

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Калькулятор для розрахунку конфігурації ПК»

Виконав студент групи ІПЗ-21-1
Зіневич Роман Русланович

Керівник роботи:
Локтікова Тамара Миколаївна

Рецензент:
Лобанчикова Надія Миколаївна

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»
Зав. кафедри інженерії програмного
забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **Зіневич Роман Русланович**

Керівник роботи: **Локтікова Тамара Миколаївна**

Тема роботи: **«Калькулятор для розрахунку конфігурації ПК»**,
затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: **об'єктом дослідження є процес створення калькулятора для розрахунку конфігурації ПК, а предметом дослідження є методи, засоби та алгоритми створення вебзастосунку для підбору компонентів ПК, інтеграція з базою даних для збереження інформації про компоненти, а також реалізація функцій перевірки сумісності та підрахунку вартості конфігурацій.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Локтікова Т. М.	27.02.2025 р.	27.02.2025 р.
2	Локтікова Т. М.	27.02.2025 р.	27.02.2025 р.
3	Локтікова Т. М.	27.02.2025 р.	27.02.2025 р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебзастосунку конфігуратора персональних комп'ютерів та пояснювальної записки. Пояснювальна записка до випускної роботи містить 67 сторінок, 26 ілюстрацій та 28 таблиць.

Виконано розробку зручного вебзастосунку для автоматизованого підбору сумісних комплектуючих ПК з урахуванням бюджету користувача та актуальних технічних характеристик.

Архітектура вебзастосунку базується на клієнт-серверній моделі з використанням сучасних технологій: React для фронтенду, Node.js та Express для бекенду, а також MongoDB для зберігання даних. Завдяки автоматичній перевірці сумісності компонентів забезпечується коректність створених збірок ПК.

Вебзастосунок має інтуїтивно зрозумілий інтерфейс, що дозволяє швидко вибирати компоненти, переглядати детальну інформацію про них і зберігати або оновлювати власні конфігурації ПК.

КЛЮЧОВІ СЛОВА: ВЕБЗАСТОСУНОК, КЛІЄНТ-СЕРВЕР, КОНФІГУРАТОР ПК, MONGODB, NODE.JS, REACT.

					ІПЗ.КР.Б-121-25-ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Зіневич Р.Р.			Калькулятор для розрахунку конфігурації ПК	Лім.	Арк.
Перевір.							3
Керівник		Локтікова Т. М.				ФІКТ Гр. ІПЗ-21-1	
Н. контр.		Тичина О.П.					
Зав. каф.		Вакалюк Т. А.					
						Аркушів	67

ABSTRACT

The bachelor's thesis consists of a web application for a personal computer configurator and an explanatory note. The explanatory note contains 67 pages, 26 illustrations, and 28 tables.

A convenient platform has been developed for automated selection of compatible PC components, taking into account the user's budget and current technical specifications.

The architecture of the application is based on a client-server model, using modern technologies: React for the frontend, Node.js and Express for the backend, and MongoDB for data storage. Thanks to automatic compatibility checking of the components, the correctness of the created PC builds is ensured.

The web application features an intuitive interface that allows users to quickly choose components, view detailed information about them, and save or update their own PC configurations.

KEYWORDS: CLIENT-SERVER, MONGODB, NODE.JS, PC CONFIGURATOR, REACT, WEB APPLICATION.

		<i>Зіневич Р.Р.</i>			<i>ІПЗ.КР.Б-121-25-ПЗ</i>	<i>Арк.</i>
		<i>Локтікова. Т. М.</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		<i>4</i>

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ КАЛЬКУЛЯТОРА КОНФІГУРАЦІЇ ПК	9
1.1. Постановка задачі.....	9
1.2. Аналіз аналогів програмного продукту	11
1.3. Вибір архітектури системи калькулятора конфігурації ПК.....	16
1.4. Обґрунтування вибору інструментальних засобів	18
Висновки до першого розділу	23
РОЗДІЛ 2. ПРОЄКТУВАННЯ КАЛЬКУЛЯТОРА ДЛЯ РОЗРАХУНКУ КОНФІГУРАЦІЇ ПК.....	25
2.1. Визначення варіантів використання системи	25
2.2. Розробка бази даних.....	32
2.3. Розробка функціональних алгоритмів роботи вебзастосунку	42
2.4. Реалізація функціоналу вебзастосунку	44
Висновки до другого розділу	51
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З КАЛЬКУЛЯТОРОМ ДЛЯ РОЗРАХУНКУ КОНФІГУРАЦІЇ ПК.....	53
3.1. Порядок встановлення та налаштування параметрів вебзастосунку	53
3.2 Структура інтерфейсу	54
3.3 Тестування вебзастосунку	58
Висновки до третього розділу	61
ВИСНОВКИ	63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	64
ДОДАТКИ	67

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПК – персональний комп'ютер

API – Application Programming Interface, інтерфейс для взаємодії клієнта і сервера

CRUD – Create, Read, Update, Delete, основні операції над даними

JWT – JSON Web Token, метод автентифікації користувача

REST API – архітектурний стиль для організації обміну даними

JSX – розширення JavaScript для опису інтерфейсів у React

NoSQL – тип баз даних, що не використовує табличну структуру, зручний для зберігання неструктурованих даних

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми пов'язана із тим, що сучасні технології не стоять на місці, і кожного року випускаються нові компоненти для персональних комп'ютерів, які постійно змінюють цей ринок. Багато користувачів стикаються з проблемою правильного вибору компонентів для своїх ПК, особливо коли мова йде про сумісність та оптимальну продуктивність. Оскільки кількість компонентів на ринку зростає, виникає необхідність у зручному інструменті, який дозволить кожному користувачеві зібрати підходящий ПК з урахуванням всіх своїх потреб і бюджету. Калькулятор для розрахунку конфігурацій ПК є важливим інструментом для тих, хто хоче оптимізувати свої придбання та зробити правильний вибір.

Метою даної роботи є розробка онлайн-калькулятора для розрахунку конфігурації ПК, який дозволить користувачам підбирати та перевіряти сумісність компонентів, а також отримувати вартість для різних конфігурацій.

Для досягнення цієї мети необхідним є виконання таких **завдань**:

- здійснити огляд й аналіз аналогів програмного продукту;
- обрати з обґрунтуванням архітектуру та інструментальні засоби реалізації вебзастосунку;
- спроектувати калькулятор із використанням вибраних засобів;
- виконати тестування калькулятора різними методиками для зменшення ймовірності помилок.

Об'єктом дослідження є процес побудови та проектування калькулятора для розрахунку конфігурації ПК.

Предметом дослідження є методи, засоби та алгоритми створення вебзастосунку для підбору компонентів ПК, інтеграція з базою даних для збереження інформації про компоненти, а також реалізація функцій перевірки сумісності та підрахунку вартості конфігурацій.

За темою випускної кваліфікаційної роботи бакалавра опубліковані тези: Зіневич Р.Р., Локтікова Т.М. Розробка калькулятора для розрахунків конфігурації ПК. Тези доповідей Всеукраїнської науково-практичної on-line конференції

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

здобувачів вищої освіти і молодих учених, присвяченої Дню науки, 12-17 травня 2025 року. Житомир: «Житомирська політехніка», 2025. С. 60.

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ КАЛЬКУЛЯТОРА КОНФІГУРАЦІЇ ПК

1.1. Постановка задачі

У сучасному світі, де технології розвиваються дуже швидко, вибір і налаштування персонального комп'ютера стає важливим і непростим завданням для багатьох людей. Це питання цікавить не лише досвідчених користувачів, але й тих, хто лише починає розбиратися в комп'ютерній техніці. Щороку з'являється багато нових комплектуючих, змінюються характеристики, зростають вимоги до продуктивності. Через це все важче самотійно підібрати комплектуючі для комп'ютера без спеціальних знань.

Тому зараз стає все важливішим мати зручні інструменти, які допомагають користувачу легко підібрати компоненти, перевірити їхню сумісність і сформувати повноцінну конфігурацію ПК. Хоч на ринку вже існує чимало різних сервісів, які дозволяють це зробити, більшість із них мають свої недоліки. Деякі з них надто складні для розуміння, інші — обмежені у функціоналі або не враховують технічні нюанси сумісності між компонентами.

Для звичайного користувача важливо, щоб інструмент був зрозумілим і простим у використанні, не вимагав додаткових налаштувань чи спеціальних знань. На жаль, багато існуючих калькуляторів конфігурації не відповідають цим вимогам. Саме тому виникає потреба у створенні нового інструменту, який би був одночасно простим, гнучким і надійним.

Метою цього проєкту є створення зручного вебзастосунку під назвою "PC Configurator", який дозволить користувачам швидко та легко обирати комплектуючі для персонального комп'ютера. Додаток надаватиме інформацію про сумісність компонентів, а також показуватиме загальну вартість зібраної системи. Особлива увага буде приділена простоті інтерфейсу та швидкості роботи.

У розробці буде використовуватись сучасний стек технологій веб-програмування, що дозволить забезпечити стабільність, продуктивність і зручність

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

у користуванні. Також буде реалізована інтеграція з базою даних, яка містить детальну інформацію про комплектуючі, їхні характеристики та ціни.

Щоб досягти поставленої мети, при розробці системи повинні вирішуватися такі завдання, серед яких:

- розробка дизайну вебсайту, яка включає створення загальної концепції дизайну вебзастосунку, а також створення UI компонентів.

- побудова архітектури клієнтської частини вебзастосунку.

- побудова архітектури серверної частини вебзастосунку.

- розробка системи авторизації та реєстрації.

- розробка основного функціоналу, яка включає реалізацію вкладки конфігурації з фільтрами для вибору кожного типу компонентів, формування остаточної збірки з підрахунком загальної вартості, можливість зберігати зібрані конфігурації в особистому профілі, створення сторінки профілю користувача з переліком його збірок, додавання функцій редагування, видалення та експорту конфігурації в PDF, реалізація кнопки публікації збірки для загального доступу, а також створення сторінки з опублікованими конфігураціями, доступної для всіх користувачів.

- тестування вебзастосунку, яке включає в себе перевірку роботи основного функціоналу, а саме додавання компонентів, збереження, видалення та публікацію збірок, тестування взаємодії з базою даних, а саме правильність зчитування інформації про компоненти та збереження користувацьких даних, а також зручності інтерфейсу в браузері.

Результатом реалізації цього проєкту стане створення вебзастосунку, за допомогою якої користувачі можуть зручно формувати конфігурації ПК, зберігати їх, редагувати, а також ділитися ними з іншими. Опубліковані збірки відображатимуться у відкритому доступі, що дозволить кожному користувачу переглядати готові конфігурації та використовувати як основу для власної.

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2. Аналіз аналогів програмного продукту

Перед початком розробки калькулятора для розрахунку конфігурацій ПК важливо провести аналіз існуючих вебзастосунків на ринку. Оцінка функціональних можливостей, переваг та недоліків цих ресурсів дозволить мінімізувати ризики та створити конкурентоспроможне рішення, яке відповідатиме актуальним потребам користувачів. Вебзастосунки для підбору компонентів ПК пропонують різноманітні функції, включаючи перевірку сумісності компонентів, порівняння цін та інтеграцію з магазинами, однак кожна з них має свої особливості.

PCPartPicker є одним з найбільш популярних вебзастосунків для підбору компонентів ПК [19]. Він дозволяє користувачам створювати конфігурації для ПК, перевіряти сумісність між компонентами та порівнювати ціни на різних магазинах. Вебзастосунок пропонує також калькулятор потужності блоку живлення та велику базу даних відгуків про продукти.

PCPartPicker має функції для порівняння компонентів, перевірки сумісності (наприклад, чи підійде материнська плата до процесора), інтеграцію з онлайн-магазинами для оновлення цін, а також надає можливість створення списків покупок. На рис. 1.1 показано сторінку конфігурації ПК PCPartPicker.

Однією з основних переваг PCPartPicker є зручність використання, інтерфейс простий і інтуїтивно зрозумілий. Інтеграція з багатьма магазинами дозволяє легко порівнювати ціни і знаходити кращі варіанти для покупки компонентів. Також вебзастосунок надає детальні відгуки користувачів, що допомагає прийняти більш обґрунтоване рішення.

Хоча вебзастосунок зручний, його функціональність дещо обмежена, особливо у порівнянні з іншими варіантами, оскільки вона не пропонує можливість кастомізації більше ніж на основі стандартних конфігурацій. Інтерфейс також може бути трохи перевантажений для новачків, оскільки багато інструментів і функцій розташовані на різних екранах.

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

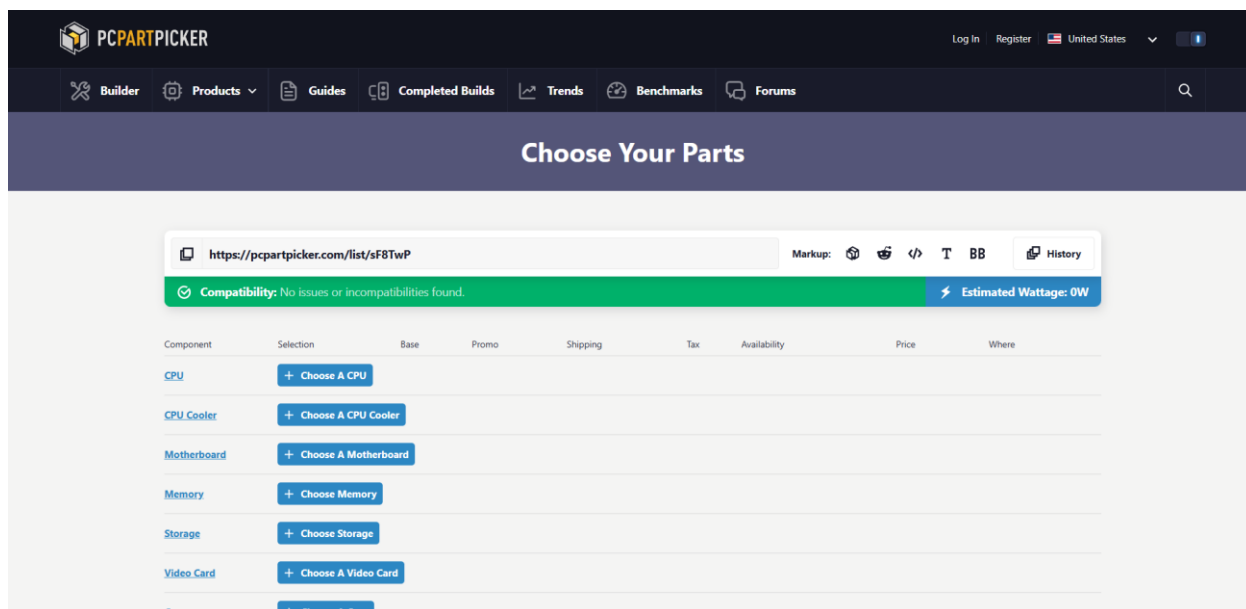


Рисунок 1.1 – Вигляд сторінки конфігурації ПК PCPartPicker

BuildMyPC є ще одним популярним вебзастосунком для підбору комплектуючих для ПК [4]. Він дозволяє створювати конфігурації, автоматично перевіряти їх сумісність і знаходити магазини, де можна придбати компоненти. Вебзастосунок має калькулятор і спеціальний режим для порівняння цін на компоненти. На рис. 1.2 показано сторінку конфігурації ПК BuildMyPC.

Основною функцією BuildMyPC є підбір компонентів за різними критеріями, включаючи бюджет та для чого потрібен ПК (ігри, робота, відеомонтаж тощо). Користувач може вказати бажаний бюджет, і система автоматично підбере оптимальні компоненти, перевірить їх сумісність та відобразить відповідні варіанти покупок.

BuildMyPC має чітку і просту навігацію, що дозволяє користувачам швидко знайти потрібні компоненти і налаштувати конфігурацію. Можливість порівняння цін на компоненти з різних магазинів і перевірка сумісності робить вебзастосунка дуже корисною для користувачів.

Вебзастосунок має обмежену базу даних компонентів порівняно з іншими ресурсами, а також не пропонує достатньо детальну інформацію про компоненти, що може ускладнити процес прийняття рішення. Окрім цього, налаштування фільтрів прості й не дають змоги робити більш детальні налаштування

конфігурацій.

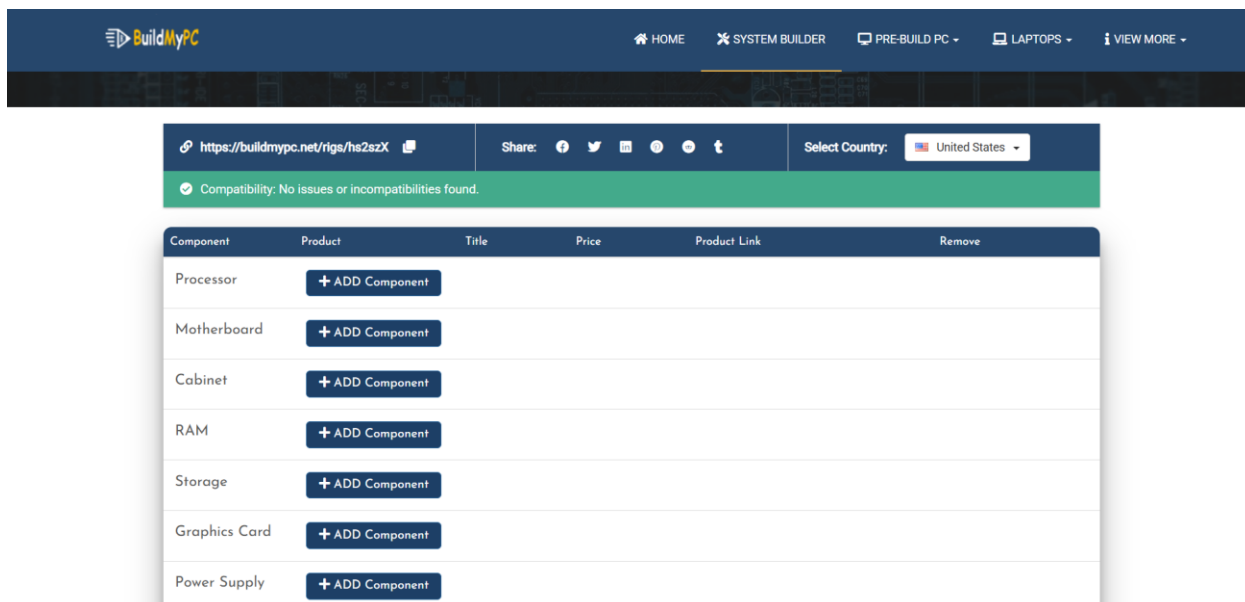


Рисунок 1.2 – Вигляд сторінки конфігурації ПК BuildMyPC

PC Builder дає змогу створити та налаштувати конфігурації ПК, використовуючи велику базу компонентів [18]. Вебзастосунок дозволяє перевіряти сумісність компонентів, порівнювати ціни, а також має функціонал для вибору оптимальних параметрів під конкретні завдання. На рис. 1.3 показано сторінку конфігурації ПК PC Builder.

PC Builder надає можливість вибору різних компонентів, після чого автоматично перевіряється їх сумісність. Вебзастосунок також дозволяє переглядати ціни на компоненти в різних магазинах, що допомагає зробити кращий вибір на основі бюджету.

Основною перевагою є проста система перевірки сумісності, що дуже корисно для користувачів, які хочуть уникнути проблем з сумісністю компонентів. Вебзастосунок також дозволяє легко порівнювати компоненти по різних параметрам, включаючи продуктивність, ціни і відгуки.

Вебзастосунок має деякі обмеження в порівнянні з іншими аналогами, а саме обмежена кількість компонентів у базі даних, а також недостатньо гнучкі налаштування для тих, хто бажає створити дуже специфічну конфігурацію.

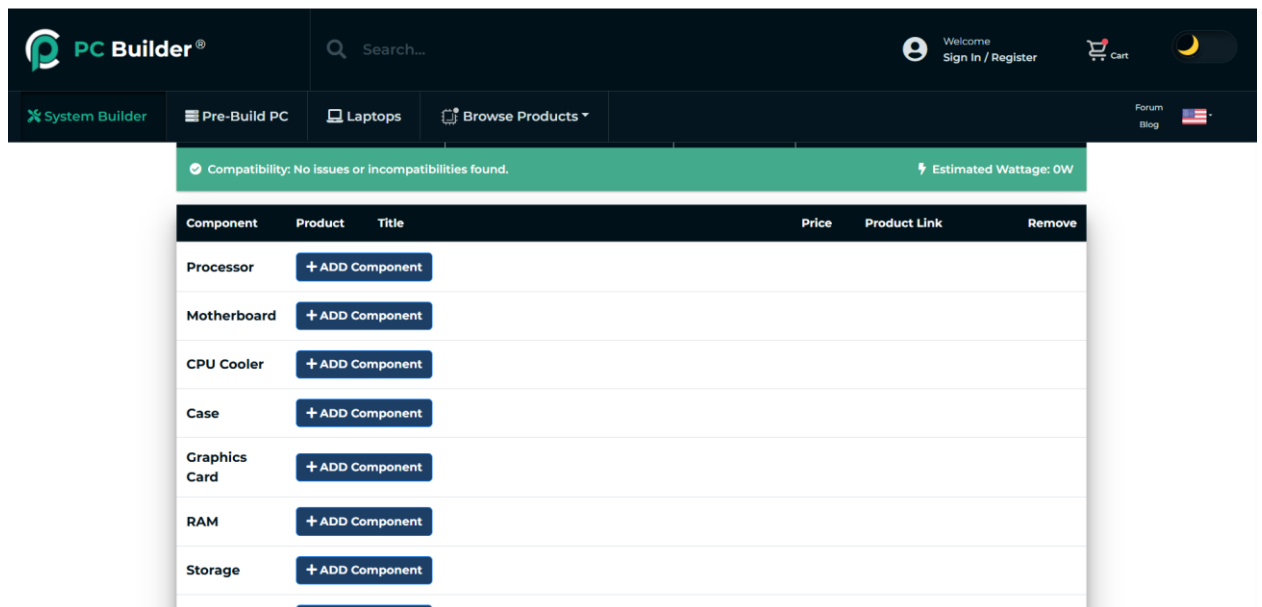


Рисунок 1.3 – Вигляд сторінки конфігурації ПК PC Builder

UserBenchmark — це не лише інструмент для тестування продуктивності ПК, але й вебзастосунок для порівняння компонентів [29]. Користувачі можуть перевіряти ефективність різних комплектуючих і порівнювати їх між собою на основі результатів тестів, які вже доступні у базі вебзастосунка. На рис. 1.4 показано сторінку конфігурації ПК UserBenchmark.

Вебзастосунок дозволяє користувачам тестувати окремі компоненти ПК, а також порівнювати їх з іншими на основі численних тестів, таких як ігрові або програмні тести. Інтерфейс дозволяє швидко знайти інформацію про сумісність та продуктивність кожного компонента.

UserBenchmark надає корисні дані про продуктивність компонентів, що дозволяє користувачам зробити вибір на основі реальних тестів, а не лише характеристик. Це значно допомагає в оцінці ефективності окремих компонентів в умовах реальної роботи.

Проте, головним недоліком є те, що вебзастосунок більше орієнтований на порівняння продуктивності, ніж на підбір компонентів для конкретних конфігурацій. Він не завжди пропонує оптимальні варіанти з урахуванням бюджету, що може зробити його менш зручним для тих, хто шукає комплексні рішення для складання ПК.

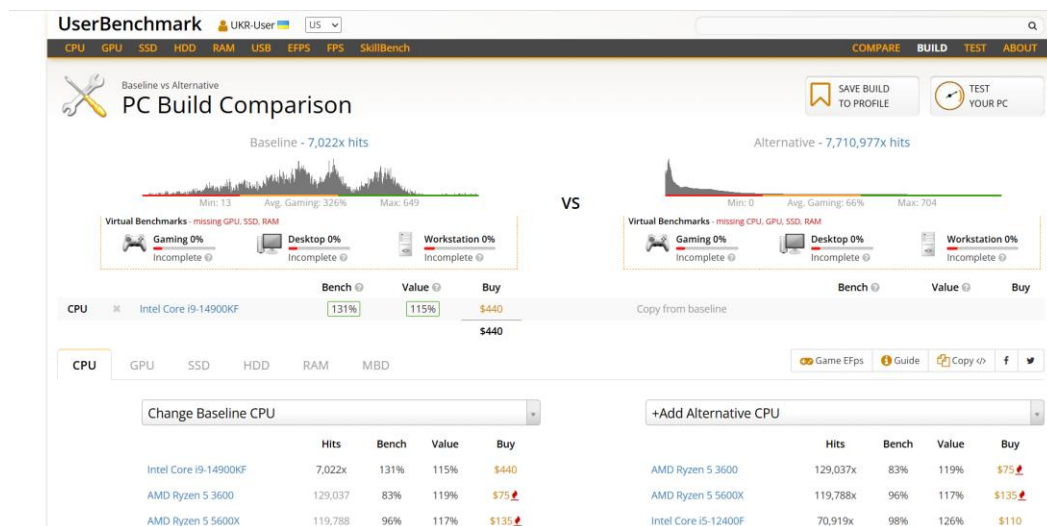


Рисунок 1.4 – Вигляд сторінки конфігурації ПК UserBenchmark

Порівняльна характеристика аналогів програмного продукту за основними параметрами представлена у табл. 1.1.

Таблиця 1.1

Характеристика	PCPartPicker	BuildMyPC	PC Builder	UserBenchmark
Інтерфейс для користувача	+	+	+	+
Перевірка сумісності	+	-	+	+/-
Інтеграція з іншими сервісами	+	+/-	+	-
Безкоштовний доступ	+	+	+/-	+/-
Збереження конфігурації	+	-	+	-
Експорт конфігурації у PDF	-	-	-	-
Публікація збірок для загального перегляду	+/-	+/-	-	-
Управління збірками у профілі	+	-	+/-	-

Більшість вебзастосунків для підбору компонентів ПК пропонують різні функції, зокрема перевірку сумісності та вибір за кількома параметрами. Втім, іноді інтерфейс виходить занадто складним, а налаштування — не зовсім зручними, що створює труднощі для тих користувачів, хто шукає простоту й швидкість у роботі.

Також наявність додаткових функцій може зробити інтерфейс більш складним для користувачів, які не потребують великої кількості налаштувань.

Водночас, відсутність певних базових можливостей, таких як перевірка сумісності або інформація про компоненти, може ускладнити вибір для користувачів, які хочуть більше інформації або мати змогу здійснити більш точний підбір компонентів.

Таким чином, сучасні вебзастосунки для підбору компонентів ПК демонструють великий потенціал, але також потребують доопрацювання для забезпечення простоти використання без втрати функціональності. Це відкриває можливості для створення більш збалансованих рішень, які поєднують зрозумілий інтерфейс та гнучкість у налаштуваннях.

Виходячи з проведених огляду й аналізу аналогів можливо сформулювати вимоги до розроблювального вебзастосунку: зручний і зрозумілий інтерфейс для користувача, перевірка сумісності компонентів, безкоштовний доступ, можливість збереження і експорту конфігурацій у PDF, а також функції публікації та управління збірками у профілі. За допомогою цього розроблювальний "PC Configurator" стане зручним та функціональним інструментом для користувачів.

1.3. Вибір архітектури системи калькулятора конфігурації ПК

Система калькулятора для розрахунків конфігурації ПК розроблена з урахуванням забезпечення гнучкості, масштабованості та надійності при роботі з великими обсягами даних і численними запитами користувачів. Проєкт включає два основні компоненти: користувацький інтерфейс та серверну частину для обробки даних. Такий поділ дозволяє досягти високої продуктивності та ефективної організації коду, розмежовуючи відповідальність між частинами системи.

Основний принцип побудови системи — це зрозуміле розмежування між клієнтською та серверною частинами. Усі складні операції, пов'язані з запитами до бази даних, створення конфігурацій ПК, перевіркою сумісності компонентів, зберіганням та обробкою даних, виконуються на сервері, а користувач отримує результати через зручний вебінтерфейс. Такий підхід забезпечує високу

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

ефективність роботи системи при одночасній мінімізації навантаження на пристрої користувачів.

Користувацький інтерфейс розроблений з використанням сучасних вебтехнологій — React у поєднанні з TypeScript, що дозволяє створювати компонентно-орієнтовану структуру з чіткою типізацією та передбачуваною логікою, а для стилізації застосовується Tailwind CSS — фреймворк, який забезпечує гнучкість в оформленні та прискорює розробку за рахунок використання готових класів. Це дозволяє керувати станом вебзастосунку, реалізовувати взаємодію в реальному часі та динамічно оновлювати дані без перезавантаження сторінки. Завдяки цьому інтерфейс залишається зручним, швидким та продуктивним.

Серверна частина забезпечує обробку запитів, перевірку даних та взаємодію з базою даних. Для її реалізації використовується Node.js, а для роботи з даними обрано MongoDB як основне сховище інформації, що дозволяє ефективно зберігати інформацію про компоненти ПК з різними властивостями.

Для забезпечення гнучкості та масштабованості в системі реалізовано програмний інтерфейс, що дозволяє зручно організувати взаємодію між клієнтом і сервером. Завдяки цьому інтерфейсу кожен компонент клієнтської частини може отримувати доступ до відповідних ресурсів і виконувати необхідні дії — від пошуку сумісних комплектуючих до збереження користувацьких збірок. Така структура дозволяє забезпечує можливість розширення функціональності в майбутньому без значних структурних змін системи.

Усі компоненти системи побудовані на принципах модульності, що дозволяє спрощувати подальший розвиток та підтримку проєкту. Кожен компонент виконує конкретні задачі функції та інтерфейси для взаємодії з іншими частинами системи, що мінімізує ризики при внесенні змін або оновлень. Це дозволяє підтримувати чисту архітектуру, зменшує залежності між компонентами та спрощує налагодження і тестування окремих частин.

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

Такий підхід значно полегшує підтримку й подальший розвиток системи, адже дозволяє ізольовано тестувати, вдосконалювати та модифікувати окремі компоненти без ризику порушення роботи інших частин проєкту. Завдяки модульній архітектурі та чіткій організації коду суттєво знижується ймовірність помилок при внесенні змін, а також спрощується інтеграція нових функцій і розширень. Кожен модуль має власну зону відповідальності, що підвищує гнучкість системи та дозволяє оперативно реагувати на нові вимоги без масштабного переписування коду.

Додатково, архітектура системи передбачає можливість адаптації, що дозволяє ефективно обслуговувати зростаючу кількість користувачів і обсяг даних. Використання сучасних технологій забезпечує сумісність з різними браузером, що підвищує доступність застосунку для широкої аудиторії. Завдяки розмежуванню логіки та організації коду спрощується інтеграція з іншими сервісами і сторонніми API у майбутньому, що відкриває додаткові можливості для розвитку функціональності та адаптації до змін у вимогах користувачів. Таким чином, обрана архітектура забезпечує не тільки стабільну і ефективну роботу системи, а й гнучкість і готовність до масштабування та змін у майбутньому, що дозволяє швидко адаптуватися до нових вимог користувачів і технічних характеристик комплектуючих ПК.

1.4. Обґрунтування вибору інструментальних засобів

Для реалізації калькулятора конфігурації ПК були обрані сучасні, надійні та ефективні інструменти, які забезпечують гнучкість, масштабованість, безпеку та пришвидшують процес розробки.

Клієнтська частина реалізована за допомогою React [22] — популярної бібліотеки для створення динамічних інтерфейсів, у поєднанні з TypeScript [26] — мовою програмування, яка додає статичну типізацію до JavaScript і спрощує розробку великих застосунків. React був обраний завдяки компонентному підходу, який дозволяє створювати багаторазово використовувані елементи інтерфейсу та

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

ефективно керувати їхнім станом. Це особливо важливо для інтерактивного калькулятора конфігурацій ПК. Віртуальний DOM забезпечує високу продуктивність при динамічній зміні інтерфейсу під час вибору користувачем різних комплектуючих.

Крім того, велика кількість розробників та широкий вибір бібліотек значно спрощують процес створення вебзастосунку. Використання TypeScript забезпечує переваги статичної типізації, що зменшує кількість помилок на етапі розробки та покращує підтримку коду при роботі зі складними структурами даних.

Для порівняння можливостей сучасних фреймворків, які можуть бути використані для створення клієнтської частини вебзастосунку, у табл. 1.2 наведено їх основні переваги та недоліки.

Таблиця 1.2

Порівняння фреймворків для клієнтської частини

Фреймворк	Переваги	Недоліки
React	Компонентний підхід	Потребує додаткових бібліотек для маршрутизації та управління станом
	Віртуальний DOM	Висока складність для початківців
	Велика спільнота та підтримка	Може бути надлишковим для простих проєктів
Vue.js	Прості в освоєнні шаблони	Менша екосистема бібліотек
	Компактність	Менша кількість розробників на ринку
	Вбудовані функції	Менша оптимізація для складних додатків

Фреймворк	Переваги	Недоліки
Angular	Повний фреймворк з усіма інструментами	Високий поріг входження
	Двостороннє зв'язування даних	Великий обсяг фінального коду
	Вбудована підтримка TypeScript	Складніша конфігурація

Для стилізації інтерфейсу використовується Tailwind CSS [25] — гнучкий фреймворк, який дозволяє швидко створювати адаптивні інтерфейси без написання власних стилів. Його обрано завдяки зручності у розробці навіть складних елементів дизайну. Tailwind дозволяє задавати стилі просто у HTML-коді через класи, що значно пришвидшує роботу та полегшує підтримку візуальної частини вебзастосунку.

Для реалізації серверної частини було обрано Node.js [17] — платформу, яка дозволяє створювати швидкі та надійні вебсервери. Її асинхронна, подієво-орієнтована архітектура дає змогу швидко обробляти запити та ефективно використовувати ресурси. Однією з ключових переваг Node.js є можливість працювати з JavaScript як на клієнті, так і на сервері, що спрощує розробку та дозволяє повторно використовувати логіку й типи даних. Крім того, велика кількість вже розроблених пакетів у репозиторії npm значно полегшує додавання необхідного функціоналу — наприклад, для роботи з базами даних або реалізації REST API.

З метою обґрунтованого вибору інструментів для реалізації серверної частини системи, у табл. 1.3 представлено порівняльну характеристику популярних серверних платформ.

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

Порівняння серверних платформ

Платформи	Переваги	Недоліки
Node.js	Асинхронна подієво-орієнтована архітектура	Менш продуктивний для CPU-інтенсивних завдань
	Єдина мова програмування (JavaScript)	Неправильне використання призводить до callback, що створює некоректність логіки програми
	Велика спільнота та прм екосистема	Менш розвинена екосистема порівняно з Java або .NET
Java EE	Висока продуктивність	Велика складність та надмірність
	Корпоративний стандарт	Потребує більше ресурсів
	Багато перевірених бібліотек	Повільніша розробка
ASP.NET	Інтеграція з екосистемою Microsoft	Вища вартість хостингу
	Високий рівень безпеки	Менша гнучкість
	Масштабованість	Складніший процес розгортання

Для роботи з даними обрано MongoDB [14] як основне сховище інформації про комплектуючі ПК. Ця документоорієнтована NoSQL база даних вибрана

завдяки її гнучкій схемі, яка ідеально підходить для зберігання різнотипних характеристик комплектуючих з різною структурою даних. MongoDB забезпечує високу продуктивність при операціях читання та зберігання даних у форматі BSON (бінарний JSON), що оптимізовано для швидкої обробки запитів. Це особливо важливо для калькулятора конфігурацій ПК, де необхідно швидко фільтрувати та обробляти значні обсяги даних про комплектуючі. Вертикальне та горизонтальне масштабування MongoDB дозволяє нарощувати систему при збільшенні каталогу комплектуючих та кількості користувачів. Для роботи з MongoDB використовується бібліотека Mongoose [15], яка забезпечує зручний інтерфейс для визначення схем даних, валідації та виконання запитів до бази даних.

Для обґрунтування вибору бази даних, що найкраще підходить для зберігання інформації про компоненти ПК, у табл. 1.4 наведено аналіз трьох популярних рішень.

Таблиця 1.4

Порівняння баз даних

Бази даних	Тип	Переваги	Недоліки
MongoDB	Нереляційна	Гнучка схема даних	Обмежені можливості для складних зв'язків
		Легке масштабування	Менша підтримка транзакцій
		Зберігання JSON-подібних документів	Займає більше місця порівняно з SQL
PostgreSQL	Реляційна	Підтримка складних запитів	Складна в налаштуванні
		Висока надійність і цілісність даних	Вища вартість хостингу
		ACID-транзакції	Менша гнучкість схеми даних
MySQL	Реляційна	Підтримка складних зв'язків	Обмеження для неструктурованих даних
		Широка документація	Проблеми з масштабуванням

Для забезпечення безпеки вебзастосунку використовуються бібліотеки bcryptjs [3] — для надійного шифрування паролів користувачів, та jsonwebtoken — для реалізації авторизації на основі JWT-токенів. Ці інструменти забезпечують захист облікових даних та безпечний доступ до функціоналу системи.

Як інтегроване середовище розробки було обрано WebStorm, що надає потужні інструменти для роботи з JavaScript, TypeScript та React. WebStorm підтримує інтелектуальне автодоповнення коду, має вбудований термінал, інтеграцію з системами контролю версій та інші функції, які суттєво пришвидшують розробку й підвищують якість програмного забезпечення.

Для тестування API використовується Postman, що дозволяє створювати, надсилати та аналізувати HTTP-запити до серверної частини. Для роботи з базою даних застосовується MongoDB, який надає зручний інтерфейс для управління колекціями та документами MongoDB.

Обраний стек технологій та інструментів забезпечує гнучкість, масштабованість і продуктивність під час розробки. Це дозволяє ефективно вирішувати завдання, пов'язані з підбором комплектуючих і перевіркою їх сумісності.

Висновки до першого розділу

У першому розділі було проведено аналіз напрямків застосування інформаційних технологій для створення калькулятора конфігурації ПК. Розглянуто кілька популярних аналогів, а саме PCPartPicker, BuildMyPC та інші, визначено їх переваги й недоліки. Це дозволило виділити актуальні потреби користувачів і сформулювати вимоги до функціональності майбутньої системи.

Обґрунтовано вибір архітектури клієнт-серверного типу з розподілом відповідальностей між частинами, що сприяє гнучкості та стабільності роботи. Для реалізації клієнтської частини обрано React з TypeScript, для серверної — Node.js, а в якості бази даних — MongoDB, яка забезпечує високу продуктивність і зручну

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

структуру зберігання. Також використано Tailwind CSS для стилізації інтерфейсу та бібліотеки bcryptjs і jsonwebtoken для захисту облікових даних.

Обраний технологічний стек створює основу для подальшої розробки ефективного, зручного й масштабованого вебзастосунку.

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

РОЗДІЛ 2. ПРОЄКТУВАННЯ КАЛЬКУЛЯТОРА ДЛЯ РОЗРАХУНКУ КОНФІГУРАЦІЇ ПК

2.1. Визначення варіантів використання системи

Вебзастосунок "PC Configurator" реалізовано за клієнт-серверною архітектурою. Уся основна логіка, включно з обробкою запитів, роботою з базою даних, автентифікацією користувачів і перевіркою прав доступу, виконується на сервері. Клієнтська частина, реалізована на базі React, відповідає за зручну візуалізацію інтерфейсу та забезпечення інтерактивної взаємодії з користувачем.

Застосунок підтримує два основних типи користувачів — гостьовий та авторизований користувач. Гості можуть переглядати доступні комп'ютерні компоненти, ознайомлюватися з опублікованими збірками, а також проходити процедури реєстрації та авторизації. Після входу або створення облікового запису користувач отримує доступ до розширеного функціоналу: створення, збереження, редагування, видалення та публікації власних конфігурацій, а також експортування їх у форматі PDF.

Такий розподіл ролей і функцій забезпечує як гнучкість, так і безпечність використання системи. Для наочної демонстрації взаємодії користувачів із системою було побудовано діаграму варіантів використання (рис. 2.1), яка відображає ролі, сценарії доступу та основні дії в межах вебінтерфейсу.

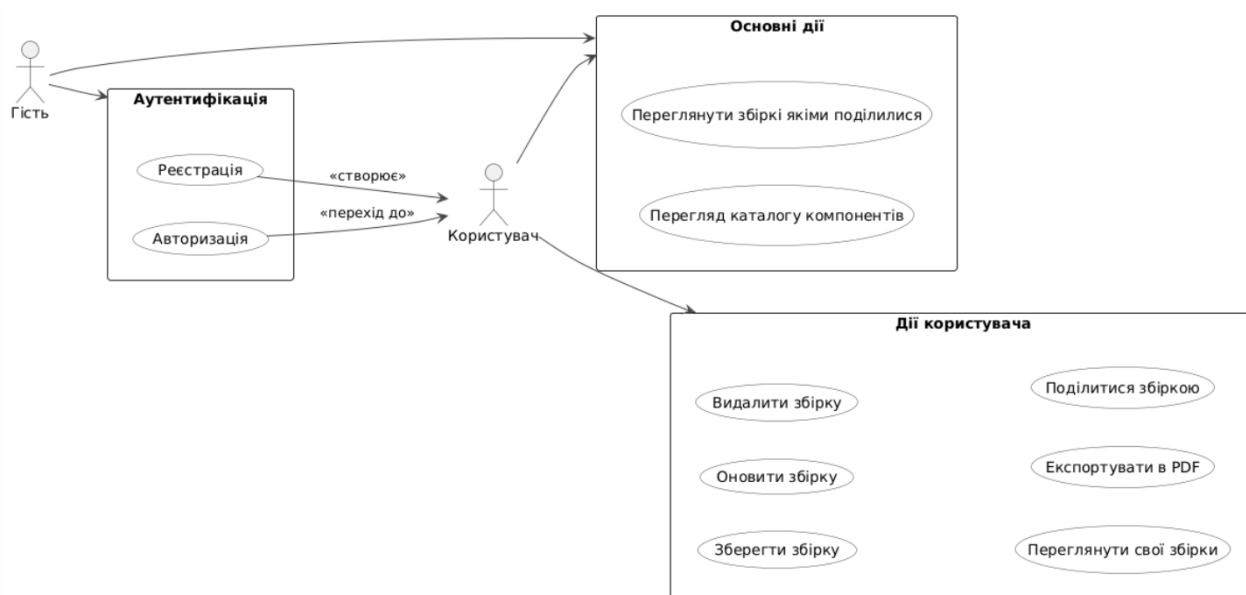


Рисунок 2.1 – Діаграма варіантів використання

Опис діаграми варіантів використання наведено в табл. 2.1 – 2.8.

Таблиця 2.1

Опис прецеденту «Вхід/Реєстрація»

Назва варіанту використання	Вхід/Реєстрація
Основні актори	Гостьовий користувач
Короткий опис	Користувач може створити новий акаунт через реєстрацію або увійти в систему за допомогою логіна і пароля, щоб отримати доступ до розширеного функціоналу вебзастосунку.

Таблиця 2.2

Опис прецеденту «Перегляд каталогу компонентів»

Назва варіанту використання	Перегляд каталогу компонентів
Основні актори	Гостьовий користувач, авторизований користувач
Короткий опис	Користувач має можливість переглядати доступні комп'ютерні компоненти з описом і ціною, без потреби авторизації.

Таблиця 2.3

Опис прецеденту «Перегляд публічних збірок»

Назва варіанту використання	Перегляд публічних збірок ПК
Основні актори	Гостьовий користувач, авторизований користувач
Короткий опис	Користувач може ознайомитися з конфігураціями, опублікованими іншими користувачами, переглядаючи їх у загальному каталозі.

Таблиця 2.4

Опис прецеденту «Створення збірки»

Назва варіанту використання	Створення власної конфігурації ПК
Основні актори	Авторизований користувач
Короткий опис	Користувач може створити нову збірку комп'ютера, вибираючи необхідні компоненти з каталогу та натиснувши на зберегти її, для подальшого використання.

Таблиця 2.5

Опис прецеденту «Оновлення збірки»

Назва варіанту використання	Оновлення конфігурації ПК через профіль
Основні актори	Авторизований користувач
Короткий опис	Користувач має можливість оновити збірку, перейшовши з профілю на сторінку конфігуратора з уже заповненими полями, внісши необхідні зміни і зберігши оновлену версію.

Таблиця 2.6

Опис прецеденту «Видалення збірки»

Назва варіанту використання	Видалення власної збірки ПК
Основні актори	Авторизований користувач
Короткий опис	Користувач може видалити свою збірку зі списку збережених конфігурацій, щоб вона більше не відображалася у його профілі.

Таблиця 2.7

Опис прецеденту «Публікація збірки»

Назва варіанту використання	Публікація конфігурації для загального доступу
Основні актори	Авторизований користувач
Короткий опис	Користувач може зробити свою збірку публічною, щоб інші користувачі могли її переглядати у загальному каталозі публічних конфігурацій.

Таблиця 2.8

Опис прецеденту «Експорт збірки у PDF»

Назва варіанту використання	Генерація PDF-файлу зі збіркою ПК
Основні актори	Авторизований користувач
Короткий опис	Користувач може згенерувати PDF-документ із повним списком компонентів своєї збірки, включно з їх назвами, типами та вартістю, для зручності збереження або друку.

Таким чином, вебзастосунок "PC Configurator" надає користувачам широкий набір функціональних можливостей — від базового ознайомлення з каталогом комп'ютерних компонентів до створення та керування власними збірками ПК. Завдяки чіткому розподілу ролей (гостьовий і авторизований користувач) та продуманій архітектурі, система дозволяє зручно налаштовувати й зберігати персональні конфігурації, а також ділитися ними з іншими користувачами. Діаграма варіантів використання наочно демонструє основні сценарії взаємодії з вебзастосунком та відображає логіку роботи вебзастосунку, дотримуючись вимог безпеки й зручності. Усі ключові дії та варіанти використання детально наведені в табл. 2.1 – 2.8.

Далі розглянемо вимоги користувачів.

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

Якщо користувач авторизований, він повинен мати можливість зареєструватися та входити у систему, переглядати каталог комп'ютерних компонентів, створювати, оновлювати та видаляти власні конфігурації ПК, переглядати вміст власних збірок, публікувати власні збірки для загального перегляду, а також експортувати конфігурації у PDF-файл.

Якщо користувач є гостьовим, він повинен мати можливість переглядати опубліковані конфігурації, ознайомлюватися з доступними комплектуючими, а також мати можливість зареєструватися та авторизуватися.

Тепер розглянемо вимоги функціоналу.

До функціональних вимог належать відображення та фільтрація комп'ютерних компонентів за категоріями (процесори, відеокарти, оперативна пам'ять тощо) та основними параметрами, реалізація авторизації та реєстрації користувачів через email і пароль, а також створення, оновлення й видалення конфігурацій ПК. Крім того, передбачено перевірку автоматичної сумісності обраних компонентів у конфігурації, можливість публікації збірок для загального перегляду та експорту цих збірок у формат PDF. Також користувачі матимуть особистий кабінет з історією створених конфігурацій.

Щодо нефункціональних вимог, то вебзастосунок має бути швидким, з часом відповіді API не більше 500 мс, а всі запити повинні бути захищені, з автентифікацією за допомогою JWT. Важливим аспектом є забезпечення можливості одночасної роботи великої кількості користувачів, підтримка роботи на основних сучасних браузерях і модульна архітектура для подальшого розширення функціоналу.

Системні вимоги включають вимоги як до серверної, так і до клієнтської частини вебзастосунку. Для серверної частини розробка має здійснюватися на операційній системі Windows 10 або новішій версії, а платформою є Node.js (версія 18 і вище) у поєднанні з базою даних MongoDB (версія 6 і вище). Що стосується клієнтської частини, вона повинна підтримувати сучасні операційні системи та

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

забезпечувати сумісність з останніми версіями основних браузерів. У ролі фронтенд-фреймворка використовується React версії 18 і вище.

Апаратні вимоги для серверної частини включають наявність процесора з мінімум двома ядрами, щонайменше 8 ГБ оперативної пам'яті та не менше 150 ГБ вільного місця на диску. Для клієнтської частини основними умовами є стабільне інтернет-з'єднання та використання браузера, який підтримує сучасні вебтехнології.

Аналіз функціональних вимог дозволив виділити основні компоненти вебзастосунку "PC Configurator". На рис. 2.2 наведено діаграму компонентів, яка відображає основні програмні модулі системи (зокрема контролери на бекенді) та їхню взаємодію. Оскільки зв'язки між цими модулями реалізовані через REST API, уся взаємодія відбувається через запити з клієнтської частини. Таким чином, клієнт виконує виклики до контролерів, які, у свою чергу, відповідають за обробку запитів та повернення даних.

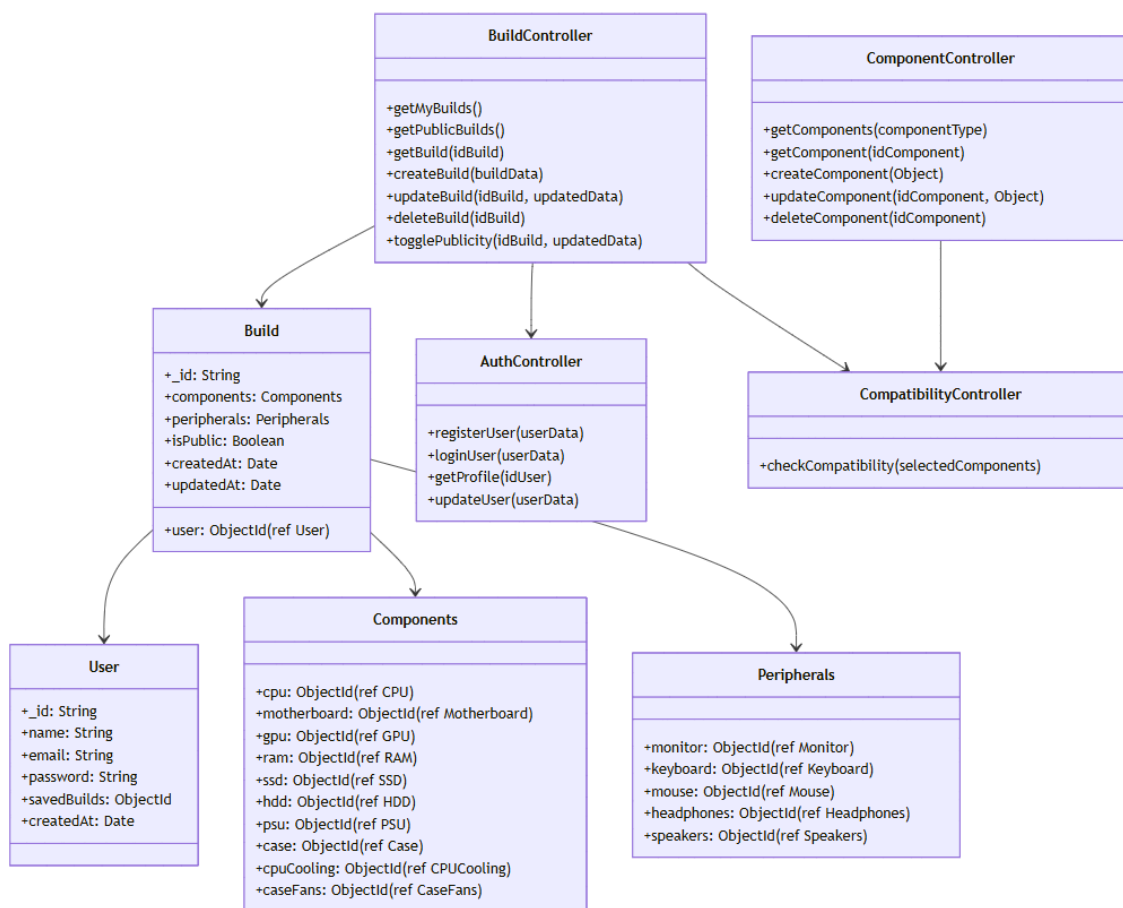


Рисунок 2.2 – Діаграма компонентів

Опишемо діаграму компонентів, яка зображена на рис. 2.2.

У вебзастосунку "PC Configurator" реалізовано кілька основних контролерів, кожен з яких відповідає за окремий функціонал. Контролер AuthController відповідає за управління обліковими записами користувачів і надає методи для реєстрації користувача (registerUser(userData)), авторизації (loginUser(userData)), отримання профілю (getProfile(idUser)).

Контролер BuildController призначений для роботи зі збірками ПК та дозволяє отримувати власні збірки користувача (getMyBuilds()), переглядати публічні збірки (getPublicBuilds()), отримувати дані про конкретну збірку (getBuild(idBuild)), створювати нові збірки (createBuild(buildData)), оновлювати (updateBuild(idBuild, updatedData)) та видаляти (deleteBuild(idBuild)) їх, а також змінювати статус публічності збірки (togglePublicity(idBuild, updatedData)).

Контролер ComponentController відповідає за управління окремими компонентами ПК та дозволяє отримувати список компонентів певного типу (getComponents(componentType)), отримувати дані про конкретний компонент (getComponent(idComponent)), а також створювати (createComponent(Object)), оновлювати (updateComponent(idComponent, Object)) та видаляти (deleteComponent(idComponent)) їх.

Для перевірки сумісності обраних компонентів використовується контролер CompatibilityController, який реалізує метод перевірки сумісності (checkCompatibility(selectedComponents)).

У системі передбачено також кілька моделей даних, що описують структуру конфігурацій ПК. Модель Build містить інформацію про збірку, зокрема власника (user), вибрані компоненти (components), периферійні пристрої (peripherals), статус публічності (isPublic), а також дату створення та оновлення. Модель User зберігає дані про користувачів, включаючи ім'я, електронну пошту, пароль, список збережених збірок (savedBuilds) і дату створення профілю. Моделі Components та Peripherals визначають зв'язки між конкретними комплектуючими та периферійними пристроями у збірці, які теж представлені вигляді моделей в коді.

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

Зв'язки між компонентами і контролерами реалізовані через API-запити. Так, BuildController використовує методи AuthController для отримання даних про користувачів, а також методи CompatibilityController для перевірки сумісності обраних компонентів. ComponentController також може звертатися до CompatibilityController для перевірки сумісності. Однак ці зв'язки реалізовані не безпосередньо у вигляді класичних залежностей, а через REST API, що забезпечує взаємодію між серверною та клієнтською частинами вебзастосунку.

2.2. Розробка бази даних

Для збереження даних у вебзастосунку використовується база даних MongoDB, яка дозволяє зберігати інформацію у вигляді гнучких документів формату BSON. Такий підхід дає змогу зберігати різні типи даних без жорсткої прив'язки до структури, що дуже зручно в процесі розробки і подальшого розширення функціоналу. У системі реалізовано 16 колекцій, які відповідають за зберігання різноманітної інформації — від даних про користувачів до компонентів ПК та зібраних конфігурацій.

Для взаємодії з базою даних застосовується бібліотека Mongoose, що забезпечує роботу з документами у вигляді класичних об'єктів, підтримує валідацію даних і захист паролів шляхом хешування. Така структура дозволяє ефективно управляти інформацією, виконувати пошук за різними критеріями та зберігати дані у логічному вигляді.

Опис кожної з колекцій бази даних наведений у табл. 2.9 – 2.24.

Таблиця 2.9

Опис колекції "User"

Поле	Тип	Призначення
username	String	Ім'я користувача.
email	String	Електронна пошта користувача.
password	String	Пароль користувача у зашифрованому вигляді.
savedBuilds	ObjectId	Масив ідентифікаторів збірок, збережених цим користувачем.
createdAt	String	Дата створення облікового запису.

Опис колекції "Build"

Поле	Тип	Призначення
buildNumber	String	Унікальний номер збірки, який формується автоматично при створенні.
name	String	Назва збірки, може бути задана вручну або автоматично.
description	String	Опис збірки, додається користувачем.
user	ObjectId	Ідентифікатор користувача, якому належить ця збірка.
isPublic	Boolean	Показник, чи є збірка публічною.
components	Object	Об'єкт, що містить всі основні компоненти ПК
components.cpu	Object	Посилання на процесор.
components.motherboard	Object	Посилання на материнську плату.
components.gpu	Object	Посилання на відеокарту.
components.ram	Object	Масив оперативної пам'яті.
components.ssd	Object	Масив SSD-накопичувачів.
components.hdd	Object	Масив HDD-дисків.
components.psu	Object	Посилання на блок живлення.
components.case	Object	Посилання на корпус.
components.cpuCooling	Object	Посилання на систему охолодження CPU.
components.caseFans	Object	Масив додаткових вентиляторів корпусу.
peripherals	Object	Об'єкт, що містить периферійні пристрої.
peripherals.monitor	Object	Масив моніторів.
peripherals.keyboard	Object	Посилання на клавіатуру.
peripherals.mouse	Object	Посилання на мишу.
peripherals.headphones	Object	Посилання на навушники.
peripherals.speakers	Object	Посилання на колонки.
totalPrice	Number	Загальна вартість збірки.
componentsPrice	Number	Вартість лише компонентів.
peripheralsPrice	Number	Вартість периферії.
createdAt	Date	Дата створення збірки.
updatedAt	Date	Дата останнього оновлення збірки.

Таблиця 2.11

Опис колекції "Case"

Поле	Тип	Призначення
name	String	Назва корпусу
manufacturer	String	Виробник
model	String	Модель корпусу
formFactor	String	Форм-фактор корпусу
motherboardSupport	String[]	Підтримка форм-факторів материнських плат
maxGPULength	Number	Макс. довжина відеокарти
maxCPUCoolerHeight	Number	Макс. висота кулера процесора
maxPSULength	Number	Макс. Довжина блоку живлення
expansionSlots	Number	Кількість слотів розширення
includedFans	Number	Встановлені вентилятори
windowedSide	Boolean	Прозора бокова панель
price	Number	Ціна
image	String	Зображення

Таблиця 2.12

Опис колекції «CaseFan»

Поле	Тип	Призначення
name	String	Назва вентилятора
manufacturer	String	Виробник
model	String	Модель
size	Number	Розмір вентилятора (80, 92, 120, 140, 200 мм)
rpm.min	Number	Мінімальні оберти в хв
rpm.max	Number	Максимальні оберти в хв
airflow	Number	Повітряний потік (CFM)
noiseLevel	Number	Рівень шуму (дБ)
pwm	Boolean	Підтримка PWM
connector	String	Тип роз'єму (3-pin, 4-pin, Molex)
rgb	Boolean	Підсвітка RGB
argb	Boolean	Підсвітка ARGB
controller	Boolean	Наявність контролера
price	Number	Ціна
image	String	Зображення (посилання)

Опис колекції «CPU»

Поле	Тип	Призначення
name	String	Назва процесора
manufacturer	String	Виробник (Intel, AMD, Other)
model	String	Модель процесора
socket	String	Сокет процесора
cores	Number	Кількість ядер
threads	Number	Кількість потоків
baseClock	Number	Базова тактова частота (ГГц)
boostClock	Number	Максимальна тактова частота (ГГц)
tdp	Number	TDP (Вт)
integratedGraphics	Boolean	Наявність вбудованої графіки
price	Number	Ціна (USD)
image	String	Посилання на зображення

Опис колекції «CPUCooling»

Поле	Тип	Призначення
name	String	Назва системи охолодження
manufacturer	String	Виробник
model	String	Модель охолодження
type	String	Тип охолодження (Air Cooler, AIO Liquid Cooler, Custom Loop)
socketSupport	Array[String]	Підтримувані сокети процесорів
tdpRating	Number	Розрахункове тепловиділення (TDP) у Вт
fanSize	Number	Розмір вентилятора (мм)
fanCount	Number	Кількість вентиляторів (за замовчуванням 1)
rpm.min	Number	Мінімальні оберти вентилятора (RPM)
rpm.max	Number	Максимальні оберти вентилятора (RPM)
height	Number	Висота охолоджувача (для Air Cooler)
radiatorSize	String	Розмір радіатора (для AIO/Custom Loop)
rgb	Boolean	Наявність RGB-підсвітки
argb	Boolean	Наявність адресної RGB-підсвітки
price	Number	Ціна в USD
image	String	Посилання на зображення продукту

Таблиця 2.15

Опис колекції «GPU»

Поле	Тип	Призначення
name	String	Назва відеокарти
manufacturer	String	Виробник відеокарти
chipManufacturer	String	Виробник графічного чіпа (NVIDIA, AMD, Intel, Other)
model	String	Модель відеокарти
vram	Number	Об'єм відеопам'яті (ГБ)
memoryType	String	Тип пам'яті (GDDR5, GDDR6 тощо)
baseClock	Number	Базова тактова частота (МГц)
boostClock	Number	Максимальна тактова частота (МГц)
tdp	Number	TDP — тепловиділення (Вт)
length	Number	Довжина відеокарти (мм)
pcieInterface	String	Інтерфейс PCI-E (PCIe 3.0, 4.0, 5.0)
powerConnectors	String	Роз'єм(и) живлення відеокарти
recommendedPSU	Number	Рекомендована потужність блока живлення (Вт)
price	Number	Ціна в USD
image	String	Посилання на зображення продукту

Таблиця 2.16

Опис колекції «HDD»

Поле	Тип	Призначення
name	String	Назва HDD накопичувача
manufacturer	String	Виробник
model	String	Модель HDD
formFactor	String	Форм-фактор (2.5» або 3.5»)
interface	String	Інтерфейс підключення (SATA II, SATA III, SAS)
capacity	Number	Об'єм диска (ГБ)
rpm	Number	Швидкість обертання шпинделя (об/хв)
cacheSize	Number	Розмір кешу (МБ)
readSpeed	Number	Швидкість читання (МБ/с)
writeSpeed	Number	Швидкість запису (МБ/с)
mtbf	Number	Середній час напрацювання на відмову (MTBF, години)

Поле	Тип	Призначення
price	Number	Ціна в USD
image	String	Посилання на зображення продукту

Таблиця 2.17

Опис колекції «Headphones»

Поле	Тип	Призначення
name	String	Назва навушників
manufacturer	String	Виробник
model	String	Модель
type	String	Тип (Over-ear, In-ear...)
purpose	String	Призначення (Gaming, Studio...)
wireless	Boolean	Бездротові
connectionType.usb	Boolean	USB підключення
microphoneIncluded	Boolean	Мікрофон у комплекті
noiseCancellation	Boolean	Шумозаглушення
batteryLife	Number	Час роботи (год)
weight	Number	Вага (г)
price	Number	Ціна (USD)
image	String	Зображення

Таблиця 2.18

Опис колекції «Keyboard»

Поле	Тип	Призначення
name	String	Назва клавіатури
manufacturer	String	Виробник
model	String	Модель
type	String	Тип (Mechanical, Membrane...)
layout	String	Розкладка (Full-size, TKL...)
switchType	String	Тип перемикачів
hotSwappable	Boolean	Можливість гарячої заміни
backlighting	String	Підсвітка (None, RGB...)
wireless	Boolean	Бездротова
connectionType.usb	Boolean	USB підключення
connectionType.bluetooth	Boolean	Bluetooth
batteryLife	Number	Час роботи (год)

Продовження табл. 2.18

Поле	Тип	Призначення
nkeyRollover	Boolean	N-key rollover підтримка
macroKeys	Number	Кількість макроклавіш
price	Number	Ціна (USD)
image	String	Зображення

Таблиця 2.19

Опис колекції «Monitor»

Поле	Тип	Призначення
name	String	Назва монітора
manufacturer	String	Виробник
model	String	Модель
size	Number	Діагональ екрану (дюйми)
resolution.width	Number	Ширина роздільної здатності
resolution.height	Number	Висота роздільної здатності
panelType	String	Тип панелі (IPS, TN...)
refreshRate	Number	Частота оновлення (Гц)
responseTime	Number	Час відгуку (мс)
brightness	Number	Яскравість (кд/м²)
contrastRatio	String	Контрастність
hdr.supported	Boolean	Підтримка HDR
hdr.type	String	Тип HDR
inputs.hdmi	Number	Кількість HDMI входів
inputs.displayPort	Number	Кількість DisplayPort входів
inputs.usbc	Number	Кількість USB-C входів
speakers	Boolean	Вбудовані колонки
price	Number	Ціна (USD)
image	String	Зображення

Таблиця 2.20

Опис колекції «Motherboard»

Поле	Тип	Призначення
name	String	Назва материнської плати
manufacturer	String	Виробник
model	String	Модель

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

Поле	Тип	Призначення
chipset	String	Чіпсет
socket	String	Сокет
formFactor	String	Форм-фактор (ATX, Mini-ITX...)
memoryType	String	Тип пам'яті (DDR3, DDR4, DDR5)
memorySlots	Number	Кількість слотів пам'яті
maxMemory	Number	Максимальний об'єм пам'яті (ГБ)
pcieSlots.x16	Number	Кількість PCIe x16 слотів
pcieSlots.x8	Number	Кількість PCIe x8 слотів
pcieSlots.x4	Number	Кількість PCIe x4 слотів
pcieSlots.x1	Number	Кількість PCIe x1 слотів
sataConnectors	Number	Кількість SATA роз'ємів
m2Slots	Number	Кількість M.2 слотів
usbPorts.usb2	Number	USB 2.0 порти
usbPorts.usb3	Number	USB 3.0 порти
usbPorts.usbc	Number	USB-C порти
wifi	Boolean	Підтримка Wi-Fi
bluetooth	Boolean	Підтримка Bluetooth
price	Number	Ціна (USD)
image	String	Посилання на зображення

Таблиця 2.21

Опис колекції «Mouse»

Поле	Тип	Призначення
name	String	Назва миші
manufacturer	String	Виробник
model	String	Модель
type	String	Тип миші (Gaming, Office, Ergonomic, Travel)
sensor	String	Тип сенсора (Optical, Laser)
dpi.min	Number	Мінімальне DPI
dpi.max	Number	Максимальне DPI
buttons	Number	Кількість кнопок
programmableButtons	Number	Програмовані кнопки (за замовчуванням 0)
wireless	Boolean	Безпроводна (true/false)
connectionType.usb	Boolean	USB з'єднання (true/false)

Поле	Тип	Призначення
connectionType.bluetooth	Boolean	Bluetooth (true/false)
connectionType.wireless24Ghz	Boolean	Радіозв'язок 2.4GHz (true/false)
batteryLife	Number	Час роботи батареї (хвилини)
weight	Number	Вага (г)
adjustableWeight	Boolean	Регульована вага (true/false)
pollingRate	Number	Частота опитування (Hz)
rgb	Boolean	RGB підсвітка (true/false)
dimensions.length	Number	Довжина (мм)
dimensions.width	Number	Ширина (мм)
dimensions.height	Number	Висота (мм)
gripType	String	Тип хвата (Palm, Claw, Fingertip, Hybrid)
handOrientation	String	Орієнтація руки (Right-handed, Left-handed, Ambidextrous)
price	Number	Ціна (USD)
image	String	URL зображення

Таблиця 2.22

Опис колекції «PSU»

Поле	Тип	Призначення
name	String	Назва блоку живлення
manufacturer	String	Виробник
model	String	Модель
power	Number	Потужність (Вт)
efficiency	String	Ефективність (80+ White, Bronze, Silver, Gold, Platinum, Titanium)
modularity	String	Модульність (None, Semi, Fully)
connectors.cpu8pin	Number	8-pin CPU конектори
connectors.cpu4pin	Number	4-pin CPU конектори
connectors.pcie8pin	Number	8-pin PCIe конектори
connectors.pcie6pin	Number	6-pin PCIe конектори
connectors.pcie16pin	Number	16-pin PCIe конектори
connectors.sata	Number	SATA конектори
connectors.molex	Number	Molex конектори

Продовження табл. 2.22

Поле	Тип	Призначення
atx3	Boolean	Підтримка ATX 3.0 (true/false)
fanSize	Number	Розмір вентилятора (мм)
formFactor	String	Форм-фактор (ATX, SFX, SFX-L, TFX, Flex ATX)
price	Number	Ціна (USD)
image	String	URL зображення

Таблиця 2.23

Опис колекції «RAM»

Поле	Тип	Призначення
name	String	Назва оперативної пам'яті
manufacturer	String	Виробник
model	String	Модель
type	String	Тип пам'яті (DDR3, DDR4, DDR5)
capacity	Number	Об'єм пам'яті (ГБ)
speed	Number	Частота пам'яті (МГц)
modules	Number	Кількість модулів у комплекті
voltage	Number	Напруга (В)
heatsink	Boolean	Наявність радіатора (true/false)
rgb	Boolean	Підсвічування RGB (true/false)
price	Number	Ціна (USD)
image	String	URL зображення

Таблиця 2.24

Опис колекції «SSD»

Поле	Тип	Призначення
name	String	Назва SSD
manufacturer	String	Виробник
model	String	Модель
formFactor	String	Форм-фактор (2.5", M.2 2280, M.2 2260, M.2 2242, PCIe Card)
interface	String	Інтерфейс (SATA III, PCIe 3.0 x4, PCIe 4.0 x4, PCIe 5.0 x4, NVMe)
capacity	Number	Об'єм SSD (ГБ)
readSpeed	Number	Швидкість читання (МБ/с)
writeSpeed	Number	Швидкість запису (МБ/с)

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

Поле	Тип	Призначення
iops.read	Number	IOPS читання
iops.write	Number	IOPS запису
tbw	Number	TBW (Терабайт на запис)
cache	Number	Кеш (МБ)
price	Number	Ціна (USD)
image	String	URL зображення

2.3. Розробка функціональних алгоритмів роботи вебзастосунку

Функціональність вебзастосунку побудована навколо логіки створення, редагування та збереження користувацьких конфігурацій ПК. Кожна дія користувача в інтерфейсі пов'язана з певним алгоритмом на сервері, який забезпечує обробку запитів, перевірку даних, зберігання інформації та взаємодію з базою даних MongoDB.

Перший базовий алгоритм — реєстрація користувача. Після введення ім'я користувача, електронної пошти та пароля дані проходять валідацію, пароль хешується за допомогою бібліотеки bcrypt, і новий обліковий запис записується до бази. Якщо пошта вже використовується, система повертає відповідну помилку. Після успішної реєстрації користувач може авторизуватися.

Авторизація реалізована за класичною схемою: після перевірки імені користувача й пароля генерується JWT-токен, який використовується для автентифікації всіх наступних запитів. Таким чином, захищено як дані, так і дії користувача, зокрема — створення чи редагування конфігурацій.

Основна функція вебзастосунку — створення конфігурації ПК. Користувач поступово обирає компоненти: процесор, материнську плату, відеокарту, оперативну пам'ять, накопичувач, блок живлення, корпус тощо і автоматично перевіряється сумісність для запобігання некоректної конфігурації. Обрана комбінація зберігається у вигляді одного об'єкта в масиві конфігурацій у профілі користувача. Кожен об'єкт містить ID компонентів, які згодом можуть

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

використовуватися для детального перегляду або оновлення.

При створенні або редагуванні конфігурації сервер перевіряє, чи всі обрані елементи існують у базі, і чи не було передано порожніх або некоректних даних. Також реалізовано функціонал перегляду збережених конфігурацій. Користувач отримує список своїх збірок, кожна з яких може бути відкрита для перегляду або редагування. У разі редагування конфігурації користувач може замінити окремі компоненти, зберігши нову версію під іншим ім'ям або перезаписавши стару.

Алгоритми на сервері побудовані з урахуванням REST-архітектури. Кожна дія (створення, читання, оновлення, видалення) відповідає окремому HTTP-методу: POST, GET, PUT, DELETE. Це забезпечує логічну й передбачувану структуру API, з якою зручно працювати як розробникам клієнта, так і стороннім сервісам у майбутньому.

Особливістю реалізації конфігуратора є автоматична перевірка сумісності компонентів у режимі реального часу. Після вибору користувачем певної деталі (наприклад, процесора), інтерфейс динамічно фільтрує всі інші категорії так, щоб показувати лише ті комплектуючі, які технічно сумісні з попереднім вибором. Це значно полегшує процес підбору конфігурації, адже користувачу не потрібно вручну перевіряти характеристики або виправляти помилки сумісності — система автоматично запобігає їх виникненню. Такий підхід робить застосунок зручним як для досвідчених користувачів, так і для новачків.

Додатково передбачено розмежування прав доступу, що є важливим аспектом безпеки та персоналізації користувацьких даних. Кожна конфігурація прив'язана до конкретного облікового запису, тому лише автентифікований власник має змогу редагувати або видаляти свої збірки. Водночас інші користувачі можуть переглядати публічні конфігурації, проте без можливості вносити будь-які зміни або видаляти їх. Такий підхід забезпечує захист авторського вмісту і збереження цілісності даних, а також гарантує, що особисті налаштування користувачів залишаються недоторканими.

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

2.4. Реалізація функціоналу вебзастосунку

Розглянемо основні етапи створення збірки на вебзастосунку. Спочатку користувач виконує вхід у систему. Для цього реалізована функція авторизації, яка надсилає запит з введеними даними і перевіряє їх на сервері:

```
export const loginUser = async (
  userData: { username: string; password: string }
): Promise<AuthResponse> => {
  try {
    const response = await $api.post<AuthResponse>('/auth/login', userData);
    return response.data;
  } catch (error: any) {
    throw error.response?.data || error;
  }
};
```

Однак перед авторизацією користувач має зареєструватися в системі. У процесі реєстрації обов'язково відбувається хешування пароля для захисту даних. Це реалізовано за допомогою бібліотеки `bcryptjs`, яка дозволяє безпечно зберігати пароль у зашифрованому вигляді:

```
UserSchema.pre('save', async function(next) {
  if (!this.isModified('password')) {
    next();
  }
  const salt = await bcrypt.genSalt(10);
  this.password = await bcrypt.hash(this.password, salt);
});

UserSchema.methods.matchPassword = async function(enteredPassword) {
  return await bcrypt.compare(enteredPassword, this.password);
};
```

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

Після успішного входу користувач переходить на головну сторінку конфігуратора збірки ПК, де відображаються поля для вибору компонентів. Для кожного типу комплектуючих надсилається окремий запит на сервер, щоб отримати список доступних компонентів:

```
const getComponentsByType = async (type: string) => {  
  try {  
    const response = await axios.get(`/components/${type}`);  
    return response.data;  
  } catch (error) {  
    console.error('Error fetching components:', error);  
    throw error;  
  }  
};
```

На сервері ці запити обробляються відповідними маршрутами API, які вибирають дані з бази і повертають клієнту:

```
exports.getComponents = async (req, res) => {  
  try {  
    const { type } = req.params;  
    const Model = getModel(type);  
    if (!Model) {  
      return res.status(400).json({ success: false, error: 'Недійсний тип компонента' });  
    }  
    const components = await Model.find();  
    res.status(200).json({  
      success: true,  
      count: components.length,  
      data: components  
    });  
  }  
};
```

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

```

    } catch (error) {
      res.status(500).json({ success: false, error: error.message });
    }
  };

```

Під час вибору компонентів користувачем виконується перевірка їх сумісності за допомогою запиту, що порівнює параметри:

```

const checkCompatibility = async (selectedComponents: any) => {
  try {
    const response = await axios.post('/compatibility', selectedComponents);
    return response.data;
  } catch (error) {
    console.error('Error checking compatibility:', error);
    throw error;
  }
};

```

На сервері цей процес виглядає так:

```

exports.getCompatibleComponents = async (req, res) => {
  try {
    const { type, selected } = req.body;
    const Model = getModel(type);
    if (!Model) return res.status(400).json({ error: 'Unknown component type' });
    const all = await Model.find();
    const compatible = all.filter((component) =>
      isCompatible(selected, component, type)
    );
    res.json(compatible);
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'Server error' });
  }
};

```

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

Після вибору кожного компонента автоматично виконується оновлення загальної вартості збірки. Система підсумовує ціни всіх обраних елементів у режимі реального часу, що дозволяє користувачу одразу бачити актуальну загальну вартість майбутньої конфігурації. Такий підхід забезпечує зручність і прозорість при плануванні бюджету під час складання ПК:

```
const getTotalPrice = () => {
  const compPrices = Object.values(selectedComponents.components)
    .flatMap(c => Array.isArray(c) ? c : c ? [c] : []);
  const periphPrices = Object.values(selectedComponents.peripherals)
    .filter((c): c is ComponentItem => c !== null);
  const all = [...compPrices, ...periphPrices];
  if (all.length === 0) return 0;
  return all.reduce((sum, item) => {
    return sum + (item?.price || 0);
  }, 0);
};
```

Коли користувач сформував збірку, він натискає кнопку «Зберегти», яка викликає запит відправки об'єкта збірки на сервер:

```
const createBuild = async (buildData: any) => {
  try {

    const response = await axios.post('/builds', buildData);
    return response.data;
  } catch (error: any) {

    console.error('Помилка при створенні збірки:', error.response?.data ||
error.message);
    throw error;
  }
};
```

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

Сервер приймає дані і зберігає їх у базі:

```
exports.createBuild = async (req, res) => {  
  try {  
    req.body.user = req.user.id;  
    const build = await Build.create(req.body);  
    res.status(201).json({  
      success: true,  
      data: build  
    });  
  } catch (err) {  
    res.status(500).json({  
      success: false,  
      error: err.message || 'Серверна помилка'  
    });  
  }  
};
```

У особистому кабінеті користувач може переглядати свої збірки, а також оновлювати їх за допомогою кнопки «Оновити»:

```
<button onClick={handleEdit} className="flex items-center gap-1 text-sm text-blue-600 hover:underline">  
<Edit className="w-4 h-4" /> Оновити  
</button>
```

Запит оновлення виконує PUT-запит на сервер:

```
const updateBuild = async (id: string, updatedData: any) => {  
  try {  
    const response = await axios.put(`/builds/${id}`, updatedData);  
    return response.data;  
  } catch (error: any) {
```

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

```

        console.error('Помилка при оновленні збірки:', error.response?.data ||
error.message);
        throw error;
    }
};

```

Для видалення збірки використовується DELETE-запит:

```

const deleteBuild = async (id: string) => {
    try {
        const response = await axios.delete(`/builds/${id}`);
        return response.data;
    } catch (error: any) {
        console.error('Помилка при видаленні збірки:', error.response?.data ||
error.message);
        throw error;
    }
};

```

Цей запит викликається у функції обробника при натисканні на кнопку «Видалити». Спочатку користувачу відображається підтвердження дії, після чого, у разі згоди, збірка видаляється, а інтерфейс оновлюється. У разі помилки виводиться відповідне повідомлення:

```

const handleDelete = async () => {
    if (window.confirm('Ви дійсно хочете видалити цю збірку?')) {
        try {
            await deleteBuild(build._id);
            if (onDelete) onDelete(build._id);
        } catch (error) {
            window.confirm('Помилка при видаленні збірки');
            console.error(error);
        }
    }
}

```

```

    }
};

```

Крім основних CRUD-операцій, реалізовано можливість керування статусом публічності збірки. Користувач може зробити свою збірку загальнодоступною або прихованою від інших користувачів. Для цього реалізовано відповідний перемикач, який надсилає оновлені дані на сервер:

```

const togglePublished = async () => {
  try {
    const updatedBuildData = formatBuildForUpdate(build, !isPublic);
    const updatedBuild = await isPublicBuild(build._id, updatedBuildData);
    setIsPublic(updatedBuild.data.isPublic);
  } catch (error) {
    window.confirm('Помилка при оновленні статусу публікації');
    console.error(error);
  }
};

```

Для зміни статусу збірок використовується окремий GET-запит:

```

const isPublicBuild = async (id: string, updatedData: any) => {
  try {
    const response = await axios.put(`/builds/${id}/publicity`, updatedData);
    return response.data;
  } catch (error: any) {
    console.error('Помилка при оновленні збірки:', error.response?.data ||
error.message);
    throw error;
  }
};

```

А для отримання списку публічних збірок використовується окремий GET-запит:

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

```
const getPublicBuilds = async () => {
  try {
    const response = await axios.get('/builds/public');
    return response.data;
  } catch (error: any) {
    console.error('Помилка при отриманні публічних збірок:', error.response?.data ||
error.message);
    throw error;
  }
};
```

Вище наведене дозволяє виводити всі загальнодоступні конфігурації, створені іншими користувачами, на окремій сторінці застосунку. Це надає можливість переглядати готові збірки, орієнтуватися на них під час створення власної, а також ділитися своєю збіркою з іншими.

Таким чином, кожен етап реалізований через окремі функції та маршрути, що забезпечує зрозумілу, масштабовану та гнучку архітектуру застосунку. Такий підхід полегшує подальший розвиток і підтримку проєкту, дозволяє легко додавати нові функціональні можливості та забезпечує надійну взаємодію між клієнтською та серверною частинами. Реалізація запитів і функцій застосунку наведена в Додатку А.

Висновки до другого розділу

У другому розділі детально спроектовано калькулятор для розрахунку конфігурації ПК. Було визначено архітектуру, створено базу даних, розроблено алгоритми роботи та реалізовано основний функціонал системи.

Обрано клієнт-серверну модель, з використанням React для інтерфейсу користувача та Node.js з Express для серверної частини. MongoDB стала основною базою даних, оскільки вона найкраще підходить для зберігання даних про компоненти ПК та користувацькі збірки.

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

Під час проєктування було приділено увагу варіантам використання вебзастосунку, щоб стало зрозумілим, як ним будуть користуватися різні типи користувачів. Також особливий акцент зроблено на безпеці: впроваджено JWT для автентифікації та bcrypt для хешування паролів.

Готова система показала, що обрані технології працюють ефективно. REST API забезпечило зручну взаємодію між клієнтом і сервером, а модульна структура коду спростила тестування й подальше вдосконалення. У результаті отримано зручний інструмент, який дозволяє користувачам швидко створювати й підбирати комп'ютерні збірки з урахуванням сумісності компонентів та бюджету.

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З КАЛЬКУЛЯТОРОМ ДЛЯ РОЗРАХУНКУ КОНФІГУРАЦІЇ ПК

3.1. Порядок встановлення та налаштування параметрів вебзастосунку

Встановлення й налаштування вебзастосунку потребує поетапного розгортання всіх його складових частин та забезпечення їхньої правильної взаємодії. Кожен компонент налаштовується окремо, після чого вони поєднуються в єдину систему, яка дозволяє стабільно й повноцінно працювати калькулятору для розрахунку конфігурацій ПК.

Для кращого розуміння архітектури проекту була створена діаграма розгортання, яка представлена на рис. 3.1.

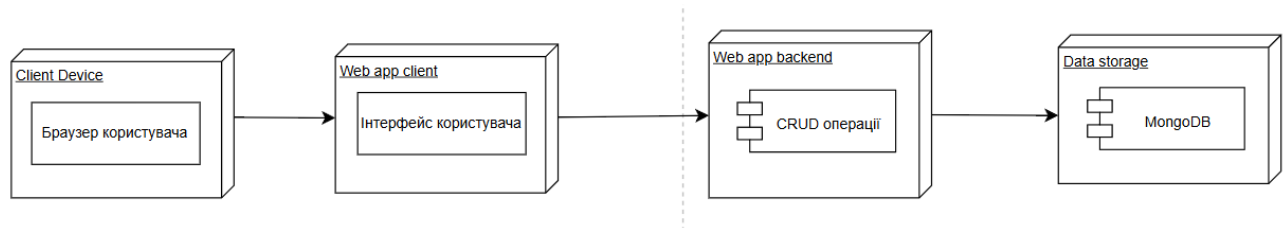


Рисунок 3.1 – Діаграма розгортання вебзастосунку конфігуратора ПК

Після створення основних модулів вебзастосунку виконується процес розгортання клієнтської та серверної частин. На клієнтській частині, реалізованій за допомогою бібліотеки React, відбувається візуалізація інтерфейсу, обробка дій користувача та формування запитів до сервера. Користувач взаємодіє з інтерфейсом через браузер, отримуючи доступ до сторінки конфігурації ПК, перегляду профілю, публічних збірок тощо.

На стороні клієнта використовуються HTTP-запити для передавання даних на сервер. Ці запити надсилаються до REST API, яке працює на сервері Node.js. Серверна частина відповідає за обробку логіки вебзастосунку: отримання переліку компонентів, створення нових збірок, їх оновлення, видалення тощо. Усі ці дії реалізовані у вигляді окремих маршрутів, які обробляють відповідні контролери.

Для зберігання даних використовується база даних MongoDB. Вона відповідає за збереження користувацьких акаунтів, параметрів збірок, списків компонентів, збірок користувачів і статусу публікації. Взаємодія з базою відбувається за допомогою стандартних CRUD-операцій.

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

Таким чином, під час розгортання вебзастосунку необхідно запустити фронтенд-додаток, серверну частину з налаштованими маршрутами та забезпечити підключення до бази даних MongoDB.

3.2 Структура інтерфейсу

Взаємодія користувача з вебзастосунком відбувається через вебінтерфейс, створений за допомогою React. Інтерфейс зручний і дозволяє виконувати всі необхідні дії для створення, редагування, перегляду та завантаження збірок персональних комп'ютерів.

Після проходження авторизації (рис. 3.2) або реєстрації (рис. 3.3) користувач потрапляє на сторінку профілю користувача. Після цього користувач може перейти на сторінку ПК Конфігуратора — тут починається процес формування нової збірки (рис. 3.4). При завантаженні цієї сторінки автоматично надсилаються запити до сервера для отримання доступних компонентів за категоріями — процесори, материнські плати, відеокарти, накопичувачі, корпуси тощо (рис. 3.5). Отримані дані відображаються у відповідних полях для вибору.

The screenshot shows a web application interface with a navigation bar at the top containing 'ПК Конфігуратор' and 'Конфігурації користувачів'. On the right side of the bar is a link 'Авторизація'. The main content area features a central white box titled 'Авторизація'. Inside this box are two input fields: 'Ім'я користувача' (with a magnifying glass icon) and 'Пароль' (with a key icon). Below these fields is a blue button labeled 'Увійти'. At the bottom of the box, there is a link 'Немає акаунту? Зареєструватися'.

Рисунок 3.2 – Сторінка авторизації

The screenshot shows a web application interface similar to the previous one. The navigation bar has 'ПК Конфігуратор' and 'Конфігурації користувачів' on the left, and a link 'Авторизація' on the right. The main content area features a central white box titled 'Реєстрація'. Inside this box are three input fields: 'Ім'я' (with a magnifying glass icon), 'Email' (with an envelope icon), and 'Пароль' (with a key icon). Below these fields is a blue button labeled 'Зареєструватися'. At the bottom of the box, there is a link 'Назад до авторизації'.

Рисунок 3.3 – Сторінка реєстрації

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

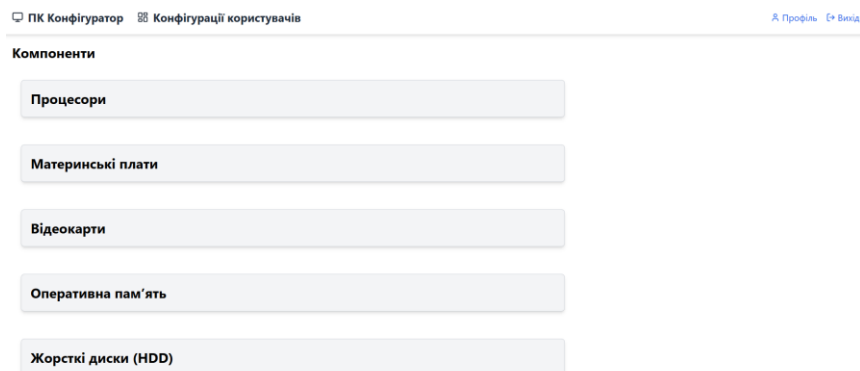


Рисунок 3.4 – Основна сторінка конфігуратора ПК

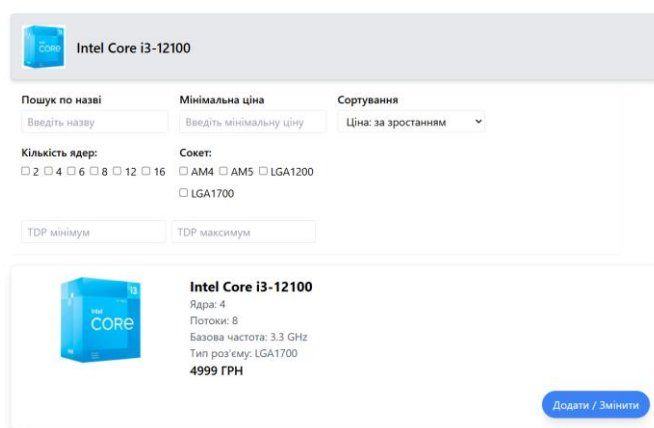


Рисунок 3.5 – Отримання компонентів з сервера

При виборі певного компонента система автоматично перевіряє його сумісність з уже обраними деталями, у разі невідповідності видаляє несумісні варіанти з подальших списків компонентів. Таким чином, користувач може обирати лише ті елементи, які гарантовано сумісні між собою.

Після того як усі необхідні комплектуючі обрано, користувач може зберегти збірку. При натисканні кнопки «Зберегти збірку», відбувається передача інформації на сервер, де створюється новий запис у базі даних (рис. 3.6).

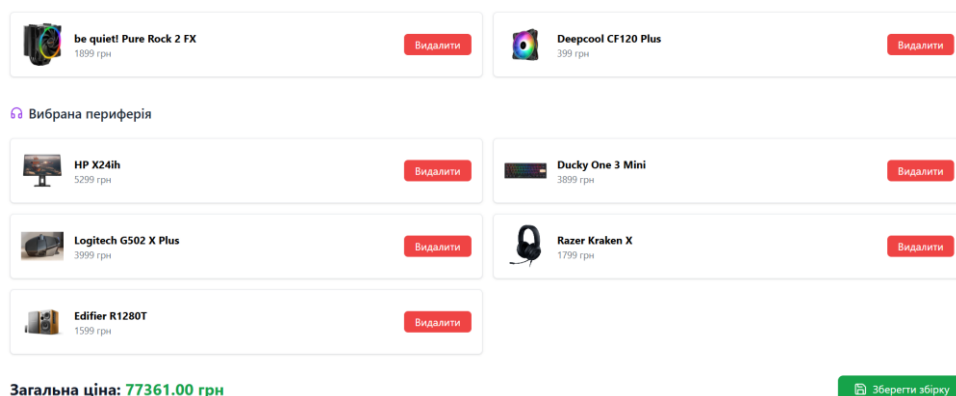


Рисунок 3.6 – Збереження збірки користувача

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

Після збереження, користувач може переглянути свої збірки у своєму профілі (рис. 3.7). У профілі відображаються всі створені ним збірки з можливістю їх оновлення, видалення або завантаження у форматі PDF.

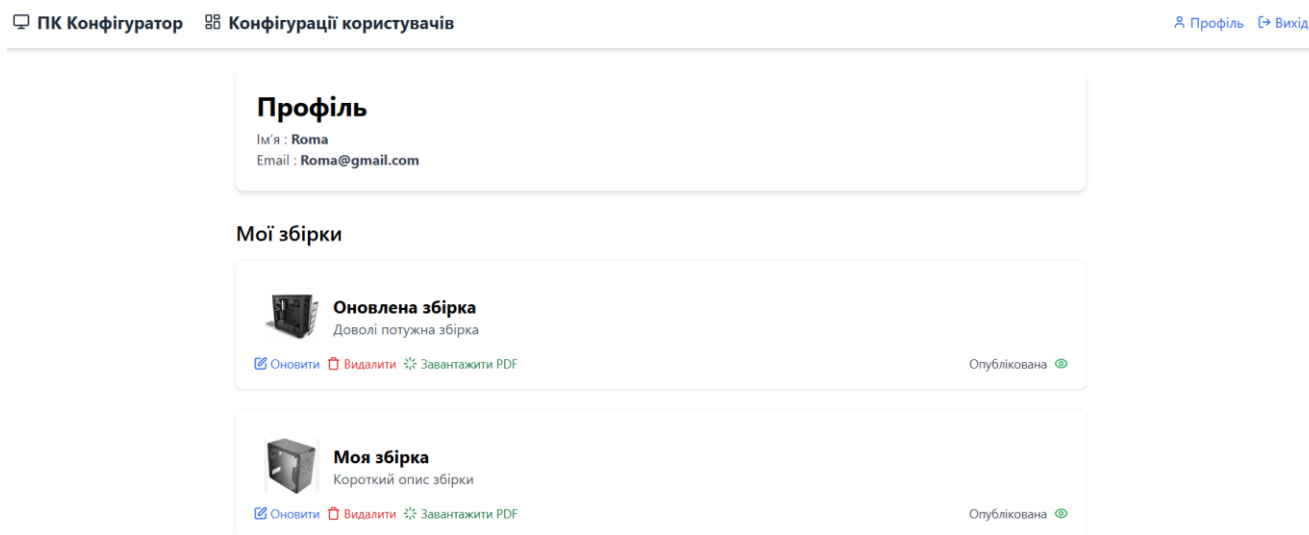


Рисунок 3.7 – Особистий кабінет користувача

Кнопка «Оновити» надає можливість змінити вже збережену конфігурацію. Після її натискання, користувача перенаправляє на сторінку конфігуратора, де попередні дані вже завантажено, а кнопка «Зберегти збірку» змінюється на «Оновити збірку» (рис. 3.8).

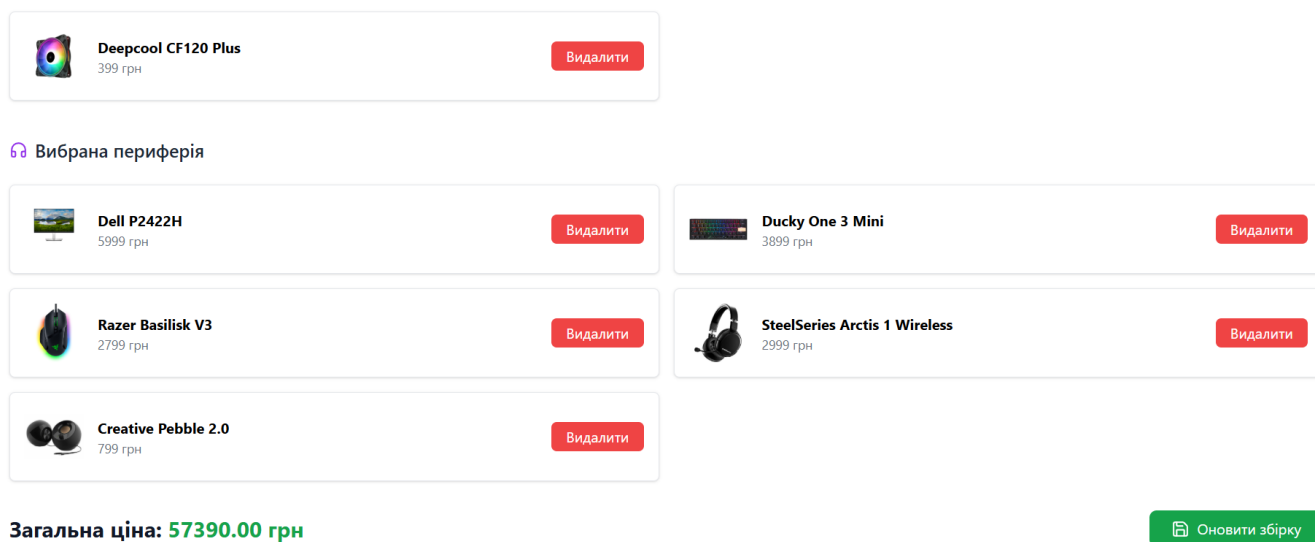


Рисунок 3.8 – Форма оновлення збірки

Для видалення збірки передбачена відповідна кнопка на картці збірки в профілі. Після підтвердження, виконується запит на видалення запису з бази даних (рис. 3.9).

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

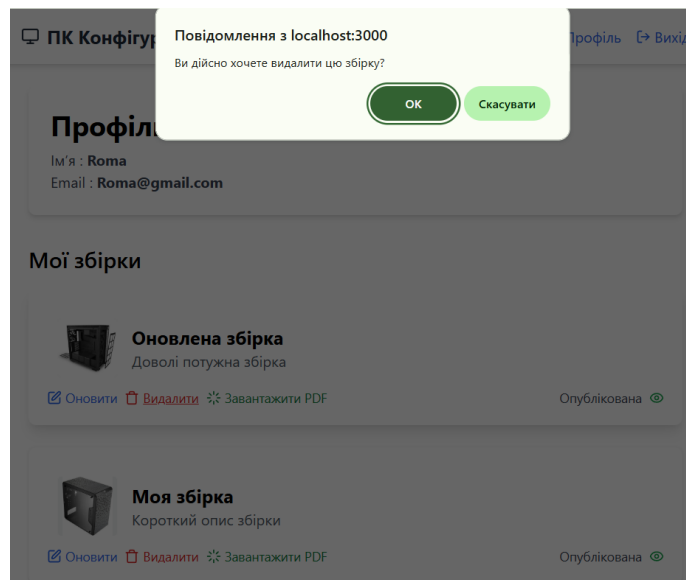


Рисунок 3.9 – Видалення збірки

Також користувач має змогу завантажити PDF-файл із описом конфігурації. Для цього реалізовано функціонал експорту, який використовує бібліотеки jsrpdf та jsrpdf-autotable. У файлі відображаються всі обрані компоненти, їх вартість та загальна сума (рис. 3.10).

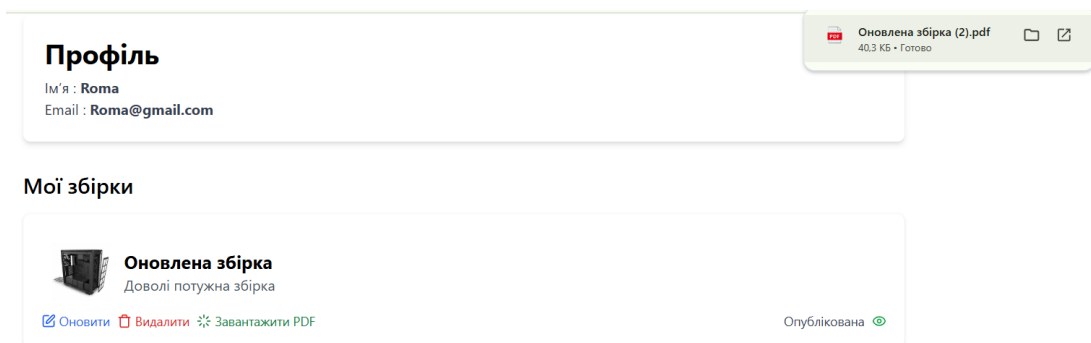


Рисунок 3.10 – Завантаження PDF-файлу з конфігурацією

Окрім перегляду власних збірок, доступна також окрема сторінка з публічними конфігураціями. Тут можна переглянути збірки, створені іншими користувачами (рис. 3.11).

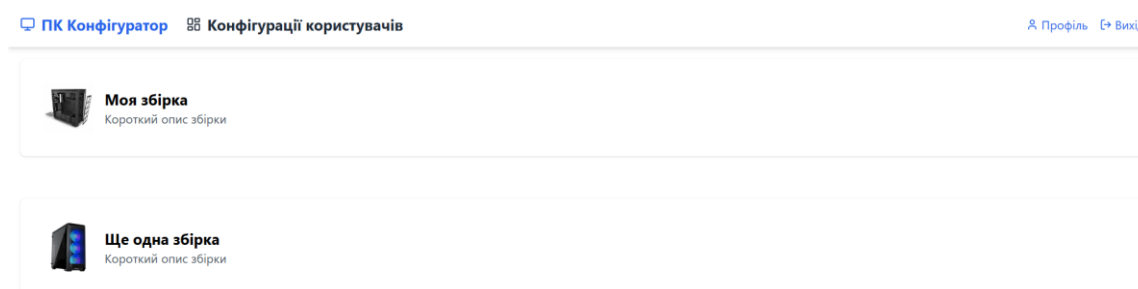


Рисунок 3.11 – Сторінка публічних збірок

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

Таким чином, структура інтерфейсу вебзастосунку забезпечує зручну та зрозумілу взаємодію користувача із системою: від авторизації, створення та налаштування збірки — до її редагування, видалення або завантаження у вигляді PDF-документа.

3.3 Тестування вебзастосунку

Для перевірки працездатності всіх функцій вебзастосунку було проведено тестування основних сценаріїв взаємодії користувача з сайтом. Метою тестування було виявлення можливих помилок в авторизації, створенні, збереженні, оновленні та видаленні збірок, а також перевірка коректності роботи механізмів перевірки введених даних.

Під час тестування авторизації було перевірено поведінку системи при введенні неправильного пароля або неіснуючого облікового запису. У разі некоректних даних авторизації система виводить відповідне повідомлення про помилку (рис. 3.12).

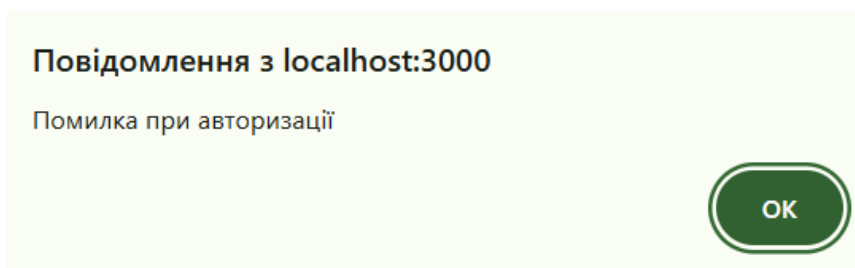


Рисунок 3.12 – Повідомлення про помилку при авторизації

Було також перевірено, що при успішному введенні логіну та пароля користувача, його перенаправляє на головну сторінку профілю (рис. 3.13).



Рисунок 3.13 – Успішна авторизація

Було протестовано поведінку системи при спробі зберегти конфігурацію без вибраних компонентів. У такому випадку зберігання не відбувається, а система виводить повідомлення про необхідність заповнення полів (рис. 3.14).

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

Повідомлення з localhost:3000

Неможливо зберегти порожню збірку. Будь ласка, додайте хоча б один компонент.

ОК

Скасувати

Рисунок 3.14 – Повідомлення про порожню конфігурацію

Також перевірено сценарій, коли неавторизований користувач намагається зберегти збірку. У цьому випадку система блокує дію і виводить повідомлення з проханням зареєструватись або увійти до облікового запису (рис. 3.15), щоб мати змогу зберігати власні конфігурації.

Повідомлення з localhost:3000

Ви ще не зареєстровані. Будь ласка, увійдіть в акаунт, щоб зберегти збірку.

ОК

Скасувати

Рисунок 3.15 – Повідомлення про необхідність авторизації для збереження збірки

Система була протестована на коректність логіки перевірки сумісності. При виборі компонентів, які не відповідають вже обраним (наприклад, процесор і материнська плата з різними сокетами), відповідні опції автоматично виключаються зі списків вибору. Це дозволяє уникнути конфліктів при складанні (рис. 3.16).

Процесори

Пошук по назві

Введіть назву

Мінімальна ціна

Введіть мінімальну ціну

Сортування

Ціна: за зростанням

Кількість ядер:

☐ 2 ☐ 4 ☐ 6 ☐ 8 ☐ 12 ☐ 16

Сокет:

☒ AM4 ☒ AM5 ☐ LGA1200

☒ LGA1700

TDP мінімум

TDP максимум

Компоненти не знайдено за заданими фільтрами.



ASRock B460M-HDV

Рисунок 3.16 – Автоматична фільтрація несумісних компонентів

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

Було протестовано оновлення раніше створеної збірки. Після натискання кнопки «Оновити збірку» у профілі, форма редагування заповнюється даними поточної конфігурації. Після внесення змін і повторного збереження оновлена версія збірки коректно заміщує попередню (рис. 3.17).

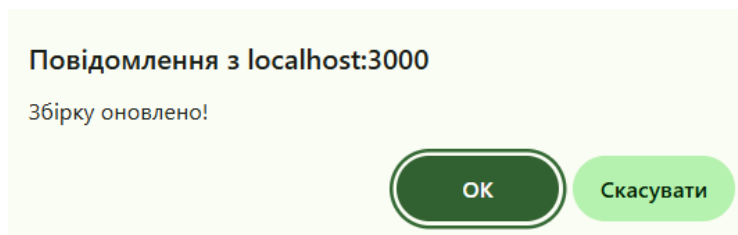


Рисунок 3.17 – Оновлення існуючої збірки

Тестування видалення показало, що після натискання кнопки «Видалити» та підтвердження дії, обрана збірка успішно зникає зі списку у профілі користувача (рис. 3.18).

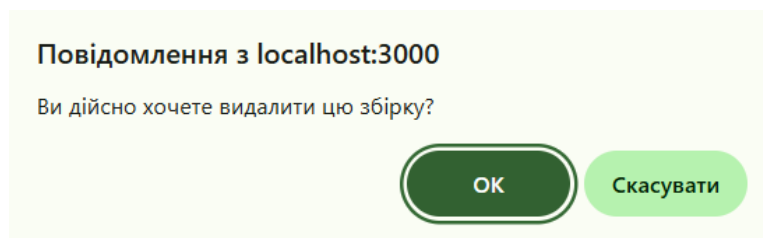


Рисунок 3.18 – Успішне видалення збірки

Тестування сторінки з публічними збірками підтвердило, що користувач може переглядати інші конфігурації, не маючи прав на їх редагування чи видалення (рис. 3.19).

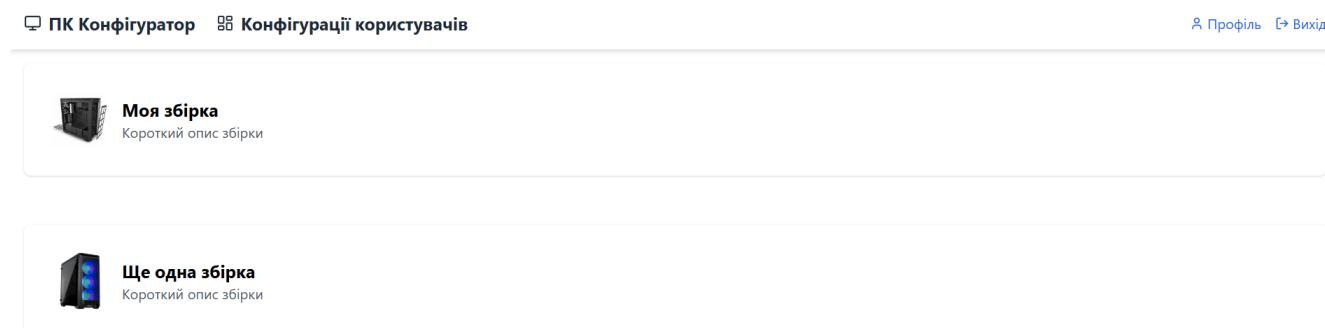


Рисунок 3.19 – Перегляд публічних збірок

Перевірено коректність формування PDF-файлу. Після натискання відповідної кнопки генерується документ, який містить таблицю з переліком усіх

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

вибраних компонентів та загальною вартістю (рис. 3.20). Для реалізації цієї функції використовувалися бібліотеки jspdf та jspdf-autotable.

Оновлена збірка

Доволі потужна збірка

Тип	Назва	Ціна
Материнська плата	ASUS B85M-G	1799 грн
Відеокарта	ASUS Phoenix GeForce GTX 1050 2GB	5200 грн
Оперативна пам'ять	TeamGroup Elite	650 грн
HDD	Seagate Barracuda 500GB	950 грн
SSD	GOODRAM CL100 Gen.2 120GB	470 грн
Блок живлення	Thermaltake Smart RGB 500W	1300 грн
Корпус	NZXT H710	799 грн
Охолодження CPU	Deepcool GAMMAXX 400 V2	899 грн
Вентилятори корпусу	Deepcool FK120	369 грн
Монітор	LG 24MP400-B	3799 грн
Навушники	Logitech H390	799 грн
Колонки	Logitech Z625	169.99 грн
Мишка	Logitech M90	250 грн

Загальна вартість: 17453.99 грн

Рисунок 3.20 – Генерація PDF-файлу з описом збірки

Таким чином, тестування вебзастосунку показало його стабільну роботу в основних сценаріях використання. Усі помилки обробляються належним чином, перевірка даних виконується надійно, а інтерфейс не дозволяє здійснити некоректні або неповні дії зі сторони користувача.

Висновки до третього розділу

У третьому розділі було детально описано процес встановлення та налаштування вебзастосунку конфігуратора ПК, а також представлено архітектуру та діаграму розгортання, яка наочно демонструє взаємозв'язок основних частин системи: клієнтської частини, сервера та бази даних.

Особливу увагу приділено інтерфейсу користувача, який відіграє важливу роль у зручності взаємодії з вебзастосунком. Його реалізовано так, щоб навіть новий користувач міг інтуїтивно зрозуміти, як працювати з вебзастосунком — від перегляду доступних публічних збірок до створення власної конфігурації з автоматичною перевіркою сумісності компонентів. Було послідовно представлено

сторінки вебзастосунку, що дозволяє показати логіку користувацької взаємодії: від реєстрації до збереження або оновлення власної збірки.

Також було проведено тестування ключових функцій, що дозволило з'ясувати, як система працює у різних ситуаціях: введення неправильного пароля, створення пустої збірки, спроба збереження без авторизації тощо. Результати тестування показали, що застосунок правильно реагує на помилки та забезпечує зворотний зв'язок з користувачем через повідомлення, що підвищує її надійність і стабільність.

Таким чином, у цьому розділі було підтверджено працездатність архітектури та інтерфейсу вебзастосунку, що свідчить про готовність продукту до подальшого використання та розвитку.

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

ВИСНОВКИ

У межах бакалаврської кваліфікаційної роботи було створено вебзастосунок для формування, перегляду та керування конфігураціями персональних комп'ютерів. Робота охоплює повний цикл розробки програмного продукту — від аналізу предметної області до тестування готового функціоналу

На початковому етапі було проведено аналіз існуючих аналогів та визначено основні потреби потенційних користувачів. Це дозволило сформулювати чіткі вимоги до системи й обґрунтувати вибір технологій, серед яких React для клієнтської частини, Node.js для серверної, MongoDB як база даних, а також Tailwind CSS для стилізації інтерфейсу. Такий набір інструментів забезпечив швидкість розробки, гнучкість у налаштуванні та зручність у користуванні.

Проектування архітектури системи, бази даних і ключових функцій дозволило реалізувати всі заплановані можливості: створення та збереження збірок ПК, автоматичну перевірку сумісності компонентів, генерацію PDF-файлів з конфігурацією, редагування та видалення зібраних конфігурацій, а також перевірку доступу на основі авторизації. Інтерфейс є інтуїтивно зрозумілим і адаптованим для потреб користувача.

Останній етап розробки був присвячений тестуванню. Було перевірено типові сценарії взаємодії: авторизація з помилками, спроба зберегти порожню збірку, недоступність функцій без реєстрації, а також загальна стабільність вебзастосунку. Всі перевірки показали, що система працює коректно та надійно обробляє помилки, надаючи користувачеві зрозумілий зворотний зв'язок.

У підсумку, розроблений застосунок повністю відповідає поставленим завданням бакалаврської роботи. Він є функціональним, зручним у використанні та готовим до подальшого вдосконалення й розширення.

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Зіневич Р.Р., Локтікова Т.М. Розробка калькулятора для розрахунків конфігурації ПК. Тези доповідей Всеукраїнської науково-практичної on-line конференції здобувачів вищої освіти і молодих учених, присвяченої Дню науки, 12-17 травня 2025 року. Житомир: «Житомирська політехніка», 2025. С. 60.
2. Axios – HTTP клієнт для браузера і Node.js [Електронний ресурс]. – Режим доступу: <https://axios-http.com/>
3. bcryptjs – бібліотека хешування паролів у JavaScript [Електронний ресурс]. – Режим доступу: <https://github.com/dcodeIO/bcrypt.js>
4. BuildMyPC – платформа для підбору комплектуючих [Електронний ресурс]. – Режим доступу: <https://buildmypc.net/>
5. CSS – Каскадні таблиці стилів [Електронний ресурс]. – Режим доступу: <https://developer.mozilla.org/en-US/docs/Web/CSS>
6. DXDigital – таблиця сумісності процесор/відеокарти [Електронний ресурс]. – Режим доступу: <https://www.dxdigitals.top/2024/03/pidbir-videokarti-pid-procesor.html>
7. E-Katalog – як вибрати процесор до відеокарти і навпаки [Електронний ресурс]. – Режим доступу: <https://ek.ua/ua/post/6417/189-how-to-choose-a-cpu-for-a-graphics-card-and-vice-versa/>
8. Express – вебфреймворк для Node.js [Електронний ресурс]. – Режим доступу: <https://expressjs.com/>
9. Git – система контролю версій [Електронний ресурс]. – Режим доступу: <https://git-scm.com/doc>
10. Intel ARK – база даних сумісності процесорів [Електронний ресурс]. – Режим доступу: <https://ark.intel.com/>
11. jsPDF – генератор PDF у JavaScript [Електронний ресурс]. – Режим доступу: <https://github.com/parallax/jsPDF>

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				64
Змн.	Арк.	№ докум.	Підпис	Дата		

- 12.jspdf-autotable – плагін для створення таблиць у jsPDF [Електронний ресурс].
– Режим доступу: <https://github.com/simonbengtsson/jsPDF-AutoTable>
- 13.Lucide – сучасна бібліотека SVG-іконок для React [Електронний ресурс]. –
Режим доступу: <https://lucide.dev/>
- 14.MongoDB – документоорієнтована база даних [Електронний ресурс]. –
Режим доступу: <https://www.mongodb.com/>
- 15.Mongoose – ORM для MongoDB у Node.js [Електронний ресурс]. – Режим
доступу: <https://mongoosejs.com/>
- 16.NPM – менеджер пакетів для JavaScript [Електронний ресурс]. – Режим
доступу: <https://www.npmjs.com/>
- 17.Node.js – середовище виконання JavaScript-коду [Електронний ресурс]. –
Режим доступу: <https://nodejs.org/>
- 18.PC Builder – сервіс складання ПК онлайн [Електронний ресурс]. – Режим
доступу: <https://www.pcbuilder.net/>
- 19.PCPartPicker – онлайн-конфігуратор ПК [Електронний ресурс]. – Режим
доступу: <https://pcpartpicker.com/>
- 20.PC.com.ua – поради щодо вибору комплектуючих для ПК [Електронний
ресурс]. – Режим доступу: <https://pc.com.ua/ua/articles/kak-podobrat-komplektujushhie-dlja-pk-vazhnye-njuansy>
- 21.Postman – інструмент для тестування API [Електронний ресурс]. – Режим
доступу: <https://www.postman.com/>
- 22.React – JavaScript бібліотека для створення інтерфейсів [Електронний
ресурс]. – Режим доступу: <https://reactjs.org/>
- 23.React DOM – API для взаємодії з DOM у React [Електронний ресурс]. – Режим
доступу: <https://reactjs.org/docs/react-dom.html>
- 24.React Router DOM – маршрутизація у React-застосунків [Електронний
ресурс]. – Режим доступу: <https://reactrouter.com/>
- 25.Tailwind CSS – утилітний CSS-фреймворк для швидкого стилювання
[Електронний ресурс]. – Режим доступу: <https://tailwindcss.com/>

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

- 26.TypeScript – типізоване доповнення до JavaScript [Електронний ресурс]. – Режим доступу: <https://www.typescriptlang.org/>
- 27.TSX – синтаксис для компонентів у React з TypeScript [Електронний ресурс]. – Режим доступу: <https://react-typescript-cheatsheet.netlify.app/>
- 28.JSON Web Token (JWT) – відкритий стандарт для безпечної передачі інформації [Електронний ресурс]. – Режим доступу: <https://jwt.io/>
- 29.UserBenchmark – сайт для тестування та порівняння компонентів ПК [Електронний ресурс]. – Режим доступу: <https://www.userbenchmark.com/>
- 30.VolynPost – як правильно підібрати комплектуючі до комп'ютерів [Електронний ресурс]. – Режим доступу: <https://www.volynpost.com/news/180774-iak-pravylno-pidibraty-komplektuyuchi-do-kompyuteriv>

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова, Т. М.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Зіневич Р.Р.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова. Т. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

```
const User = require('../models/User');
const jwt = require('jsonwebtoken');

const generateToken = (id) => {
  return jwt.sign({ id }, process.env.JWT_SECRET, {
    expiresIn: '30d'
  });
};

// Регістрація
exports.register = async (req, res) => {
  try {
    const { username, email, password } = req.body;

    const existingUser = await User.findOne({ $or: [{ email }, { username }] });
    if (existingUser) {
      return res.status(400).json({
        success: false,
        message: 'Користувач з таким email або логіном вже існує'
      });
    }

    const user = await User.create({
      username,
      email,
      password
    });

    res.status(201).json({
      success: true,
      token: generateToken(user._id),
      user: {
        id: user._id,
        username: user.username,
        email: user.email
      }
    });
  } catch (error) {
    res.status(500).json({
      success: false,
      message: 'Помилка сервера',
      error: error.message
    });
  }
};
```

// Авторизація

```
exports.login = async (req, res) => {
  try {
    const { username, password } = req.body;

    const user = await User.findOne({ username }).select('+password');
    if (!user) {
      return res.status(401).json({
        success: false,
        message: 'Невірний логін або пароль'
      });
    }

    const isMatch = await user.matchPassword(password);
    if (!isMatch) {
      return res.status(401).json({
        success: false,
        message: 'Невірний логін або пароль'
      });
    }

    res.status(200).json({
      success: true,
      token: generateToken(user._id),
      user: {
        id: user._id,
        username: user.username,
        email: user.email
      }
    });
  } catch (error) {
    res.status(500).json({
      success: false,
      message: 'Помилка сервера',
      error: error.message
    });
  }
};
```

// Отримання профілю

```
exports.getProfile = async (req, res) => {
  try {
    const user = await User.findById(req.user.id).populate('savedBuilds');

    res.status(200).json({
```

```

        success: true,
        user
    });
} catch (error) {
    res.status(500).json({
        success: false,
        message: 'Помилка сервера',
        error: error.message
    });
}
};

```

```

const getModel = (type) => {
    const models = {
        'cpu': require('../models/CPU'),
        'gpu': require('../models/GPU'),
        'motherboard': require('../models/Motherboard'),
        'ram': require('../models/RAM'),
        'storage-ssd': require('../models/SSD'),
        'storage-hdd': require('../models/HDD'),
        'case': require('../models/Case'),
        'psu': require('../models/PSU'),
        'cpu-cooling': require('../models/CPU Cooling'),
        'case-fan': require('../models/Case Fans'),
        'monitor': require('../models/Monitor'),
        'keyboard': require('../models/Keyboard'),
        'mouse': require('../models/Mouse'),
        'headphones': require('../models/Headphones'),
        'speakers': require('../models/Speakers'),
    };

    return models[type] || null;
};

// Отримати всі компоненти певного типу
exports.getComponents = async (req, res) => {
    try {
        const { type } = req.params;
        const Model = getModel(type);

        if (!Model) {
            return res.status(400).json({ success: false, error: 'Недійсний тип компонента'
        });
    }
}

```

```

const components = await Model.find();

res.status(200).json({
  success: true,
  count: components.length,
  data: components
});
} catch (error) {
  res.status(500).json({ success: false, error: error.message });
}
};

// Отримати конкретний компонент
exports.getComponent = async (req, res) => {
  try {
    const { type, id } = req.params;
    const Model = getModel(type);

    if (!Model) {
      return res.status(400).json({ success: false, error: 'Недійсний тип компонента'
});
    }

    const component = await Model.findById(id);

    if (!component) {
      return res.status(404).json({ success: false, error: 'Компонент не знайдено' });
    }

    res.status(200).json({ success: true, data: component });
  } catch (error) {
    res.status(500).json({ success: false, error: error.message });
  }
};

// Створити новий компонент
exports.createComponent = async (req, res) => {
  try {
    const { type } = req.params;
    const Model = getModel(type);

    if (!Model) {
      return res.status(400).json({ success: false, error: 'Недійсний тип компонента'
});
    }
  }

```

```

    const component = await Model.create(req.body);

    res.status(201).json({ success: true, data: component });
  } catch (error) {
    if (error.name === 'ValidationError') {
      const messages = Object.values(error.errors).map(val => val.message);
      return res.status(400).json({ success: false, error: messages });
    }
    res.status(500).json({ success: false, error: error.message });
  }
};

// Оновити компонент
exports.updateComponent = async (req, res) => {
  try {
    const { type, id } = req.params;
    const Model = getModel(type);

    if (!Model) {
      return res.status(400).json({ success: false, error: 'Недійсний тип компонента'
});
    }

    const component = await Model.findByIdAndUpdate(id, req.body, {
      new: true,
      runValidators: true
    });

    if (!component) {
      return res.status(404).json({ success: false, error: 'Компонент не знайдено' });
    }

    res.status(200).json({ success: true, data: component });
  } catch (error) {
    if (error.name === 'ValidationError') {
      const messages = Object.values(error.errors).map(val => val.message);
      return res.status(400).json({ success: false, error: messages });
    }
    res.status(500).json({ success: false, error: error.message });
  }
};

// Видалити компонент
exports.deleteComponent = async (req, res) => {
  try {

```

```

const { type, id } = req.params;
const Model = getModel(type);

if (!Model) {
  return res.status(400).json({ success: false, error: 'Недійсний тип компонента'
});
}

const component = await Model.findByIdAndDelete(id);

if (!component) {
  return res.status(404).json({ success: false, error: 'Компонент не знайдено' });
}

res.status(200).json({ success: true, data: {} });
} catch (error) {
  res.status(500).json({ success: false, error: error.message });
}
};
const Build = require('../models/Build');
require('../models/CPU');
require('../models/GPU');
require('../models/Motherboard');
require('../models/RAM');
require('../models/SSD');
require('../models/HDD');
require('../models/PSU');
require('../models/Case');
require('../models/CPU Cooling');
require('../models/Case Fans');
require('../models/Monitor');
require('../models/Keyboard');
require('../models/Mouse');
require('../models/Headphones');
require('../models/Speakers');

// Отримати всі збірки поточного користувача
exports.getMyBuilds = async (req, res) => {
  try {
    const builds = await Build.find({ user: req.user.id })
      .populate('components.cpu')
      .populate('components.motherboard')
      .populate('components.gpu')
      .populate('components.ram')
      .populate('components.ssd')
      .populate('components.hdd')

```

```

        .populate('components.psu')
        .populate('components.case')
        .populate('components.cpuCooling')
        .populate('components.caseFans');

    res.status(200).json({
        success: true,
        count: builds.length,
        data: builds
    });
} catch (err) {
    console.error(err);
    res.status(500).json({
        success: false,
        error: 'Серверна помилка'
    });
}
};

// Отримати всі публічні збірки
exports.getPublicBuilds = async (req, res) => {
    try {
        const builds = await Build.find({ isPublic: true })
            .populate('user', 'name')
            .populate('components.cpu')
            .populate('components.motherboard')
            .populate('components.gpu')
            .populate('components.ram')
            .populate('components.ssd')
            .populate('components.hdd')
            .populate('components.psu')
            .populate('components.case')
            .populate('components.cpuCooling')
            .populate('components.caseFans');

        res.status(200).json({
            success: true,
            count: builds.length,
            data: builds
        });
    } catch (err) {
        res.status(500).json({
            success: false,
            error: 'Серверна помилка'
        });
    }
};

```

```
    }  
  };  
};
```

// Отримати збірку за ID

```
exports.getBuild = async (req, res) => {
```

```
  try {
```

```
    const build = await Build.findById(req.params.id)
```

```
      .populate('user', 'name')
```

```
      .populate('components.cpu')
```

```
      .populate('components.motherboard')
```

```
      .populate('components.gpu')
```

```
      .populate('components.ram')
```

```
      .populate('components.ssd')
```

```
      .populate('components.hdd')
```

```
      .populate('components.psu')
```

```
      .populate('components.case')
```

```
      .populate('components.cpuCooling')
```

```
      .populate('components.caseFans')
```

```
      .populate('peripherals.monitor')
```

```
      .populate('peripherals.keyboard')
```

```
      .populate('peripherals.mouse')
```

```
      .populate('peripherals.headphones')
```

```
      .populate('peripherals.speakers');
```

```
    if (!build) {
```

```
      return res.status(404).json({
```

```
        success: false,
```

```
        error: `Збірку з ID ${req.params.id} не знайдено`
```

```
      });
```

```
    }
```

```
    if (!build.isPublic && build.user._id.toString() !== req.user.id.toString()) {
```

```
      return res.status(403).json({
```

```
        success: false,
```

```
        error: 'Немає доступу до перегляду цієї збірки'
```

```
      });
```

```
    }
```

```
    res.status(200).json({
```

```
      success: true,
```

```
      data: build
```

```
    });
```

```
  } catch (err) {
```

```
    console.error(err);
```

```
    res.status(500).json({
```

```
        success: false,  
        error: 'Серверна помилка'  
    });  
}  
};
```

```
// Створити нову збірку  
exports.createBuild = async (req, res) => {  
    try {  
        req.body.user = req.user.id;  
  
        const build = await Build.create(req.body);  
  
        res.status(201).json({  
            success: true,  
            data: build  
        });  
    } catch (err) {  
        res.status(500).json({  
            success: false,  
            error: err.message || 'Серверна помилка'  
        });  
    }  
};
```

```
// Оновити збірку  
exports.updateBuild = async (req, res) => {  
    try {  
        let build = await Build.findById(req.params.id);  
  
        if (!build) {  
            return res.status(404).json({  
                success: false,  
                error: `Збірку з ID ${req.params.id} не знайдено`  
            });  
        }  
  
        if (build.user._id.toString() !== req.user.id) {  
            return res.status(403).json({  
                success: false,  
                error: 'Користувач не має прав на оновлення цієї збірки'  
            });  
        }  
    }  
};
```

```

    build = await Build.findByIdAndUpdate(req.params.id, req.body, {
      new: true,
      runValidators: true
    });

    res.status(200).json({
      success: true,
      data: build
    });
  } catch (err) {
    res.status(500).json({
      success: false,
      error: err.message || 'Серверна помилка'
    });
  }
};

```

// Видалити збірку

```

exports.deleteBuild = async (req, res) => {
  try {
    const build = await Build.findById(req.params.id);

    if (!build) {
      return res.status(404).json({
        success: false,
        error: `Збірку з ID ${req.params.id} не знайдено`
      });
    }

    if (build.user._id.toString() !== req.user.id) {
      return res.status(403).json({
        success: false,
        error: 'Користувач не має прав на видалення цієї збірки'
      });
    }

    await build.deleteOne();

    res.status(200).json({
      success: true,
      data: {}
    });
  } catch (err) {
    res.status(500).json({
      success: false,
      error: err.message || 'Серверна помилка'
    });
  }
};

```

```

    });
  }
};

// Змінити статус публічності збірки
exports.togglePublicity = async (req, res) => {
  try {
    const build = await Build.findById(req.params.id);

    if (!build) {
      return res.status(404).json({
        success: false,
        error: `Збірку з ID ${req.params.id} не знайдено`
      });
    }

    if (build.user._id.toString() !== req.user.id) {
      return res.status(403).json({
        success: false,
        error: 'Користувач не має прав на зміну цієї збірки'
      });
    }

    build.isPublic = !build.isPublic;
    await build.save();

    res.status(200).json({
      success: true,
      data: build
    });
  } catch (err) {
    res.status(500).json({
      success: false,
      error: err.message || 'Серверна помилка'
    });
  }
};

const isCompatible = (selected, target, type) => {
  if (!selected || Object.keys(selected).length === 0) {
    return true;
  }

  const {
    motherboard,
    cpu,

```

```
    ram,  
    gpu,  
    psu,  
    case: pcCase,  
    'cpu-cooling': cooler,  
    'storage-hdd': hdd,  
    'storage-ssd': ssd,  
    'case-fan': caseFans  
  } = selected;
```

```
// CPU сумісність з Motherboard  
if (type === 'cpu') {  
  if (motherboard && motherboard.socket && target.socket !==  
motherboard.socket) return false;  
}  
if (type === 'cpu') {  
  if (cooler &&  
    Array.isArray(cooler.socketSupport) &&  
    !cooler.socketSupport.map(s =>  
s.trim().toLowerCase()).includes(target.socket.trim().toLowerCase())  
  ) return false;  
}  
  
if (type === 'motherboard') {  
  if (cpu && cpu.socket && target.socket !== cpu.socket) return false;  
  
  if (ram) {  
    if (target.memoryType !== ram.type) return false;  
    if (ram.capacity && ram.capacity > target.memorySlots) return false;  
  }  
  
  if (pcCase && pcCase.motherboardSupport &&  
!pcCase.motherboardSupport.includes(target.formFactor)) return false;  
}  
  
// RAM сумісність з Motherboard  
if (type === 'ram') {  
  if (motherboard && motherboard.memoryType && target.type !==  
motherboard.memoryType) return false;  
}  
  
// GPU сумісність з Case  
if (type === 'gpu') {  
  if (pcCase && target.length && pcCase.maxGPULength && target.length >  
pcCase.maxGPULength) return false;
```

```

    if (psu && target.recommendedPSU && psu.power < target.recommendedPSU)
return false;

    if (psu && target.powerConnectors) {
        const gpuPins = target.powerConnectors?.match(/\d+/g)?.map(Number) || [];
        const requiredPins = {
            6: 0,
            8: 0,
            16: 0
        };
        gpuPins.forEach(pin => {
            if (requiredPins[pin] !== undefined) requiredPins[pin]++;
        });

        const connectors = psu.connectors || {};

        if (
            (connectors.pcie6pin || 0) < requiredPins[6] ||
            (connectors.pcie8pin || 0) < requiredPins[8] ||
            (connectors.pcie16pin || 0) < requiredPins[16]
        ) {
            return false;
        }
    }

    if (motherboard) {
        const hasX16Slot = motherboard.pcieSlots?.x16 > 0;
        if (!hasX16Slot) return false;
    }
}

// Cooler сумісність з CPU
if (type === 'cpu-cooling') {
    if (
        cpu &&
        target.socketSupport &&
        !target.socketSupport.map(s =>
s.trim().toLowerCase()).includes(cpu.socket.trim().toLowerCase())
    ) {
        return false;
    }
    if (pcCase && target.height && pcCase.maxCPUCoolerHeight && target.height >
pcCase.maxCPUCoolerHeight) return false;
}

```

```

// PSU сумісність з GPU
if (type === 'psu') {
  if (gpu) {
    if (target.power < gpu.recommendedPSU) return false;

    const gpuPins = gpu.powerConnectors?.match(/\d+/g)?.map(Number) || [];
    const requiredPins = {
      6: 0,
      8: 0,
      16: 0
    };
    gpuPins.forEach(pin => {
      if (requiredPins[pin] !== undefined) requiredPins[pin]++;
    });

    const connectors = target.connectors || {};

    if (
      (connectors.pcie6pin || 0) < requiredPins[6] ||
      (connectors.pcie8pin || 0) < requiredPins[8] ||
      (connectors.pcie16pin || 0) < requiredPins[16]
    ) {
      return false;
    }
  }

  if (pcCase && target.fanSize && pcCase.maxPSULength && target.fanSize <
pcCase.maxPSULength) return false;
}

// Case сумісність з Motherboard
if (type === 'case') {
  if (motherboard && motherboard.formFactor && target.motherboardSupport &&
!target.motherboardSupport.includes(motherboard.formFactor)) return false;
}

// HDD/SSD сумісність з Motherboard
if ((type === 'storage-hdd' || type === 'storage-ssd') && motherboard) {
  const isM2 = target.formFactor?.toLowerCase().includes('m.2');
  const isNVMe = target.interface?.toLowerCase().includes('pcie');

  if (type === 'storage-hdd') {
    if (!motherboard.sataConnectors || motherboard.sataConnectors < 1) return false;
  }

  if (type === 'storage-ssd') {

```

```

    if (isM2 && isNVMe) {
      if (!motherboard.m2Slots || motherboard.m2Slots < 1) return false;
    } else {
      if (!motherboard.sataConnectors || motherboard.sataConnectors < 1) return
false;
    }
  }
}

// Case-fan сумісність з корпусом
if (type === 'case-fan') {
  if (pcCase) {
    if (target.size) {
      if (pcCase.dimensions) {
        const supportedSizes = [
          ...(pcCase.dimensions.size || []),
        ];
        if (!supportedSizes.includes(target.size)) return false;
      }
    }
  }
}

// Motherboard сумісність з Case
if (type === 'motherboard' && pcCase && pcCase.motherboardSupport &&
!pcCase.motherboardSupport.includes(target.formFactor)) return false;

return true;
};

```

```

exports.getCompatibleComponents = async (req, res) => {
  try {
    const { type, selected } = req.body;
    const Model = getModel(type);

    if (!Model) return res.status(400).json({ error: 'Unknown component type' });

    const all = await Model.find();

    const compatible = all.filter((component) =>
      isCompatible(selected, component, type)
    );

    res.json(compatible);
  } catch (err) {

```

```

        console.error(err);
        res.status(500).json({ error: 'Server error' });
    }
};
const BuildCardProfil = ({ build, onDelete }: { build: any, onDelete?: (id: string) => void }) => {
    const [isPublic, setIsPublic] = useState(build.isPublic);

    const navigate = useNavigate();

    const handleEdit = () => {
        navigate('/', {
            state: {
                mode: 'edit',
                buildData: build
            }
        });
    };

    const togglePublished = async () => {
        try {
            const updatedBuildData = formatBuildForUpdate(build, !isPublic);
            const updatedBuild = await isPublicBuild(build._id, updatedBuildData);

            setIsPublic(updatedBuild.data.isPublic);
        } catch (error) {
            window.confirm('Помилка при оновленні статусу публікації');
            console.error(error);
        }
    };

    const handleDelete = async () => {
        if (window.confirm('Ви дійсно хочете видалити цю збірку?')) {
            try {
                await deleteBuild(build._id);
                if (onDelete) onDelete(build._id);
            } catch (error) {
                window.confirm('Помилка при видаленні збірки');
                console.error(error);
            }
        }
    };

    return (
        <div className="bg-white p-5 rounded-lg shadow hover:shadow-lg transition">
            <BuildInfo
                name={build.name}

```

```

        description={build.description}
        imageUrl={build.components.case?.image}
        build={build}
      />
      <div className="flex items-center justify-between">
        <div className="flex gap-2">
          <button onClick={handleEdit} className="flex items-center gap-1 text-sm text-blue-600 hover:underline">
            <Edit className="w-4 h-4" /> Оновити
          </button>
          <button onClick={handleDelete} className="flex items-center gap-1 text-sm text-red-600 hover:underline">
            <Trash className="w-4 h-4" /> Видалити
          </button>
          <button onClick={() => generateBuildPDF(build)} className="flex items-center gap-1 text-sm text-green-700 hover:underline">
            <Loader className="w-4 h-4" /> Завантажити PDF
          </button>
        </div>
        <div>
          <button onClick={togglePublished} className="flex items-center gap-2 text-sm text-gray-700 hover:text-blue-600">
            <span>
              {isPublic ? 'Опублікована' : 'Не опублікована'}
            </span>
            {isPublic
              ? <Eye className="w-4 h-4 text-green-600" />
              : <EyeOff className="w-4 h-4 text-gray-400" />}
          </button>
        </div>
      </div>
    </div>
  );
};

```