  if (incompatibilities[category] && incompatibilities[category]()) {
    selected[category] = null
    activeComponentId.value = null


    if (!selected[activeCategory.value]) {
      activeComponentId.value = null
    }
    notifyWithUndo('Компонент несумісний. Несумісний компонент було скинуто.', 'warning')
    return
  }
  selected[activeCategory.value] = item;
  activeComponentId.value = item.id;
  updateComponentDetails();
  notify('Компонент успішно додано до збірки!', 'success');
};
const toggleBuild = (item) => {
  const category = activeCategory.value;
  if (selected[category]?.id === item.id) {
    delete selected[category];
    activeComponentId.value = null;
    notify('Компонент видалено зі збірки.', 'info');
    return;
  }
  addToBuild(item);
};
const clearBuild = () => {
  Object.keys(selected).forEach(key => delete selected[key]);
  notify('Збірку очищено.', 'info');
};
const comparison = reactive(loadComparison())
const showModal = ref(false)
const openModal = () => {
  showModal.value = true
}
const closeModal = () => {
  showModal.value = false
}
function loadComparison() {
  try {
    const stored = localStorage.getItem('comparison')
    return stored ? JSON.parse(stored) : createEmptyComparison()
  } catch {
    return createEmptyComparison()
  }
}
const totalPrice = computed(() => {
  return Object.values(selected).reduce((sum, item) => {
    return sum + (item?.price || 0)
  }, 0)
})
function createEmptyComparison() {
  return Object.fromEntries(categories.map(c => [c, []]))
}
function saveComparison() {
  localStorage.setItem('comparison', JSON.stringify(comparison))
}
const toggleCompare = (category, item) => {
  const list = comparison[category]
  const index = list.findIndex(i => i.id === item.id)
  if (index !== -1) {
    list.splice(index, 1)
  } else {
    if (list.length >= 5) {
      notify('⚠ Можна порівнювати лише до 5 компонентів', 'warning')
      return
    }
    list.push(item)
  }
  saveComparison()
}
const isInComparison = (category, item) => {
  return comparison[category]?.some(i => i.id === item.id)
}
const getCategoryIcon = (category) => {
  const icons = {
    cpus: '/icons/gpu.png',
    motherboards: '/icons/gpu.png',
    memory: '/icons/gpu.png',
  };
  return icons[category] || 'public/icons/gpu.png';
};
const filters = {
  motherboards: () => {
    let result = props.motherboards;
    if (selected.cpu) result = result.filter(mb => isCpuCompatibleWithMotherboard(selected.cpu, mb));
    if (selected.memory) result = result.filter(mb => isMemoryCompatibleWithMotherboard(mb, selected.memory));
    if (selected.case) result = result.filter(mb => isMotherboardCompatibleWithCase(mb, selected.case));
```

```
        if (selected.internal_hard_drive) result = result.filter(mb
=>
isDriveCompatibleWithMotherboard(selected.internal_hard_
drive, mb));
        return result;
    },
    cpus: () => {
        let result = props.cpus;
        if (selected.cpu_coolers) {
            result = result.filter(cpu =>
isCoolerCompatibleWithCpu(selected.cpu_coolers, cpu));
        }
        if (selected.motherboards) {
            result = result.filter(cpu =>
isCpuCompatibleWithMotherboard(cpu,
selected.motherboards));
        }
        return result;
    },
    cpu_coolers: () => {
        let result = props.cpu_coolers;
        if (selected.cpus) {
            result = result.filter(cooler =>
isCoolerCompatibleWithCpu(cooler, selected.cpus));
        }
        if (selected.cases) {
            result = result.filter(cooler =>
isCoolerCompatibleWithCase(cooler, selected.cases));
        }
        return result;
    },
    memory: () => {
        return selected.motherboards ? props.memory.filter(mem
=>
isMemoryCompatibleWithMotherboard(selected.motherboar
ds, mem)) : props.memory
    },
        cases: () => {
            let result = props.cases;
            if (selected.video_card) {
                result = result.filter(c =>
isGpuCompatibleWithCase(selected.video_card, c));
            }

            if (selected.power_supply) {
                result = result.filter(c =>
isPsuCompatibleWithCase(selected.power_supply, c));
            }
            if (selected.cpu_cooler) {
                result = result.filter(c =>
isCoolerCompatibleWithCase(selected.cpu_cooler, c));
            }
            if (selected.motherboard) {
                result = result.filter(c =>
isMotherboardCompatibleWithCase(selected.motherboard,
c));
            }
            if (selected.internal_hard_drive) {
                result = result.filter(c =>
isDriveCompatibleWithCase(selected.internal_hard_drive,
c));
            }
            return result;
        },
    video_cards: () => {
        return selected.cases ? props.video_cards.filter(gpu =>
isGpuCompatibleWithCase(gpu, selected.cases)) :
props.video_cards
    },
    power_supplies: () => {
        let result = props.power_supplies;
        if (selected.video_card) {
            result = result.filter(psu =>
isGpuCompatibleWithPsu(selected.video_card, psu));
        }
        if (selected.case) {
            result = result.filter(psu =>
isPsuCompatibleWithCase(psu, selected.case));
        }
        result = result.filter(psu =>
            isPsuWattageSufficient(psu, {
                cpu: selected.cpu,
                gpu: selected.video_card,
                memory: selected.memory,
                internal_hard_drives: selected.internal_hard_drive
            })
        );
        return result;
    },
    internal_hard_drives: () => {
        let result = props.internal_hard_drives;
        if (selected.motherboard) {
            result = result.filter(d =>
isDriveCompatibleWithMotherboard(d,
selected.motherboard));
        }
        if (selected.case) {
            result = result.filter(d =>
isDriveCompatibleWithCase(d, selected.case));
        }

        return result;
    },
}
const filteredComponents = computed(() => {
    return filters[activeCategory.value]?.() || []
})
const dependencyWarning = computed(() => {
    if (activeCategory.value === 'memory' &&
!selected.motherboards) {
        return '! Виберіть материнську плату, щоб бачити
сумісну пам’ять.'
    }
    if (activeCategory.value === 'cpu_coolers' &&
!selected.cpus) {
        return '! Виберіть процесор, щоб бачити сумісні
кулери.'
    }
    if (activeCategory.value === 'cases' &&
!selected.video_cards) {
        return '! Виберіть відеокарту, щоб бачити сумісні
корпуси.'
    }
    return null
})
const showDependencyWarning = computed(() =>
Boolean(dependencyWarning.value))
const addToCart = async (item) => {
    try {
        const response = await fetch('/cart/add-component', {
            method: 'POST',
            headers: {
                'X-CSRF-TOKEN':
document.querySelector('meta[name="csrf-token"]').content,
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({component: {id: item.id, name:
item.name, price: item.price}}),
```

```
        })
        const data = await response.json()
        if (data.success) {
            notify(` ${item.name} додано до кошика`, 'success')
        } else {
            notify(` ${item.name} не вдалося додати до
кошика`, 'error')
        }
    } catch {
        notify(' Помилка ', 'error')
    }
}

const requiredComponents = [
    'cpus',
    'motherboards',
    'memory',
    'video_cards',
    'power_supplies',
    'cases',
    'internal_hard_drives'];


const addCustomConfig = async () => {
    for (const component of requiredComponents) {
        if (!selected[component]) {
            const componentLabel = categoryLabels[component] ||
component;
            notify(`⚠ Необхідно вибрати ${componentLabel}`,
'warning');
            return;
        }
    }

    if (!Object.keys(selected).length) {
        notify("⚠ Спочатку оберіть хоча б один компонент",
'warning');
        return;
    }

    try {
        const response = await fetch('/cart/add-custom-config', {
            method: 'POST',
            headers: {
                'X-CSRF-TOKEN':
document.querySelector('meta[name="csrf-token"]').content,
                'Content-Type': 'application/json',
            },
            body:                          JSON.stringify({config:
Object.values(selected)}),
        });
        const data = await response.json();
        if (data.success) {
            notify(`${data.success}`, 'success');
        } else {
            notify('Помилка при додаванні', 'error');
        }
    } catch {
        notify('Помилка', 'error');
    }
};

const clearComparison = (category) => {
    comparison[category] = [];
    notify('Порівняння очищено.', 'info');
    showModal.value = false;
}; const fieldsPerCategory = {
    cpus: [
        'core_count',
        'core_clock',
        'boost_clock',
        'smt',
        'tdp',
        'socket',
        'graphics',
        'price',
    ],
    motherboards: [
        'socket',
        'chipset',
        'memory_type',
        'max_memory',
        'memory_slots',
        'form_factor',
        'price',
        'color',
    ],
    memory: [
        'type',
        'capacity',
        'speed',
        'cas_latency',
        'first_word_latency',
        'modules',
        'voltage',
        'price',
        'price_per_gb',
        'color',
    ],
    video_cards: [
        'chipset',
        'memory',
        'boost_clock',
        'core_clock',
        'power_connectors',
        'length',
        'interface',
        'price',
        'color',
        'wattage',
    ],
    power_supplies: [
        'wattage',
        'efficiency',
        'pcie_connectors',
        'modular',
        'form_factor',
        'type',
        'price',
        'color',
    ],
    cases: [
        'supported_motherboard_form_factors',
        'gpu_max_length',
        'cooler_max_height',
        'type',
        'external_volume',
        'side_panel',
        'internal_35_bays',
        'psu',
        'price',
        'color',
    ],
    cpu_coolers: [
        'tdp_support',
        'supported_sockets',
        'height',
        'noise_level',
```

```
    'rpm',
    'size',
    'price',
    'color',
  ],
  wireless_network_cards: [
    'protocol',
    'max_speed_mbps',
    'interface',
    'bluetooth_support',
    'form_factor',
    'price',
    'color',
    'name',
  ],
  internal_hard_drives: [
    'capacity',
    'type',
    'interface',
    'read_speed',
    'write_speed',
    'speed',
    'cache',
    'form_factor',
    'price_per_gb',
    'price',
    'name',
    'wattage',
  ],
}


const fieldsForCategory = (category) => fieldsPerCategory[category] || []
const fieldLabels = {
  cpus: {
    price: 'Ціна',
    core_count: 'Кількість ядер',
    core_clock: 'Базова частота',
    boost_clock: 'Частота Boost',
    tdp: 'TDP',
    graphics: 'Вбудована графіка',
    smt: 'Підтримка SMT',
    socket: 'Сокет'
  },
  motherboards: {
    price: 'Ціна',
    socket: 'Сокет',
    form_factor: 'Форм-фактор',
    max_memory: 'Максимальна пам’ять',
    memory_slots: 'Слоти пам’яті',
    color: 'Колір',
    chipset: 'Чипсет',
    memory_type: 'Тип пам’яті'
  },
  memory: {
    price: 'Ціна',
    speed: 'Швидкість (МГц)',
    modules: 'Кількість модулів',
    price_per_gb: 'Ціна за ГБ',
    color: 'Колір',
    first_word_latency: 'Затримка першого слова',
    cas_latency: 'CAS Latency',
    type: 'Тип пам’яті',
    capacity: 'Об’єм',
    voltage: 'Напруга'
  },
  video_cards: {
    price: 'Ціна',
    chipset: 'Чипсет',
    memory: 'Об’єм пам’яті',
    core_clock: 'Базова частота ядра',
    boost_clock: 'Boost частота',
    color: 'Колір',
    length: 'Довжина (мм)',
    power_connectors: 'Конектори живлення',
    interface: 'Інтерфейс',
    wattage:'Потужність'
  },
  power_supplies: {
    price: 'Ціна',
    type: 'Тип',
    efficiency: 'Енергоефективність',
    wattage: 'Потужність (Вт)',
    modular: 'Модульність',
    color: 'Колір',
    pcie_connectors: 'PCIe конектори',
    form_factor: 'Форм-фактор'
  },
  cases: {
    price: 'Ціна',
    type: 'Тип корпуса',
    color: 'Колір',
    psu: 'Блок живлення',
    side_panel: 'Бічна панель',
    external_volume: 'Зовнішній об’єм',
    internal_35_bays: 'Внутрішні 3.5" відсіки',
    gpu_max_length: 'Макс. довжина GPU',
    cooler_max_height: 'Макс. висота кулера',
    supported_motherboard_form_factors: 'Підтримка форм-факторів материнських плат'
  },
  cpu_coolers: {
    price: 'Ціна',
    rpm: 'Оберти вентилятора (RPM)',
    noise_level: 'Рівень шуму (дБ)',
    color: 'Колір',
    size: 'Розмір',
    supported_sockets: 'Підтримувані сокети',
    tdp_support: 'Підтримка TDP',
    height: 'Висота (мм)'
  },
  wireless_network_cards: {
    name: 'Назва',
    price: 'Ціна',
    protocol: 'Протокол',
    interface: 'Інтерфейс',
    color: 'Колір',
    form_factor: 'Форм-фактор',
    bluetooth_support: 'Підтримка Bluetooth',
    max_speed_mbps: 'Макс. швидкість (Мбіт/с)'
  },
  internal_hard_drives: {
    name: 'Назва',
    price: 'Ціна',
    capacity: 'Об’єм',
    price_per_gb: 'Ціна за ГБ',
    type: 'Тип накопичувача',
    cache: 'Кеш-пам’ять',
    form_factor: 'Форм-фактор',
    interface: 'Інтерфейс',
    speed: 'Швидкість обертання (RPM)',
    read_speed: 'Швидкість читання (МБ/с)',
    write_speed: 'Швидкість запису (МБ/с)',
    wattage:'Потужність Вт'
  }
}
```