Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Програмне забезпечення віртуального конфігуратора комп'ютерів загального призначення
Назва теми

КвРКІ. 200124.01.23 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент IV курсу, група КІ2-20-1 _____ В. В. Швалюк
Підпис, Ініціали, прізвище

Керівник _____ М. О. Слободян
Підпис, дата, Ініціали, прізвище

Нормоконтролер _____ С. М. Лисенко
Підпис, дата, Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О. Говорущенко
Підпис, Ініціали, прізвище

«24» червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень БАКАЛАВР

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ
Зав. кафедри Т.О.Говорущенко

"10" 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Швалюк Вадим Володимирович
Прізвище, ім'я, по батькові студента

1. Тема проєкту (роботи) Програмне забезпечення віртуального конфігуратора комп'ютерів загального призначення

Керівник проєкту (роботи) Слободян М. О.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 15.02.2024 р. № 8

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2024 р.

3. Вихідні дані до проєкту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз предметної області згідно теми роботи

Проєктування програмного забезпечення віртуального конфігуратора комп`ютерів загального призначення

Реалізація та тестування програмного забезпечення віртуального конфігуратора комп'ютерів загального призначення

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Архітектура ПЗ проєкту 1

Архітектура ПЗ проєкту 2

Архітектура БД для проєкту

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийня
Нормоконтроль	Лисенко С. М., професор кафедри КІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІС		

7. Дата видачі завдання «_10_»_____01_____2024 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітк
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2024	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2024	виконан
3	Робота над розділом 1 – аналіз предметної області згідно теми роботи	01.03.2024	виконан
4	Робота над розділом 2 – проєктування програмного забезпечення віртуального конфігуратора комп'ютерів загального призначення	01.04.2024	виконан
5	Робота над розділом 3 – реалізація та тестування програмного забезпечення віртуального конфігуратора комп'ютерів загального призначення	29.04.2024	виконан
6	Оформлення пояснювальної записки згідно вимог	25.05.2024	виконан
7	Попередній захист ВКР	26.05.2024	виконан
8	Захист ВКР на засіданні ЕК	24.06.2024	

Студент _____ В. В. Швалюк
Підпис Ініціали, прізвище

Керівник роботи _____ О. М. Слободян
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ. 200124.01.23 ПЗ	Пояснювальна записка	56		
			Графічні матеріали			
2		КвРКІ. 200124.01.23 Е8	Архітектура ПЗ проєкту 1	1		Копія
						А2
3		КвРКІ. 200124.01.23 Е8	Архітектура ПЗ проєкту 2	1		Копія
						А2
4		КвРКІ. 200124.01.23 Е8	Архітектура БД для	1		Копія
			проєкту			А2

Зм	Арк	№ докум	Підпис	Дата	КвРКІ. 200124.01.23 ВР		
Розробив		Швалюк В.В.			Програмне забезпечення віртуального конфігуратора комп'ютерів загального призначення Відомість роботи	Літера	Аркуш	Аркушів
Перевір.		Слободян М.О				У	1	1
						ХНУ, КІ2-20-1		
Н. контр.		Лисенко С.М.						
Затв.		Говорущенко Т.О.		24.06				

АНОТАЦІЯ

Тема кваліфікаційної роботи: *«Програмне забезпечення віртуального конфігуратора комп'ютерів загального призначення».*

Автор роботи: *Швалюк Вадим Володимирович.*

Керівник роботи: *Слободян Максим Олегович.*

Пояснювальна записка: *56 с., 22 рис., 2 табл., 5 дод., 50 джерел.*

Графічна частина: *3 креслення.*

КОНФІГУРАТОР КОМП'ЮТЕРІВ, ІНФОРМАЦІЙНА СИСТЕМА, АРХІТЕКТУРА, БАЗА ДАНИХ.

Метою даної роботи є розробка програмного забезпечення, яке б дозволяло користувачам з легкістю конфігурувати комп'ютери, враховуючи останні досягнення в області комп'ютерних технологій та компонентів. Основні завдання, вирішені в роботі, включають аналіз існуючих рішень віртуальних конфігураторів, визначення вимог до програмного забезпечення, розробку архітектури та інтерфейсу користувача, програмування функціональності конфігуратора та тестування отриманого продукту.

У процесі розробки було використано сучасні методології програмування та інструментальні засоби, що дозволило створити зручний у використанні та функціональний продукт. Результати тестування показали високу ефективність та надійність розробленого програмного забезпечення.

Дана робота має значення для ринку комп'ютерних технологій, оскільки вона сприяє оптимізації процесу вибору та налаштування комп'ютерних систем, забезпечує користувачам доступ до індивідуалізованих рішень та підвищує загальну інформованість про доступні комп'ютерні компоненти та їх сумісність.

Підпис студента

30.05.2024

Дата

ЗМІСТ

Вступ 4
1 Аналіз предметної області згідно теми роботи 5
1.1 Обґрунтування актуальності теми роботи 5
1.2 Огляд функціональних можливостей існуючого програмного забезпечення для віртуального конфігурування комп'ютерів 9
1.3 Дослідження та аналіз шляхів вдосконалення програмного забезпечення віртуального конфігурування комп'ютерів 10
1.4 Аналіз вимог до програмного забезпечення та постановка технічного завдання 13
1.5 Висновки до першого розділу 20
2 Проєктування програмного забезпечення віртуального конфігуратора комп'ютерів загального призначення 21
2.1 Опис компонентів комп'ютерної системи загального призначення з позиції автоматизованого конфігурування 21
2.2 Проєктування бази даних програмного забезпечення віртуального конфігуратора 22
2.3 Розробка архітектури та вибір стеку технологій 24
2.5 Висновок до другого розділу 32
3 Реалізація та тестування програмного забезпечення віртуального конфігуратора комп'ютерів загального призначення 33
3.1 Розробка бази даних віртуального конфігуратора 33
3.2 Розробка серверної частини 42
3.3 Розробка та опис інтерфейсу користувача 46
3.4 Розгортання та тестування програмного забезпечення 50
3.5 Висновки до третього розділу 57

Висновки 59
Перелік джерел посилань 60
Додаток А Архітектура ПЗ 1 64
Додаток Б Архітектура ПЗ 2 65
Додаток В Архітектура БД 66
Додаток Г Скріншоти роботи додатку 67
Додаток Д Лістинги коду 70

ВСТУП

Сучасний розвиток інформаційних технологій і зростаючі вимоги до персоналізації комп'ютерних систем стимулюють інтерес до розробки ефективних інструментів для конфігурації комп'ютерів. У цьому контексті, програмне забезпечення віртуального конфігуратора комп'ютерів загального призначення є актуальним напрямком дослідження, яке відповідає потребам ринку та задовольняє вимоги користувачів до індивідуалізації та оптимізації комп'ютерних систем.

Актуальність теми обумовлена постійним розвитком комп'ютерних технологій та необхідністю створення зручних і функціональних інструментів для персоналізованого підбору комп'ютерних компонентів. Це дозволить користувачам не лише оптимізувати вартість і продуктивність комп'ютерних систем, але й забезпечити їх максимальну сумісність і ефективність.

Мета кваліфікаційної роботи полягає у розробці програмного забезпечення для віртуального конфігуратора комп'ютерів, яке забезпечить ефективний вибір та налаштування компонентів комп'ютерних систем відповідно до потреб користувачів. Для досягнення цієї мети були поставлені наступні завдання:

1. Провести аналіз існуючих рішень в області конфігурації комп'ютерів.
2. Визначити вимоги до програмного забезпечення віртуального конфігуратора.
3. Розробити архітектуру та інтерфейс користувача для програмного забезпечення.
4. Запрограмувати та протестувати функціонал віртуального конфігуратора.

Практичне значення результатів роботи виражається у можливості їх використання для оптимізації процесу вибору та налаштування комп'ютерних систем, зниження витрат часу та ресурсів на підбір компонентів, а також підвищення інформованості користувачів щодо доступних опцій і новинок ринку комп'ютерної техніки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЗГІДНО ТЕМИ РОБОТИ

1.1 Обґрунтування актуальності теми роботи

Сучасний світ інформаційних технологій постійно еволюціонує, а потреби користувачів у персоналізованих рішеннях зростають з кожним днем. Це призводить до того, що розробники і компанії повинні впроваджувати нові та інноваційні підходи для задоволення цих зростаючих вимог. Одним з найбільш перспективних напрямків є створення комп'ютерних систем загального призначення, які можуть бути налаштовані під індивідуальні потреби кожного користувача.

В умовах широкого впровадження таких систем виникає необхідність у розробці ефективних та зручних інструментів для їх конфігурації. Це означає, що користувачі повинні мати можливість легко налаштовувати свої системи, вибираючи оптимальні параметри для своїх завдань і бюджету. Вирішення цієї задачі лежить у створенні віртуальних конфігураторів.

Віртуальні конфігуратори – це спеціалізоване програмне забезпечення, яке дозволяє користувачам вибудовувати системи відповідно до їхніх вимог. Завдяки таким конфігураторам, користувачі можуть з легкістю підбирати компоненти та параметри своїх систем, забезпечуючи оптимальний баланс між продуктивністю та вартістю. Це значно спрощує процес налаштування, дозволяючи навіть користувачам без глибоких технічних знань створювати ефективні рішення.

Крім того, віртуальні конфігуратори можуть використовувати різні алгоритми та аналітичні інструменти для рекомендації найбільш підходящих варіантів. Це робить процес налаштування ще більш зручним та точним, що особливо важливо у світі, де швидкість і точність прийняття рішень стають все більш критичними. Впровадження таких інструментів дозволяє не тільки підвищити ефективність роботи з комп'ютерними системами, але й значно знизити витрати часу та ресурсів на їх конфігурацію.

Отже, розвиток віртуальних конфігураторів є важливим кроком у напрямку задоволення зростаючих потреб користувачів у сучасному світі інформаційних технологій. Це дозволяє створювати більш персоналізовані, ефективні та зручні у використанні рішення, що сприяє подальшому розвитку та впровадженню інноваційних технологій у різних сферах життя [1].

Актуальність теми дипломної роботи полягає в потребі ринку у програмних продуктах, що забезпечують гнучкість, швидкість вибору компонентів та ефективність їх взаємодії. Сучасний ринок інформаційних технологій характеризується швидкими темпами розвитку та постійними змінами. Користувачі прагнуть отримувати продукти, які не лише відповідають їхнім поточним потребам, але й мають потенціал для адаптації до майбутніх викликів та нових тенденцій.

Зі стрімким розвитком обчислювальної техніки, важливість програмного забезпечення для конфігурації комп'ютерів суттєво зросла. Сучасні користувачі потребують інструментів, які дозволяють швидко і легко обирати необхідні компоненти, забезпечуючи оптимальне співвідношення продуктивності та вартості. Крім того, це програмне забезпечення повинно гарантувати ефективну взаємодію між обраними компонентами, щоб уникнути несумісності та максимізувати продуктивність системи.

Одним з ключових аспектів є гнучкість програмних рішень. Це означає, що програми для конфігурації повинні мати можливість швидко адаптуватися до нових апаратних компонентів та програмних технологій, які з'являються на ринку. Здатність оперативно інтегрувати нові тенденції та технологічні інновації є важливою перевагою для користувачів, оскільки це дозволяє їм завжди мати актуальні інструменти для конфігурації своїх систем.

Адаптивність програмного забезпечення є ще одним критично важливим фактором. Програми повинні мати можливість підлаштовуватися під індивідуальні вимоги користувачів, враховуючи їх специфічні потреби та вподобання.

Це включає як можливість налаштування інтерфейсу, так і забезпечення підтримки різних мов програмування, операційних систем та апаратних платформ.

З урахуванням постійно зростаючих вимог до швидкості та ефективності, програмне забезпечення для конфігурації комп'ютерів має бути не лише функціональним, але й високопродуктивним. Це передбачає використання передових алгоритмів та методів оптимізації, що дозволяють мінімізувати час, необхідний для налаштування системи, та максимально ефективно використовувати наявні ресурси.

Таким чином, тема дипломної роботи є актуальною, оскільки відповідає на виклики сучасного ринку та спрямована на створення програмних продуктів, які здатні забезпечити користувачам гнучкість, швидкість вибору компонентів та ефективність їх взаємодії. Це, у свою чергу, сприятиме подальшому розвитку інформаційних технологій та підвищенню рівня задоволеності користувачів.

Основною метою розробки такого програмного забезпечення є створення інструменту, який дозволить користувачам швидко та ефективно вибирати необхідні компоненти комп'ютерних систем, налаштовувати їх під конкретні завдання та оптимізувати їхню роботу. Програма буде враховувати широкий спектр факторів, включаючи продуктивність, сумісність, вартість та енергоспоживання компонентів.

Актуальність кваліфікаційної роботи обумовлена не лише нагальними потребами ринку, але й необхідністю створення передових інструментів, що спрощують орієнтацію у складному технологічному просторі. Сучасний користувач має справу з величезною кількістю опцій та конфігурацій, що може викликати труднощі при виборі оптимального рішення. Розроблене програмне забезпечення дозволить спростити цей процес, надаючи користувачам інтуїтивно зрозумілий інтерфейс та рекомендації на основі аналізу потреб.

Крім того, впровадження таких інструментів сприятиме підвищенню продуктивності та ефективності використання комп'ютерних систем. За допомогою програмного забезпечення користувачі зможуть:

- швидко обирати оптимальні компоненти реалізуючи алгоритми підбору, які будуть враховувати як технічні характеристики, так і фінансові обмеження користувачів, що дозволить знаходити найкращі рішення для кожного конкретного випадку;

- налаштовувати системи під конкретні задачі засобами програмного забезпечення, що дозволить здійснювати налаштування системи, адаптуючи її для максимальної продуктивності у відповідних умовах експлуатації;

- оцінювати та порівнювати варіанти кінцевими користувачами, які зможуть легко порівнювати різні конфігурації та вибирати найбільш підходящу з них, що зменшить ризики вибору неефективних рішень.

Базові функції програмного забезпечення показані схематично на рисунку 1.1.

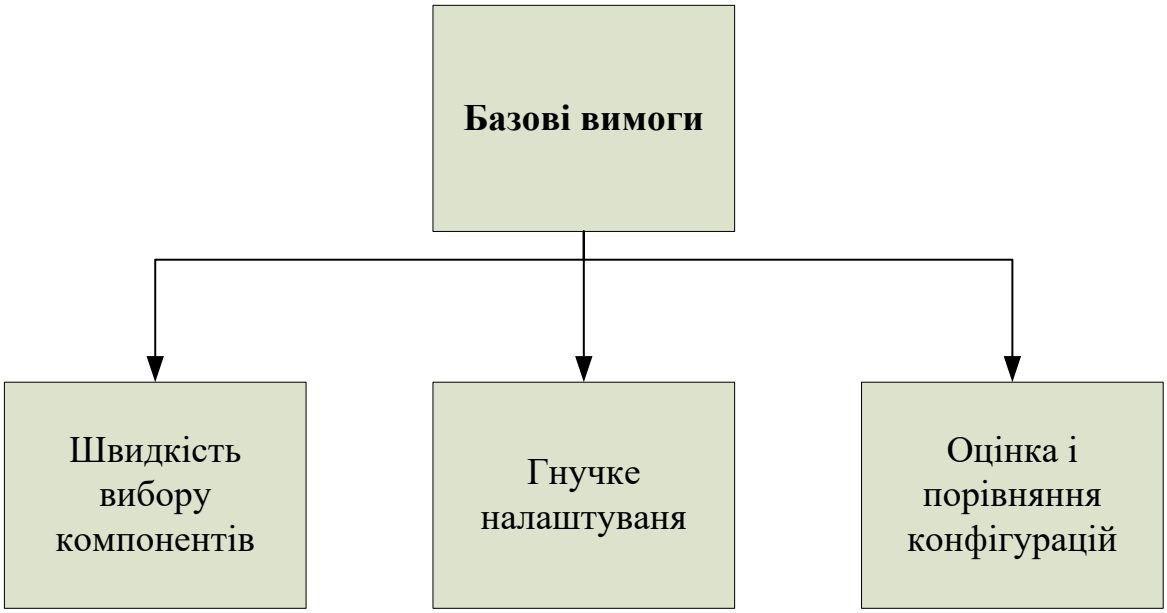


Рисунок 1.1 – Базові вимоги до функцій програмного забезпечення

Таким чином, розробка програмного забезпечення, що полегшує вибір та налаштування комп'ютерних систем, є важливим кроком у напрямку задоволення потреб ринку та створення інструментів, які підвищують продуктивність та

ефективність використання технологій. Це сприятиме подальшому розвитку інформаційних технологій та забезпеченню високого рівня задоволеності користувачів.

1.2 Огляд функціональних можливостей існуючого програмного забезпечення для віртуального конфігурування комп'ютерів

Серед існуючих програмних продуктів для віртуального конфігурування комп'ютерів можна виділити низку систем, що надають користувачам можливість самостійного вибору компонентів для збірки персонального комп'ютера [2]. Ці програмні продукти забезпечують ряд функціональних можливостей, які спрощують процес конфігурації та адаптації системи під індивідуальні потреби користувача.

Перш за все, це графічні інтерфейси користувача, які зроблені інтуїтивно зрозумілими та легкими у використанні. Вони дозволяють користувачам в режимі реального часу визначати та модифікувати складові елементи комп'ютера, такі як процесор, оперативна пам'ять, накопичувачі, відеокарти, тощо. Така інтерактивність є ключовою для ефективного вибору необхідних компонентів. Наступним важливим аспектом є це системи рекомендацій, які засновані на алгоритмах штучного інтелекту та машинного навчання. Вони аналізують попередні вибори користувачів та визначають найбільш популярні або ефективні комбінації компонентів, пропонуючи оптимальні рішення для задоволення їхніх потреб. Ще однією ключовою особливістю є можливість здійснення порівняльного аналізу різних конфігурацій за допомогою вбудованих інструментів. Це дозволяє користувачам оцінювати ефективність та цінність кожного компонента в контексті загальної системи. Наступна функціональна можливість полягає у злагодженій взаємодії з постачальниками компонентів та оновлювальними сервісами.

Сучасні конфігуратори забезпечують доступ до останніх даних про ціни, наявність компонентів на складі та їх технічні характеристики, що значно спрощує процес придбання [3]. Також варто виділити високий рівень кастомізації та підтримки користувачів, що включає технічну підтримку, довідкові матеріали, туторіали та форуми для обміну досвідом.

1.3 Дослідження та аналіз шляхів вдосконалення програмного забезпечення віртуального конфігурування комп'ютерів

В процесі дослідження існуючих систем віртуального конфігурування комп'ютерів, стає зрозумілим, що попри те, що багато з них вже пропонують значний спектр функціональних можливостей, існує ряд потенційних шляхів для їх удосконалення. Цей розділ присвячено виявленню таких шляхів та аналізу можливостей для покращення існуючих рішень роблячи їх більш адаптивними, гнучкими та відповідними до сучасних вимог користувачів [4].

Перший напрямок вдосконалення програмного забезпечення віртуального конфігурування комп'ютерів полягає у розширенні та оновленні бази даних компонентів, що має на меті надати користувачам актуальну та всеохоплюючу інформацію. Сучасні ринки IT-компонентів швидко змінюються, з новими продуктами, що випускаються на щоквартальній, а іноді й щомісячній основі. Відтак, база даних, яка регулярно оновлюється, є критично важливою для забезпечення того, щоб користувачі могли вибрати найновіші та найефективніші компоненти для своїх систем [5]. Така база даних має включати різноманітність параметрів: від базових характеристик, таких як частота процесора та розмір оперативної пам'яті, до більш специфічних деталей, як, наприклад, підтримка технологій віртуалізації або сумісність з певними інтерфейсами.

Крім того, можливість фільтрації та порівняння компонентів на основі різних критеріїв, таких як ціна, продуктивність, енергоефективність, конфігуратор надає користувачам можливість зробити більш інформований та виважений вибір.

Це означає, що користувачі зможуть відфільтровувати доступні компоненти за заданими параметрами, що дозволить швидко знайти ті, які найкраще відповідають їхнім потребам та фінансовим можливостям.

Ключові функції програмного забезпечення включають фільтрування компонентів:

- за ціною, що полягає у можливості користувачів встановлювати цінові межі, в яких вони готові розглядати компоненти, що дозволить уникнути варіантів, які перевищують їхній бюджет;
- за характеристиками продуктивності компонентів, таких як частота процесора, обсяг оперативної пам'яті, пропускна здатність системних шин та інші параметри, що визначають продуктивність системи;
- за енергоефективністю, що надає користувачеві функцію вибору компоненти з оптимальним рівнем енергоспоживання з метою зниження витрат на електроенергію та забезпечення екологічності системи.

Критерії фільтрування комп’ютерних компонентів засобами програмного конфігуратора зображено на рисунку 1.2.

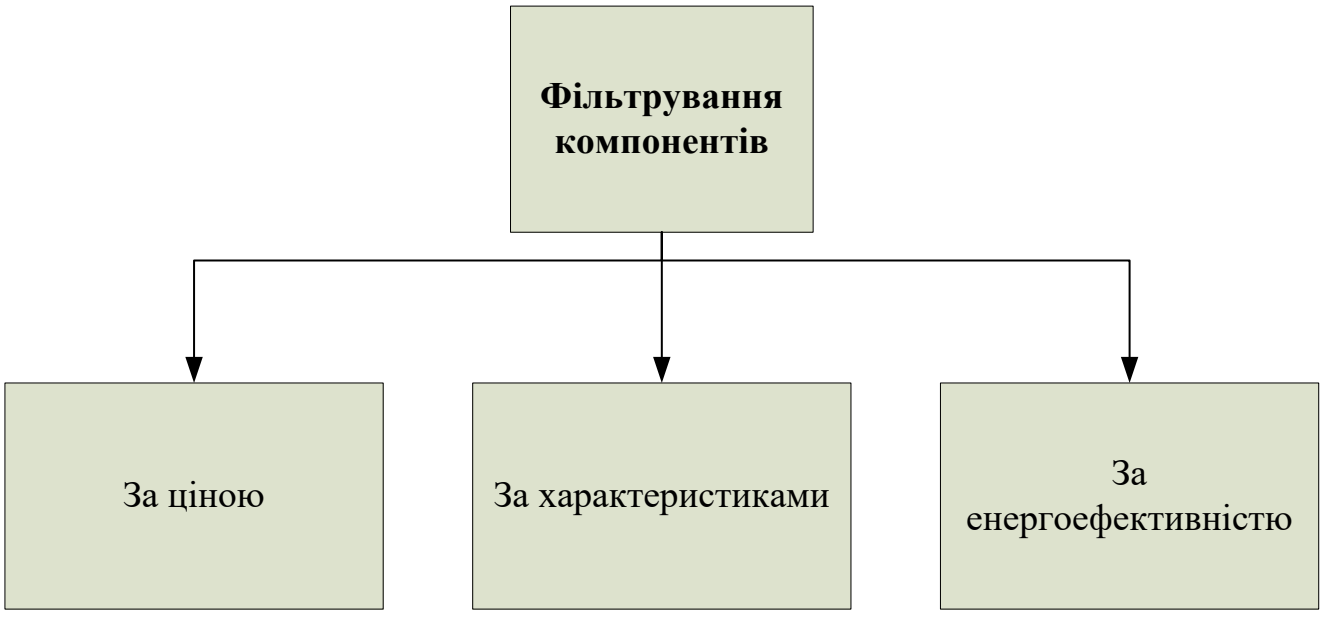


Рисунок 1.2 – Критерії фільтрування комп’ютерних компонентів

За допомогою функції порівняння компонентів користувачі зможуть вибирати кілька компонентів для порівняння їх характеристик у зручному форматі. Це дозволить детально аналізувати переваги та недоліки кожного варіанту та приймати більш обґрунтовані рішення. Також до числа додаткових функції користувачів відноситься можливість ведення історії відгуків користувачів, що включатиме базу даних з відгуками реальних користувачів, що допоможе зрозуміти практичні аспекти використання компонентів та їх надійність. Підтримка рейтингової системи продуктів на основі відгуків та експертних оцінок дозволить швидко оцінити популярність та якість товарів.

На додачу до функціональних вимог конфігуратора уваги заслуговують також його нефункціональні вимоги, які покращать користування програмою з точки зору кінцевого користувача. До нефункціональних вимог належать загальна інформованість та можливість детально аналізувати і порівнювати компоненти на основі різних критеріїв забезпечує користувачам доступ до всебічної інформації, що сприяє прийняттю більш виважених рішень. Економія часу користувача шляхом реалізації автоматизованих функцій фільтрації та порівняння, що зменшить час, необхідний для пошуку та аналізу компонентів, що дозволить користувачам швидше знаходити оптимальні для себе рішення. Функція персоналізації полягає у можливості враховувати індивідуальні особливості та потреби користувачів, створюючи індивідуальні конфігурації, що відповідають конкретним потребам, забезпечуючи підвищену ефективність системи. З метою зниження ризиків необхідно передбачити доступ до відгуків та рейтингів компонентів з метою уникнути придбання неякісних частин, або таких, що не відповідають потребам.

Таким чином, розробка програмного забезпечення з можливістю фільтрації та порівняння компонентів, а також включенням оглядів та рейтингів, значно підвищить якість вибору користувачів, забезпечуючи їм всебічну та актуальну інформацію для прийняття обґрунтованих рішень. Це сприятиме створенню більш

ефективних та оптимальних комп'ютерних систем, відповідно до індивідуальних потреб кожного користувача.

Стратегія підтримки прийняття рішень полягає у розгортанні модулів машинного навчання, здатних аналізувати дії користувачів та удосконалювати системи рекомендацій [6]. Застосування таких технологій може якісно покращити рівень взаємодії користувачів з конфігураторами, роблячи її більш персоналізованою та ефективною. Системи машинного навчання можуть виявляти звички та вподобання користувачів, передбачати потенційні варіанти, що їх можуть зацікавити, та прогнозувати майбутні потреби на основі аналізу великих даних. Зокрема, такі системи можуть пропонувати конфігурації, які оптимально поєднують продуктивність та вартість, виходячи з історичних даних про попит та відгуки користувачів. Це дозволяє не тільки підвищити задоволення користувача, але й зробити більш точні та індивідуально підібрані пропозиції, що враховують специфічні потреби та бюджет кожного користувача.
В результаті, використання модулів машинного навчання може сприяти збільшенню лояльності клієнтів спонукати їх продовжувати використання даного програмного забезпечення.

1.4 Аналіз вимог до програмного забезпечення та постановка технічного завдання

Цей розділ присвячений аналізу вимог до програмного забезпечення і є ключовим етапом у процесі розробки програмного продукту. На основі аналізу зібраних в попередньому розділі вхідних даних, визначимо головні функціональних та нефункціональних вимоги віртуального конфігуратора.

Функціональні вимоги описують конкретні дії користувача з відповідною реакцією конфігуратора, яку має забезпечити програмне забезпечення:

- користувачі мають можливість використовувати інтерактивну кнопку розширення списку комплектуючих для розгортання додаткових опцій або категорій компонентів;

- механізм авторизація користувачів з наданням системою можливості користувачам щодо виконання реєстрації та входу за допомогою логіну та пароля.

- забезпечення можливості зберігати, переглядати та ділитися збірками компонентів шляхом реалізації модуля збереження збірок;

- фільтрація та параметризований пошук компонентів за критеріями;

- оформлення замовлення та інтеграція з Інтернет-магазинами;

Нефункціональні вимоги конфігуратора стосуються тих аспектів системи, що визначають якість графічного інтерфейсу та користувацький досвід. До дизайну графічного інтерфейсу висувається вимога щодо візуальної привабливості, та інтуїтивно зрозумілого розташування елементів керування з часом відгуку системи не більше 2 секунд та коректним відображенням та функціонуванням на різних пристроях і платформах. З метою покращення доступу користувача до усіх функцій також необхідно забезпечити ефективну та інтуїтивно розуміти карту сайту та меню навігації із забезпеченням технологій для захисту персональних даних користувачів.

Отже, за результатами аналізу головних функціональних та нефункціональних вимог до програмного забезпечення, була виконана постановка технічного завдання на розробку віртуального конфігуратора комп'ютерів загального призначення. Метою розробки є автоматизація процесу конфігурування комп'ютерів користувачами. Кінцевим результатом розробки є програмне забезпечення віртуального конфігуратора, розгорнуте на сервері з доступом до мережі Інтернет. Головні функції конфігуратора відповідають вищеописаним вимогам. Повнота вимог забезпечується аналізом існуючих програмних продуктів, які частково виконують ці функції, а також аналізом спеціалізованих вимог в процесі дослідження предметної області.

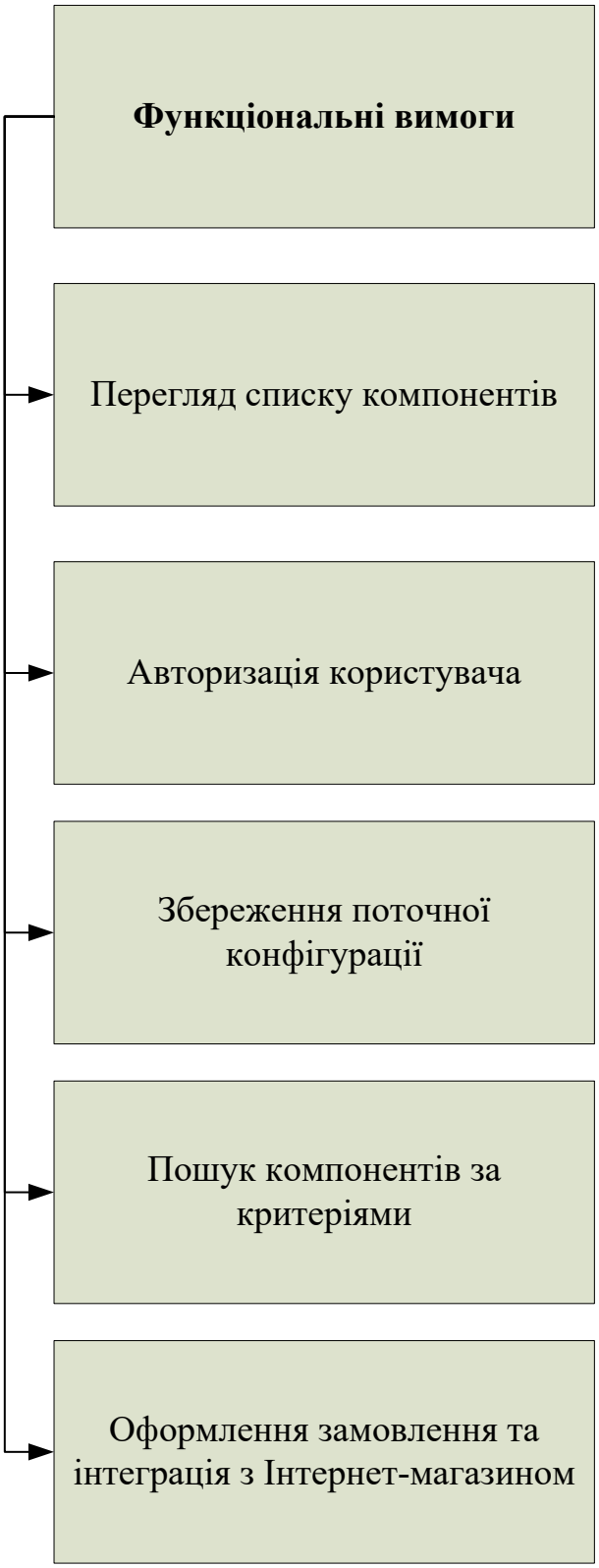


Рисунок 1.3 – Функціональні вимоги до програмного забезпечення

При проєктуванні інтерфейсу користувача віртуального конфігуратора комп'ютерів ключовою метою є забезпечення не тільки функціональності, але й

високої естетичної привабливості та інтуїтивно зрозумілого дизайну, що разом сприяють наданню найкращого користувацького досвіду.

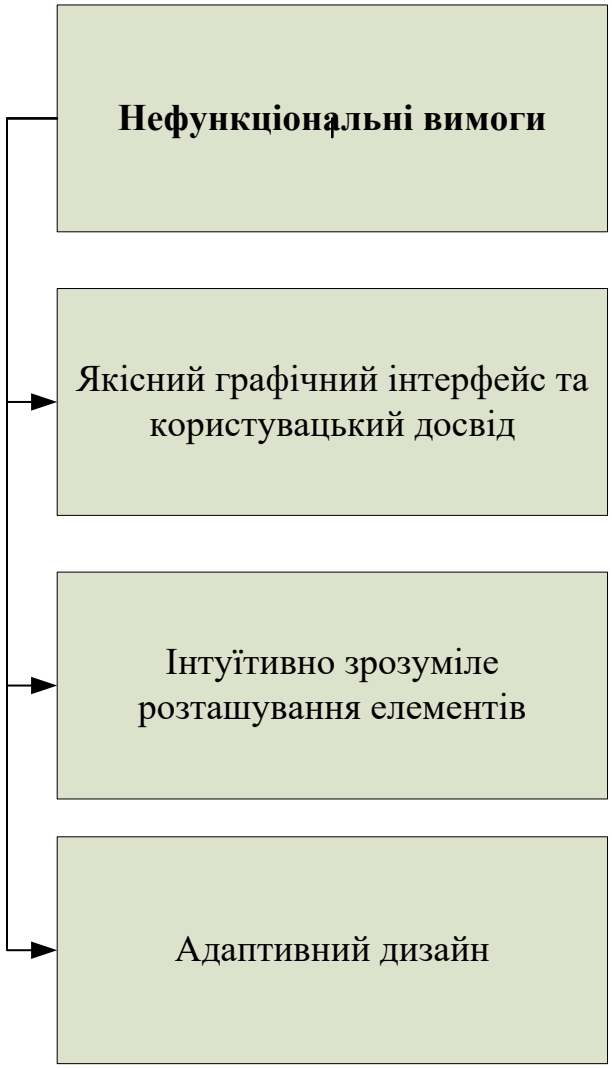


Рисунок 1.4 – Нефункціональні вимоги до програмного забезпечення

Центральна ідея полягає у створенні зрозумілого та ефективного користувацького інтерфейсу, який дозволяє користувачам легко керувати процесом вибору та кастомізації продукту.

Для цього важливо втілити дизайн, який би був сучасним і мінімалістичним, з чистими лініями, великим білим простором, чіткими шрифтами та яскравими, але не перевантаженими кольорами, що разом створюють візуально приємне

середовище. Це дозволить користувачам зосередитись на головній задачі - конфігурації комп'ютера, не відволікаючись на зайві елементи. Логіка навігації в інтерфейсі віртуального конфігуратора комп'ютерів визначає спосіб, яким користувачі переходять між різними етапами процесу конфігурації, забезпечуючи зручність та легкість використання. Для досягнення цього мети, кожен крок конфігурації має бути чітко визначений та доступний користувачу, а прогрес у конфігурації має бути легко відстежуваним, дозволяючи користувачам зрозуміти, на якому етапі вони знаходяться та що ще залишилося до завершення. Для забезпечення зручності використання та відстеження прогресу конфігурації, важливо використовувати інтерактивні елементи управління. Наприклад, слайдери можуть використовуватися для налаштування параметрів, таких як обсяг пам'яті чи потужність процесора. Випадаючі списки можуть допомагати користувачам вибирати компоненти, такі як відеокарти чи жорсткі диски, зі списку доступних опцій. Кнопки для вибору опцій дозволяють користувачам вибирати додаткові функції або аксесуари для свого комп'ютера. Ці інтерактивні елементи повинні забезпечувати миттєву реакцію на дії користувача, щоб вони могли швидко та ефективно взаємодіяти з додатком. Забезпечення такого швидкого та безперебійного взаємодії є ключовим для створення задоволеного користувача та позитивного враження від використання віртуального конфігуратора.

Забезпечення адаптивного дизайну для віртуального конфігуратора комп'ютерів є критично важливим, оскільки користувачі можуть отримувати доступ до нього з різних пристроїв з різними розмірами екранів та характеристиками. Адаптивний дизайн забезпечує зручне та ефективне використання додатку на будь-яких пристроях, від маленьких мобільних пристроїв до великих настільних моніторів.

При розробці адаптивного дизайну слід враховувати різні фактори, такі як розмір екрану, орієнтація пристрою, роздільна здатність тощо. Крім того, слід

враховувати те, що користувачі можуть використовувати конфігуратор як в портретному, так і в альбомному режимах, тому важливо забезпечити оптимальний дизайн для обох варіантів. Одним із ключових аспектів адаптивного дизайну є візуалізація виборів та конфігурації у реальному часі. Користувачі мають бачити результати своїх дій негайно, щоб мати можливість оцінити їх та внести необхідні зміни. Це може включати відображення зображень або 3D-моделей вибраних компонентів, що допомагає користувачам краще розуміти вплив їхніх виборів на загальну конфігурацію та вартість системи. Такий підхід не лише забезпечує користувачам впевненість у своїх виборах, але й сприяє покращенню їхнього розуміння процесу конфігурації комп'ютера. Завершення проектування інтерфейсу користувача віртуального конфігуратора є важливим етапом, що спрямований на створення високоякісного та задовільного досвіду для користувачів. Основні цілі цього етапу включають створення зручного, функціонального та естетично привабливого інтерфейсу, який сприяє легкості вибору та замовлення ідеальної конфігурації комп'ютера. Перш за все, інтерфейс повинен бути зручним для використання. Це означає, що всі елементи управління та навігації мають бути доступні та легкі у використанні. Користувачі повинні легко розуміти, як взаємодіяти з додатком, щоб вони могли швидко та ефективно здійснювати свої вибори. Далі, інтерфейс повинен бути функціональним, що означає, що він повинен відповідати усім потребам користувачів. Всі необхідні функції та можливості повинні бути легко доступні, а процес конфігурації повинен бути покроковим та інтуїтивно зрозумілим.

Крім того, естетичний аспект також має велике значення. Інтерфейс повинен бути приємним для очей, зі спеціальним акцентом на дизайн та візуальну привабливість. Елементи дизайну, такі як кольори, шрифти та макет, повинні бути гармонійно злагодженими та створювати позитивний естетичний враження. Загалом, завершення проектування інтерфейсу користувача віртуального конфігуратора спрямоване на створення якісного та задовільного досвіду, який

допомагає користувачам з легкістю обирати та замовляти ідеальну конфігурацію комп'ютера.

З урахуванням функціональних та нефункціональних вимог, а також вимог до інтерфейсу користувача, була розроблена модель варіантів використання програмного забезпечення віртуального конфігуратора комп’ютерів, що зображена на рисунку 1.5.

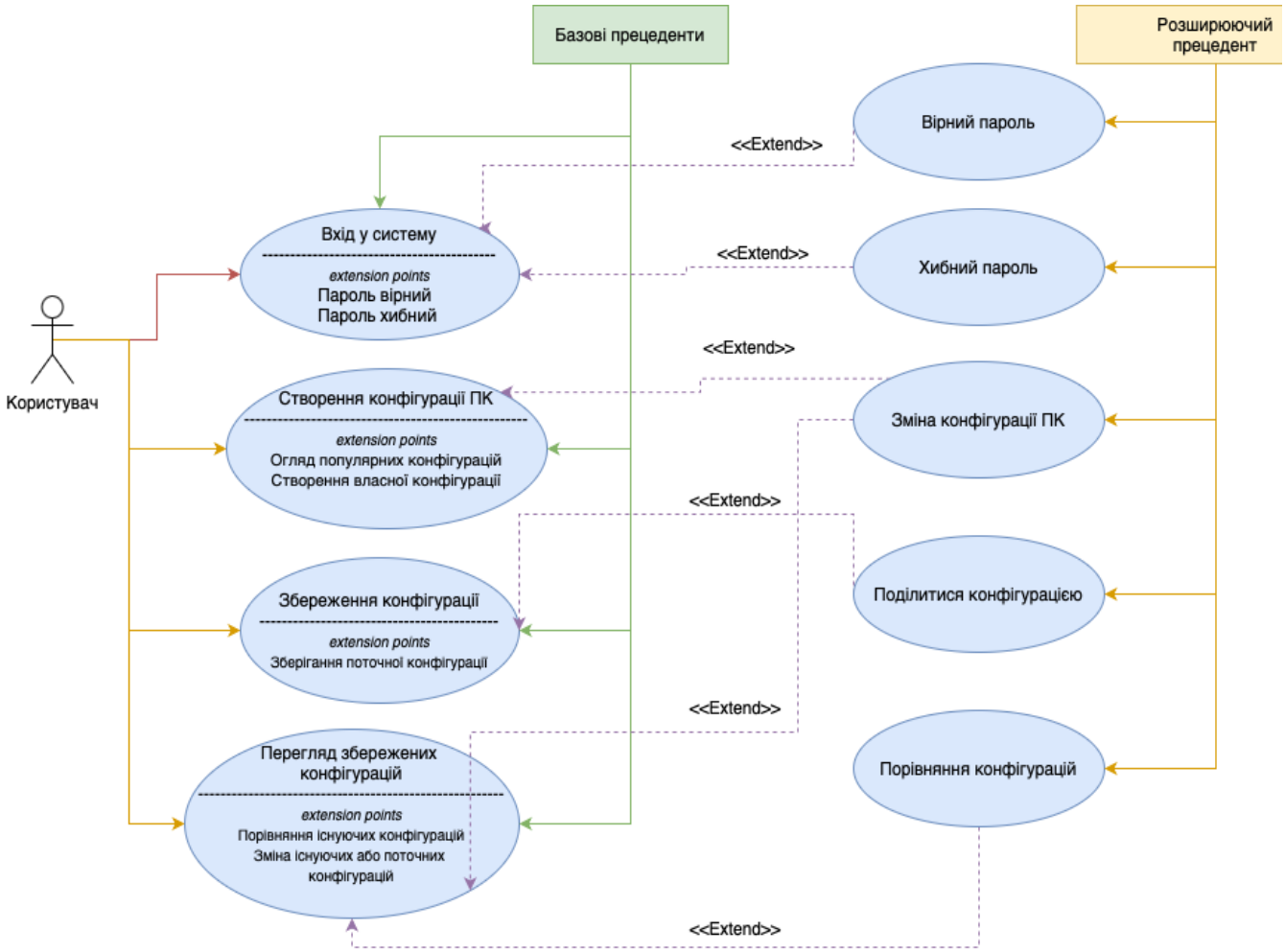


Рисунок 2.4 – Модель варіантів використання віртуального конфігуратора

Деталі реалізації складаються з опису архітектури системи (Додаток Б та Додаток В), обґрунтування вибору технологій і платформи, а також технічного

процесу, який буде застосований в розробці з метою розширення проєкту до такого, над яким доцільно проводити роботу командою розробників.

Критерії прийнятності в технічному завданні полягають у відповідності кінцевого програмного продукту поставленим вимогам щодо функціоналу, коректність проходження тестовим сценаріям та експертній оцінці фахівця з предметної області.

Календарний план виконання кваліфікаційної роботи розроблено з урахуванням навчального плану, який складається з формулювання завдання, виконання етапів завдання, які представлені розділами кваліфікаційної роботи.

1.5 Висновки до першого розділу

Перший розділ дипломної роботи був присвячений детальному аналізу предметної області щодо програмного забезпечення для віртуального конфігурування комп'ютерів, визначенню актуальності теми, огляду існуючих рішень на ринку, а також дослідженню можливостей для їх удосконалення.

З аналізу актуальності теми випливає, що зростання потреби в індивідуалізованих комп'ютерних системах вимагає створення вдосконалених інструментів для їх конфігурації. Огляд існуючих рішень вказує на наявність значного простору для інновацій та покращень, зокрема в частинах користувацького інтерфейсу, алгоритмів рекомендацій, безпеки, та інтеграції з електронною комерцією.

Висновки з дослідження шляхів удосконалення підкреслюють важливість розвитку гнучких та масштабованих систем, здатних адаптуватися до швидко мінливих технологій і вимог користувачів. Це включає інтеграцію передових технологій, таких як машинне навчання, та використання даних для створення більш персоналізованих рішень.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВІРТУАЛЬНОГО КОНФІГУРАТОРА КОМП'ЮТЕРІВ ЗАГАЛЬНОГО ПРИЗНАЧЕННЯ

2.1 Опис компонентів комп'ютерної системи загального призначення з позиції автоматизованого конфігурування

Для ефективної розробки віртуального конфігуратора комп'ютерів необхідно провести опис загальної структури та компонентів, які входять до складу стандартної комп'ютерної системи. Такий аналіз є критично важливим, оскільки на ньому базується модель та взаємодія даних предметної області проєкту. Основні компоненти, вибір яких відіграє головну роль під час модульного конфігурування комп'ютерів, представлені відповідними позиціями в роздрібних та гуртових пропозиціях постачальників із сортуванням за ціною за одиницю товару та описом тактико-технічних характеристик.

В таблиці 2.1 наведено перелік компонентів сучасного комп'ютера та їхніх характеристик.

Таблиця 2.1 – Характеристики комп'ютерних компонентів

Компонент	Опис характеристик
1	*2*
Центральний Процесор (ЦП)	Тип роз'єму, кількість ядер, тип пам'яті, сімейство процесорів, частота, особливості (наприклад, підтримка турбобусту, вбудоване графічне ядро)
Оперативна пам'ять (RAM)	Тип пам'яті (наприклад, DDR4, DDR5), максимальна підтримувана швидкість, можливості модуля (наприклад, ECC пам'ять)
Зберігання даних (HDD/SSD)	Тип пристрою (HDD або SSD), інтерфейс (наприклад, SATA, NVMe), швидкість читання/запису, обсяг

Кінець таблиці 2.1 – Характеристики компонентів

1	*2*
Відеокарта (GPU)	Інтегрована або дискретна, модель, об'єм пам'яті, тип пам'яті, тактова частота, підтримка API (DirectX, Vulkan)
Материнська плата	Сумісність з процесором, типи пам'яті, кількість слотів для розширень, наявні порти (USB, Ethernet, аудіо порти)
Блок живлення	Потужність у ватах, кількість роз'ємів для підключення компонентів, стандарти ефективності (80 Plus)
Корпус	Тип корпусу (наприклад, ATX, microATX), кількість місць для встановлення вентиляторів, дизайн, вага

Отже, дані, що зведені в таблицю 2.1, будуть використані для розробки сутностей предметної області, що слугуватиме базою для розробки функцій користувача програмного забезпечення.

2.2 Проєктування бази даних програмного забезпечення віртуального конфігуратора

Під час розробки програмного забезпечення для віртуального конфігуратора комп'ютерів, етап проєктування база даних відіграє важливу роль, адже саме вона є центральним сховищем усієї інформації, яка обробляється та надається користувачу [13]. Розробка такої бази даних виконана на основі аналізу бізнес-логіки конфігуратора з урахування багатьох факторів, таких як швидкість доступу до даних, гнучкість для майбутньої інтеграції з іншими системами, безпека інформації та здатність до масштабування.

В контексті віртуального конфігуратора, база даних спроектована так, щоб підтримувати великий спектр компонентів комп'ютера відповідно до таблиці 1.2, дозволяючи легко додавати, оновлювати або вилучати елементи в міру зміни

асортименту продуктів. Вона також має забезпечувати можливість ефективного пошуку та вибору компонентів згідно з різними критеріями, такими як ціна, характеристики, виробник, та інші важливі параметри.

З огляду на вищезазначене, структура бази даних має бути розроблена з урахуванням аспектів нормалізації даних та ретельного планування зв'язків між таблицями, що допомагає уникнути дублювання даних та забезпечити їхню цілісність [14]. Важливим аспектом під час проєктування бази даних також є забезпечення захисту від несанкціонованого доступу до даних, особливо з урахуванням персональних даних користувачів та інформації про платіжні транзакції (реалізація цих функцій виходить за рамки кваліфікаційної роботи, але може також бути реалізована в подальшій роботі над проєктом). З метою забезпечення гнучкості щодо подальшого розширення програмного забезпечення схема бази даних має бути спроектована таким чином, щоб легко вносити зміни у структуру даних, не порушуючи існуючу структуру та працездатність.

Отже, з урахуванням основних принципів проєктування та вимог до бази даних, представимо перелік таблиць та схему зв'язків бази даних віртуального конфігуратора, що наведені в таблиці 2.2 та зображені на рисунку 2.1 відповідно.

Таблиця 2.2 – Перелік таблиць бази даних проєкту

Назва таблиці	Опис
1	*2*
Збірка	Для кожної збірки вона зберігає ID, назву збірки, опис, статус готовності
Компоненти	У цій таблиці ведеться реєстрація всіх можливих компонентів, які можна вибрати для збірки
Типи компонентів	Ця таблиця визначає категорії або типи компонентів
component_attributes	Вона зберігає характеристики для кожного компоненту
Користувач	Вона зберігає особисту інформацію користувачів, їх роль у системі та стан автентифікації

Кінець таблиці 2.2 – Перелік таблиць бази даних проєкту

1	*2*
Права та Права користувача	Ці дві таблиці разом забезпечують ролеву модель управління доступом користувачів до системи
Цільові атрибути	Ця таблиця використовується для визначення набору атрибутів, обов'язкових для кожного типу компоненту
Конфігуратор компонентів	Вона є зв'язуючою таблицею між builds і components

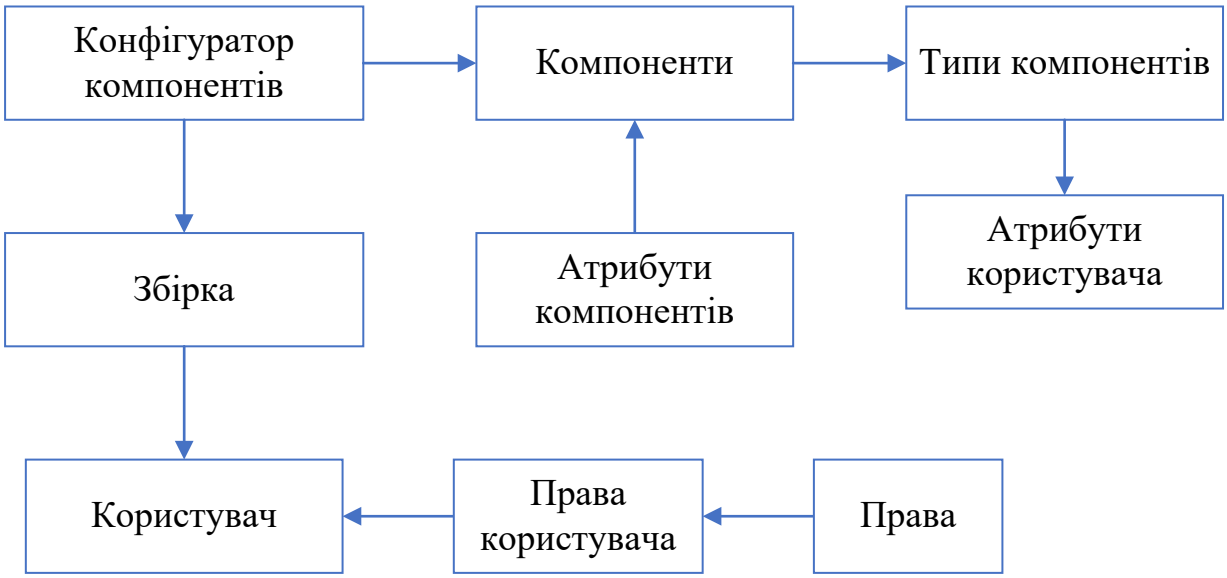


Рисунок 2.1 – Схема зв’язків бази даних

Отже, на рисунку 2.1 зображені сутності (прототипи таблиць) бази даних проєкту із позначеними зв’язками. До головних таблиць належать: таблиця компонентів, таблиця типів компонентів, таблиця з відповідними атрибутами, а також таблиця цільових атрибутів, що задані користувачем. Допоміжні таблиці – це таблиця користувачів та таблиці з правами доступу та ролями користувачів.

Ця модель буде використана в наступному розділі в якості основи для програмної реалізації бази даних віртуального конфігуратора комп’ютерів.

2.3 Розробка архітектури та вибір стеку технологій

Під час розробки архітектури віртуального конфігуратора комп'ютерів важливо врахувати низку ключових аспектів, які впливають на якість та ефективність системи. Відповідно до потреб користувачів та зростання обсягу даних, система повинна мати можливість масштабування. Це означає, що вона повинна бути готова працювати з великою кількістю користувачів одночасно та обробляти збільшення обсягу даних без втрати продуктивності. Також надзвичайно важливим аспектом при проєктуванні будь-якої системи є забезпечення безпеки, особливо якщо така система повинна збирати, зберігати та обробляти конфіденційну інформацію користувачів. Тому віртуальний конфігуратор комп'ютерів повинен мати вбудовані засоби безпеки для захисту від зловмисників, а також забезпечити конфіденційність даних користувачів. Крім того, важливим факторами є загальна надійність та стабільність роботи системи, тому її програмна архітектура повинна передбачати стійкість до помилок та відновлюватися після виникнення відмов без втрати даних. В контексті взаємодії програмної системи з користувачем, до інтерфейсу користувача ставиться вимога зручності у використанні та інтуїтивної зрозумілості.

З урахуванням вищеописаних вимог до архітектури системи, вибір стеку технологій React для фронтенду та Laravel для бекенду є обґрунтованим, оскільки обидва ці фреймворки мають велику популярність, широкі можливості та активну спільноту розробників. Фреймворк React забезпечує динамічний та ефективний інтерфейс користувача, тоді як Laravel забезпечує швидку розробку, безпеку та надійність серверної частини додатку. Такий підхід дозволить розробити повнофункціональний веб-додаток з сучасним інтерфейсом та надійною серверною частиною, що відповідає вимогам до масштабованості, безпеки, стабільності та зручності користування [19].

Фронтенд віртуального конфігуратора комп'ютерів, що побудований на основі React, забезпечує динамічне та інтерактивне середовище користувача завдяки його компонентно-орієнтованому підходу [18]. Кожен компонент в React є самодостатньою одиницею, що керує власним станом та рендером, що дозволяє

розробникам будувати складні інтерфейси з простих частин. Управління станом у додатках React є головним аспектом при розробці складних та великих за обсягом додатків. Під час взаємодії з додатком користувачі змінюють стан, ініціюючи різні події та взаємодію з елементами інтерфейсу. Подальше управління станом дозволяє зберігати, оновлювати та синхронізувати ці зміни стану між різними компонентами додатку. Бібліотеки Redux та Context API, допомагають керувати цим станом, забезпечуючи однозначність та передбачуваність дій користувача. Бібліотека Redux надає інструменти для керування станом та забезпечує однозначність шляхом зберігання всього стану додатку в одному централізованому сховищі (store) і застосовує зміни до цього стану за допомогою чистих функцій, що називаються редукторами. Альтернативним способом керування станом у React-додатках є Context API, який дозволяє передавати дані вниз по дереву компонентів без потреби вручну передавати їх через кожен проміжний компонент. У контексті віртуального конфігуратора, де користувачі можуть одночасно взаємодіяти з багатьма налаштуваннями та опціями, ефективне керування станом стає критично важливим. Використання Redux або Context API допомагає забезпечити, що всі компоненти додатку будуть завжди відображати актуальний стан, уникнути конфліктів та забезпечити зручну та приємну взаємодію користувачів з додатком.

Використання мови JSX (JavaScript XML), що є розширенням синтаксису для JavaScript, дозволяє писати код, який виглядає майже як HTML, проте він інтерпретується як виклики функцій JavaScript. Це робить код більш зрозумілим і читабельним для розробників. Завдяки JSX, React-компоненти можуть бути представлені у вигляді дерева вузлів, що відповідають структурі веб-сторінки.

Структурна діаграма додатку React зображена на рисунку 2.2.

Актуальність вибору технології React обумовлена підтримкою віртуальної моделі документу (Document Object Model, DOM). При зміні даних у React, він не вносить безпосередні зміни до реального DOM. Замість цього, React використовує віртуальний DOM, який є копією реального DOM, і порівнює зміни, які потрібно

зробити, зі змінами, які відбулися у віртуальному DOM. Після цього React виконує оптимізовані зміни до реального DOM, що робить його оновлення ефективним та швидким.

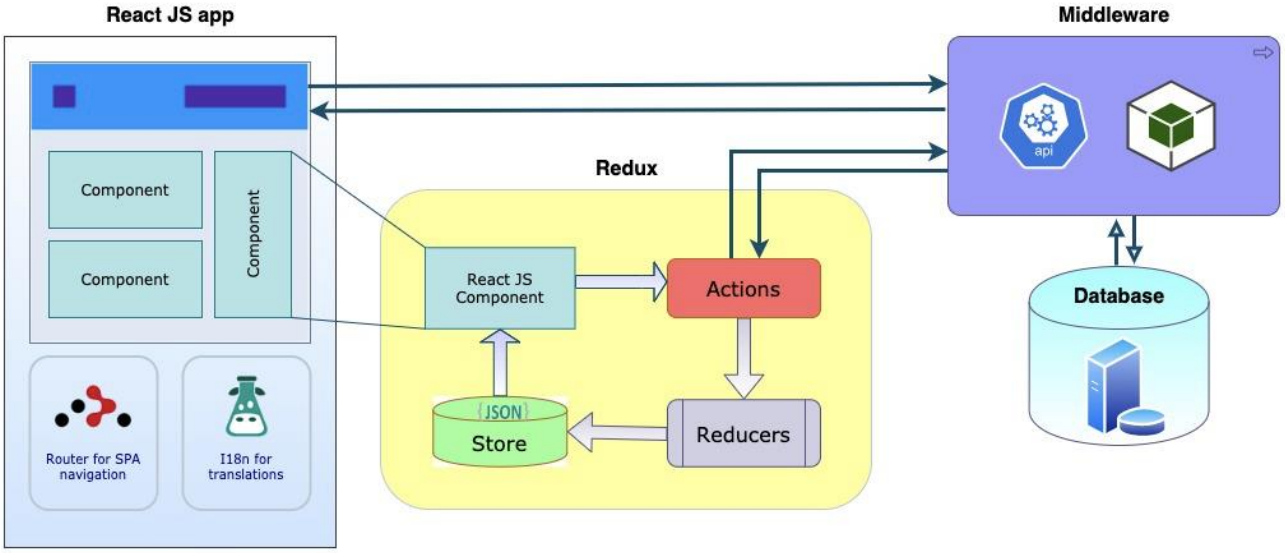


Рисунок 2.2 – Структурна діаграма додатку React

Така модель реакція на зміни даних дозволяє програмі оновлювати лише ті частини інтерфейсу, які дійсно змінилися, не перерисовуючи весь інтерфейс. Це призводить до значного підвищення швидкодії інтерфейсу, оскільки реактивні оновлення DOM роблять його більш ефективним та оптимізованим [20].

Отже, для розробки віртуального конфігуратора комп'ютерів використання фреймфорку React надає унікальні можливості, що дозволяють користувачам бачити результати своїх дій миттєво. Це означає, що користувачі можуть в реальному часі спостерігати за змінами у збірці комп'ютера: наприклад, вони можуть відслідковувати оновлення вартості збірки з кожним доданим або зміненим компонентом, не чекаючи на перезавантаження сторінки. Інтерактивність та миттєва зміна стану додатку є ключовими елементи для створення високоякісного досвіду користувача. Використання React дозволяє реалізувати таку функціональність без затримок та забезпечити швидкість реакції на дії користувачів. Завдяки віртуальному DOM та механізмам оптимізації

відображення, React забезпечує ефективну роботу з динамічним вмістом, що робить його ідеальним вибором для створення веб-додатків сучасного зразка, таких як віртуальні конфігуратори комп'ютерів.

Для реалізації серверної частини програмного забезпечення було обрано мову PHP та фреймворк Laravel. Серверна частина (бекенд) віртуального конфігуратора на базі Laravel є головним компонентом системи, який відповідає за обробку запитів від клієнтської частини (фронтенду) програмного забезпечення, а також за збереження та обробку даних, роботу із в базою даних, а також надає інтерфейс для взаємодії іншими компонентами. Використання обраної технології для розробки серверної частини дозволяє впроваджувати сучасні підходи, забезпечуючи високий рівень абстракції та гнучкість у взаємодії з базою даних. Це особливо важливо для складних систем, таких як конфігуратори, де необхідно ефективно зберігати та оновлювати великий обсяг даних.

Архітектура Laravel базується на шаблоні Model-View-Controller (MVC), завдяки чому фреймворк забезпечує чітке відокремлення логіки додатку. Контролери відповідають за обробку запитів та взаємодію з моделями, які представляють структуру та логіку бази даних. Моделі дозволяють гнучко взаємодіяти з базою даних, забезпечуючи доступ до неї через високорівневі методи. Відомо, що рівень представлень використовується рідше у контексті API-орієнтованого додатку з фронтендом на React, однак він може бути задіяний для рендерингу шаблонів електронної пошти або в інших завданнях, де необхідно генерувати HTML-код на сервері для відображення користувачу. Такий підхід дозволяє ефективно розділити логіку додатку та подати дані користувачам у зручній формі.

Головним компонентом Laravel модуль Eloquent ORM, який надає багатофункціональний інтерфейс для взаємодії з базою даних, дозволяючи виконувати запити та маніпуляції з даними за допомогою об'єктно-орієнтованого

підходу [22]. Це спрощує розробку, підтримку коду та знижує ймовірність помилок.

Автентифікація та авторизація в Laravel вирішуються за допомогою вбудованих функцій, які можуть бути налаштовані та розширені відповідно до специфічних вимог віртуального конфігуратора. Це включає забезпечення безпечного доступу до API, верифікацію користувачів та управління правами доступу до різних частин системи [23]. Система кешування Laravel забезпечує можливість підвищення продуктивності додатку, кешуючи часто запитувані дані та результати обчислень. Використання черг дозволяє оптимізувати виконання тривалих процесів, таких як масова розсилка листів або складні обчислення, переміщуючи їх у фоновий процес.

Інтегровані засоби для тестування в Laravel, включаючи PHPUnit для модульного тестування та Dusk для браузерного тестування, дозволяють автоматизувати перевірку якості коду та забезпечити його надійність перед розгортанням. Усі ці аспекти створюють міцну основу для бекенду, який зможе надійно підтримувати роботу інтерактивного та функціонально багатого фронтенду віртуального конфігуратора.

Архітектурні шари: Архітектура додатку може бути розбита на наступні шари:

- презентаційний рівень (фронтенд),
- бізнес-логіка (контролери),
- доступ до даних (Eloquent ORM),
- рівень моделі (Models).

Презентаційний рівень побудований на базі React для розробки SPA (Single Page Application), що надає користувачам інтерактивний інтерфейс для конфігурації комп'ютерів. Цей рівень відповідальний за відображення інтерфейсу користувачам та обробку користувацьких подій [24]. Логіка, яка обробляє запити від фронтенду, виконує бізнес-правила та маніпулює даними. Вона також забезпечує валідацію та авторизацію дій користувачів. Фреймворк Laravel

використовує Eloquent ORM для спрощення взаємодії з базою даних. Цей рівень дозволяє легко отримувати, вставляти та оновлювати дані в базі без необхідності писати складні SQL-запити. Рівень моделі визначає структуру даних і їхні відносини. Моделі відображають таблиці бази даних і є зв'язуючою ланкою між базою даних і бізнес-логікою додатку.

Забезпечення безпеки додатку має відбуватися на кількох рівнях. На рівні фронтенду це може бути захист від XSS та CSRF атак. Функціонал Laravel надає вбудовані засоби для захисту від таких вразливостей, які можна інтегрувати у ваш фронтенд. Для комунікації між фронтендом та бекендом буде розроблено RESTful API, який повинен бути добре задокументований та використовувати HTTP методи для обробки CRUD операцій (створення, читання, оновлення, видалення).

Обрані засоби підтримують автоматизоване тестування, що є важливим для забезпечення якості коду. Система неперервної інтеграції (CI/CD) може бути використана для автоматичного тестування та розгортання додатку. Загалом, комбінація React і Laravel є вдалим рішенням для створення сучасного веб-додатку віртуального конфігуратора, який буде масштабованим, безпечним та зручним для користувачів.

У розділі представлено опис стеку технологій та засобів розробки, використаних для створення програмного забезпечення віртуального конфігуратора комп'ютерів загального призначення. Проект складається з двох основних частин: фронтенду та бекенду, кожна з яких використовує сучасні інструменти та технології для ефективної розробки та підтримки.

Для розробки фронтенд-частини веб-додатку використовувалась бібліотека React, яка дозволяє створювати інтерактивні користувацькі інтерфейси з використанням компонентного підходу. React сприяє підвищенню швидкості розробки та легкості внесення змін завдяки реактивному оновленню DOM. Розробка велася на мовах програмування JavaScript та TypeScript, останній з яких забезпечує строгу типізацію та підвищену безпеку коду.

Додатково використовувались такі бібліотеки та фреймворки як Ant Design для створення візуально привабливих та функціональних компонентів інтерфейсу, а також React Router для управління маршрутизацією у додатку. Запити до сервера оброблялись за допомогою бібліотеки Axios. За управління станами запитів використовувався React Query, що дозволяє ефективно керувати асинхронними запитами та кешуванням даних.

Для забезпечення якості коду та його відповідності стандартам використовувались такі інструменти як ESLint та Prettier. ESLint допомагає підтримувати чистоту коду, ідентифікувати потенційні помилки та неефективні шаблони, тоді як Prettier забезпечує єдність форматування коду.

Бекенд-частина проєкту була розроблена з використанням фреймворку Laravel, який базується на мові програмування PHP. Цей фреймворк вибрано через його розширені можливості управління бізнес-логікою, що є критично важливим для середовищ розробки складних додатків. Laravel надає розробникам різноманітні інструменти для ефективної роботи з базами даних через Eloquent ORM, що дозволяє здійснювати запити та маніпуляції з даними в базі даних на високому рівні абстракції. Крім того, Laravel забезпечує потужні засоби для маршрутизації запитів, що дозволяє з легкістю визначати маршрути за допомогою зручних засобів Laravel Router. Це сприяє чіткій організації взаємодії між клієнтом і сервером і забезпечує гнучке управління трафіком. Також Laravel включає в себе комплексні механізми для управління сесіями та автентифікації, що є невід'ємною частиною будь-якого захищеного веб-додатку. Laravel Mix, ще один ключовий інструмент у стеку Laravel, використовувався для збірки та оптимізації фронтенд ресурсів, таких як CSS і JavaScript. Він дозволяє інтегрувати та модифікувати різні ресурси, покращуючи загальну продуктивність веб-додатку. Це значно спрощує процес роботи розробників, дозволяючи їм концентруватися на логіці додатку, не хвилюючись про низькорівневі деталі реалізації функціональності.

Завдяки такому вибору технологій, бекенд-частина проекту вирізняється високою продуктивністю, безпекою та легкістю у супроводі та масштабуванні, що робить Laravel одним з найкращих виборів для розробки складних і функціонально багатих веб-додатків. Таким чином, вибір стеку технологій був обумовлений потребою в швидкій розробці, масштабованості проекту та забезпеченні високого рівня безпеки та ефективності взаємодії з користувачем. Використання сучасних інструментів та фреймворків дозволяє ефективно вирішувати поставлені задачі та підтримувати високий рівень продуктивності розробки.

2.5 Висновок до другого розділу

В рамках другого розділу дипломної роботи було проведено детальне проєктування програмного забезпечення для віртуального конфігуратора комп'ютерів загального призначення. Було розглянуто ключові компоненти системи, які включають архітектуру бази даних, вибір технологічного стеку, розробку користувацького інтерфейсу, а також заходи забезпечення безпеки та продуктивності. Модель бази даних виявилась ефективною для зберігання і управління даними завдяки своїй реляційній структурі та гнучкості в модифікації схеми. Використання Laravel як основи для бекенду забезпечило підтримку MVC архітектури та ORM для взаємодії з базою даних та вбудовані функції для автентифікації та авторизації, що критично важливо для безпеки системи. Для розробки клієнтської частини обрано фреймворк React, що дозволило створити динамічний та інтуїтивно зрозумілий інтерфейс, який адаптується під різні пристрої та розміри екранів. В ході проєктування користувацького інтерфейсу було зосереджено на створенні робочого середовища з навігацією та візуалізацією вибору компонентів. Це забезпечує користувачам змогу без зусиль керувати процесом конфігурації, знижуючи ризики потенційних помилок та підвищуючи задоволеність від використання системи користувачем.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВІРТУАЛЬНОГО КОНФІГУРАТОРА КОМП’ЮТЕРІВ ЗАГАЛЬНОГО ПРИЗНАЧЕННЯ

3.1 Розробка бази даних віртуального конфігуратора

Розробка бази даних для віртуального конфігуратора комп'ютерів здійснювалася з урахуванням необхідності забезпечення гнучкості конфігурації компонентів, зберігання історії складаних конфігурацій та управління користувацькими профілями. Структура бази даних віртуального конфігуратора комп'ютерів розроблена з урахуванням необхідності забезпечити ефективне зберігання та швидкий доступ до даних про компоненти, користувачів та конфігурації систем.

На рисунку 3.1 показано список створених таблиць бази даних у вікні інтерфейсу PHPMyAdmin, а фізична схема з відображенням зв’язків показана на рисунку 3.2. Порівнюючи кінцевий варіант схеми із моделлю даних, яка була описана в розділі 2 та зображена на рисунку 2.1, варто відмітити розширений перелік створений в процесі розробки таблиць.

Наведено опис таблиць бази даних у відповідності до розробленої схеми та моделі даних програмного забезпечення.

Центральною для фіксації користувацьких збірок є таблиця builds. Для кожної збірки вона зберігає унікальний ідентифікатор (ID), назву збірки, опис, статус готовності (чи готова збірка до використання або ж вона все ще в процесі налаштування), а також ідентифікатор користувача, який цю збірку створив. Важливим є поле description, де може бути вказано призначення ПК (ігровий, для роботи, для дизайну тощо) та інші важливі примітки. Для реєстрація всіх можливих компонентів, які можна вибрати для збірки використовується таблиця components. Вона містить таку інформацію, як назва компоненту, детальний опис, зокрема технічні параметри, вартість, виробник, а також інші характеристики.

Поле type_id зв'язує кожен компонент з таблицею component_types, дозволяючи класифікувати компоненти по категоріям, таким як процесори, відеокарти тощо. Таблиця 'component_types' визначає категорії або типи компонентів, що дозволяє системі віртуального конфігуратора фільтрувати та представляти користувачам тільки релевантні варіанти під час вибору. Такий підхід полегшує пошук і вибір, а також спрощує процес додавання нових типів компонентів у майбутньому. Для збереження характеристик було створено таблицю component_attributes, яка дозволяє забезпечити детальну специфікацію та забезпечує додаткову інформацію для алгоритмів порівняння та рекомендацій системи.

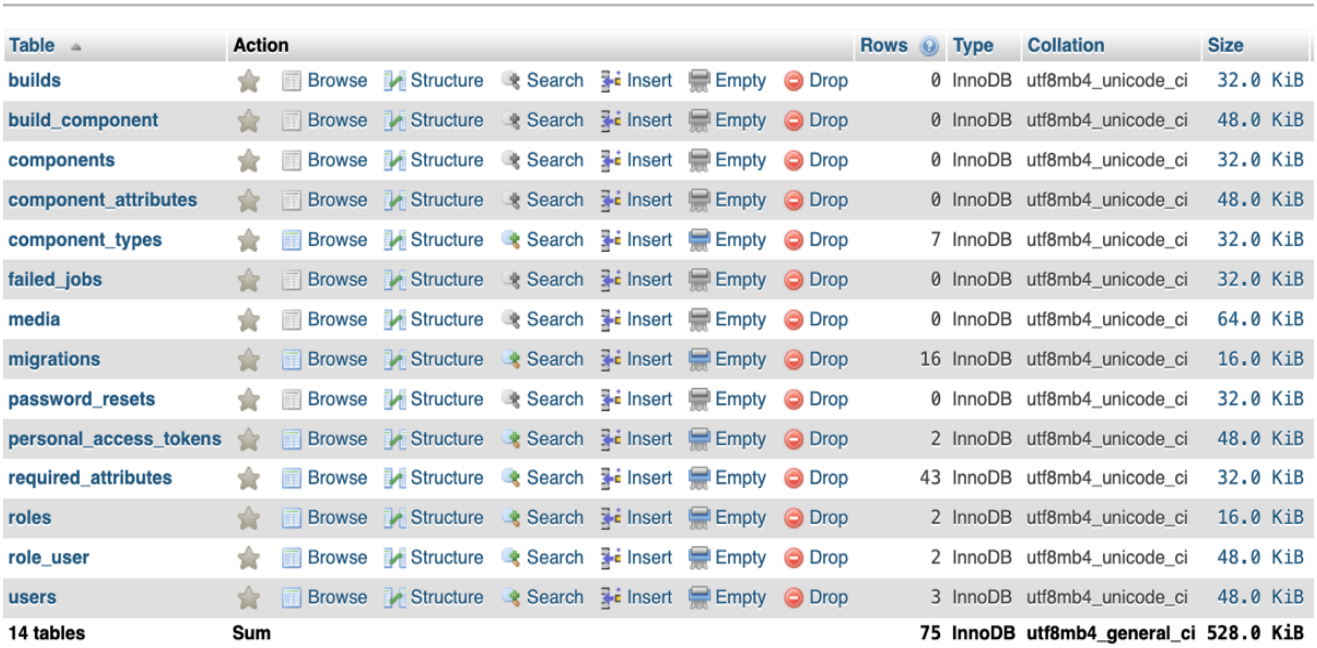


Table	Action	Rows	Type	Collation	Size
builds	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB
build_component	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	48.0 KiB
components	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB
component_attributes	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	48.0 KiB
component_types	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_unicode_ci	32.0 KiB
failed_jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB
media	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	64.0 KiB
migrations	Browse Structure Search Insert Empty Drop	16	InnoDB	utf8mb4_unicode_ci	16.0 KiB
password_resets	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB
personal_access_tokens	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_unicode_ci	48.0 KiB
required_attributes	Browse Structure Search Insert Empty Drop	43	InnoDB	utf8mb4_unicode_ci	32.0 KiB
roles	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_unicode_ci	16.0 KiB
role_user	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_unicode_ci	48.0 KiB
users	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	48.0 KiB
14 tables	Sum	75	InnoDB	utf8mb4_general_ci	528.0 KiB

Рисунок 3.1 – Структура таблиць БД

До службових таблиць з параметрами користувачів відносяться таблиці users, roles, role_user. Таблиця користувачів (users) є важливою для забезпечення безпеки та індивідуалізації досвіду кожного користувача. Вона зберігає особисту інформацію користувачів, їх роль у системі та стан автентифікації. Ролеву модель управління доступом забезпечують таблиці roles та role_user. Вони визначають, які ролі існують у системі (наприклад, адміністратор, зареєстрований користувач, гість тощо) та які користувачі відповідають цим ролям [16]. Це дозволяє

налаштувати дозволи на певні дії в системі, забезпечуючи гранулювання прав доступу та підвищення безпеки.

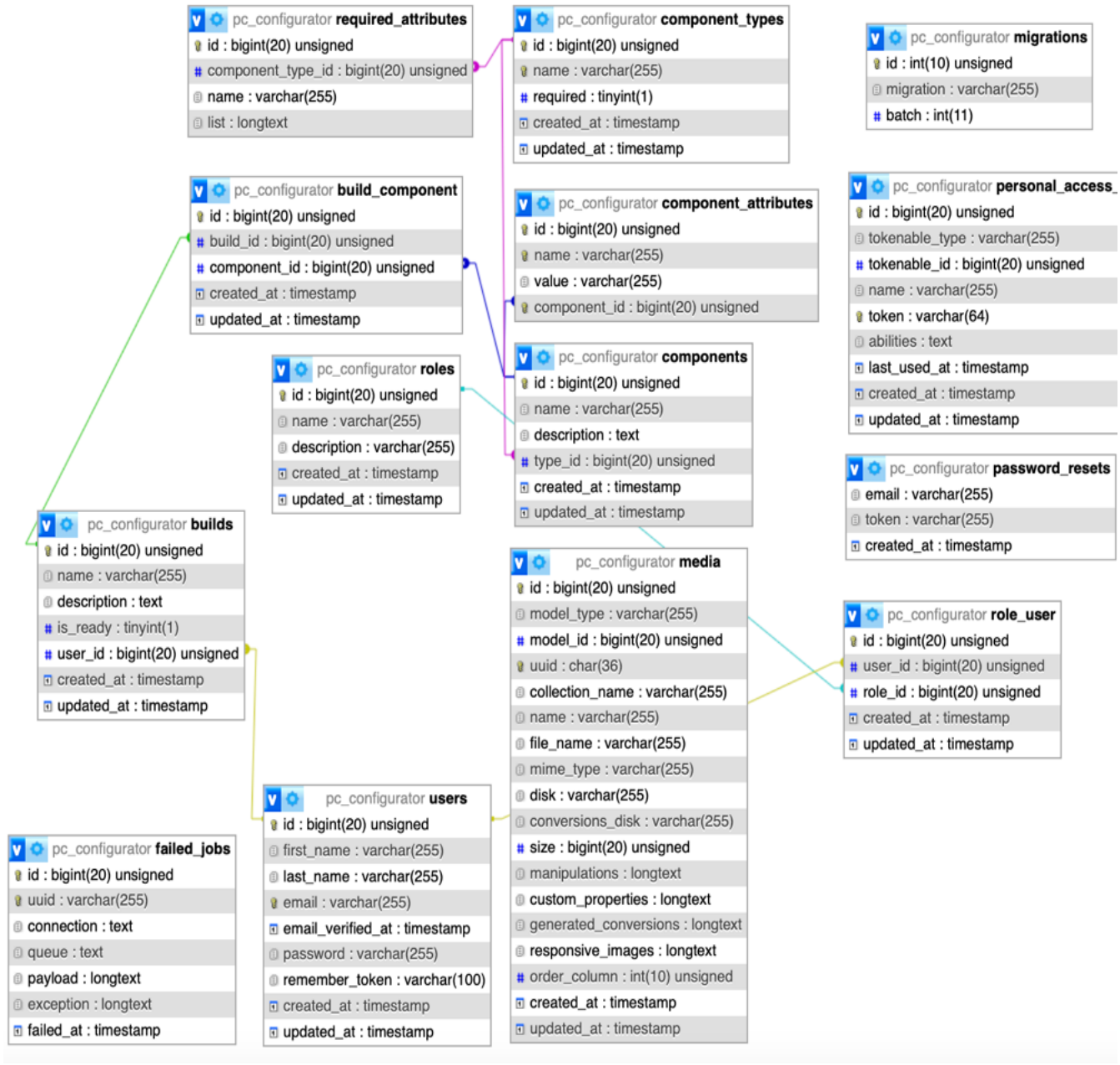


Рисунок 3.2 – Схема бази даних

Відповідно до фізичної схеми бази даних, яка зображена на рисунку 3.2, таблиця users є головною таблицею, що зберігає особисті дані користувачів системи. Кожен користувач має унікальний ідентифікатор та пов'язані атрибути, які включають ім'я, прізвище, електронну пошту та хешований пароль. Важливим

елементом є зв'язок users із таблицею roles, який визначає рівень доступу користувача в системі. Визначення ролей, які можуть бути призначені користувачам, міститься в таблиці roles, забезпечуючи гнучке управління правами та доступами до різних частин системи.

Інформації про складені користувачами конфігурації комп'ютерів зберігається в таблиці builds, яка включає перелік характеристик про кожен компонент, який був вибраний користувачем, та зберігає статус готовності конфігурації. Водночас основною таблицею яка містить інформацію про всі компоненти, доступні для вибору в конфігураторі, є таблиця components. Кожен запис у цій таблиці включає назву компонента, опис та зв'язок з типом компонента в таблиці component_types, яка зберігає категорії або класифікації компонентів, такі як процесор, материнська плата, відеокарта тощо, що дозволяє системі диференціювати компоненти та застосовувати специфічні правила валідації або сумісності. Характеристики кожного компонента комп’ютера, наприклад, об'єм пам'яті для оперативної пам'яті або частоту процесора, зберігаються в таблиці component_attributes. Зв'язуючою таблицею між таблицями builds і components є таблиця build_component, в якій міститься інформація про відстеження конкретних компонентів, що входять до складу кожної конфігурації. Набір атрибутів, які є обов'язковими для кожного типу компоненту, зберігається в таблиці required_attributes. Ця таблиця забезпечує систему інформацією про те, які параметри необхідно зазначити при створенні або редагуванні конфігурації.

Додаткові системні таблиці, такі як migrations, password_resets, failed_jobs, personal_access_tokens, media, виконують важливі функції управління інфраструктурою, безпекою, інтеграцією засобів зв'язку та обробки зображень, що в цілому сприяє підвищенню стабільності та безпеки роботи віртуального конфігуратора. Кожна із цих таблиць спроектована таким чином, щоб забезпечити не тільки поточні потреби користувацької аудиторії, але й можливість гнучкого розширення та легкої інтеграції з іншими системами, а також здатність швидко адаптуватися до змін у галузі комп'ютерних технологій.

Проектуючи базу даних, ми створюємо фундамент для всіх подальших рівнів віртуального конфігуратора, від взаємодії з користувачем до аналітики даних і прийняття рішень на основі зібраної інформації.

Для ефективної взаємодії з базою даних під час розробки програмного забезпечення віртуального конфігуратора комп'ютерів використовувалась складна SQL-логіка. Це дозволило забезпечити оперативний доступ до даних та їхнє ефективне управління, інтегруючи стандартні CRUD операції, що охоплюють створення, читання, оновлення, та видалення даних. Особливу увагу було приділено складним JOIN операціям, які дозволяли збирати детальну інформацію про компоненти системи, використовуючи дані з таблиць, таких як builds, components, component_types і component_attributes. Для вибіркового пошуку компонентів за специфікаціями застосовувались запити з умовами WHERE, а складні запити, які аналізують сумісність компонентів, таких як тип процесора та материнська плата, демонструють взаємодію між різними таблицями і логічні умови.

Ініціалізація бази даних та її первинне наповнення здійснювалися за допомогою сідерів Laravel, які виконували автоматичне наповнення бази даних даними для тестування. Клас UserFactory використовувався для генерації тестових профілів користувачів, що включало створення імен, електронних адрес, та хешованих паролів, що є ключовим для тестування системи автентифікації та управління користувацькими профілями. Також, клас RequiredAttributeSeeder використовувався для наповнення бази даних детальною інформацією про обов'язкові атрибути компонентів, забезпечуючи структуру, необхідну для валідації конфігурацій.

Лістинг класу User Factory:

```
namespace Database\Factories;
use Illuminate\Database\Eloquent\Factories\Factory;
use Illuminate\Support\Str;
use Illuminate\Support\Facades\Hash;
```

```
/** * Клас UserFactory використовується для автоматичного створення тестових профілів користувачів. */
class UserFactory extends Factory
{
    /**
     * Определяє стандартний стан моделі.
     * @return array
     */
    public function definition()
    {
        return [
            'first_name' => $this->faker->firstName(),
            'last_name' => $this->faker->lastName(),
            'email' => $this->faker->unique()->safeEmail(),
            'email_verified_at' => now(),
            'password'  =>  Hash::make('password'),     // Стандартний пароль для всіх тестових профілів
            'remember_token' => Str::random(10),
        ];
    }
    /**
     * Позначає, що email користувача не повинен бути підтвердженим.
     *                                                  @return \Illuminate\Database\Eloquent\Factories\Factory
     */
    public function unverified()
    {
        return $this->state(function (array $attributes) {
            return [
```

```
                'email_verified_at' => null,
            ];
        });
    }
}
```

Налаштування та оптимізація бази даних були виконані для забезпечення високої продуктивності та надійності роботи конфігуратора, де ретельна настройка параметрів з'єднання та оптимізація виконання запитів сприяли підвищенню ефективності взаємодії з базою даних.

Лістинг конфігураційного файлу Database.php:

```
<?php
use Illuminate\Support\Str;
return [
    'default' => env('DB_CONNECTION', 'mysql'),
    'connections' => [
        'sqlite' => [
            'driver' => 'sqlite',
            'url' => env('DATABASE_URL'),
            'database'        =>        env('DB_DATABASE',
database_path('database.sqlite')),
            'prefix' => '',
            'foreign_key_constraints'                    =>
env('DB_FOREIGN_KEYS', true),
        ],
        'mysql' => [
            'driver' => 'mysql',
            'url' => env('DATABASE_URL'),
            'host' => env('DB_HOST', '127.0.0.1'),
            'port' => env('DB_PORT', '3306'),
            'database' => env('DB_DATABASE', 'forge'),
```

```
            'username' => env('DB_USERNAME', 'forge'),
            'password' => env('DB_PASSWORD', ''),
            'unix_socket' => env('DB_SOCKET', ''),
            'charset' => 'utf8mb4',
            'collation' => 'utf8mb4_unicode_ci',
            'prefix' => '',
            'prefix_indexes' => true,
            'strict' => true,
            'engine' => null,
            'options'  =>  extension_loaded('pdo_mysql')  ?
array_filter([
                PDO::MYSQL_ATTR_SSL_CA                    =>
env('MYSQL_ATTR_SSL_CA'),
            ]) : [],
        ],
    ],
    'migrations' => 'migrations',
    'redis' => [
        'client' => env('REDIS_CLIENT', 'phpredis'),
        'options' => [
            'cluster' => env('REDIS_CLUSTER', 'redis'),
            'prefix'        =>        env('REDIS_PREFIX',
Str::slug(env('APP_NAME', 'laravel'), '_').'_database_'),
        ],
        'default' => [
            'url' => env('REDIS_URL'),
            'host' => env('REDIS_HOST', '127.0.0.1'),
            'password' => env('REDIS_PASSWORD'),
            'port' => env('REDIS_PORT', '6379'),
            'database' => env('REDIS_DB', '0'),
```

```
        ],
        'cache' => [
            'url' => env('REDIS_URL'),
            'host' => env('REDIS_HOST', '127.0.0.1'),
            'password' => env('REDIS_PASSWORD'),
            'port' => env('REDIS_PORT', '6379'),
            'database' => env('REDIS_CACHE_DB', '1'),
        ],
    ],
];
```

Конфігураційний файл config/database.php є вихідним пунктом для встановлення параметрів з'єднання з різними СКБД. В цьому файлі можна вказати, наприклад, інформацію про хост, порт, ім'я бази даних, ім'я користувача та пароль для доступу до СКБД. Також тут визначаються додаткові параметри, такі як тип кодування – зазвичай utf8mb4, яке є стандартом для сучасних веб-додатків, що підтримують різноманітні символи. Параметри префіксів таблиць дозволяють уникати конфліктів імен при використанні однієї бази даних для кількох проектів.

Індексація є одним з найефективніших способів оптимізації запитів до бази даних. Шляхом створення індексів для ключових стовпців, часто використовуваних у запитах, зменшується час пошуку необхідної інформації. Важливо вибирати стовпці для індексації розумно, оскільки надмірна індексація може сповільнити операції вставки, оновлення та видалення через необхідність відновлення індексів. Для зменшення навантаження на базу даних при високій частоті однотипних запитів застосовується кешування. Redis використовується як швидкий кеш-сервер, що зберігає результати запитів у пам'яті. Це дозволяє миттєво отримувати відповіді на часті запити, не навантажуючи при цьому базу даних повторними дорогими операціями читання. Кешування особливо ефективне для даних, що не змінюються часто, але часто запитуються – наприклад, списки

доступних компонентів або типів компонентів у конфігураторі. Управління оптимізацією та кешуванням вимагає ретельного моніторингу та аналізу продуктивності системи, щоб забезпечити баланс між швидкодією і актуальністю даних. За допомогою таких інструментів як Query Builder та Eloquent ORM у Laravel, можна детально визначити, які саме запити повинні кешуватися, та настроїти термін їх зберігання в кеші.

Вищеописані структура та методи роботи з базою даних були вибрані для забезпечення високої гнучкості та масштабованості віртуального конфігуратора комп'ютерів, дозволяючи користувачам з легкістю створювати та модифікувати їх конфігурації згідно з їх потребами.

3.2 Розробка серверної частини

Розробка серверної частини програми віртуального конфігуратора комп'ютерів на базі фреймворку Laravel була зосереджена на створенні чіткої, логічно організованої структури проєкту. Ця структура забезпечувала основу для модульного підходу до розробки, дозволяючи окремо працювати над кожною частиною системи та ефективно інтегрувати її з іншими компонентами.

Ключовим елементом архітектури були класи, які відображали основні елементи комп'ютерної конфігурації. Кожен такий клас був ретельно спроєктований з властивостями, що відповідали технічним характеристикам цих компонентів, наприклад, тактова частота процесора, кількість ядер, тип і обсяг пам'яті, типи роз'ємів та форм-фактори для материнських плат тощо. Методи класів дозволяли здійснювати такі операції, як пошук за параметрами, порівняння характеристик та керування залежностями між різними компонентами при складанні конфігурації.

Лістинг створеного компонента CPU.php:

```
<?php
namespace App\Core\Components\CPU;
use App\Core\Components\Component;
```

```
use App\Core\Components\RAM\RAMType;
class CPU extends Component
{
    public readonly array $attributes;
    public readonly string $name;
    public function __construct()
    {
        $this->name = 'cpu';
        $this->attributes = [
            'cors' => [],
            'threads' => [],
            'socket' => [],
            'memory_type'        =>        ['list'        =>
array_column(RAMType::cases(), 'value')],
            'energy_consumption' => []
        ];
    }
}
```

Для вибірки та фільтрації даних за певними критеріями були розроблені спеціалізовані інструменти, такі як Eloquent Query Scopes та кастомні фільтри, що вбудовувалися в класи моделей. Це дозволило створювати складні запити з високою гнучкістю і швидкістю, мінімізуючи повторення коду та полегшуючи подальше масштабування проекту. Інтеграція цих компонентів із базою даних була виконана через механізм Object-Relational Mapping (ORM) засобами Eloquent, який дозволяв елегантно та безпосередньо взаємодіяти з базою даних, не залучаючи складний SQL-синтаксис. Це забезпечило не лише швидкість розробки, а й високий рівень абстракції, що сприяло підтримці коду та його читабельності.

Маршрутизація в Laravel є фундаментальною частиною дизайну будь-якого веб-додатку, вона визначає, як додаток відповідає на різні HTTP запити від користувачів. У контексті віртуального конфігуратора комп'ютерів, файл routes/api.php служить картою, що спрямовує запити до відповідних контролерів, таких як BuildController, ComponentController, і ComponentTypeController.

Лістинг компоненту Api.php:

```
<?php
*/
// COMPONENTS
Route::prefix('components')->group(function () {
    Route::get('/', [ComponentController::class, 'index']);
    Route::post('/',           [ComponentController::class,
'store'])->middleware(['role:admin', 'auth:sanctum']);
    Route::post('/{id}',       [ComponentController::class,
'update'])->middleware(['role:admin', 'auth:sanctum']);;
    Route::delete('/{id}',     [ComponentController::class,
'destroy'])->middleware(['role:admin', 'auth:sanctum']);;
});
// COMPONENT TYPES
Route::get('/component-types/{type}/attributes',
[ComponentTypeController::class, 'getRequiredAttributes']);
Route::get('/component-types/',
[ComponentTypeController::class, 'index']);
// AUTH
Route::post('/login', [AuthController::class, 'login']);
Route::post('/register',           [AuthController::class,
'register']);
Route::get('/logout', [AuthController::class, 'logout']);
// USER
Route::middleware(['auth:sanctum'])->group(function () {
```

```
    Route::post('/user',            [UserController::class,
'update']);
    Route::get('/users', [UserController::class, 'index']);
    Route::get('/users/{id}',       [UserController::class,
'show'])->where('id', '[0-9]+');
    Route::get('/users/me',         [UserController::class,
'getAuthUser']);
});
})->name('local.temp');
```

Кожен маршрут у цьому файлі асоційований з певним HTTP методом (наприклад, GET, POST, PUT, DELETE) і вказує на метод в контролері, який має бути викликаний при цьому запиті.

Таким чином, коли користувач здійснює запит до API, наприклад, для отримання списку компонентів або створення нової конфігурації, Laravel використовує визначення маршрутів для визначення, який контролер і метод повинен обробити цей запит.

Контролери є посередниками між запитом користувача та логікою додатку. Вони приймають вхідні дані запиту, обробляють їх з використанням різних сервісів і повертають відповіді.

Наприклад, ComponentController може використовувати ComponentService для роботи з даними компонентів – здійснювати пошук в базі даних, виконувати валідацію даних, отриманих від користувача, та ініціювати створення нових записів або оновлення існуючих.

Лістинг компоненту ComponentService:

```
<?php
class ComponentService
{
    public      function      getComponents(ComponentQuery
$componentQuery)
```

```
    {
        $components = Component::filter($componentQuery)
            ->with(['attributes', 'type'])
            ->get();
        return $components->map(fn($component) => GetComponentDto::fromModel($component))->values();
    }
```

ComponentService та інші подібні сервіси є частиною бізнес-логіки програми і відповідають за виконання конкретних операцій з даними. Вони інкапсулюють код, необхідний для маніпуляції даними, що робить цей код легшим для підтримки та модифікації.

Такий підхід дозволяє зберігати контролери неперевантаженими надмірним функціоналом поміщаючи в них лише ту логіку, яка необхідна для обробки запитів і відповідей. Це також сприяє розділенню відповідальності в коді, де кожен компонент системи відповідає лише за свою частину роботи.

Особлива увага приділялася безпеці додатку, включаючи автентифікацію користувачів, валідацію даних та очистку введення (sanitization).

Для цього використовувалися вбудовані функції Laravel, такі як валідатори запитів та middleware для перевірки автентифікації.

В цілому, серверна частина віртуального конфігуратора була розроблена з урахуванням принципів чистого коду, гнучкості та розширюваності, що забезпечувало надійну та ефективну основу для функціонування всієї системи.

3.3 Розробка та опис інтерфейсу користувача

Розробка інтерфейсу користувача для віртуального конфігуратора комп'ютерів фокусується на забезпеченні інтуїтивної взаємодії та зручності для користувачів під час збірки їхніх ідеальних систем.

Використання сучасних технологій та підходів у дизайні дозволило створити ефективний та естетично привабливий інтерфейс. Першим кроком у взаємодії з конфігуратором є процес автентифікація.

Форма входу, яка забезпечує простий та безпечний спосіб входу в систему, зображена на рисунку 3.3.

Користувач може ввести свої логін і пароль, а також пройти процес реєстрації. Цей компонент розроблено з особливою увагою до безпеки та зручності використання.

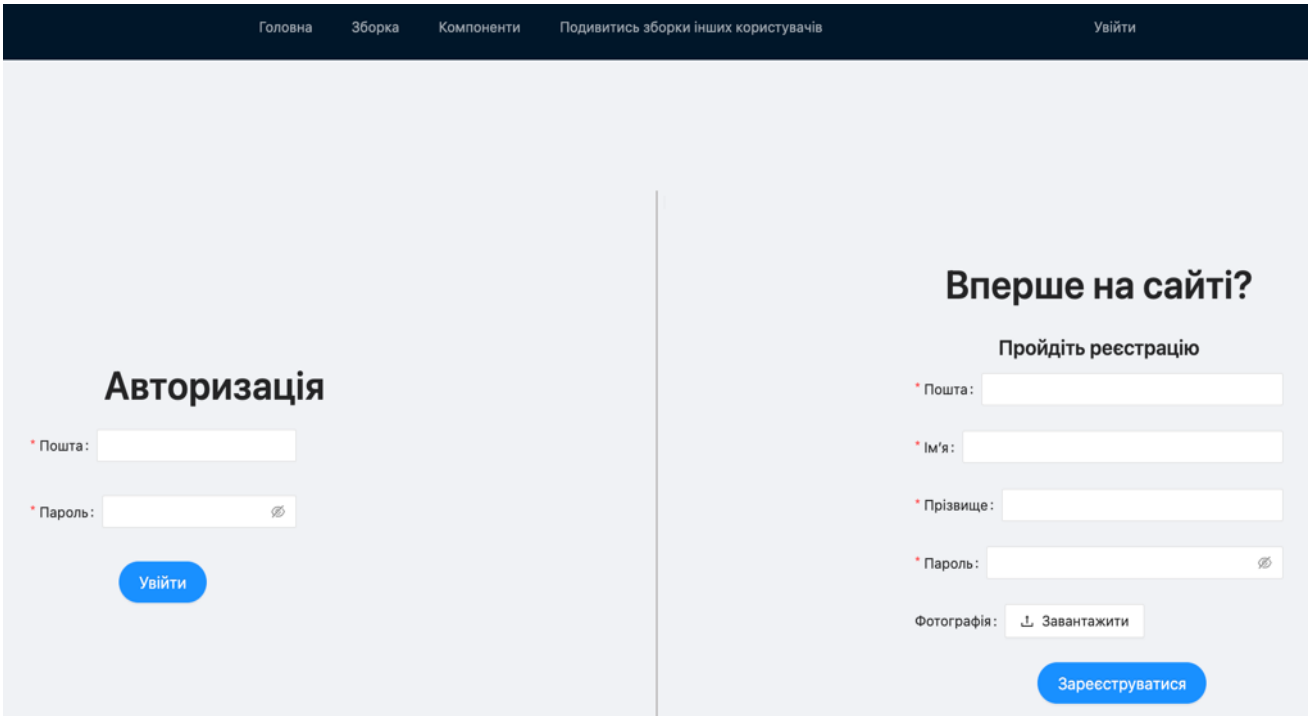


Рисунок 3.3 – Сторінка логіну і реєстрації

На наступному кроці користувач переходить до вибору основних компонентів для конфігурування комп'ютерів.

Інтерфейс дозволяє вибирати між різними опціями, такими як процесори, відеокарти, і материнські плати.

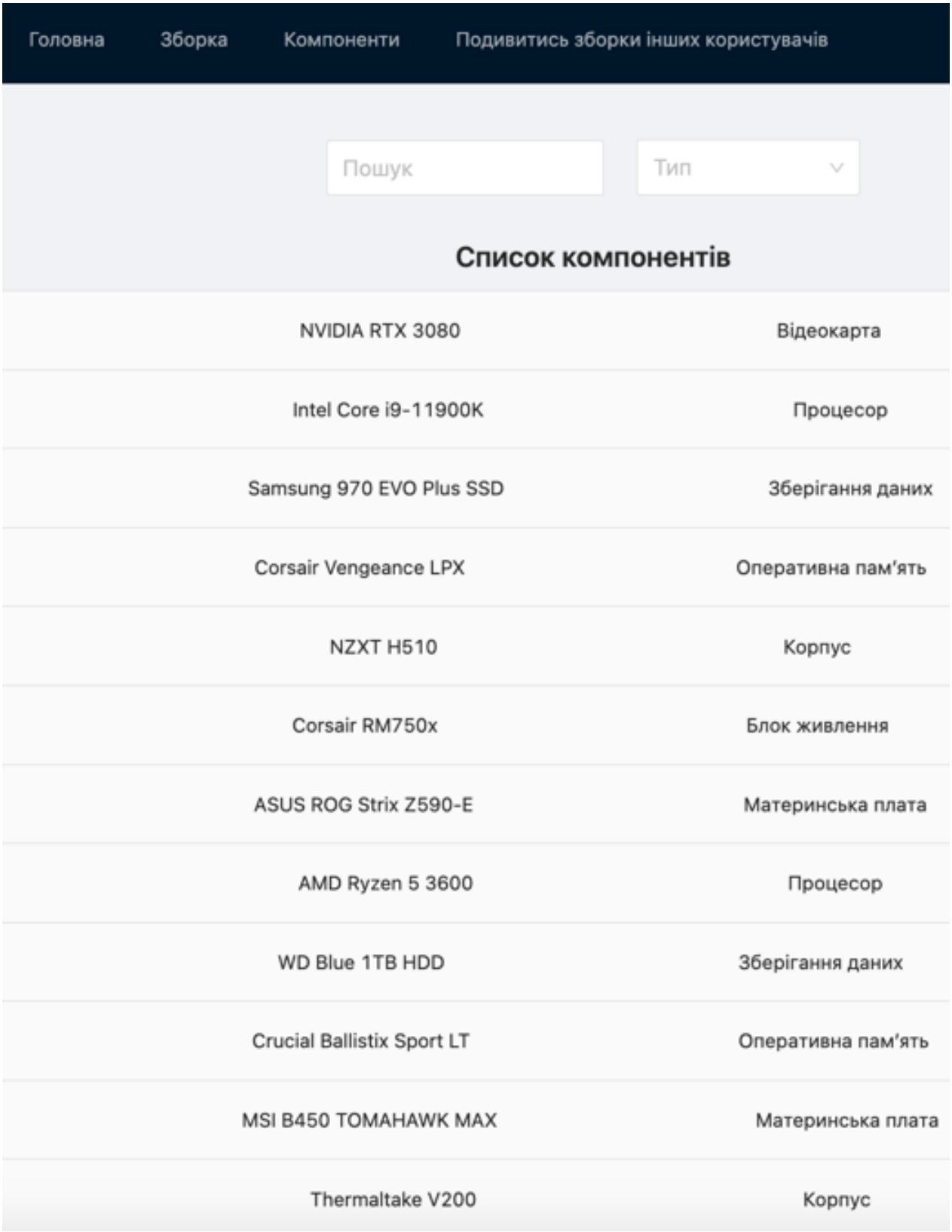


Рисунок 3.4 – Список компонентів

Ці компоненти представлені у формі списків, що показано на рисунку 3.4, з можливістю детального опису та специфікацій, забезпечуючи користувачам усю необхідну інформацію для інформованого вибору (див. рисунок 3.5).

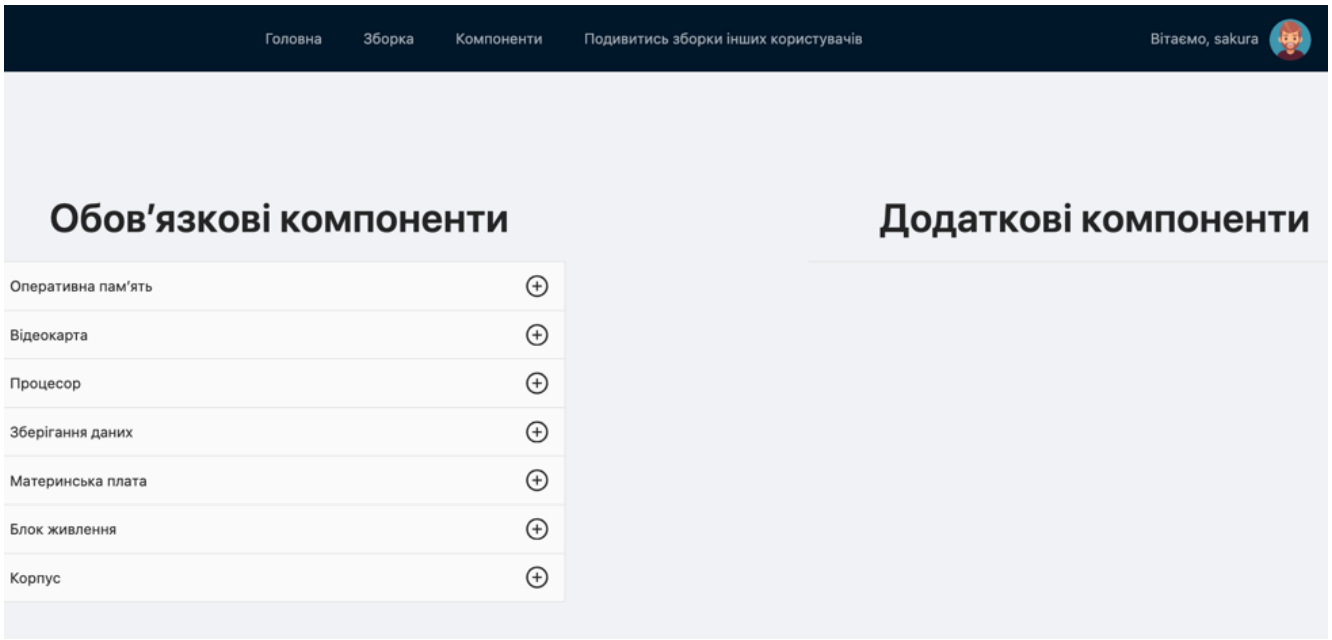


Рисунок 3.5 – Створений компонент

Після вибору основних компонентів, користувач є змогу переглядати та керувати списком вибраних елементів, як це показано на рисунку 3.6. Ця частина інтерфейсу відображає усі компоненти, які були додані до конфігурації, дозволяючи легко модифікувати або видаляти елементи перед фінальним замовленням.

Це важливий крок, що дозволяє користувачам фіналізувати свої конфігурації перед покупкою.

Решта елементів графічного інтерфейсу віртуального конфігуратора подано в Додатку Г.

Цілісність і зрозумілість інтерфейсу сприяють кращому залученню користувачів і підвищенню їх задоволеності від процесу збору власного комп'ютера. Кожен компонент інтерфейсу був ретельно спроектований для

забезпечення оптимальної взаємодії з користувачем, використовуючи сучасні веб-технології для створення гнучких користувацьких інтерфейсів.

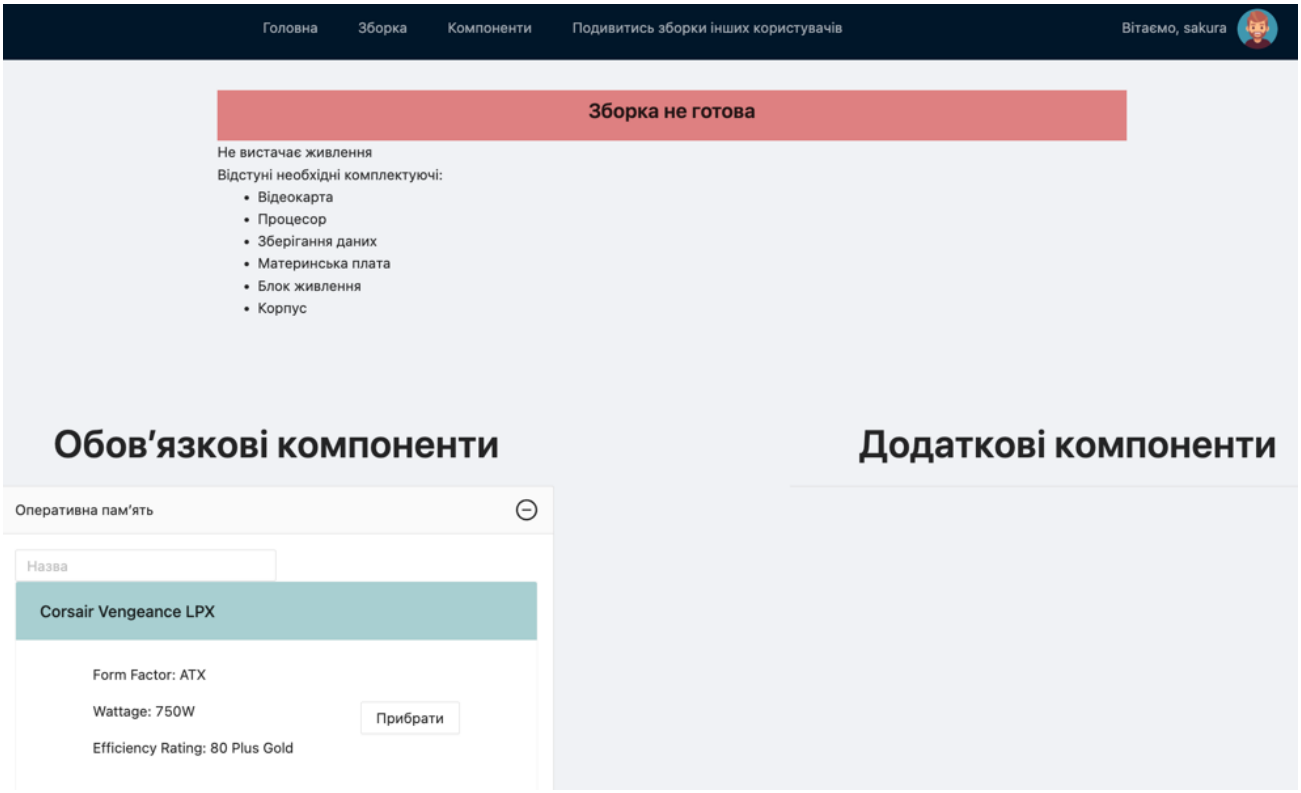


Рисунок 3.6 – Приклад використання

3.4 Розгортання та тестування програмного забезпечення

Для тестування розробки програмного забезпечення віртуального конфігуратора комп'ютерів було застосовано серверну платформу XAMPP, інтерфейс панелі керування якої показано на рисунку 3.7.

Платформа XAMPP є інтегрованим пакетом, що включає веб-сервер Apache, систему керування базою даних (СКБД) MySQL, а також інтерпретатори мов програмування PHP та Perl.

Ця платформа забезпечує швидке встановлення та налаштування робочого середовища, що дозволяє розробнику заощадити час, на конфігурацію кожного з компонентів окремо. Однією з основних переваг платформи XAMPP є її кросплатформеність.

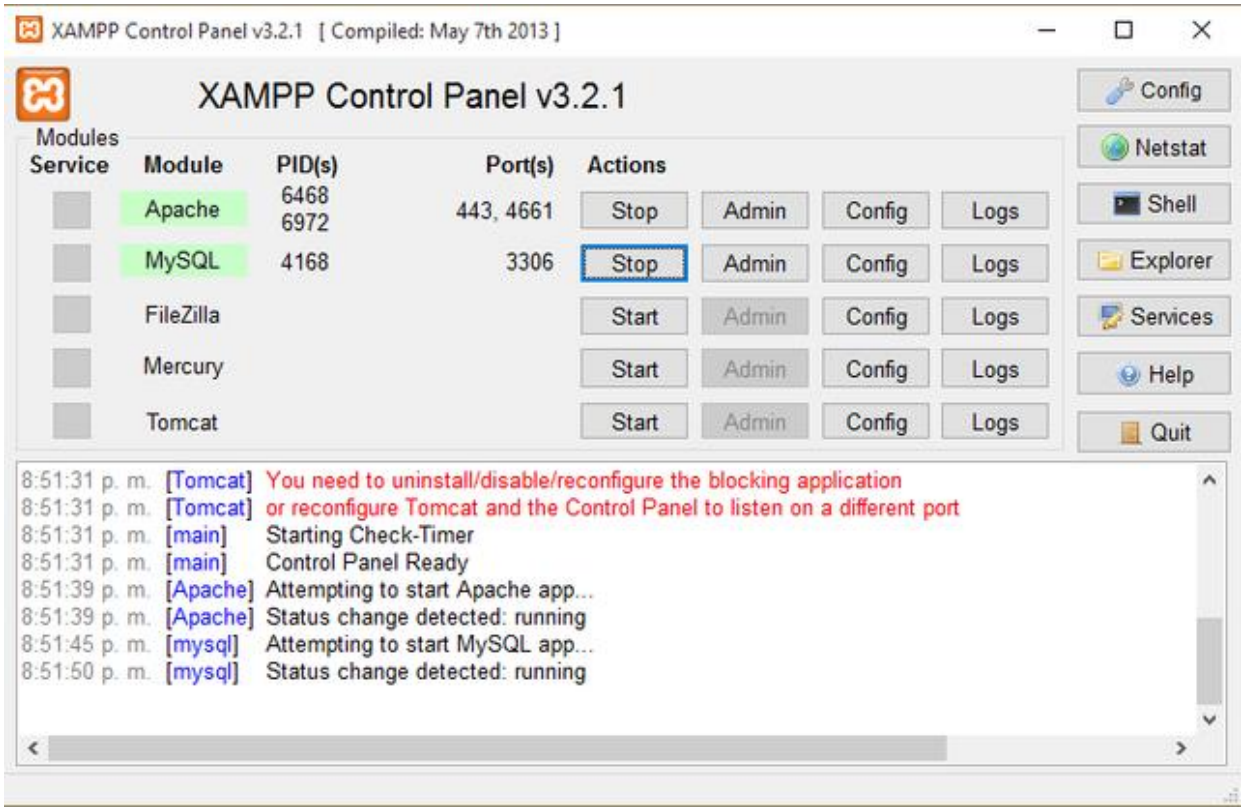


Рисунок 3.7 – Інтерфейс панелі керування XAMPP

Ця платформа підтримується на більшості операційних систем (ОС), таких як Windows, Linux, і macOS, що дозволяє розробникам використовувати однакове середовище незалежно від ОС розробника. Така універсальність істотно спрощує процес розгортання та подальшого супроводу програмного забезпечення.

Крім того, XAMPP має велику спільноту користувачів та розробників, які постійно оновлюють платформу, виправляючи помилки та додаючи нові функції.

Це створює сприятливе середовище для швидкого отримання допомоги або консультацій, а також доступу до великої кількості навчальних матеріалів та ресурсів.

Широке використання XAMPP у спільноті також означає, що багато поширених проблем та їх рішення вже добре документовані, що може сприяти ефективнішому вирішенню технічних питань під час розробки.

В контексті роботи, XAMPP був вибраний як серверна платформа з огляду на те, що цільовим середовищем є веб-додаток, який вимагає стабільного та

надійного сервера з підтримкою сучасних технологій веб-розробки. Вибір XAMPP як платформи для розгортання дозволив розробникам використовувати такі інструменти як PHPMyAdmin для управління базами даних, а також впроваджувати сучасні бекенд-рішення, написані на PHP.

За допомогою XAMPP можна імплементувати комплексну архітектуру веб-додатку, що забезпечує високу продуктивність, безпеку та легкість у супроводженні.

Розгортання програмного забезпечення на локальному рівні є ключовим етапом підготовки до розробки та тестування додатків.

У випадку проектів на базі PHP, особливо тих, які побудовані з використанням фреймворку Laravel, перший крок полягає у встановленні та налаштуванні стеку програмного забезпечення, як-от XAMPP. XAMPP надає уніфіковане середовище, яке об'єднує Apache як веб-сервер, MySQL як систему управління базами даних, а також PHP як мову програмування.

Опишемо процес встановлення програмного забезпечення XAMPP для розгортання середовища розробки і тестування.

1. Завантаження останньої версії XAMPP із офіційного сайту та виконання інсталяційного файлу з вибором компонентів для встановлення.

2. Визначення директорії для інсталяції (зазвичай C:\xampp для ОС Windows).

3. Запуск XAMPP Control Panel для запуску Apache та MySQL (Рисунок 3.8).

4. Копіювання прикладу **.**env.example файла і створення нового .env файлу.

6. Введення деталей для підключення до бази даних, таких як ім'я бази, користувач та пароль.

7. Налаштування інших змінних середовища, таких як додаткові конфігурації пошти, шляхів для логування тощо.

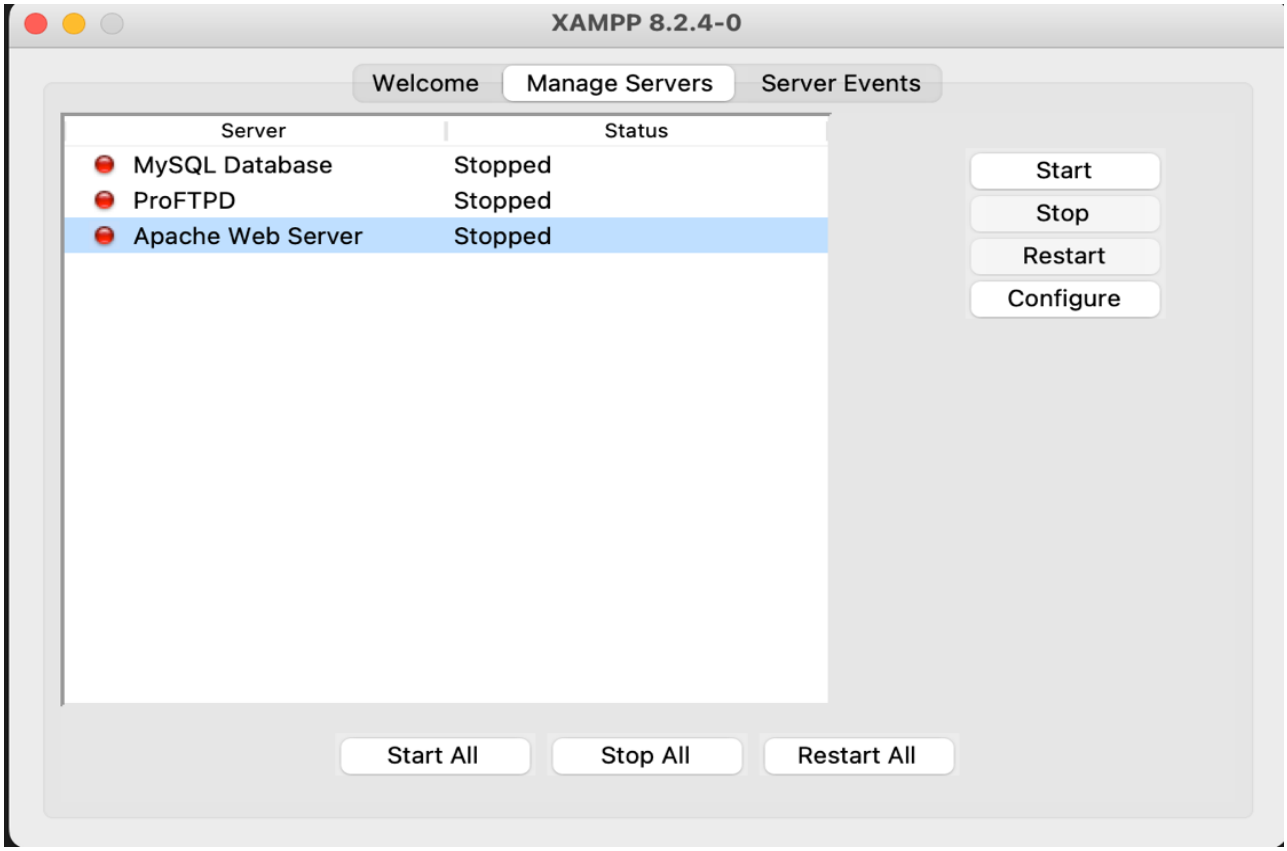


Рисунок 3.8 – Інтерфейс XAMPP

Зазвичай, XAMPP налаштовується таким чином, що сервер може бути з параметрами за замовчуванням. Однак, у деяких випадках може знадобитись додаткове налаштування, наприклад, для встановлення віртуального хосту, який забезпечує доступ до локального проекту за власним URL.

Фрагмент конфігураційного файлу наведено в лістингу:

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=
APP_DEBUG=true
APP_URL=http://localhost
APP_NAME=Laravel
APP_ENV=local
APP_KEY=
```

```
APP_DEBUG=true
APP_URL=http://localhost
LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=pc_configurator
DB_USERNAME=root
DB_PASSWORD=
BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
```

Для встановлення залежностей використано менеджер пакетів Composer, використовуючи командний рядок або термінал та перейшовши до кореневої директорії проєкту. Composer прочитає файл composer.json та встановить всі необхідні PHP бібліотеки та пакети, що потрібні для проекту.

Для створення структури бази даних було використано інструменти міграції Laravel, які запускаються командою php artisan migrate. Це дозволяє автоматично створити таблиці з необхідною структурою. Після успішного виконання міграцій можна розпочати сам процес розробки або тестування.

Тестування програмного забезпечення в Laravel є систематичним процесом, який забезпечує високий рівень впевненості у якості та стабільності коду. Із використанням вбудованих засобів, як PHPUnit та трейт RefreshDatabase, Laravel дозволяє розробникам впроваджувати комплексні тести для перевірки різних аспектів API. Трейт RefreshDatabase гарантує, що для кожного тесту буде

створено свіжу копію бази даних, з якою буде проводитись ізольоване тестування. Це означає, що зміни в базі даних, внесені під час одного тесту, не впливатимуть на інші тести, запобігаючи таким чином помилкам, що можуть виникнути через пошкодження даних.

Тестовий Клас APITest створює основу для написання тестових випадків, дозволяючи розробникам описувати сценарії, за яких відбувається взаємодія з API. Тестові методи всередині класу APITest, такі як testRegister, testLogin, testRegisterWithErrorUserExist, виконують HTTP-запити до відповідних кінцевих точок API і перевіряють, чи результати відповідають очікуваним. Наприклад:

- testRegister перевіряє, що при надсиланні правильних даних для реєстрації користувача, API відповідає статусом 201 (Created) і видає токен доступу;
- testLogin валідує можливість входу в систему з вірними обліковими даними, переконуючись, що відповідь містить токен доступу та має статус 200 (OK);
- testRegisterWithErrorUserExist використовується для тестування відповіді системи на спробу реєстрації користувача з уже існуючим email, переконуючись, що API повертає помилку зі статусом 404 (Not Found) або іншим статусом помилки, який вказує на конфлікт даних.

Ці тести є прикладами автоматизованих перевірок, які можуть бути інтегровані у процес неперервної інтеграції (CI) для постійного моніторингу якості коду.

Вони можуть бути викликані через інтерфейс командного рядка за допомогою команди php artisan test аб о можуть бути заплановані та автоматизовані за допомогою інструментів CI, таких як Jenkins, GitLab CI або GitHub Actions.

За допомогою цих інструментів можна налаштувати сценарії, які будуть запускати тести автоматично після кожного коміту або злиття змін в репозиторій, забезпечуючи постійну перевірку стану проєкту та швидке виявлення та виправлення помилок.

Крім стандартних тестів, які перевіряють взаємодію з API, Laravel також підтримує функціональне тестування, яке імітує поведінку користувача в браузері.

З використанням Laravel Dusk розробники можуть створювати тести для автоматичного керування браузером та перевірки елементів веб-сторінок, надсилання форм, кліків по посиланнях та інших дій користувача.

Результати тестування показані на рисунку 3.9.

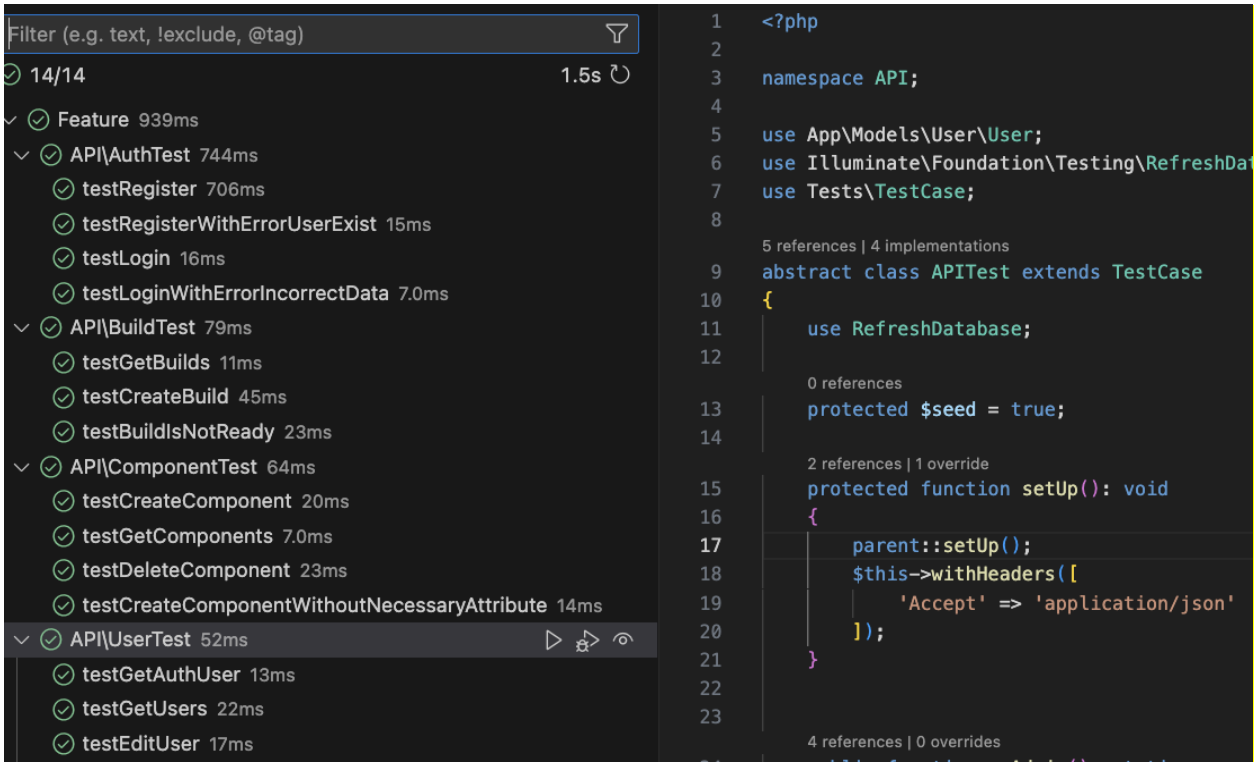


Рисунок 3.9 – Результати тестування

Коди тестових сценаріїв знаходяться у Додатку Д. Тестування у Laravel не обмежується лише бекендом. Vue.js, React та інші JavaScript фреймворки, які можуть бути використані для розробки фронтенду, також мають свої інструменти для модульного та кінцевого тестування.

Це дозволяє охопити тестуванням повний стек технологій, використовуваних у проєкті.

Підсумовуючи, розгорнутий процес тестування є важливим для підтримання високого рівня якості коду. Він допомагає виявити та усунути помилки на ранніх стадіях розробки та забезпечує впевненість у тому, що нові функції не порушують існуючу функціональність.

Це створює основу для стабільного та надійного програмного забезпечення, яке може бути легко адаптоване та розширене з часом.

3.5 Висновки до третього розділу

У третьому розділі було детально розглянуто аспекти реалізації та тестування програмного забезпечення для віртуального конфігуратора комп'ютерів загального призначення.

Важливим моментом стала побудова надійної серверної платформи на основі Laravel, яка забезпечила стабільну основу для всієї системи та ефективне управління базою даних.

Опис технологічного стеку та засобів розробки дав змогу глибше зрозуміти вибір інструментів та їхню роль у створенні проекту. Висвітлення реалізації програмних компонентів конфігуратора підтвердило важливість модульної архітектури, яка дозволяє гнучко додавати та модифікувати функціональні частини системи.

Розгортання локального сервера з використанням XAMPP та налаштування середовища через файл .env продемонстрували ефективність використання стандартизованих середовищ, що спрощує процес розробки та подальшої підтримки проекту.

Тестування зайняло особливе місце у процесі розробки.

Використання автоматизованих тестів, що виконуються через PHPUnit, та трейтів Laravel, таких як RefreshDatabase, забезпечило якість коду та його відповідність поставленим вимогам.

Проведені тести охопили функціонал API, від реєстрації та автентифікації користувачів до створення та управління конфігураціями комп'ютерів, що гарантує надійність розробленої системи..

ВИСНОВКИ

Завершуючи роботу на тему «Програмне забезпечення віртуального конфігуратора комп'ютерів загального призначення», можемо зробити висновки:

1. Процес розробки та реалізації віртуального конфігуратора комп'ютерів був зосереджений на створенні зручного, інтуїтивно зрозумілого та функціонального інтерфейсу, який дозволяє користувачам легко вибирати та комбінувати компоненти для збірки індивідуальних комп'ютерних систем. Реалізація такого проекту вимагала глибокого аналізу потреб користувачів та ринку комп'ютерних компонентів, а також забезпечення можливості легкого оновлення та масштабування продукту.

2. В рамках проєкту була створена серверна архітектура, що базується на потужному фреймворку Laravel, який надав інструменти для ефективної роботи з базою даних, авторизації користувачів та обробки запитів до API. Вибір Laravel як основи для бекенду був зумовлений його високою продуктивністю, гнучкістю та зручністю у розробці.

3. Фронтенд-частина, реалізована за допомогою технології React, що дало змогу створити динамічний користувацький інтерфейс, який відповідає всім сучасним стандартам дизайну завдяки модульній структурі додатку та використанню компонентно-орієнтованого підходу до розробки.

Протягом усього процесу особлива увага приділялася тестуванню: від одиничних модульних тестів до комплексного функціонального та інтеграційного тестування. Завдяки використанню автоматизованих інструментів тестування, таких та неперервної інтеграції, було забезпечено високу надійність програмного продукту та швидке виявлення помилок, що в свою чергу сприяло підвищенню якості кінцевого продукту.

Отже, розроблене програмне забезпечення віртуального конфігуратора комп'ютерів відповідає поставленим вимогам та може бути застосування в приватних цілях та для вирішення задач бізнесу щодо автоматизації процесу конфігурування комп’ютерів загального призначення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Barnett R. C. Preventing Web Attacks with Apache. NJ : Pearson Education, 2006. 448 с. ISBN 978-0321321289.
2. Schlossnagle G. Advanced PHP Programming. NJ : Pearson Education, 2004. 224 с. ISBN 978-0672325614.
3. Zandstra M. PHP Objects, Patterns, and Practice. NY : Apress, 2013. 536 с. ISBN 978-1430259278.
4. Shiflett C. Essential PHP Security. CA : O'Reilly Media, 2005. 100 с. ISBN 978-0596006563.
5. Сімпсон К. You Don't Know JS: ES6 & Beyond. CA : O'Reilly Media, 2015. 278 с. ISBN 978-1491904244.
6. Майерс М. A Smarter Way to Learn JavaScript. The new tech-assisted approach that requires half the effort. Self-published, 2014. 254 с. ISBN 978-1497408180.
7. Osmani A. Learning JavaScript Design Patterns. CA : O'Reilly Media, 2012. 254 с. ISBN 978-1449331818.
8. Elliott E. Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries. CA : O'Reilly Media, 2014. 254 с. ISBN 978-1491950296.
9. Закас Н. С. Understanding ECMAScript 6: The Definitive Guide for JavaScript Developers. No Starch Press, 2016. 352 с. ISBN 978-1593277574.
10. Flanagan D. JavaScript: The Definitive Guide. CA : O'Reilly Media, 2020. 706 с. ISBN 978-1491952023.
11. Ullman L. PHP and MySQL for Dynamic Web Sites. CA : Peachpit Press, 2017. 696 с. ISBN 978-0134301846.
12. Nixon R. Learning PHP, MySQL & JavaScript. CA : O'Reilly Media, 2018. 832 с. ISBN 978-1491978917.

13. Rauschmayer A. Exploring ES6: Upgrade to the Next Version of JavaScript. Self-published, 2016. Online resource.

14. McConnell S. Code Complete: A Practical Handbook of Software Construction. WA : Microsoft Press, 2004. 960 c. ISBN 978-0735619678.

15. Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. CA : New Riders, 2014. 216 c. ISBN 978-0321965516.

16. Marcotte E. Responsive Web Design. NY : A Book Apart, 2011. 150 c. ISBN 978-0984442577.

17. Verou L. CSS Secrets: Better Solutions to Everyday Web Design Problems. CA : O'Reilly Media, 2015. 392 c. ISBN 978-1449372637.

18. Duckett J. PHP & MySQL: Novice to Ninja. UK : SitePoint, 2017. 690 c. ISBN 978-0994346988.

19. Wenz C. PHP Phrasebook. IN : Addison-Wesley, 2005. 480 c. ISBN 978-0672328707.

20. Nixon R. Modern PHP: New Features and Good Practices. CA : O'Reilly Media, 2015. 268 c. ISBN 978-1491905012.

21. Svein Nordbotten: Introduction to Development of Dynamic Web Applications, 2007. 159 c. Online resource.

22. John Moore Williams: The Modern Web Design Process, 2016. 124 c. Online resource.

23. Jason Beaird: The Principles of Beautiful Web Design, 2007. 29 c. Online resource.

24. Shriram Krishnamurthi: Programming Languages: Application and Interpretation, 2007. 376 c. Online resource.

25. M. Ben-Ari: Understanding Programming Languages, 2006. 322 c. Online resource.

26. Michael Mendez: The Missing Link. An Introduction to Web Development and Programming, 2014. 303 c. Online resource.

27. Computerworld: The A-Z of Programming Languages, 2008-2010. 127 с. Online resource.

28. Marijn Haverbeke: Eloquent JavaScript, 2018. 463 с. Online resource.

29. Dr. Axel Rauschmayer: JavaScript For Impatient Programmers, 2022. 379 с. Online resource.

30. Mario Lurig: PHP Reference: Beginner to Intermediate PHP5, 2008. 163 с. ISBN: 978-1-4357-1590-5.

31. Tarandeep Kaur: Cloud Computing, 2013. 327 с. ISBN: 978- 93-94068-33-9.

32. Jaydip Sen: Cloud computing – architecture and applications, 2017. 142 с. ISBN: 978-953-51-3244-8.

33. J.D. Lasica, dentity in the Age of Cloud Computing – The next-generation Internet'simpact on business, governance and social interaction, 2009. 110 с. ISBN: 0-89843-505-6

34. Stuttard D., Pinto M. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws. NY : Wiley, 2020. 912 с. ISBN 978-1119687359.

35. Ballad T., Confer W. Securing PHP Web Applications. MA : Syngress, 2008. 304 с. ISBN 978-1597495081.

36. McDonald M., McGovern J. et al. Web Security for Developers: Real Threats, Practical Defense. CA : O'Reilly Media, 2022. 542 с. ISBN 978-1492079515.

37. Bergmann S., Priebsch S. Foundations of PHP for Web Developers. NY : Apress, 2015. 372 с. ISBN 978-1484210438.

38. Бэнкс А., Порселло Є. Learning React: Functional Web Development with React and Redux. CA : O'Reilly Media, 2017. 350 с. ISBN 978-1491954621.

39. Bergmann S., Priebsch S. PHPUnit Pocket Guide. CA : O'Reilly Media, 2005. 120 с. ISBN 978-0596101039.

40. Sklar D., Trachtenberg A. PHP Cookbook. CA : O'Reilly Media, 2014. 820 с. ISBN 978-1449363758.

41. Welling L., Thomson L. PHP and MySQL Web Development. CA : Addison-Wesley, 2017. 672 с. ISBN 978-0134804075.

42. Harold Abelson and Gerald Jay Sussman: Structure and Interpretation of Computer Programs, JavaScript Edition, 2022. 642 с. Online resource.

43. Snyder C., Southwell M., Owad T. PHP Security. CA : O'Reilly Media, 2005. 428 с. ISBN 978-0596006563.

44. Snyder C., Myer T., Southwell M. Pro PHP Security: From Application Security Principles to the Implementation of XSS Defenses. NY : Apress, 2006. 704 с. ISBN 978-1590595084.

45. J N Ndunagu,B C Mbam,J N Ndunagu: Website Design and Programming, 2010. 138 с. Online resource.

46. Allan Jerjas,Sten Andersson,Lansner Anders: Building Blocks of Responsive Web Design, 2013. 36 с. Online resource.

47. Gustavo Rossi,Matias Urbieta,Damiano Distante: 25 Years of Model Driven Web Engineering What we Achieved What is Missing, 2016. 29 с. Online resource.

48. Mike Grant, Zachary Palmer, Scott Smith: Principles of Programming Languages, 2020. 160 с. Online resource.

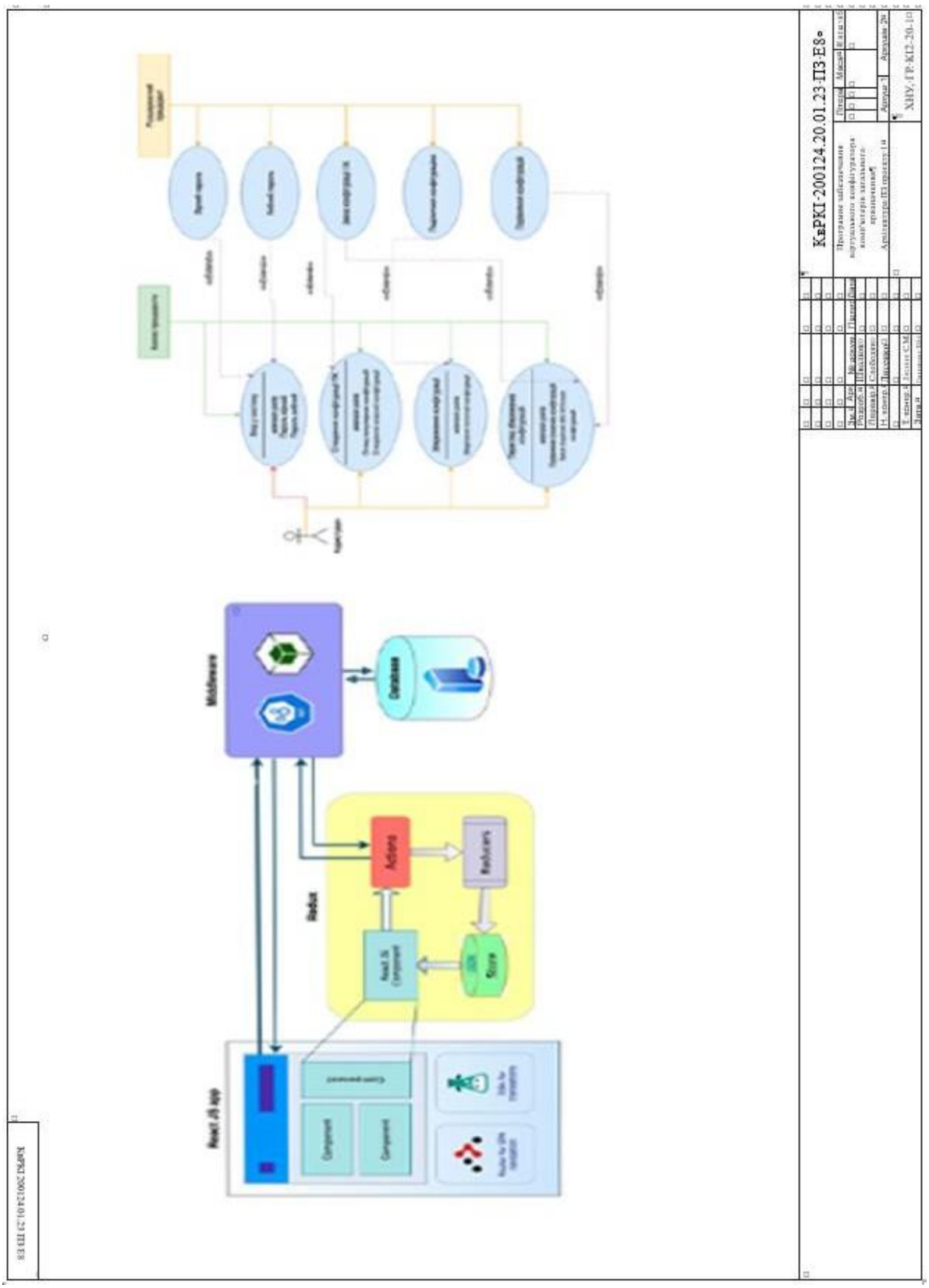
49. Theo Lynn, John G. Mooney, Brian Lee and Patricia Takako Endo: The Cloud-to-Thing Continuum, 2020. 183 с. ISBN: 978-3-030-41110-7.

50. Dr. E. Saravana Kumar, Dr. Selvaraj Kesavan, Dr. R. Ch. A Naidu, Dr. Senthil Kumar R And Latha: Comprehensive Analysis of Cloud based Databases, 2021. 15 с. Online resource.

ДОДАТОК А

(обов'язковий)

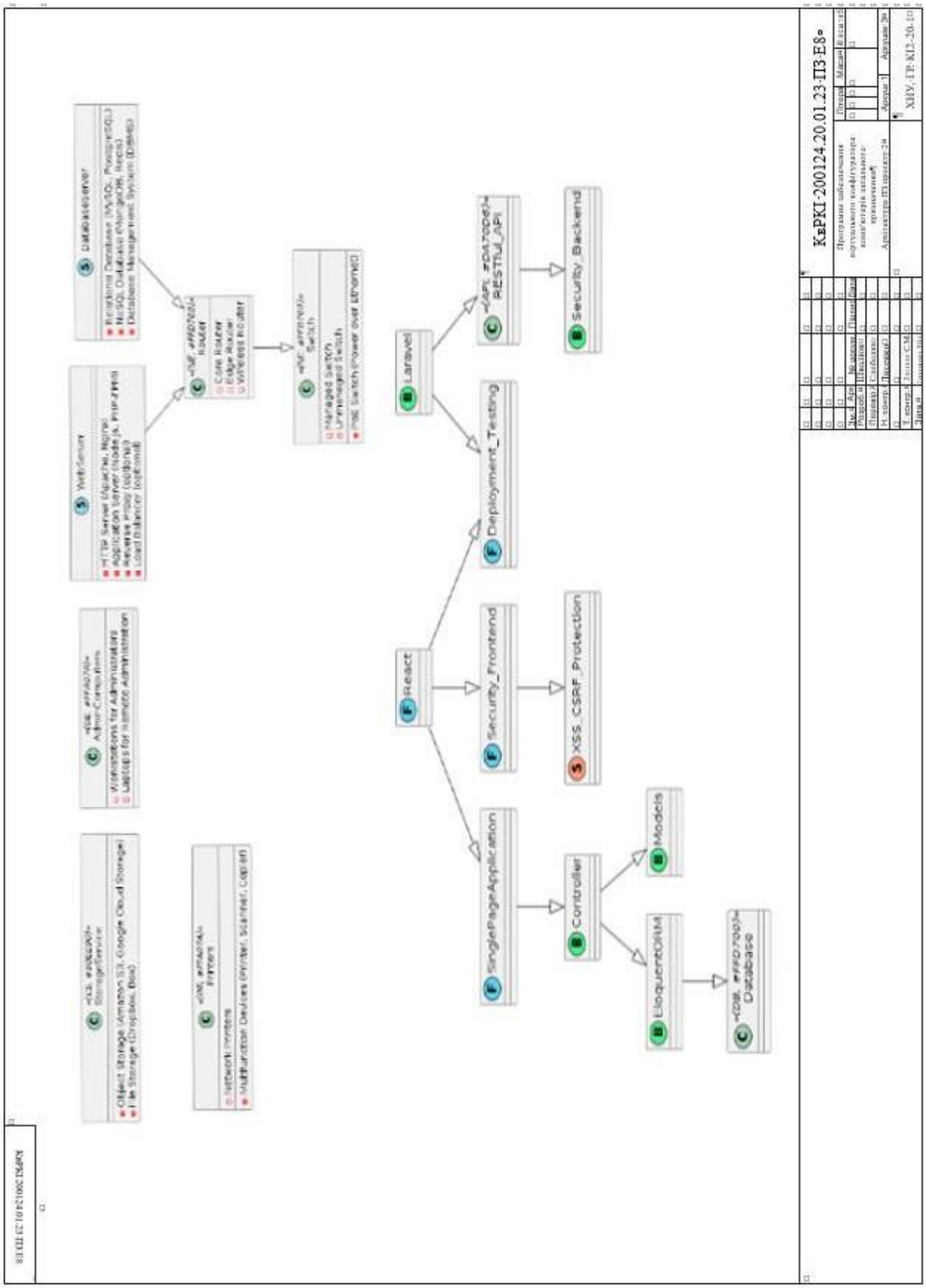
Копія креслення «Архітектура ПЗ проєкту 1»



ДОДАТОК Б

(обов'язковий)

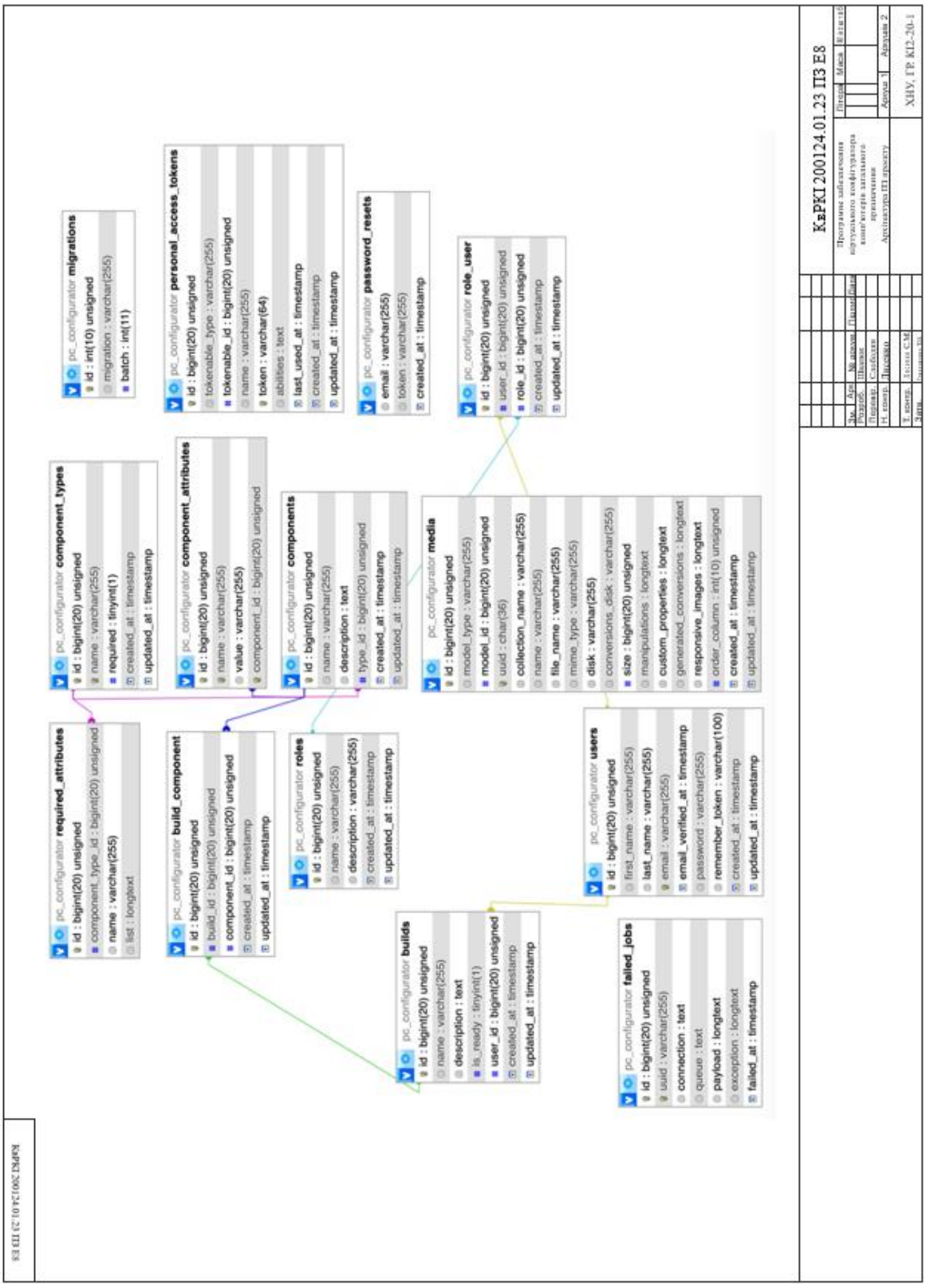
Копія креслення «Архітектура ПЗ проєкту 2»



ДОДАТОК В

(обов'язковий)

Копія креслення «Архітектура БД для проєкту»



ДОДАТОК Г

Скріншоти роботи додатку

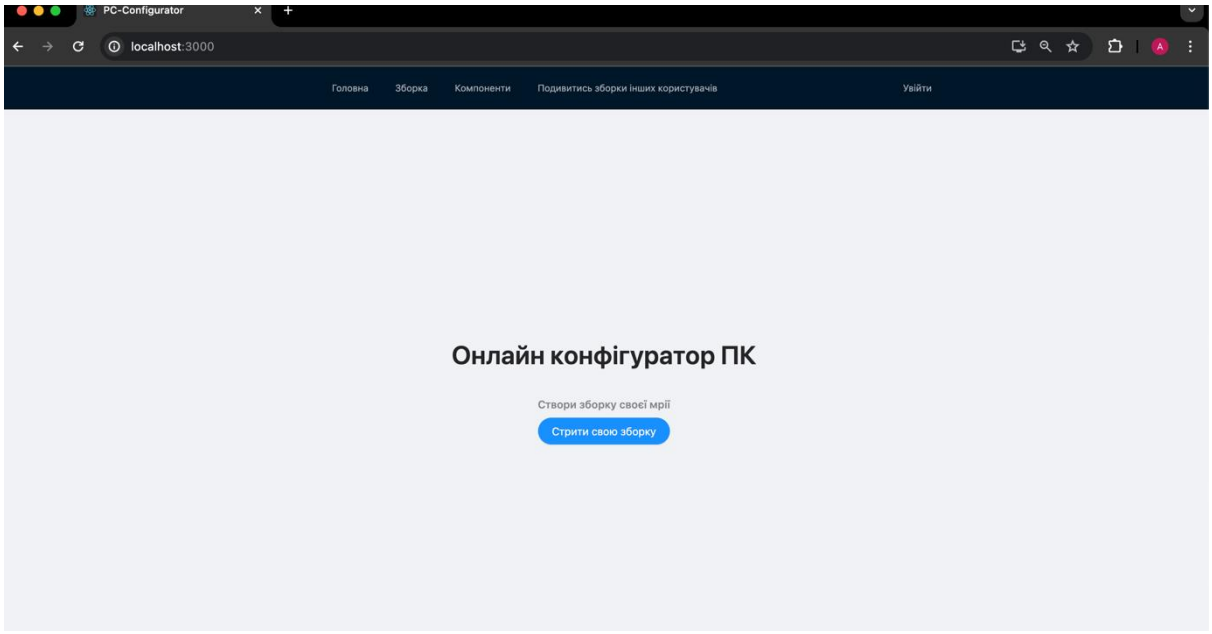


Рисунок Г.1 – Головна сторінка

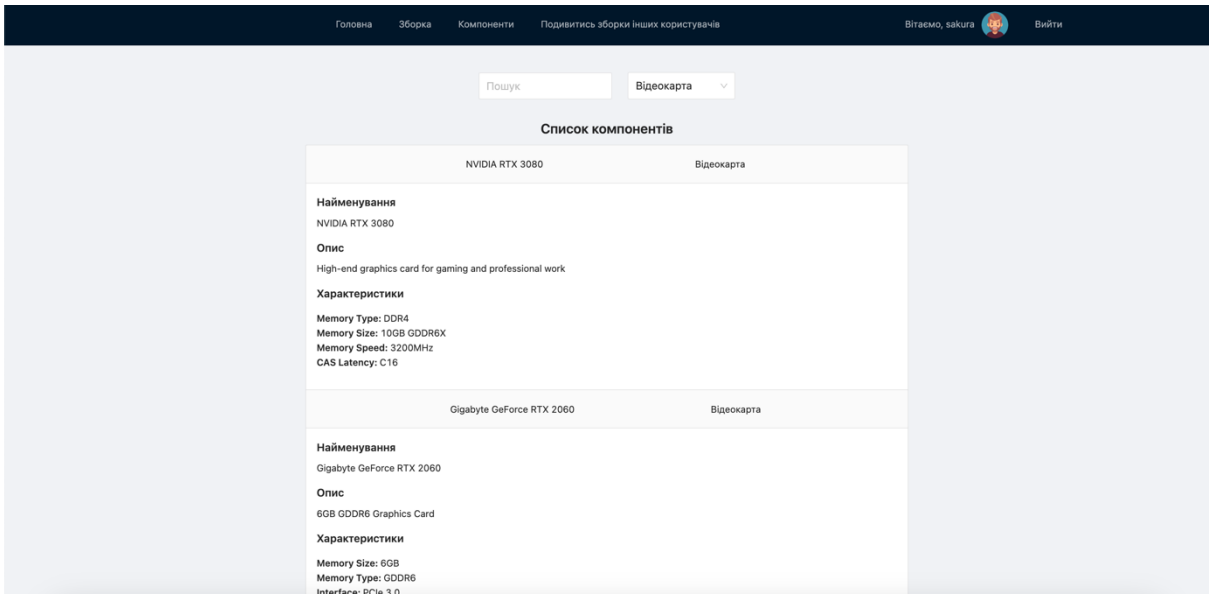


Рисунок Г. 3 – Порівняння компонентів

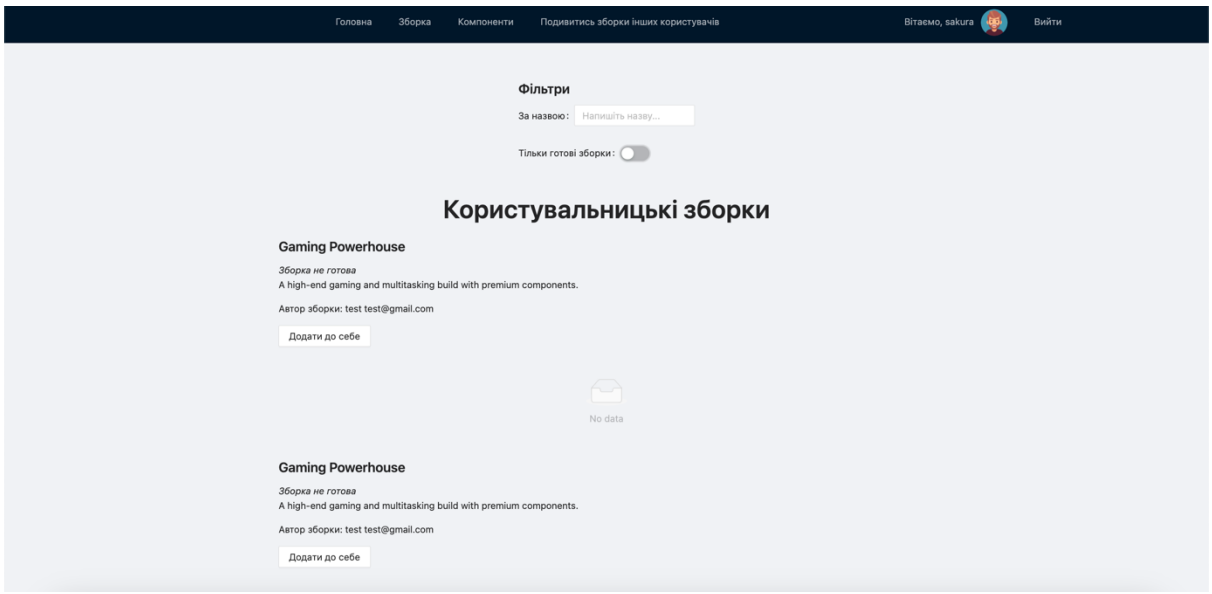


Рисунок Г. 4 – Сторінка перегляду користувацьких збірок

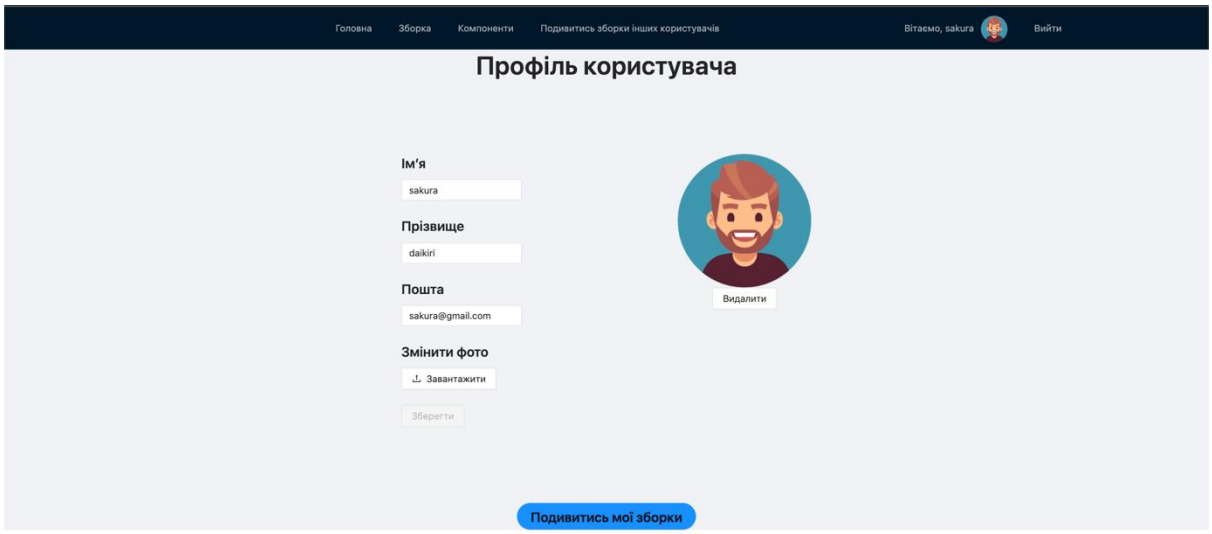


Рисунок Г. 5 – Сторінка користувача

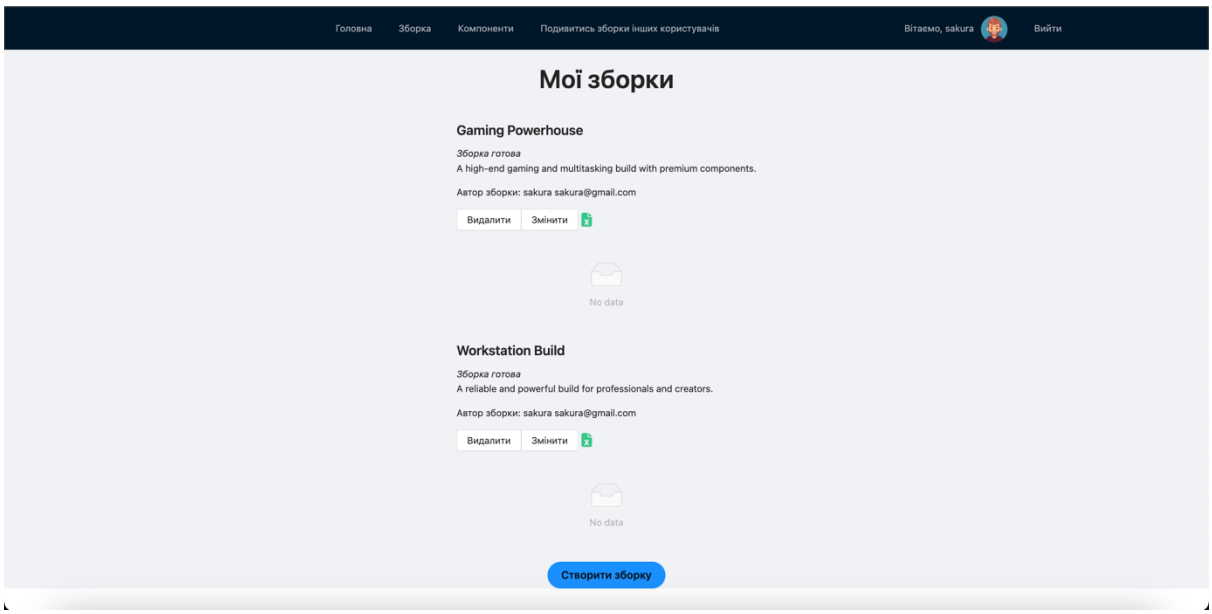


Рисунок Г. 6 – Сторінка «Мої збірки»

ДОДАТОК Д

Лістинг коду

Конфігуратор файлу eve

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=pc_configurator
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1
```

```
REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailhog
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS="hello@example.com"
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

App.php

```
<?php
```

```
use Illuminate\Support\Facades\Facade;

return [

    /*
    |--------------------------------------------------------------------------
    | Application Name
    |--------------------------------------------------------------------------
    |
    | This value is the name of your application. This value is used when the
    | framework needs to place the application's name in a notification or
    | any other location as required by the application or its packages.
    |
    */

    'name' => env('APP_NAME', 'Laravel'),

    /*
    |--------------------------------------------------------------------------
    | Application Environment
    |--------------------------------------------------------------------------
    |
    | This value determines the "environment" your application is currently
    | running in. This may determine how you prefer to configure various
    | services the application utilizes. Set this in your ".env" file.
    |
    */

    'env' => env('APP_ENV', 'production'),
```

```
/*
|--------------------------------------------------------------------------
| Application Debug Mode
|--------------------------------------------------------------------------
|
| When your application is in debug mode, detailed error messages with
| stack traces will be shown on every error that occurs within your
| application. If disabled, a simple generic error page is shown.
|
*/

'debug' => (bool)env('APP_DEBUG', false),

/*
|--------------------------------------------------------------------------
| Application URL
|--------------------------------------------------------------------------
|
| This URL is used by the console to properly generate URLs when using
| the Artisan command line tool. You should set this to the root of
| your application so that it is used when running Artisan tasks.
|
*/

'url' => env('APP_URL', 'http://localhost'),

'asset_url' => env('ASSET_URL'),
```

```
/*
|--------------------------------------------------------------------------
| Application Timezone
|--------------------------------------------------------------------------
|
| Here you may specify the default timezone for your application, which
| will be used by the PHP date and date-time functions. We have gone
| ahead and set this to a sensible default for you out of the box.
|
*/

'timezone' => 'UTC',

/*
|--------------------------------------------------------------------------
| Application Locale Configuration
|--------------------------------------------------------------------------
|
| The application locale determines the default locale that will be used
| by the translation service provider. You are free to set this value
| to any of the locales which will be supported by the application.
|
*/

'locale' => 'en',

/*
|--------------------------------------------------------------------------
| Application Fallback Locale
```

```
|--------------------------------------------------------------------------
|
| The fallback locale determines the locale to use when the current one
| is not available. You may change the value to correspond to any of
| the language folders that are provided through your application.
|
*/

'fallback_locale' => 'en',

/*
|--------------------------------------------------------------------------
| Faker Locale
|--------------------------------------------------------------------------
|
| This locale will be used by the Faker PHP library when generating fake
| data for your database seeds. For example, this will be used to get
| localized telephone numbers, street address information and more.
|
*/

'faker_locale' => 'en_US',

/*
|--------------------------------------------------------------------------
| Encryption Key
|--------------------------------------------------------------------------
|
| This key is used by the Illuminate encrypter service and should be set
```

```
| to a random, 32 character string, otherwise these encrypted strings
| will not be safe. Please do this before deploying an application!
|
*/

'key' => env('APP_KEY'),

'cipher' => 'AES-256-CBC',

/*
|--------------------------------------------------------------------------
| Autoloaded Service Providers
|--------------------------------------------------------------------------
        |
| The service providers listed here will be automatically loaded on the
| request to your application. Feel free to add your own services to
| this array to grant expanded functionality to your applications.
|
*/

'providers' => [

    /*
     * Laravel Framework Service Providers...
     */
    Illuminate\Auth\AuthServiceProvider::class,
    Illuminate\Broadcasting\BroadcastServiceProvider::class,
    Illuminate\Bus\BusServiceProvider::class,
    Illuminate\Cache\CacheServiceProvider::class,
```

```
        Illuminate\Foundation\Providers\ConsoleSupportServiceProvider::class,
        Illuminate\Cookie\CookieServiceProvider::class,
        Illuminate\Database\DatabaseServiceProvider::class,
        Illuminate\Encryption\EncryptionServiceProvider::class,
        Illuminate\Filesystem\FilesystemServiceProvider::class,
        Illuminate\Foundation\Providers\FoundationServiceProvider::class,
        Illuminate\Hashing\HashServiceProvider::class,
        Illuminate\Mail\MailServiceProvider::class,
        Illuminate\Notifications\NotificationServiceProvider::class,
        Illuminate\Pagination\PaginationServiceProvider::class,
        Illuminate\Pipeline\PipelineServiceProvider::class,
        Illuminate\Queue\QueueServiceProvider::class,
        Illuminate\Redis\RedisServiceProvider::class,
        Illuminate\Auth\Passwords\PasswordResetServiceProvider::class,
        Illuminate\Session\SessionServiceProvider::class,
        Illuminate\Translation\TranslationServiceProvider::class,
        Illuminate\Validation\ValidationServiceProvider::class,
        Illuminate\View\ViewServiceProvider::class,

        /*
         * Package Service Providers...
         */
        /*
         * Application Service Providers...
         */
        App\Providers\AppServiceProvider::class,
        App\Providers\AuthServiceProvider::class,
        // App\Providers\BroadcastServiceProvider::class,
        App\Providers\EventServiceProvider::class,
```

```
        App\Providers\RouteServiceProvider::class,

    ],

    /*
    |--------------------------------------------------------------------------
    | Class Aliases
    |--------------------------------------------------------------------------
    |
    | This array of class aliases will be registered when this application
    | is started. However, feel free to register as many as you wish as
    | the aliases are "lazy" loaded so they don't hinder performance.
    |
    */

    'aliases' => Facade::defaultAliases()->merge([
        // ...
    ])->toArray(),

];
```

Auth.php

```
<?php

return [

    'defaults' => [
        'guard' => 'web',
```

```
        'passwords' => 'users',
    ],
'guards' => [
        'web' => [
            'driver' => 'session',
            'provider' => 'users',
        ],
    ],
  'providers' => [
        'users' => [
            'driver' => 'eloquent',
            'model' => \App\Models\User\User::class,
        ],
    ],
    'passwords' => [
        'users' => [
            'provider' => 'users',
            'table' => 'password_resets',
            'expire' => 60,
            'throttle' => 60,
        ],
    ],
    'password_timeout' => 10800,

];
```

Broadcasting.php

```
<?php
```

```
return [
  'default' => env('BROADCAST_DRIVER', 'null'),
  'connections' => [

      'pusher' => [
          'driver' => 'pusher',
          'key' => env('PUSHER_APP_KEY'),
          'secret' => env('PUSHER_APP_SECRET'),
          'app_id' => env('PUSHER_APP_ID'),
          'options' => [
              'cluster' => env('PUSHER_APP_CLUSTER'),
              'useTLS' => true,
          ],
          'client_options' => [
              // Guzzle client options: https://docs.guzzlephp.org/en/stable/request-options.html
          ],
      ],

      'ably' => [
          'driver' => 'ably',
          'key' => env('ABLY_KEY'),
      ],

      'redis' => [
          'driver' => 'redis',
          'connection' => 'default',
      ],
```

```
        'log' => [
            'driver' => 'log',
        ],

        'null' => [
            'driver' => 'null',
        ],

    ],

];
```

Cache.php

```
<?php

use Illuminate\Support\Str;

return [
    'default' => env('CACHE_DRIVER', 'file'),
    'stores' => [

        'apc' => [
            'driver' => 'apc',
        ],

        'array' => [
            'driver' => 'array',
```

```
    'serialize' => false,
],

'database' => [
    'driver' => 'database',
    'table' => 'cache',
    'connection' => null,
    'lock_connection' => null,
],

'file' => [
    'driver' => 'file',
    'path' => storage_path('framework/cache/data'),
],

'memcached' => [
    'driver' => 'memcached',
    'persistent_id' => env('MEMCACHED_PERSISTENT_ID'),
    'sasl' => [
        env('MEMCACHED_USERNAME'),
        env('MEMCACHED_PASSWORD'),
    ],
    'options' => [
        // Memcached::OPT_CONNECT_TIMEOUT => 2000,
    ],
    'servers' => [
        [
            'host' => env('MEMCACHED_HOST', '127.0.0.1'),
            'port' => env('MEMCACHED_PORT', 11211),
```

```
                'weight' => 100,
            ],
        ],
    ],

    'redis' => [
        'driver' => 'redis',
        'connection' => 'cache',
        'lock_connection' => 'default',
    ],

    'dynamodb' => [
        'driver' => 'dynamodb',
        'key' => env('AWS_ACCESS_KEY_ID'),
        'secret' => env('AWS_SECRET_ACCESS_KEY'),
        'region' => env('AWS_DEFAULT_REGION', 'us-east-1'),
        'table' => env('DYNAMODB_CACHE_TABLE', 'cache'),
        'endpoint' => env('DYNAMODB_ENDPOINT'),
    ],

    'octane' => [
        'driver' => 'octane',
    ],

],
'prefix' => env('CACHE_PREFIX', Str::slug(env('APP_NAME', 'laravel'), '_').'_cache_'),

];
```

Cors.php

```
<?php

return [

  'paths' => ['api/*', 'sanctum/csrf-cookie'],

    'allowed_methods' => ['*'],

    'allowed_origins' => ['*'],

    'allowed_origins_patterns' => [],

    'allowed_headers' => ['*'],

    'exposed_headers' => [],

    'max_age' => 0,

    'supports_credentials' => true,

];
```

Database.php

```
<?php

use Illuminate\Support\Str;
```

```
return [

    'default' => env('DB_CONNECTION', 'mysql'),

  'connections' => [

        'sqlite' => [
            'driver' => 'sqlite',
            'url' => env('DATABASE_URL'),
            'database' => env('DB_DATABASE', database_path('database.sqlite')),
            'prefix' => '',
            'foreign_key_constraints' => env('DB_FOREIGN_KEYS', true),
        ],

        'mysql' => [
            'driver' => 'mysql',
            'url' => env('DATABASE_URL'),
            'host' => env('DB_HOST', '127.0.0.1'),
            'port' => env('DB_PORT', '3306'),
            'database' => env('DB_DATABASE', 'forge'),
            'username' => env('DB_USERNAME', 'forge'),
            'password' => env('DB_PASSWORD', ''),
            'unix_socket' => env('DB_SOCKET', ''),
            'charset' => 'utf8mb4',
            'collation' => 'utf8mb4_unicode_ci',
            'prefix' => '',
            'prefix_indexes' => true,
            'strict' => true,
```

```
        'engine' => null,
        'options' => extension_loaded('pdo_mysql') ? array_filter([
            PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'),
        ]) : [],
    ],

    'pgsql' => [
        'driver' => 'pgsql',
        'url' => env('DATABASE_URL'),
        'host' => env('DB_HOST', '127.0.0.1'),
        'port' => env('DB_PORT', '5432'),
        'database' => env('DB_DATABASE', 'forge'),
        'username' => env('DB_USERNAME', 'forge'),
        'password' => env('DB_PASSWORD', ''),
        'charset' => 'utf8',
        'prefix' => '',
        'prefix_indexes' => true,
        'search_path' => 'public',
        'sslmode' => 'prefer',
    ],

    'sqlsrv' => [
        'driver' => 'sqlsrv',
        'url' => env('DATABASE_URL'),
        'host' => env('DB_HOST', 'localhost'),
        'port' => env('DB_PORT', '1433'),
        'database' => env('DB_DATABASE', 'forge'),
        'username' => env('DB_USERNAME', 'forge'),
        'password' => env('DB_PASSWORD', ''),
```

```
            'charset' => 'utf8',
            'prefix' => '',
            'prefix_indexes' => true,
        ],

    ],

    'migrations' => 'migrations',

    'redis' => [

        'client' => env('REDIS_CLIENT', 'phpredis'),

        'options' => [
            'cluster' => env('REDIS_CLUSTER', 'redis'),
            'prefix' => env('REDIS_PREFIX', Str::slug(env('APP_NAME', 'laravel'), '_').'_database_'),
        ],

        'default' => [
            'url' => env('REDIS_URL'),
            'host' => env('REDIS_HOST', '127.0.0.1'),
            'password' => env('REDIS_PASSWORD'),
            'port' => env('REDIS_PORT', '6379'),
            'database' => env('REDIS_DB', '0'),
        ],

        'cache' => [
            'url' => env('REDIS_URL'),
```

```
            'host' => env('REDIS_HOST', '127.0.0.1'),
            'password' => env('REDIS_PASSWORD'),
            'port' => env('REDIS_PORT', '6379'),
            'database' => env('REDIS_CACHE_DB', '1'),
        ],

    ],

];
```

Filesystems.php

```
<?php

return [

    'default' => env('FILESYSTEM_DISK', 'local'),

    'disks' => [

        'local' => [
            'driver' => 'local',
            'root' => storage_path('app'),
            'throw' => false,
        ],

        'public' => [
            'driver' => 'local',
            'root' => storage_path('app/public'),
```

```
            'url' => env('APP_URL').'/storage',
            'visibility' => 'public',
            'throw' => false,
        ],

        's3' => [
            'driver' => 's3',
            'key' => env('AWS_ACCESS_KEY_ID'),
            'secret' => env('AWS_SECRET_ACCESS_KEY'),
            'region' => env('AWS_DEFAULT_REGION'),
            'bucket' => env('AWS_BUCKET'),
            'url' => env('AWS_URL'),
            'endpoint' => env('AWS_ENDPOINT'),
            'use_path_style_endpoint'    =>    env('AWS_USE_PATH_STYLE_ENDPOINT',
false),
            'throw' => false,
        ],

    ],

    'links' => [
        public_path('storage') => storage_path('app/public'),
    ],

];
```

Hashing.php

```
<?php
```

```
return [

    'driver' => 'bcrypt',

    'bcrypt' => [
        'rounds' => env('BCRYPT_ROUNDS', 10),
    ],

    'argon' => [
        'memory' => 65536,
        'threads' => 1,
        'time' => 4,
    ],

];
```

Logging.php

```
<?php

use Monolog\Handler\NullHandler;
use Monolog\Handler\StreamHandler;
use Monolog\Handler\SyslogUdpHandler;

return [

    'default' => env('LOG_CHANNEL', 'stack'),
```

```
'deprecations' => env('LOG_DEPRECATIONS_CHANNEL', 'null'),

'channels' => [
    'stack' => [
        'driver' => 'stack',
        'channels' => ['single'],
        'ignore_exceptions' => false,
    ],

    'single' => [
        'driver' => 'single',
        'path' => storage_path('logs/laravel.log'),
        'level' => env('LOG_LEVEL', 'debug'),
    ],

    'daily' => [
        'driver' => 'daily',
        'path' => storage_path('logs/laravel.log'),
        'level' => env('LOG_LEVEL', 'debug'),
        'days' => 14,
    ],

    'slack' => [
        'driver' => 'slack',
        'url' => env('LOG_SLACK_WEBHOOK_URL'),
        'username' => 'Laravel Log',
        'emoji' => ':boom:',
        'level' => env('LOG_LEVEL', 'critical'),
    ],
```

```
        'papertrail' => [
            'driver' => 'monolog',
            'level' => env('LOG_LEVEL', 'debug'),
            'handler' => env('LOG_PAPERTRAIL_HANDLER', SyslogUdpHandler::class),
            'handler_with' => [
                'host' => env('PAPERTRAIL_URL'),
                'port' => env('PAPERTRAIL_PORT'),
                'connectionString'                                          =>
'tls://'.env('PAPERTRAIL_URL').':'.env('PAPERTRAIL_PORT'),
            ],
        ],

        'stderr' => [
            'driver' => 'monolog',
            'level' => env('LOG_LEVEL', 'debug'),
            'handler' => StreamHandler::class,
            'formatter' => env('LOG_STDERR_FORMATTER'),
            'with' => [
                'stream' => 'php://stderr',
            ],
        ],

        'syslog' => [
            'driver' => 'syslog',
            'level' => env('LOG_LEVEL', 'debug'),
        ],

        'errorlog' => [
```

```
            'driver' => 'errorlog',
            'level' => env('LOG_LEVEL', 'debug'),
        ],

        'null' => [
            'driver' => 'monolog',
            'handler' => NullHandler::class,
        ],

        'emergency' => [
            'path' => storage_path('logs/laravel.log'),
        ],
    ],

];
```

Mail.php

```
<?php

return [

    'default' => env('MAIL_MAILER', 'smtp'),

    'mailers' => [
        'smtp' => [
            'transport' => 'smtp',
            'host' => env('MAIL_HOST', 'smtp.mailgun.org'),
            'port' => env('MAIL_PORT', 587),
```

```
        'encryption' => env('MAIL_ENCRYPTION', 'tls'),
        'username' => env('MAIL_USERNAME'),
        'password' => env('MAIL_PASSWORD'),
        'timeout' => null,
    ],

    'ses' => [
        'transport' => 'ses',
    ],

    'mailgun' => [
        'transport' => 'mailgun',
    ],

    'postmark' => [
        'transport' => 'postmark',
    ],

    'sendmail' => [
        'transport' => 'sendmail',
        'path' => env('MAIL_SENDMAIL_PATH', '/usr/sbin/sendmail -bs -i'),
    ],

    'log' => [
        'transport' => 'log',
        'channel' => env('MAIL_LOG_CHANNEL'),
    ],

    'array' => [
```

```
            'transport' => 'array',
        ],

        'failover' => [
            'transport' => 'failover',
            'mailers' => [
                'smtp',
                'log',
            ],
        ],
    ],

    'from' => [
        'address' => env('MAIL_FROM_ADDRESS', 'hello@example.com'),
        'name' => env('MAIL_FROM_NAME', 'Example'),
    ],

    'markdown' => [
        'theme' => 'default',

        'paths' => [
            resource_path('views/vendor/mail'),
        ],
    ],

];
```

Queve.php

```
<?php

return [

    'default' => env('QUEUE_CONNECTION', 'sync'),

    'connections' => [

        'sync' => [
            'driver' => 'sync',
        ],

        'database' => [
            'driver' => 'database',
            'table' => 'jobs',
            'queue' => 'default',
            'retry_after' => 90,
            'after_commit' => false,
        ],

        'beanstalkd' => [
            'driver' => 'beanstalkd',
            'host' => 'localhost',
            'queue' => 'default',
            'retry_after' => 90,
            'block_for' => 0,
            'after_commit' => false,
        ],
```

```
        'sqs' => [
            'driver' => 'sqs',
            'key' => env('AWS_ACCESS_KEY_ID'),
            'secret' => env('AWS_SECRET_ACCESS_KEY'),
            'prefix' => env('SQS_PREFIX', 'https://sqs.us-east-1.amazonaws.com/your-account-id'),
            'queue' => env('SQS_QUEUE', 'default'),
            'suffix' => env('SQS_SUFFIX'),
            'region' => env('AWS_DEFAULT_REGION', 'us-east-1'),
            'after_commit' => false,
        ],

        'redis' => [
            'driver' => 'redis',
            'connection' => 'default',
            'queue' => env('REDIS_QUEUE', 'default'),
            'retry_after' => 90,
            'block_for' => null,
            'after_commit' => false,
        ],

    ],

    'failed' => [
        'driver' => env('QUEUE_FAILED_DRIVER', 'database-uuids'),
        'database' => env('DB_CONNECTION', 'mysql'),
        'table' => 'failed_jobs',
    ],
```

```
];
```

Sanctum.php

```
<?php

use Laravel\Sanctum\Sanctum;

return [

    'stateful' => explode(',', env('SANCTUM_STATEFUL_DOMAINS', sprintf(
        '%s%s',
        'localhost,localhost:8000,localhost:3000,127.0.0.1,127.0.0.1:8000,::1',
        Sanctum::currentApplicationUrlWithPort()
    ))),

    'guard' => ['web'],

    'expiration' => 60,

    'middleware' => [
        'verify_csrf_token' => App\Http\Middleware\VerifyCsrfToken::class,
        'encrypt_cookies' => App\Http\Middleware\EncryptCookies::class,
    ],

];
```

Services.php

```
<?php

return [

    'mailgun' => [
        'domain' => env('MAILGUN_DOMAIN'),
        'secret' => env('MAILGUN_SECRET'),
        'endpoint' => env('MAILGUN_ENDPOINT', 'api.mailgun.net'),
        'scheme' => 'https',
    ],

    'postmark' => [
        'token' => env('POSTMARK_TOKEN'),
    ],

    'ses' => [
        'key' => env('AWS_ACCESS_KEY_ID'),
        'secret' => env('AWS_SECRET_ACCESS_KEY'),
        'region' => env('AWS_DEFAULT_REGION', 'us-east-1'),
    ],

];
```

Session.php

```
<?php

use Illuminate\Support\Str;

return [
```

```
'driver' => env('SESSION_DRIVER', 'file'),

'lifetime' => env('SESSION_LIFETIME', 120),

'expire_on_close' => false,

'encrypt' => false,

'files' => storage_path('framework/sessions'),

'connection' => env('SESSION_CONNECTION'),

'table' => 'sessions',

'store' => env('SESSION_STORE'),

'lottery' => [2, 100],

'cookie' => env(
    'SESSION_COOKIE',
    Str::slug(env('APP_NAME', 'laravel'), '_').'_session'
),

'path' => '/',

'domain' => env('SESSION_DOMAIN'),

'secure' => env('SESSION_SECURE_COOKIE'),
```

```
    'http_only' => true,

    'same_site' => 'lax',

];
```

View.php

```
<?php

return [

    'paths' => [
        resource_path('views'),
    ],

    'compiled' => env(
        'VIEW_COMPILED_PATH',
        realpath(storage_path('framework/views'))
    ),

];
```

Код папки tests

Код Тест файлу APITest.php

```
<?php

namespace API;

use App\Models\User\User;
use Illuminate\Foundation\Testing\RefreshDatabase;
```

```
use Tests\TestCase;

abstract class APITest extends TestCase
{
    use RefreshDatabase;

    protected $seed = true;

    protected function setUp(): void
    {
        parent::setUp();
        $this->withHeaders([
            'Accept' => 'application/json'
        ]);
    }


    public function asAdmin(): static
    {
        $token = $this->createAdminToken();
        $this->withHeaders(['Authorization' => "Bearer $token"]);
        return $this;
    }

    public function asUser(): static
    {
        $token = $this->createUserToken();
        $this->withHeaders(['Authorization' => "Bearer $token"]);
        return $this;
    }

    public function createUserToken()
    {
        $user = User::updateOrCreate([
            'email' => 'user@test.ru',
        ], [
            'first_name' => 'user',
            'last_name' => 'user',
            'password' => '123123'
        ]);

        $user->roles()->create(['name' => 'user']);
```

```
        return $user->createToken('accessToken')->plainTextToken;
    }

    public function createAdminToken()
    {
        $user = User::updateOrCreate([
            'email' => 'admin@test.ru',
        ], [
            'first_name' => 'admin',
            'last_name' => 'admin',
            'password' => '123123'
        ]);

        $user->roles()->create(['name' => 'admin']);

        return $user->createToken('accessToken')->plainTextToken;
    }
}
```

Код Тест файлу AuthTest.php

```
<?php


namespace API;

use Illuminate\Testing\Fluent\AssertableJson;

class AuthTest extends APITest
{

    public function testRegister(): void
    {
        $userData =
            [
                'email' => 'test' . random_int(1, 100000) . '@example.ru',
                'password' => '123123',
                'lastName' => 'John',
                'firstName' => 'Doe',
            ];
```

```
        $response = $this->post('api/register', $userData);
        $response
            ->assertStatus(201)
            ->assertJson(
                fn(AssertableJson $json) => $json->has('accessToken')
            );
    }

    public function testRegisterWithErrorUserExist(): void
    {
        $userData =
            [
                'email' => 'test' . random_int(1, 100000) . '@example.ru',
                'password' => '123123',
                'lastName' => 'John',
                'firstName' => 'Doe',
            ];


        $this->post('api/register', $userData);

        $response = $this->post('api/register', $userData);

        $response->assertStatus(404);
    }

    public function testLogin(): void
    {
        $userData =
            [
                'email' => 'test@example.ru',
                'password' => '123123',
                'firstName' => 'John',
                'lastName' => 'Doe',
            ];

        $this->post('api/register', $userData);

        $response = $this->post('api/login', ['password' => $userData['password'],
'email' => $userData['email']]);

        $response
```

```
            ->assertStatus(200)
            ->assertJson(
                fn(AssertableJson $json) => $json->has('accessToken')
            );
    }

    public function testLoginWithErrorIncorrectData(): void
    {
        $userData =
            [
                'email' => 'test@example.ru',
                'password' => '1231234',
            ];
        $response = $this->post('api/login', $userData);

        $response->assertStatus(401);
    }
}
```

Код Тест файлу BuildTest.php

```
<?php

namespace API;

use App\Http\Controllers\Build\dto\CheckBuildResult;
use App\Http\Controllers\Build\dto\CreateBuildDto;
use App\Http\Controllers\Build\dto\GetBuildDto;
use App\Http\Controllers\Component\dto\CreateAttributeDto;
use App\Http\Controllers\Component\dto\CreateComponentDto;
use App\Services\Component\ComponentService;
use Illuminate\Testing\Fluent\AssertableJson;

class BuildTest extends APITest
{

    private readonly ComponentService $componentService;

    protected function setUp(): void
    {
        parent::setUp();
        $this->componentService = $this->app->make(ComponentService::class);
```

```
    }

    public function testGetBuilds()
    {
        $response = $this->get('api/builds/');

        $response->assertStatus(200)->assertJson(
            function (AssertableJson $json) {
                if ($json->toArray()) {
                    $json->first(
                        fn(AssertableJson $json) => $json-
>hasAll(getClassProperties(GetBuildDto::class))
                    );
                }
            }
        );
    }

    public function testCreateBuild()
    {
        $build = $this->getTestBuild();

        $response = $this->asUser()->post('api/builds', (array)$build);

        $response->assertStatus(201)->assertJson(
            fn(AssertableJson $json) => $json-
>hasAll(getClassProperties(GetBuildDto::class))
        );
    }

    public function testBuildIsNotReady()
    {
        $build = $this->getTestBuild();

        $response = $this->asUser()->post('api/builds/check', (array)$build);

        $response->assertStatus(200)
            ->assertJson(['isReady' => false])
            ->assertJson(
                fn(AssertableJson $json) => $json-
>hasAll(getClassProperties(CheckBuildResult::class))
            );
```

```
    }

    private function getTestBuild()
    {
        $createComponentDto = new CreateComponentDto(
            name: 'Test component',
            description: 'this is test component',
            type: 'ram',
            photo: null,
            attributes: CreateAttributeDto::collection([
                new CreateAttributeDto(name: 'number_of_modules', value: 1),
                new CreateAttributeDto(name: 'frequency', value: 3200),
                new CreateAttributeDto(name: 'capacity', value: 16),
                new CreateAttributeDto(name: 'memory_type', value: 'ddr3'),
            ])
        );


        $component                    =                    $this->componentService-
>createComponent($createComponentDto);
        return new CreateBuildDto(
            name: 'Test build',
            description: 'this is a test build',
            componentsIds: [$component->id]
        );
    }
}
```

Код Тест файлу ComponentTest.php

```
<?php

namespace API;

use App\Http\Controllers\Component\dto\CreateAttributeDto;
use App\Http\Controllers\Component\dto\CreateComponentDto;
use App\Http\Controllers\Component\dto\GetComponentDto;
use Illuminate\Testing\Fluent\AssertableJson;

class ComponentTest extends APITest
{
```

```
    public function testCreateComponent(): void
    {
        $component = $this->getTestComponent();

        $response    =    $this->asAdmin()->post('/api/components',    $component-
>toArray());


        $response->assertStatus(201)
            ->assertJson(fn(AssertableJson         $json)         =>         $json-
>hasAll(getClassProperties(GetComponentDto::class))
            );
    }


    public function testGetComponents(): void
    {
        $response = $this->get('/api/components');

        $response->assertStatus(200);

        $response->assertJson(function (AssertableJson $json) {
            if ($json->toArray()) {
                $json->first(fn(AssertableJson       $json)       =>       $json-
>hasAny(getClassProperties(GetComponentDto::class)));
            }
        }
        );
    }

    public function testDeleteComponent()
    {
        $component     =     $this->asAdmin()->post('/api/components',     $this-
>getTestComponent()->toArray());

        $response      =      $this->asAdmin()->delete('/api/components/'      .
$component['id']);

        $response->assertStatus(200);
    }

    public function testCreateComponentWithoutNecessaryAttribute(): void
```

```
    {
        $component = $this->getTestComponent();

        unset($component->attributes[0]);

        $response    =    $this->asAdmin()->post('/api/components',    $component-
>toArray());

        $response->assertStatus(422);
    }


    private function getTestComponent(): CreateComponentDto
    {
        return new CreateComponentDto(
            name: 'Test component',
            description: 'this is test component',
            type: 'ram',
            photo: null,
            attributes: CreateAttributeDto::collection([
                new CreateAttributeDto(name: 'number_of_modules', value: 1),
                new CreateAttributeDto(name: 'frequency', value: 3200),
                new CreateAttributeDto(name: 'capacity', value: 16),
                new CreateAttributeDto(name: 'memory_type', value: 'ddr3'),
            ])
        );
    }
}
```

Код Тест файлу UserTest.php

```
<?php

namespace API;

use App\Http\Controllers\User\dto\EditUserDto;
use App\Http\Controllers\User\dto\GetUserDto;
use Illuminate\Testing\Fluent\AssertableJson;

class UserTest extends APITest
{
    public function testGetAuthUser(): void
```

```
    {
        $response = $this->asUser()->get('api/users/me');

        $response->assertStatus(200)->assertJson(fn(AssertableJson $json) =>
$json->hasAll(getClassProperties(GetUserDto::class)));
    }

    public function testGetUsers(): void
    {
        $response = $this->asUser()->get('api/users');

        $response->assertStatus(200)->assertJson(
            fn(AssertableJson $json) => $json->first(
                fn(AssertableJson $json) => $json->hasAll(getClassProperties(GetUserDto::class))
            ));
    }

    public function testEditUser()
    {
        $updateUserDto = new EditUserDto(null, 'Nick', null, null);

        $response = $this->asUser()->post('api/user', (array)$updateUserDto);

        $response->assertStatus(200)->assertJson(
            fn(AssertableJson $json) => $json->hasAll(getClassProperties(GetUserDto::class)))->assertJson(['firstName' => 'Nick']
        );
    }
}
```

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016377451

Дата перевірки:
20.06.2024 10:54:47 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
24.06.2024 11:20:20 EEST

ID користувача:
100005591

Назва документа: Швалюк_Програмне забезпечення віртуального конфігуратора комп'ютерів загального пр...

Кількість сторінок: 64 **Кількість слів:** 11708 **Кількість символів:** 94307 **Розмір файлу:** 2.30 MB **ID файлу:** 1016185918

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

8.68%
Схожість

Найбільша схожість: 4.04% з Інтернет-джерелом (https://akademik.del.ac.id/11320039/PSW-PROYEK-KEL.04/commit/9af...

6.81% Джерела з Інтернету	304	Сторінка 66
2.9% Джерела з Бібліотеки	172	Сторінка 68

0% Цитат

Не знайдено жодних цитат

Посилання	1	Сторінка 68

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	13 сторінок

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

	Документ		Сумарний збіг по Базі Даних	
ID: 131690 Назва: БКР Програмне забезпечення віртуального конфігуратора комп'ютерів загального призначення Додано в БД: 2024-06-20 Автора: В. В. Швалюк Керівники: М. О. Слободян Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	73006	1085	2178 (3%)	30 (3%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Швалюк Вадим Володимирович

Тема: Програмне забезпечення віртуального конфігуратора комп'ютерів загального призначення

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 64

1. Короткий зміст роботи та прийнятих рішень:
Метою роботи є розробка програмного забезпечення, яке б дозволяло користувачам з легкістю конфігурувати комп'ютери, враховуючи останні досягнення в області комп'ютерних технологій та компонентів. Основні завдання, вирішені в роботі, включають аналіз існуючих рішень віртуальних конфігураторів, визначення вимог до програмного забезпечення, розробку архітектури та інтерфейсу користувача, програмування функціональності конфігуратора та тестування отриманого продукту.

2. Висновок про відповідність роботи дипломному завданню:
Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи:

Перший розділ дипломної роботи був присвячений аналізу предметної області програмного забезпечення для віртуального конфігурування комп'ютерів, визначенню актуальності теми, огляду існуючих рішень на ринку, а також дослідженню можливостей для їх удосконалення. В рамках другого розділу дипломної роботи було проведено детальне проектування програмного забезпечення для віртуального конфігуратора комп'ютерів загального призначення. Ми розглянули ключові компоненти системи, які включають архітектуру бази даних, вибір технологічного стеку, розробку користувацького інтерфейсу, а також заходи забезпечення безпеки та продуктивності. У третьому розділі було детально

розглянуто ключові аспекти реалізації та тестування програмного забезпечення для віртуального конфігуратора комп'ютерів загального призначення. Важливим моментом стала побудова надійної серверної платформи на основі Laravel, яка забезпечила стабільну основу для всієї системи та ефективне управління базою даних.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: відсутність інструкції користувача програмного забезпечення віртуального корфугуратора.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на достатньому рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Стецюк М. В.
др. філософії кф. кібербезпеки

"20" червня 2024 р. _____ (підпис)

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Швалюка Вадима Володимировича

ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

22 квітня 2024 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ
КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Програмне забезпечення віртуального конфігуратора комп'ютерів загального призначення

Автор: Швалюк Вадим Володимирович

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Слободян Максим Олегович

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;

2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до оформлення лістингів програмного коду, де використовуються нерегламентовані комбінації латинських літер, спеціальних символів та цифр.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 8.68% і адресується до 476 першоджерел, що, з урахуванням наведених обгрунтувань, відповідає характеру унікального проєкту і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ М. О. Слободян

Гарант ОП _____ С. М. Лисенко

Завідувач кафедри КІІС _____ Т. О. Говорущенко