Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Цегельника Олексія Володимировича
Прізвище, ім'я, по-батькові студента(ки)

на здобуття ступеня вищої освіти Бакалавра

Вебзастосунок для публікації 3D-друкованих проєктів
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

КвРІПЗ.2201127.01.06.ПЗ

Виконав студент ІІІ курсу, група ІПЗс-22-1 _____ Олексій ЦЕГЕЛЬНИК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник асистент _____ _____ В'ячеслав БОЙКО
Науковий ступінь, вчене звання Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер канд.техн.наук, доцент _____ Оксана ЯШИНА
Науковий ступінь, вчене звання Підпис Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Леонід БЕДРАТЮК
Підпис Ім'я, ПРІЗВИЩЕ

11 червня 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
_____ Л. П. Бедратюк
2.01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Цегельника Олексія Володимирович
Прізвище, ім'я, по батькові студента

1. Тема роботи Вебзастосунок для публікації 3D-друкованих проєктів

Керівник роботи Бойко В'ячеслав Олександрович, асистент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 07.02.2025 р. № 23
2. Строк подання студентом роботи на кафедру 01.06.2025 р.
3. Вихідні дані до роботи Матеріали до кваліфікаційної роботи
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області і постановка задачі, проектування програмного забезпечення, програмна реалізація та тестування програмного забезпечення
5.Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Презентаційні матеріали (слайди, 14шт.)

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Яшина О.М., канд.техн.наук, доцент		
Антиплагіат	Форкун Ю. В., доцент		

7. Дата видачі завдання «02» січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1 Ознайомлення з тематикою кваліфікаційних робіт (КвР), визначення та узгодження індивідуальної теми	01.12 – 31.12.2024	
2 Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розробка технічного завдання	01.01 – 20.02.2025	
3 Проектування програмного забезпечення	21.02 – 20.03.2025	
4 Програмна реалізація з використанням відповідних засобів розробки. Тестування ПЗ	21.03 – 30.04.2025	
5 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог стандартів	01.05 – 25.05.2025	
6 Попередній захист КвР	Травень 2025	Згідно графіка
7 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05 – 30.05.2025	
8 Здача КвР на кафедру; підготовка КвР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КвР	з 01.06.2025	

Студент _____ Олексій ЦЕГЕЛЬНИК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник роботи _____ В'ячеслав БОЙКО
Підпис Ім'я, ПРІЗВИЩЕ

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Вебзастосунок для публікації 3D-друкованих проєктів».

Автор роботи: Цегельник Олексій Володимирович.

Керівник роботи: Бойко В'ячеслав Олександрович.

Пояснювальна записка: 85 с., 35 рис., 3 табл., 3 дод., 30 джерела.

Графічна частина: 14 слайдів.

3D-ПРОЄКТ, ВЕБЗАСТОСУНОК, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ФРЕЙМВОРК, СЕРЕДОВИЩЕ РОЗРОБКИ, ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

Мета кваліфікаційної роботи: проектування, реалізація та тестування вебзастосунку для публікації 3D-друкованих проєктів.

В ході кваліфікаційної роботи було здійснено аналіз предметної області, визначено функціональні та нефункціональні вимоги, а також проаналізовано вже існуючі розробки в сфері програмного забезпечення для публікування 3D-проєктів. Також було спроектовано вебзастосунок, здійснено його реалізацію та тестування.

Вебзастосунок реалізовано в середовищі програмування PHPStorm. Проект написаний на фреймворку Laravel мовою програмування PHP. Для динамічного контенту використаний фреймворк VueJS.

Результатом кваліфікаційної роботи став розроблений вебзастосунок, що дозволяє здійснювати перегляд, та публікацію 3D проектів з будь-якої точки світу, де є підключення до мережі інтернет. На перспективу, можна інтегрувати сторонні сервіси для зберігання 3D-об'єктів, наприклад, Google Drive або Dropbox та оптимізувати інтерфейс для роботи на мобільних пристроях.

10 червня
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.2201127.01.06.ПЗ	Пояснювальна записка	85		
2	А4		Завдання на кваліфікаційну роботу	1		
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні матеріали	14		
5	А3	КвРІПЗ.2201127.01.06.Е8	Схема бази даних	1		

КвРІПЗ.2201127.01.06.ВД

Виконав	Цегельник О.В.	01.06	Вебзастосунок для публікації 3D-друкованих проєктів Відомість документів
Керівник	Бойко В.О.	01.06	
Н. контр.	Яшина О.М.	01.06	
Зав. каф.	Бедратюк Л.П.	01.06	

Літ.	Арк.	Аркушів
	1	1

ХНУ, ІПЗс-22-1

ЗМІСТ

ВСТУП

У сучасному світі 3D-друк став не лише популярним захопленням, а й вагомим інструментом для вирішення прикладних завдань у різних сферах діяльності. Ця технологія активно застосовується як у побуті – для реалізації любительських проєктів, навчання, так і в професійному середовищі, де вона знаходить своє місце від стартапів до масштабних промислових та інженерних проєктів. Однією з ключових переваг 3D-друку є можливість швидко створювати прототипи, а також унікальні деталі, які важко або навіть неможливо виготовити за допомогою традиційних технологій виробництва.

Актуальність теми обумовлена стрімким зростанням доступності 3D-принтерів для широкого кола користувачів, розширенням спільнот зацікавлених у цій технології та появою нових запитів на зручні цифрові інструменти для обміну і зберігання 3D-проєктів. Сьогодні у світі вже існують популярні платформи, такі як Thingiverse, Cults3D, MyMiniFactory, які дозволяють користувачам знаходити, завантажувати та публікувати 3D-моделі. Проте більшість з них мають закритий вихідний код, обмежені можливості взаємодії з іншими користувачами, а інтерфейси цих сервісів часто є досить складними для новачків. Крім того, часто виникають труднощі з адаптацією таких платформ під індивідуальні потреби користувачів, що значно обмежує їхню функціональність для розробників або ентузіастів, які прагнуть створити власний сервіс на локальному сервері або у навчальному середовищі.

Особливо гострою є потреба серед початківців у простому, інтуїтивному способі пошуку, завантаження та збереження цікавих моделей. Водночас більш досвідчені користувачі бажають мати можливість не лише зберігати свої напрацювання, а й ділитися власними розробками, формувати персональне портфоліо та отримувати зворотній зв’язок від спільноти.

У рамках дипломної роботи буде виконанна розробка власного вебзастосунку, який забезпечить зручний інтерфейс для публікації, перегляду та збереження 3D-проєктів. Для реалізації застосунку будуть використані сучасні

вебтехнології, зокрема Laravel для серверної частини та клієнтської частини, Vue.js для динамічних компонентів на клієнтській стороні, а також база даних MySQL для зберігання інформації. Новий вебзастосунок орієнтований як на початківців, які шукають готові рішення для 3D-друку, так і на авторів моделей, які бажають поділитись своїми роботами.

Мета дипломної роботи – розробити функціональний вебзастосунок для публікації, збереження та перегляду 3D-друкованих моделей, що забезпечить простий та інтуїтивний інтерфейс, а також повноцінну систему взаємодії з користувачами**.**

В ході виконання дипломної роботи потрібно виконати такі завдання:

– виконати детальний аналіз сучасного стану технології 3D-друку, визначити основні сфери її застосування, а також з’ясувати потреби і запити потенційних користувачів;

– провести комплексний огляд існуючих вебресурсів та платформ для публікації і обміну 3D-проєктами, виявити їхні переваги та суттєві обмеження;

– сформулювати функціональні та нефункціональні вимоги до майбутнього вебзастосунку з урахуванням виявлених потреб користувачів і характеристик конкурентних продуктів;

– розробити детальне технічне завдання на розробку вебзастосунку, а також обрати оптимальну архітектуру системи, що забезпечить масштабованість, надійність та зручність у подальшій експлуатації;

– створити дизайн користувацького інтерфейсу, який відповідатиме сучасним вимогам UX/UI, забезпечить зручність і простоту навігації для користувачів різного рівня;

– реалізувати повноцінний вебзастосунок згідно з технічним завданням, інтегрувавши серверну, клієнтську частини та базу даних;

– провести тестування функціональності розробленої системи, а також оцінити зручність використання застосунку з точки зору кінцевих користувачів, здійснити коригування за результатами тестування.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області

На сьогодні, попит на 3D друк зростає у багатьох сферах та слугує багатьом цілям у різних галузях, впроваджуючи нові підходи проектування, виробництва та внесення змін у продукції.

Однією із сфер застосування є освітня галузь, де такий друк забезпечує цікавий і практичний досвід навчання. Він дозволяє здобувачам освіти розробляти і друкувати нові проекти, що розширює набір інструментів для навчання та дає більше динаміки в навчальному середовищі. Викладачам набагато легше демонструвати складні геометричні форми, або ж рухомі механізми, що в свою чергу полегшує засвоєння навчальних матеріалів, оскільки наочне візуальне сприйняття є набагато кращим та ефективнішим. Ця технологія також вимагає навчання персоналу та студентів різним компетенція, таким як принципам проектування та робота з принтером, що покращує їхні навички для майбутніх можливостей працевлаштування.

3D проекти є важливими для дизайнерів, архітекторів, інженерів та художників, тому що це сприяє інноваціям та творчості. Ця технологія долає обмеження, що накладаються традиційними методами виробництва, дозволяючи художникам, дизайнерам та винахідникам досліджувати нові сфери можливостей, що в свою чергу забезпечує додавання нових унікальних дизайнів та технологічних рішень.

Ще одна важлива роль такого друку – це значне прискорення процесу створення прототипів, дозволяючи виконавцям проводити швидкі ітерації та модифікації прототипів, набагато швидше впроваджувати вдосконалення, а також тестувати прототипи забезпечуючи конкурентну перевагу на ринках, які надають пріоритет швидкості та адаптивності. Економічна ефективність полягає у відкиданні необхідності у великих запасах і може зменшити фінансові витрати пов’язані з аутсорсингом.

З точки зору сталого розвитку, 3D-друк вирізняється мінімальним утворенням відходів. Ця технологія виробляє лише матеріал, необхідний для конкретної деталі, що призводить до значного скорочення промислових відходів і зменшення викидів у навколишнє середовище. Можливість створювати предмети на замовлення заохочує відхід від масового виробництва, що ще більше позитивно впливає на екологію.

Для 3D друку переважно застосовуються пластикові матеріали завдяки їхній легкості та різноманітності складів, що дає змогу створювати деталі з індивідуальних матеріалів, які забезпечують специфічні властивості, такі як термостійкість, вища міцність або водонепроникність, що в свою чергу ще більше розширює універсальність застосування.

3D-друк дозволяє створювати складні органічні форми, які набагато легші, ніж деталі, створені традиційним способом. Він став популярним хобі завдяки своїй невисокій вартості та простому доступу.

Сервіси для публікації 3D-друкованих проектів слугують комплексною платформою, призначеною для навчання, підтримки та сприяння впровадженню технологій 3D-друку в різних галузях, зокрема в освіті та бізнесі. Ця технологія створила революцію у виробничих та дизайнерських процесах, пропонуючи інноваційні рішення, які оптимізовують та вирішують такі проблеми, як високі виробничі витрати, управління запасами та обмеження традиційних методів виробництва. Це дає змогу значно швидше створювати прототипи, впроваджувати на вимогу нові зміни у виробництво. 3D друк не лише розширює можливості творчого самовираження, але й значно підвищує ефективність виконання робіт.

Значення платформи для розміщення 3D друкованих проектів полягає в його здатності сприяти розвитку культури інновацій та співпраці. Він слугує хабом для викладачів та студентів, надаючи їм можливість публікування ресурсних файлів для друку, особливо для напрямків дизайну. Підприємці можуть використовувати різноманітні ресурси для надання індивідуальних продуктів, тим самим підвищуючи лояльність та залученість клієнтів. Крім того,

таких спосіб має трансформаційний потенціал, що дозволяє компаніям залишатись конкурентоспроможними в умовах швидких ринкових змін.

Незважаючи на численні переваги, широке впровадження технологій 3D друку має свої недоліки, такі як регуляторна невизначеність, матеріальні обмеження, проблеми контролю якості та питання інтелектуальної власності. Ці виклики можуть бути особливо важливими для малого бізнесу, який може стикнутись із початковими витратами і складністю інтеграції 3D друку у своїй діяльності. Таким чином, добре структурований вебзастосунок для публікації 3D друкованих проектів (хаб) стає необхідним для підтримки та подолання таких проблем.

Підсумовуючи, ресурсний центр відіграє важливу роль у коректному доступі до цієї трансформаційної технології, враховуючи як переваги, так і виклики, які вона створює. Він відіграє ключову роль у формуванні майбутнього виробництва та дизайну в різних секторах, що в кінцевому підсумку сприяє економічному зростанню, підвищенню якості освіти та креативності.

3D проект – це файл з відповідним розширенням (зазвичай у таких форматах, як .stl, .wrl, .ply), який надає принтеру інструкції для виконання, вказуючи в які сторони рухати соплом для правильного та коректного друку готового елементу. Перед тим як передати проєктний файл на 3D-друк, його необхідно підготувати у відповідному середовищі розробки. Цей етап є критично важливим, адже правильне моделювання визначає кінцеву якість та функціональність надрукованого об'єкта.

Як досвідчені фахівці, так і ентузіасти можуть створювати власні 3D проекти, використовуючи спеціалізовані програми для тривимірного моделювання. Ці середовища надають широкий набір інструментів для проектування, редагування та підготовки моделей до друку.

Серед найпопулярніших середовищ для 3D-моделювання можна виділити Tinkercad (рис. 1.1.1) та SelfCAD (рис. 1.1.2). Вони відзначаються зручним інтерфейсом та широкими можливостями для роботи з просторовими об'єктами.

Tinkercad є особливо зручним для новачків завдяки інтуїтивному інтерфейсу та спрощеним інструментам. SelfCAD, у свою чергу, пропонує розширений набір функцій для більш досвідчених користувачів, дозволяючи працювати з більш складними моделями та деталізацією

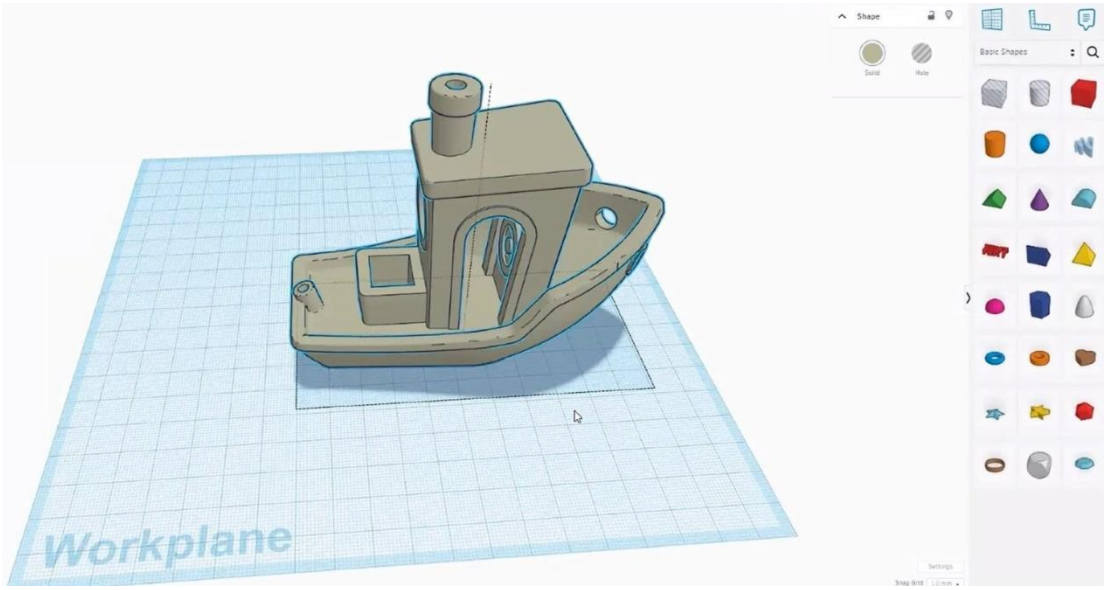


Рисунок 1.1.1 – Середовище розробки 3D проектів Tinkercad

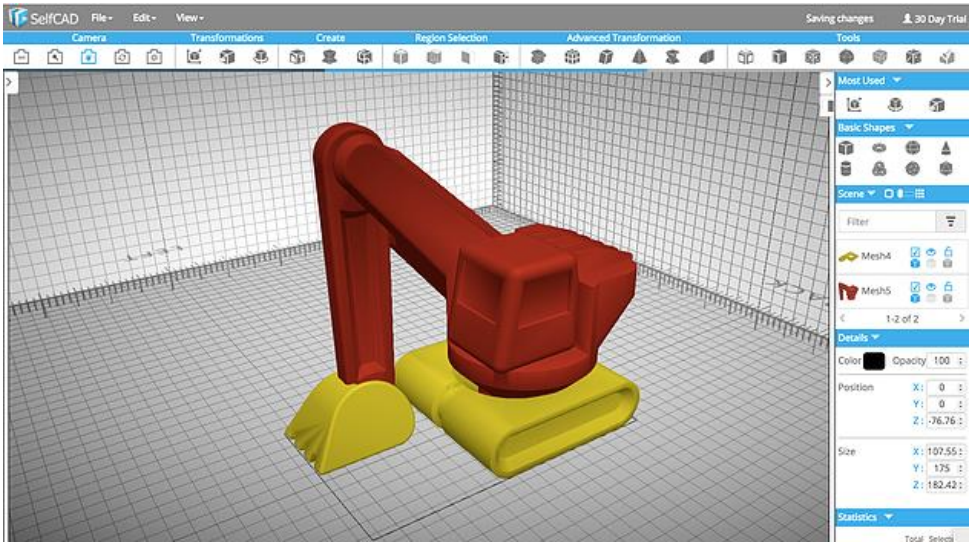


Рисунок 1.1.2 – Середовище розробки 3D проектів SelfCAD

Для пошуку вже створених 3D-проєктів існують спеціалізовані онлайн-платформи та централізовані сховища, які надають можливість обміну й поширення моделей між користувачами.

Такі ресурси дозволяють авторам ділитися своїми проектами, завантажувати файли у відкритий доступ, отримувати зворотний зв'язок, а також відстежувати популярність власних робіт. Інші користувачі, у свою чергу, можуть завантажувати готові 3D-моделі для власних потреб, використовувати їх у своїх проектах або ж редагувати під свої вимоги.

Окрім цього, такі платформи зазвичай підтримують функції взаємодії між користувачами, що дозволяє коментувати проекти, ставити оцінки, залишати відгуки та навіть обговорювати деталі друку чи доопрацювання моделей. Це сприяє розвитку активної спільноти, де кожен може знайти натхнення, отримати корисні поради або поділитися власним досвідом.

Централізовані додатки для публікації та збереження 3D проектів допомагає вирішити чималу низку завдань у сфері 3D-моделювання, візуалізації та спільної роботи над проектами. Це не просто архів моделей, а потужний інструмент для управління цифровими активами, що дозволяє:

- упорядкувати творчий процес – замість зберігання десятків окремих файлів на локальних дисках абр в розрізнених хмарних сервісах, користувачі отримують єдине місце для зберігання, зручного пошуку та управління версіями;
- спрощення демонстрації проектів – веб-платформа дозволяє ділитися інтерактивними 3D моделями з колегами, друзями без потреби у встановленні спеціального програмного забезпечення;
- оптимізація командної роботи – колективи дизайнерів, архітекторів та розробників можуть одночасно слідкувати за проектом та коментувати його зміни;
- збереження та підтримка моделей – у хабі можна створювати резервні копії проектів, запобігаючи втраті важливих даних через збій обладнання чи випадкове видалення;

– інтеграція з іншими цифровими інструментами, наприклад імпорт та експорт моделей у різних форматах, а також синхронізація з 3D-редакторами чи сервісами для підготовки до друку, що значно спрощує робочий процес.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Будь-який процес розробки нового рішення вимагає ґрунтованого аналізу вже наявних альтернатив. Без цього неможливо об’єктивно оцінити стан ринку, зрозуміти переваги та недоліки існуючих рішень, а також виявити ключові особливості розроблених програмних забезпечень, що в свою чергу дає можливість обґрунтовано підходити до формування вимог до власного продукту.

Комплексний аналіз дозволяє систематизувати інформацію про функціональні можливості, архітектурні особливості та організацію інформаційних потоків у межах предметної області. Це сприяє прийняттю зважених рішень щодо розробки нового рішення, орієнтованого на сучасні технологічні виклики та потреби користувачів.

Таким чином, дослідження наявних платформ є необхідним кроком для розробки ефективного та конкурентоспроможного сервісу, який відповідатиме сучасним вимогам і забезпечуватиме високу якість роботи з 3D проектами.

Сучасні платформи для публікації та обміну 3D моделями надають користувачам можливість демонструвати свої роботи, ділитися ними з клієнтами та спільнотою, а також інтегрувати 3D контент у веб-сайти або VR/AR середовища. Всі вони різняться між собою функціоналом, можливостями інтеграції, типами підтримуваних моделей і орієнтованістю на певні галузі, такі як дизайн, архітектура, геймдев, та інші.

Кожна платформа несе мету принести певну користь користувачу та заточена під певний функціонал. Деякі сервіси пропонують інтерактивність, підтримку 3D друку, чи навіть наявність можливості продажу моделей.

Першим прикладом є сайт https://www.thingiverse.com (рисунок 1.2.1).

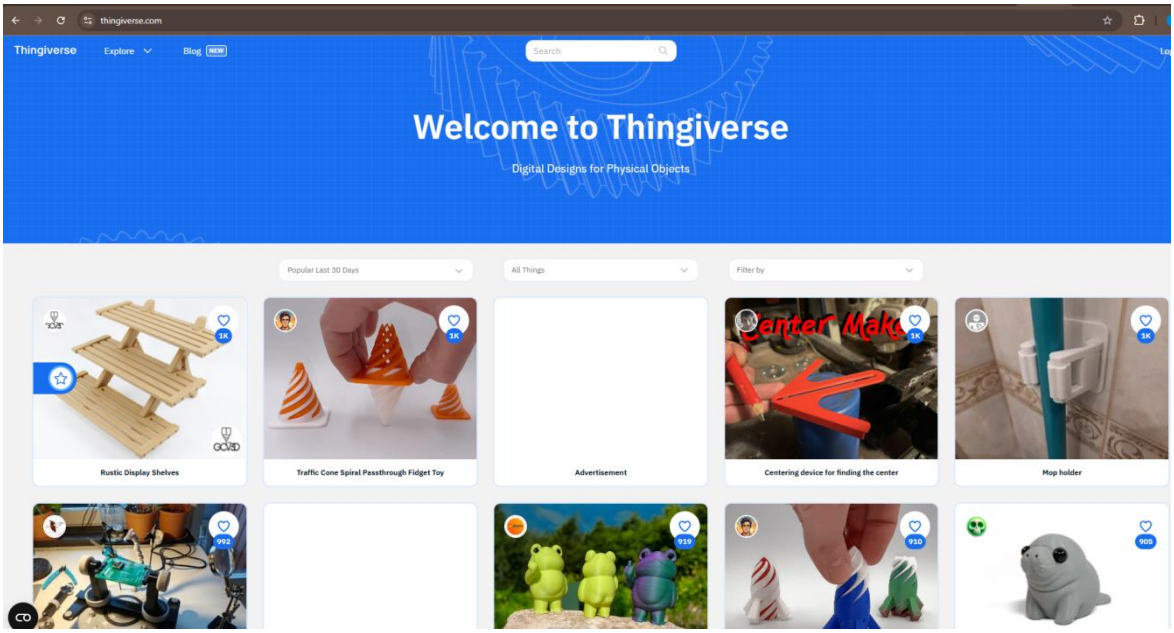


Рисунок 1.2.1 – Вигляд веб сервісу www.thingiverse.com

Дана платформа більше орієнтована на користувачів. Вона містить чималу бібліотеку моделей, які доступні до завантаження, змін та використання для друку. Основними перевагами окремо можна виділити:

- безкоштовний доступ абсолютно до усіх моделей;
- більш детальний опис щодо правильності параметрів та формату друку певних моделей;
- можливість завантаження готових до друку STL-файлів.

Однак, дана платформа також містить ряд недоліків. Серед них:

- відсутність підтримки анімації та інтерактивного перегляду моделей у реальному часі;
- обмежений вибір форматів файлів (головним є лише STL);
- через відсутність контролю, якість публікованих матеріалів може бути низькою.

Наступним прикладом є сайт https://www.turbosquid.com (рисунок 1.2.2). Даний ресурс є одним із найстаріших, та більше орієнтована на професійних дизайнерів, аніматорів, розробників ігор, які шукають високоякісні ресурси. З головних переваг є величезна бібліотека моделей, структур і матеріалів, висока якість контенту, оскільки усі файли перевіряються перед продажем.

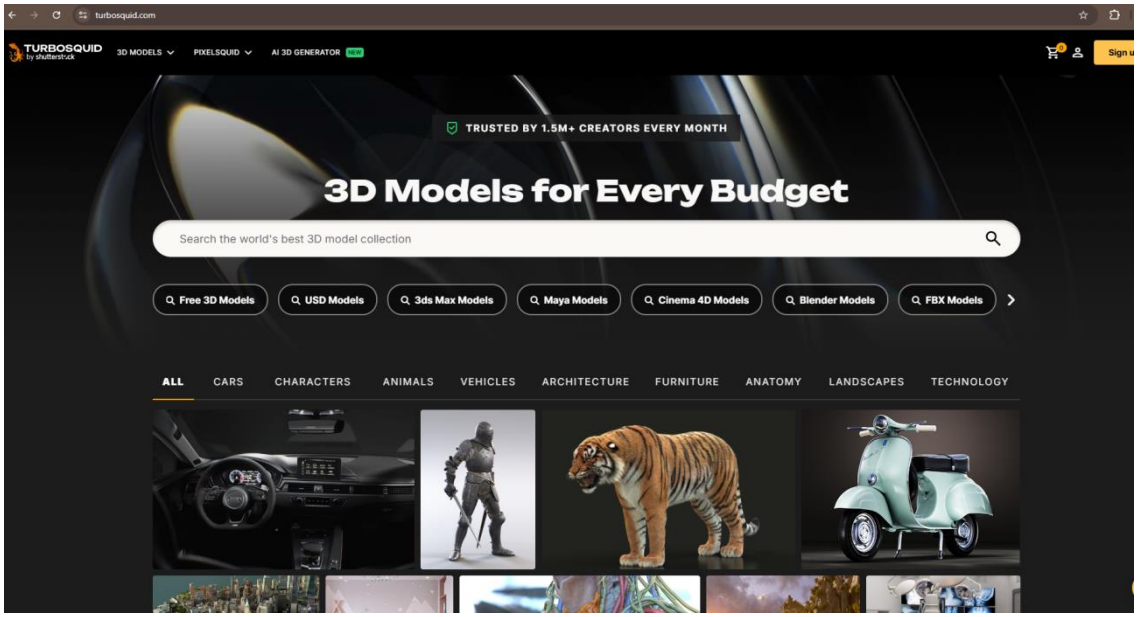


Рисунок 1.2.2 – Вигляд веб сервісу www.turbosquid.com

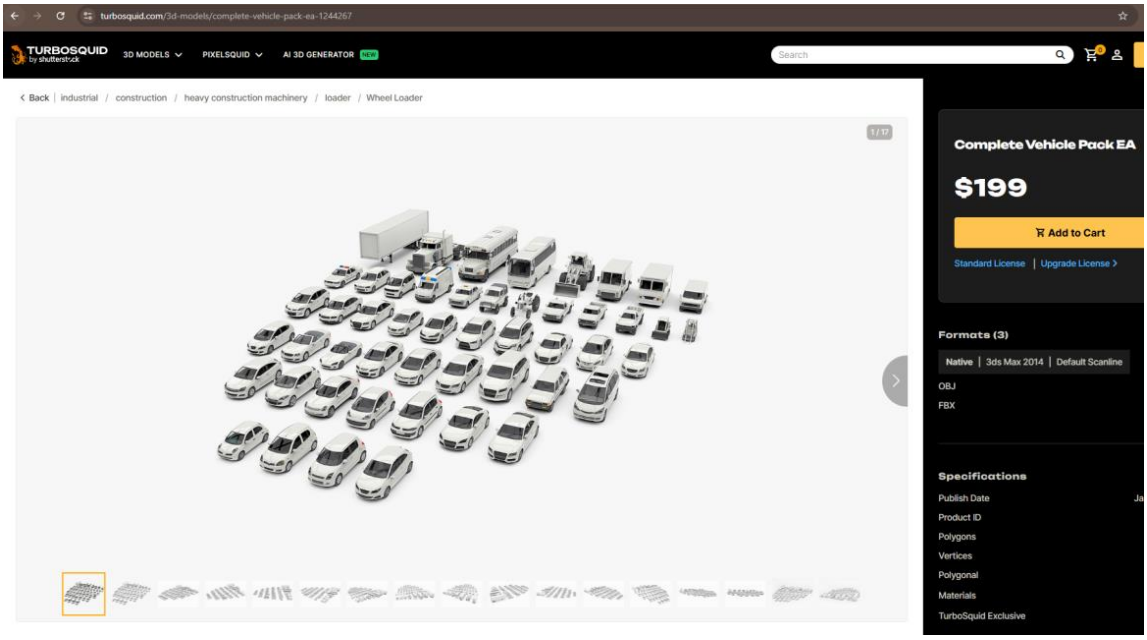


Рисунок 1.2.3 – Платні матеріали з веб сервісу www.turbosquid.com

Також даний сайт поєднує у собі підтримку платних та безкоштовних моделей, а що найголовніше – політику компенсації у випадку невідповідності файлу опису. З недоліків можна відмітити відсутність інтерактивного перегляду моделей у реальному часі, зависокі ціни та акцент на продажу моделей (рис. 1.2.3), що ускладнює співпрацю з початківцями, які шукають більш простих, але в той же час якісних матеріалів.

Одним із найкращих веб ресурсів та найбільш популярних для публікації 3D моделей є Sketchfab (https://sketchfab.com/) (рис. 1.2.4). Він має найбільш гармонійний дизайн, більш зручне використання фільтрів для пошуку різноманітних моделей. Серед усіх перелічених веб ресурсів, саме цей має функціонал для інтерактивного перегляду 3D об’єктів у браузері без потреби у додаткових плагінах. Тобто користувачі можуть взаємодіяти з моделями безпосередньо у веб-браузері, що робить платформу зручнішою у використанні, оскільки вона пропонує більш наглядно працювати із публікованими модулями. Самі моделі можна вбудовувати у веб-сайти, ділитися через посилання, а також продавати на вбудованому маркетплейсі.

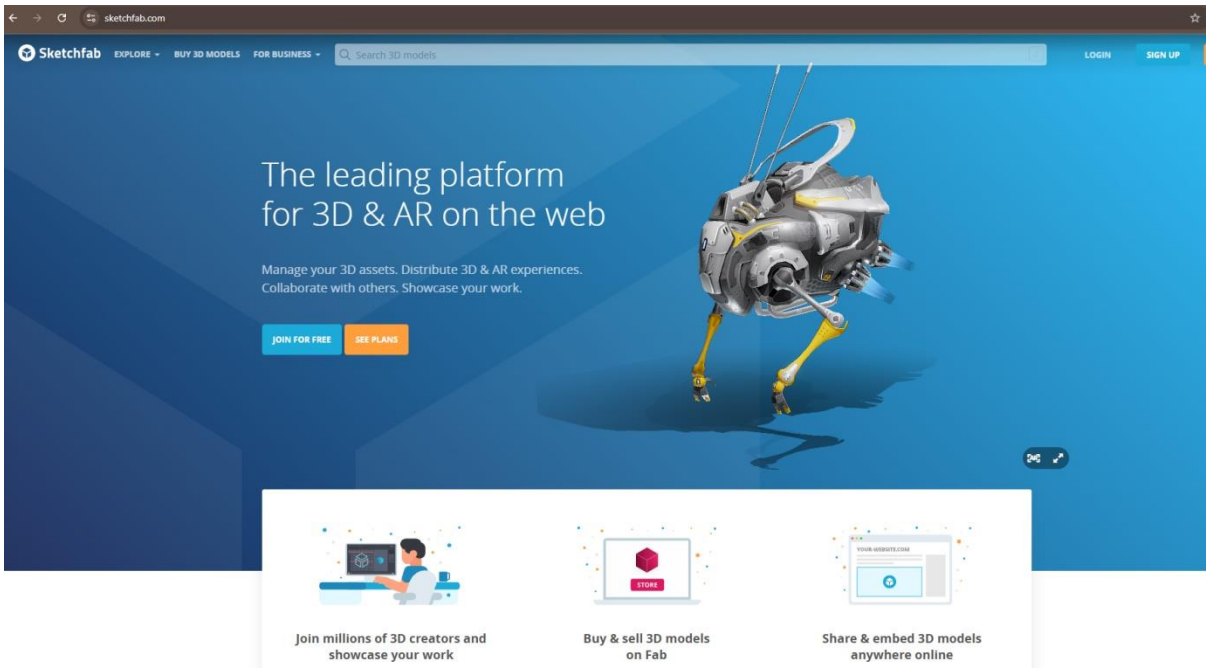


Рисунок 1.2.4 – Вигляд веб сервісу sketchfab.com

Більше того, даний веб сервіс підтримує більше 50 форматів ресурсних файлів (FBX, OBJ, STL, та інші). Завдяки такій широкій підтримці форматів користувачі можуть легко завантажувати, переглядати та редагувати моделі, незалежно від програмного забезпечення, у якому вони були створені.

Також, він підтримує VR/AR, що дозволяє використовувати окуляри доповненої реальності для реалістичності відтворення 3D моделі у просторі.

1.3 Визначення вимог до програмного забезпечення та розробка технічного завдання

Під час розробки будь-якого програмного забезпечення, важливим етапом є визначення вимог та їх опрацювання. Головним завданням є здійснення опису учасників процесу взаємодії із веб за стосунком, тобто визначити їхні ролі для кожної із сторін та описати які саме дії будуть доступні для кожної із ролей.

Найголовнішою роллю, яка буде взаємодіяти є адміністратор, який повинен контролювати наповнення (контент, моделі), а також адмініструвати користувачів. Адміністратор – це користувач, який заходить у веб застосунок через спеціальну адмін. панель.

Наступною важливою роллю є користувачі – тобто всі люди, які заходять на платформу, та авторизовуються. Завдяки авторизації у веб застосунку, користувачі перестають бути просто гостями (простими відвідувачами) і будуть мати більші можливості, у порівнянні із ними. Також у список доступних дій для авторизованих користувачів входить більш глибокий пошук, включаючи більший спектр фільтрів, можливість публікації власних моделей, коментування та вподобання існуючих публікованих моделей іншими користувачами, а також збереження моделей в розділ «улюблені».

Найменш функціональною роллю є гість, тобто не зареєстровані чи авторизовані відвідувачі веб застосунку. Для такої ролі буде доступно лише реєстрація, переглядання існуючих опублікованих моделей, а також використання простої групи фільтрів.

Для успішного функціонування системи, повинен бути розроблений функціонал для зареєстрованого користувача. Деталі такого функціоналу є наступними:

- авторизація на веб ресурсі;
- зручне меню з розширеною групою фільтрів;
- перегляд існуючих опублікованих моделей;
- публікування власних моделей;
- додавання нових динамічних параметрів при створенні нової моделі;

– додавання нових динамічних значень для параметрів при створенні нової моделі;

– редагування власних моделей;

– видалення власних моделей;

– коментування моделей інших користувачів;

– вподобання моделей інший користувачів;

– збереження моделей в улюблену категорію.

Функціонал для адміністратора:

– crud усіх наявних користувачів, тобто можливість створювати нових, редагувати та видаляти існуючі акаунти;

– crud усіх наявних публікацій, дати змогу редагувати існуючі публікації, створювати нові, та видаляти не актуальні публікації;

– crud усіх коментарів до існуючих публікацій, адміністратор повинен мати змогу видаляти коментарі, які порушують порядок та взаємоповагу;

– crud усіх динамічних атрибутів для проектів;

– crud усіх значень динамічних атрибутів для проектів.

Коли розробляється будь-яка система, зокрема веб додаток для публікації 3D друкованих проектів, необхідно чітко визначити вимоги до неї. Загалом існує 2 загальних типи: функціональні та нефункціональні вимоги. Вони визначають наскільки кінцевий продукт відповідає очікуванням від поставленої задачі. Точний опис майбутньої системи дає змогу розробникам чітко спланувати графік виконання робіт та полегшує дотримання дедлайнів, оскільки атомарно поділені завдання набагато легші для усвідомлення поставленої завдання та реальну оцінку часу, скільки необхідно для її виконання.

За загальний опис самої сутті системи, тобто що саме вона повинна виконувати відповідають функціональні вимоги. Вони визначають, який конкретно функціонал має бути реалізований для виконання основних завдань користувачів:

– авторизація (зареєстрованих) та окремо реєстрація користувачів у веб за стосунку;

– вивід усіх наявних моделей;

– пагінація – розділення виводу даних на декілька сторінок;

– деталізований вивід кожної моделі з можливістю коментування та вподобання;

– створення окремих вподобаних списків для користувачів;

– можливість пошуку моделі по назві чи ключових словах;

– надання створення нових гнучких атрибутів при публікації нової моделі;

– надання створення нових гнучких значень атрибутів при публікації нової моделі.

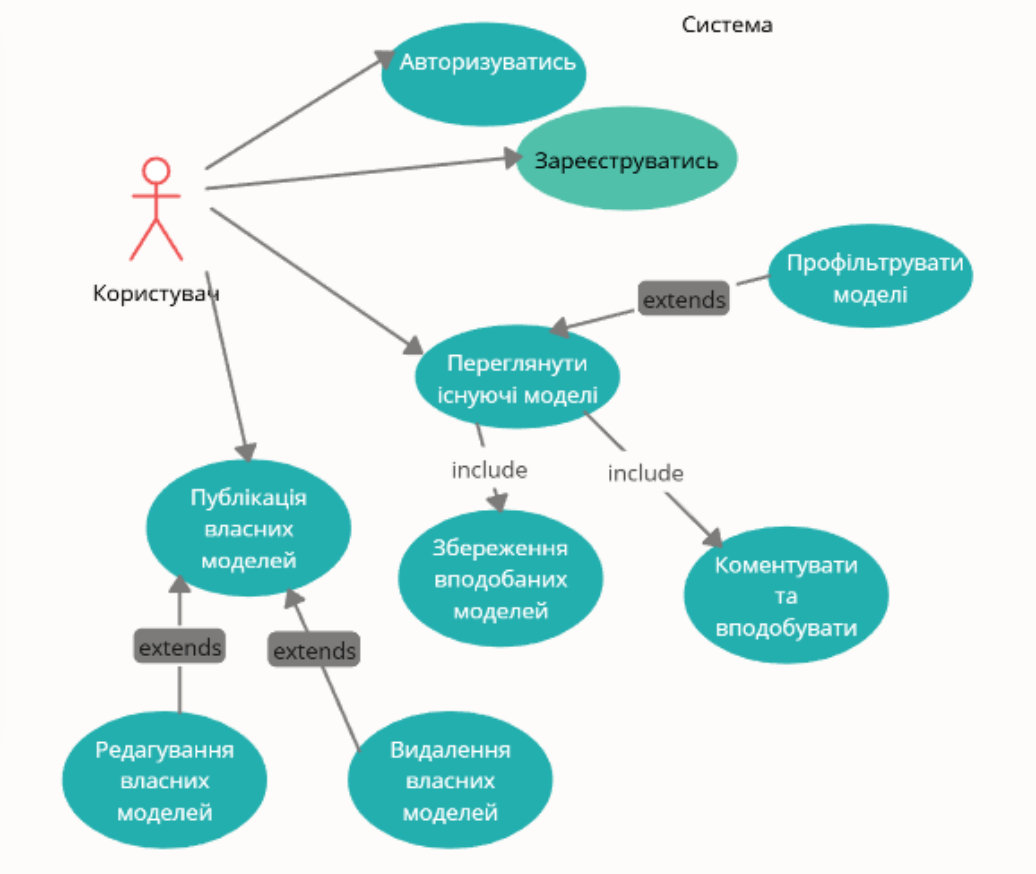


Рисунок 1.3.1 – Діаграма Use Case для вебзастосунку для публікації 3D друкованих проектів

На рисунку 1.3.1 зображено діаграму використання (Use Case) для вебзастосунку для публікації 3D друкованих проектів, де відображено ролі та відповідні можливості для кожної з ролей.

На рисунку 1.3.2 зображено діаграму використання (Use Case) для ролі адміністратора. Оскільки адміністратор буде мати повні права, відповідно такий користувач буде мати змогу повністю контролювати головні сутності застосунку, а саме користувачів, публікові моделі, коментарі та динамічні атрибути з динамічнними значеннями. Для повного контролю дана роль повинна могти виконувати CRUD дії, тобто створення нових сутностей, та редагування чи видалення існуючих.

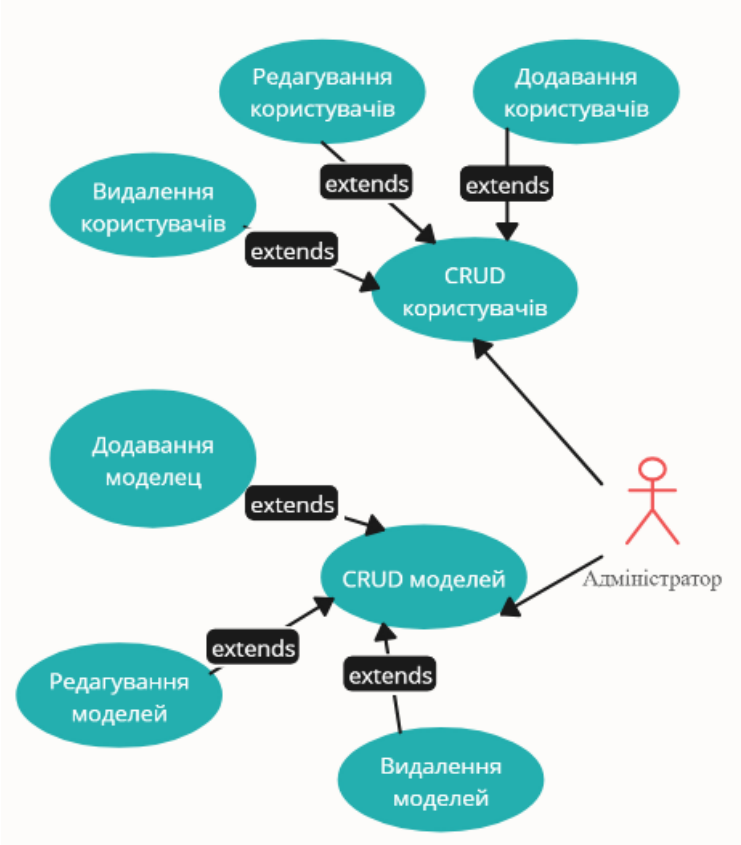


Рисунок 1.3.2 – Діаграма Use Case для ролі адміністратора

За основний опис роботи системи, зокрема її продуктивність, рівень безпеки, масштабованість, стабільність та зручність використання відповідають нефункціональні вимоги. Вони визначають якість роботи системи та її

обмеження, без опису конкретного функціоналу. Натомість визначають загальні обмеження під час експлуатації. Більш детальний опис таких вимог:

- середній час відповіді системи на усі запити від користувачів;
- надійність та продуманість від майбутніх помилок, а також швидкість виправлення помилок, якщо такі стануться;
- пропускна здатність, а саме можливість обробляти велику кількість запитів за певний проміжок часу, або ж балансування вхідного трафіку;
- доступність для усіх користувачів з різних пристроїв.

Одним із ключових факторів не функціональних вимог є вимоги до надійності: швидке відновлення у разі виникнення помилок, відновлення бази даних, щоб не втратити існуючі дані, моніторинг стану додатку, а саме завантаження, використанні пам’яті, процесора, та наявних ресурсів.

Наступним кроком потрібно визначити вимоги до інтерфейсу:

- простий та зрозумілий для користувача без надлишкового навантаження;
- відображені елементи керування повинні бути відображені відповідно до наданої ролі;
- сторінки, на які звичайні користувачі не будуть мати доступу повинні мати валідацію та повертати на головну сторінку.

Також важливо виділити вимоги, які повинні бути поставленими до контенту, для того аби забезпечити якість, зручність використання та відповідність до ролі. Основна структура виводу даних публікованих моделей буде мати наступний вигляд:

- метадані – назва, автор, дата завантаження, опис;
- зображення готової моделі;
- детальний опис усіх характеристик моделі;
- секція для завантаження матеріалів у відповідному форматі;
- секція для з усіма динамічними атрибутами для проекту, що надасть більше динамічної інформації;

– секція з коментарями та окремим полем для авторизованих користувачів для додавання даних нових коментарів;

– секція з кількістю вподобань моделі за відображення кнопок для оцінки проекту;

– на головній сторінці для користувачів повинен бути доступний рядок пошуку по ключових словах моделей;

– повинна бути окрема секція, де будуть зберігатись усі проекти, які користувач уже зберіг;

– якщо користувач авторизований, він буде мати змогу залишати коментарі та оцінювати публіковані моделі іншими авторами за 5-ти бальною шкалою (5 зірочок);

– при створенні нової моделі, користувач зможе динамічно додавати нові атрибути, якщо існуючий набір атрибутів не містить необхідних. В подальшому, всі створенні динамічно атрибути будуть доступні для усіх інших моделей.

У даному розділі було глибоко проаналізована предметна область, що дало змогу краще зрозуміти суть задачі та потреби майбутніх користувачів. Аналіз наявних рішень показав, які підходи, функціонал та інтерфейси вже використовуються, де вони ефективні, і де можна вдосконалити певні речі. Були визначені функціональні та нефункціональні вимоги, в яких чітко окреслений функціонал відповідно до ролей, щоб забезпечити коректне проектування та подальшу розробку.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Проектування архітектури програмного забезпечення

Проектування є одним із ключових етапів розробки програмного забезпечення, оскільки саме цей етап визначає структуру, логіку роботи та взаємодії компонентів. На цьому етапі важливо закласти основу архітектури, що в свою чергу впливає на продуктивність, масштабованість та зручність для подальшого розвитку. Гарне та структурне проектування вирішує ряд проблем:

- зменшення ризиків помилок на етапі розробки;
- спрощення інтеграцій нових функцій;
- підвищення ефективності взаємодії користувачів із сервісом;
- забезпечення зручної підтримки та розвитку програмного продукту.

Процес проектування включає в себе визначення основних компонентів системи, функції та способи обміну даними. Окрім визначення загальної логіки системи, необхідно вибрати необхідні технології для реалізації. Це стосується серверної частини: вибір мови програмування, відповідного фреймворка, архітектури серверу, так і клієнтської частини: інтерфейс користувача, формат представлення даних, адаптивність дизайну. Важливим етапом є розробка проекту бази даних, яка забезпечить правильність збереження даних, надійність та швидкість обробки інформації.

У рамках цього проєкту проаналізовано кілька можливих архітектур програмного забезпечення:

– монолітна архітектура передбачає, що всі компоненти системи розміщуються в одному модулі. Такий підхід спрощує початкову розробку, але створює складнощі при масштабуванні та супроводі: будь-яка зміна потребує повторного збирання всієї системи, що ускладнює підтримку;

– мікросервісна архітектура базується на поділі системи на незалежні сервіси, кожен з яких виконує окрему функцію. Цей підхід є високоефективним для великих і розподілених проєктів, проте потребує значних ресурсів для

налаштування взаємодії між сервісами, організації мережевої безпеки та моніторингу;

– клієнт-серверна архітектура дозволяє чітко розмежувати клієнтську частину (інтерфейс користувача) та серверну частину (бізнес-логіка та робота з даними). У межах цього проєкту було обрано саме клієнт-серверну архітектуру (рисунок 2.1.1), оскільки вона найкраще відповідає визначеним вимогам – забезпечення масштабованості, безпеки, продуктивності та підтримки декількох типів клієнтських пристроїв. Додаток буде складатись із двох частин:

- клієнтська (frontend, зовнішній вигляд) відповідає за взаємодію із користувачами та системою;
- серверна.

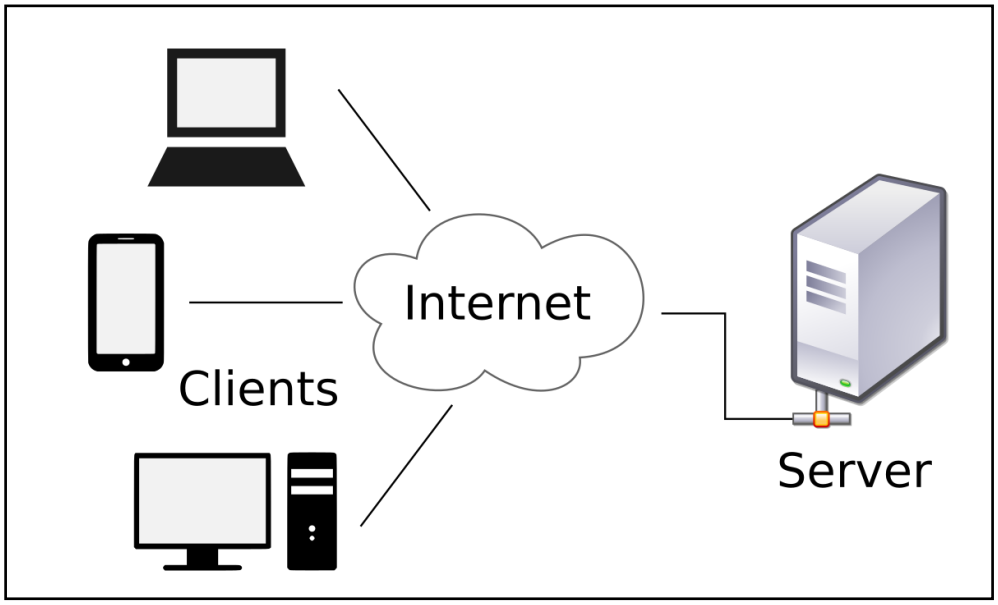


Рисунок 2.1.1 – Схема клієнт-серверної архітектури

Серверна частина буде слугувати для опрацювання вхідних запитів від користувачів та містити основну бізнес-логіку. Тобто буде тримати в собі моделі – опис сутностей, комунікацію з базою даних збереження та відображення файлів, валідацію даних, опис ендпоінтів – вхідних точок для API запитів, початкові статичні ресурси для відображення на фронтовій стороні (стилі, картинки), макети сторінок. Також серверна сторона буде відповідати за

авторизацію користувачів для забезпечення коректної роботи для різних ролей користувачів.

Передача даних між цими вдома компонентами буде можлива завдяки API (application programming interface), а саме REST API. Це протокол даних, який призначений для передачі даних. Основними типами запитів є GET та POST запити. На відмінну від GET, POST запит може містити корисне навантаження, наприклад JSON структуру даних, які серверу потрібно отримати, провести валідацію та обробити.

Серед нефункціональних вимог, що вплинули на вибір архітектури, можна виділити:

- підтримка роботи з мобільних і десктопних пристроїв;
- легкість масштабування для збільшення кількості користувачів;
- забезпечення швидкої обробки запитів;
- потреба в розмежуванні прав доступу до функціоналу;
- модульність і підтримка розширення функціоналу.

Мова програмування займає одне з центральних позицій усього додатку. Від вибору мови залежить швидкість та час написання функціоналу, швидкість виконання коду та складність написаної програми. Для даного проекту буде використано мову програмування PHP, оскільки вона швидша та продуктивніша, ніж Python, має простіший синтаксис з можливістю динамічної типізації у порівнянні із C#, а також більш структурована мова для написання зручного коду в порівнянні із JavaScript. Також PHP має змогу працювати з файлами, у які можна додавати код окрім стандартної мови розмітки HTML.

Писати додаток з нуля не є коректним рішенням, оскільки існує чимало готових фреймворків, які пропонують різноманітний функціонал, а також численну кількість сторонніх плагінів для розширення початкового функціоналу. Готові фреймворки пропонують вже готові механізми авторизації, робота з базою даних на низькому рівні (створення з’єднання, надсилання запитів, створення транзакцій, готові ORM конструкції замість прямого SQL коду), механізми для створення нових сутностей та маршрутизації (створення

ендпоінтів для API). Найбільш популярними фреймворками є Laravel, Symphony та Yii2. Серед усіх перелічених фреймворків найбільш оптимальним є саме Laravel. В першу чергу, він містить вбудований пакетний менеджер Composer, що спрощує процес встановлення сторонніх пакетів інших вендорів для забезпечення правильного функціонування. Даний фреймворк має простішу структуру та забезпечує більш швидку розробку нового функціоналу. Додатковою перевагою є окреме розширення Laravel Backpack – це розширення, яке пропонує зручну адмін панель для реалізації CRUD дій (створення, редагування та видалення сутностей). Більше того, Laravel підтримує інтеграцію із JavaScript фреймворком VueJS, що дає змогу розробляти динамічні компоненти для реактивної взаємодії на фронтовій частині з клієнтами. На рисунку 2.1.2 зображена основна структура даного фреймворку:

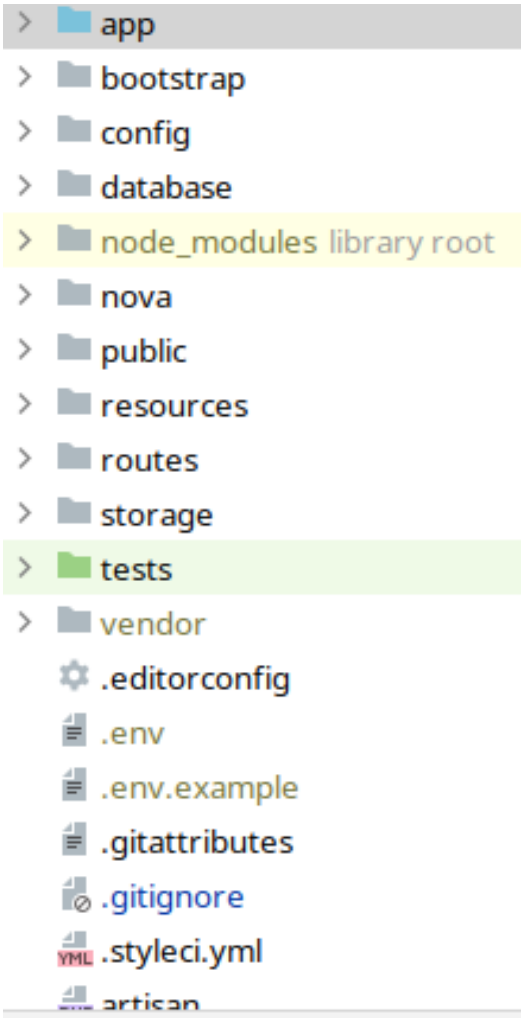


Рисунок 2.1.2 – Структура фреймворку Laravel

Laravel – це сучасний PHP-фреймворк, який орієнтований на зручність розробки, масштабованість і дотримання принципів чистої архітектури. Однією з ключових особливостей фреймворку є підтримка об’єктно-орієнтованої парадигми програмування, що дозволяє будувати додаток на основі класів, чітко розподілених за ролями. У цьому проекті логіка буде реалізована у вигляді класів, які розміщуватимуться в директорії app/, як того вимагає структура Laravel.

В основі будь-якого Laravel-проекту лежить шаблон MVC (Model-View-Controller). Цей підхід дозволяє розділити додаток на окремі рівні: модель відповідає за роботу з базою даних, контролер – за логіку обробки запитів, а представлення (view) – за те, як саме інформація буде подана користувачу. Така структуризація коду значно полегшує його підтримку, масштабування та командну розробку.

У цьому проекті класи моделей, які описують сутності бази даних, успадковують функціонал від базового класу Model. Це означає, що кожна модель автоматично отримує доступ до методів взаємодії з базою – таких як пошук, фільтрація, створення, оновлення або видалення записів. Контролери, своєю чергою, будуть наслідувати стандартний клас Controller, що надає зручні інструменти для обробки HTTP-запитів, виклику відповідних сервісів, передачі даних у view або формування відповіді у форматі JSON.

Маршрутизація в Laravel реалізована у вигляді окремого файлу routes/web.php, у якому задаються всі правила обробки запитів. Кожен маршрут містить шлях, HTTP-метод (GET, POST, тощо) і вказівку, який саме метод якого контролера слід викликати. Це дозволяє чітко розуміти, який код буде виконано у відповідь на запит користувача до певного URL-адресу. Наприклад, при переході на сторінку перегляду проектів, маршрут спрямує запит на метод index() контролера, який отримає дані з бази через модель та передасть їх у відповідне представлення.

Для зручності розробки, підтримки та масштабування проєкту у структурі бази даних використовуються міграції у вигляді спеціальних PHP-файлів, які

програмно описують створення, зміну або видалення таблиць, полів та індексів. Кожна зміна структури бази даних фіксується у вигляді окремого міграційного файлу, що дозволяє відтворювати ці зміни на будь-якому середовищі лише за допомогою однієї команди. Такий підхід гарантує узгодженість структури БД між усіма учасниками команди та середовищами розробки, тестування і продакшну. Крім того, це значно знижує ймовірність людських помилок при ручному внесенні змін до бази даних, полегшує відкат змін у разі помилок і забезпечує зручне ведення історії змін. Застосування міграцій є критично важливим для командної розробки, автоматизації розгортання та підтримки довгострокової стабільності проекту.

Щоб зробити архітектуру додатку ще більш гнучкою, у проекті реалізовується підхід із використанням сервісних класів і репозиторіїв. Це дозволяє відокремити логіку обробки бізнес-процесів від логіки взаємодії з базою даних, що позитивно впливає на читабельність коду, спрощує тестування й майбутнє розширення функціональності. Наприклад, замість того щоб безпосередньо звертатися до моделі в контролері, контролер викликатиме метод сервісу, який у свою чергу взаємодіє з репозиторієм.

Окрім серверної логіки, у проекті активно використовуються JavaScript-скрипти, які забезпечують динамічну поведінку інтерфейсу. Вони розташовані в папці public/js і відповідають за оновлення контенту на сторінці без необхідності повного її перезавантаження. Це особливо актуально для таких дій, як фільтрація, сортування, додавання елементів у список або навігація через пагінацію. У поєднанні з бібліотекою Axios, яка дозволяє виконувати асинхронні HTTP-запити, користувачі отримують плавний досвід взаємодії з системою: дані підвантажуються у реальному часі, сторінка не мерехтить, а інтерфейс залишається стабільним і привабливим.

Ще одним функціональним компонентом проекту є використання Backpack CRUD. Це розширення для швидкої та зручної реалізації адміністративної панелі, що значно пришвидшує процес розробки, забезпечує зручний UI адміністрування усіх сутностей, як це показано на рисунку 2.1.3.

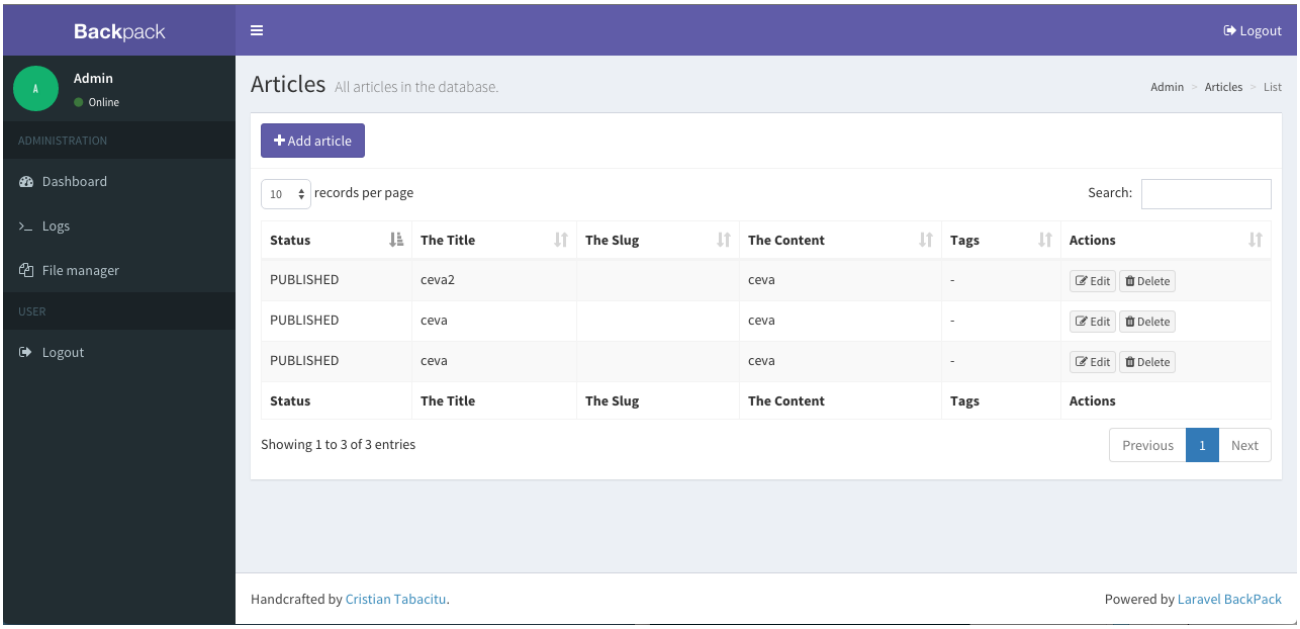


Рисунок 2.1.3 – Адмін панель Laravel Backpack

Базова архітектура буде доповнена цим функціоналом, що дозволить виконувати CRUD-операції (створення, редагування, видалення) над сутностями.

2.2 Проектування структур даних

Структури даних – це спосіб організації інформації для її ефективного зберігання, обробки, передачі та використання. Вони відіграють ключову роль у процесі розробки програмного забезпечення, оскільки правильний вибір та реалізація структур значно впливають на продуктивність і зручність роботи системи. Саме цей етап є одним із найважливіших та найбільш ресурсомістких у процесі проектування.

Головним сховищем даних у системі є база даних. На даному етапі необхідно детально описати структуру всієї бази даних, визначити її основні сутності, атрибути та зв’язки між ними. У подальшому базу слід формалізувати, тобто привести її до третьої нормальної форми (3НФ), що забезпечить оптимізацію, усунення надлишкових даних і підвищення ефективності її використання.

Для початку потрібно визначити основні сутності, що будуть взаємодіяти між собою. В даному проекті є 2 головні таблиці: проект та користувач, які будуть між собою взаємодіяти. Усі поля даних таблиць наведено на рисунку 2.2.1

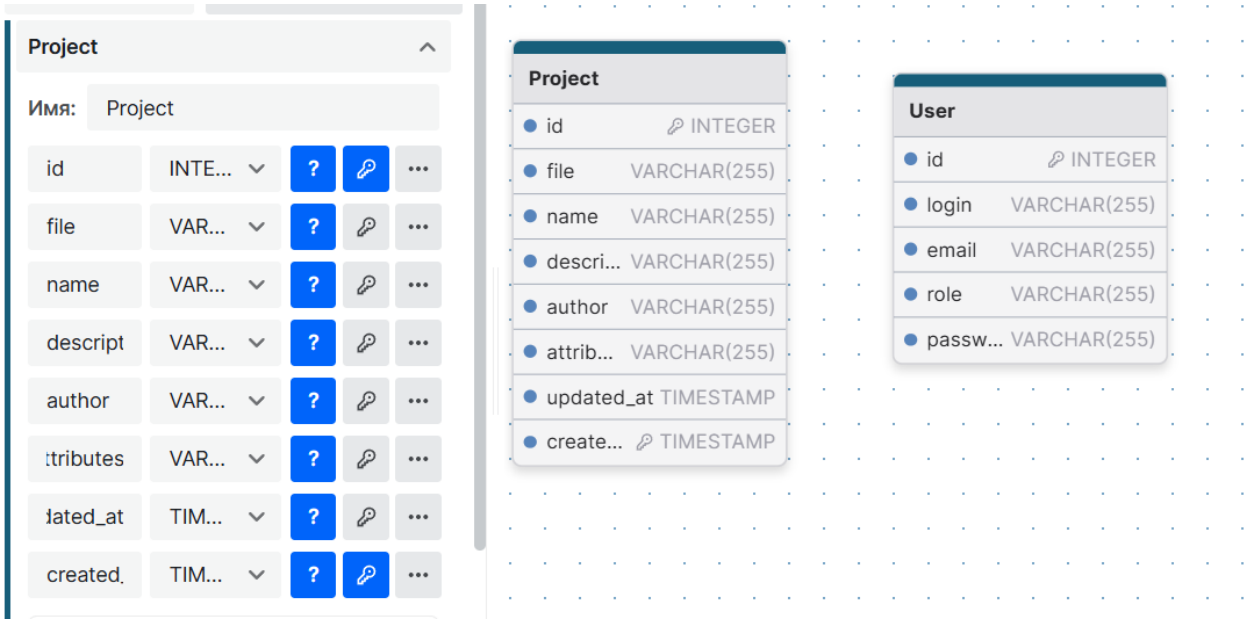


Рисунок 2.2.1 – Головні таблиці з описом полів

Перша нормальна форма передбачає, що всі атрибути в таблиці повинні містити лише неподільні (атомарні) значення. Це означає, що складні поля, такі як «ПІБ», «адреса» та подібні, необхідно розбити на окремі атрибути (наприклад, «ім'я», «прізвище», «по батькові» або «місто», «вулиця», «номер будинку»). Виходячи з цього, можна стверджувати, що наша база даних відповідає першій нормальній формі.

Друга нормальна форма вимагає, щоб база вже перебувала у першій нормальній формі, а також щоб усі неключові атрибути залежали тільки від повного первинного ключа, а не від його частини. Для цього необхідно чітко визначити первинний ключ кожної таблиці та впевнитися, що всі інші атрибути описують саме його

У нашій базі даних виявлено, що деякі поля містять значення, які не залежать безпосередньо від основного ключа, що свідчить про необхідність

створення окремих таблиць. Зокрема, поле «attributes» потребує реалізації у вигляді окремої структури таблиць для забезпечення гнучкої та динамічної системи збереження атрибутів. Для цього створюється три таблиці: «project_entities», «project_entities_attributes» та «project_attributes», які зображені на рисунку 2.2.2. Такий підхід дозволяє уникнути надлишкових даних та підвищити узгодженість інформації у базі.

Опис процесу, за яким буде працювати ця схема: загалом у таблиці project_entities будуть зберігатись усі можливі динамічні атрибути (наприклад сфера застосування). Таблиця project_entity_attributes буде містити усі можливі варіанти для динамічних атрибутів (наприклад сфери застосування – у побуті, у саду, в парку). Третя таблиця project_attributes і є проміжною між усіма проектами та усіма динамічними атрибутами. Тобто умовно буде змережено, що проект X має сферу застосування у побуті, в саду та парку (3 опції існуючого атрибута). Таким чином досягається динамічне створення нових атрибутів, опцій для них, та визначення, які атрибути будуть притаманні певним проектам.

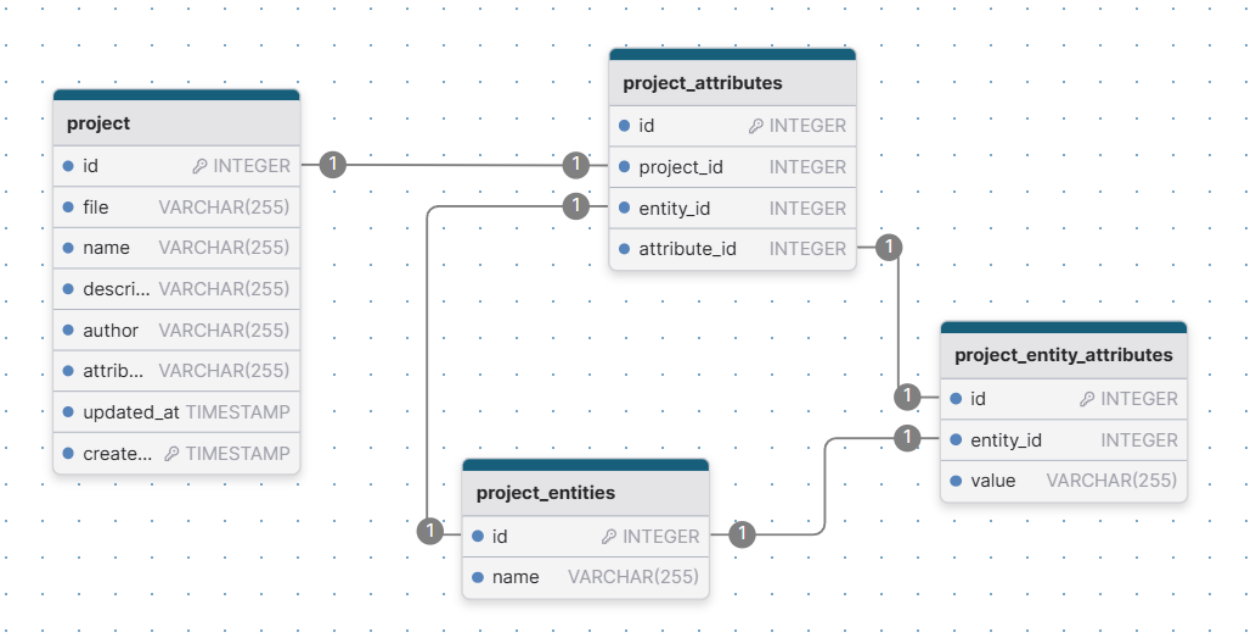


Рисунок 2.2.2 – Організація таблиць для динамічних атрибутів

Для приведення бази даних до другої нормальної форми, необхідно провести декомпозицію таблиці «project» на окремі логічні сутності, такі як

«likes», «comments» та «saved». Такий розподіл дозволяє уникнути надлишкових даних, покращити логіку зберігання інформації та знизити ризик виникнення аномалій під час оновлення, вставки або видалення записів. Кожна з цих сутностей може існувати окремо, без прямої залежності від інших, оскільки вони є частиною взаємодії користувачів із проектами, але вони можуть бути оброблені окремо, а їхня структура буде гнучкою для масштабування.

Розподілення на окремі сутності дає змогу зменшити дублювання даних, полегшити їх обробку та у разі необхідності додавати нові сутності без ускладнення існуючої структури. Завдяки цьому, додавання нових сутностей в майбутньому, таких як «project_category» чи «project_rating», буде значно зручнішим і практичнішим, оскільки всі дані будуть добре структуровані та взаємопов'язані.

Для приведення бази даних до другої нормальної форми, важливо також здійснити декомпозицію таблиці «користувачі», виділивши з неї окрему сутність – «юзер ролі». Такий підхід дозволяє уникнути дублювання даних, забезпечує логічну цілісність інформації та сприяє більш чіткій структуризації. Замість того щоб зберігати роль користувача безпосередньо в основній таблиці, всі ролі виносяться в окрему сутність, яка пов’язана з таблицею «користувачі» через зовнішній ключ. Це дає змогу не лише гнучко керувати правами доступу, але й легко масштабувати систему, наприклад – додавати нові типи ролей без необхідності змінювати структуру основної таблиці.

Таблиця «користувачі» міститиме основну інформацію про користувачів, таку як ім'я, прізвище, електронна пошта. У свою чергу, таблиця «юзер ролі» міститиме інформацію про ролі користувачів, такі як адміністратор, чи звичайний користувач та інші. Таким чином, ролі користувачів будуть описані в окремій таблиці, що дасть змогу уникнути ситуацій, коли одна таблиця містить змішану інформацію про користувачів та їхні ролі, що сприяє збереженню логічної чистоти та зручності в обробці даних. Крім того, це полегшить масштабування системи в майбутньому, якщо буде потрібно додати нові ролі. На рисунку 2.2.3 зображена база, приведена до другої нормальної форми.

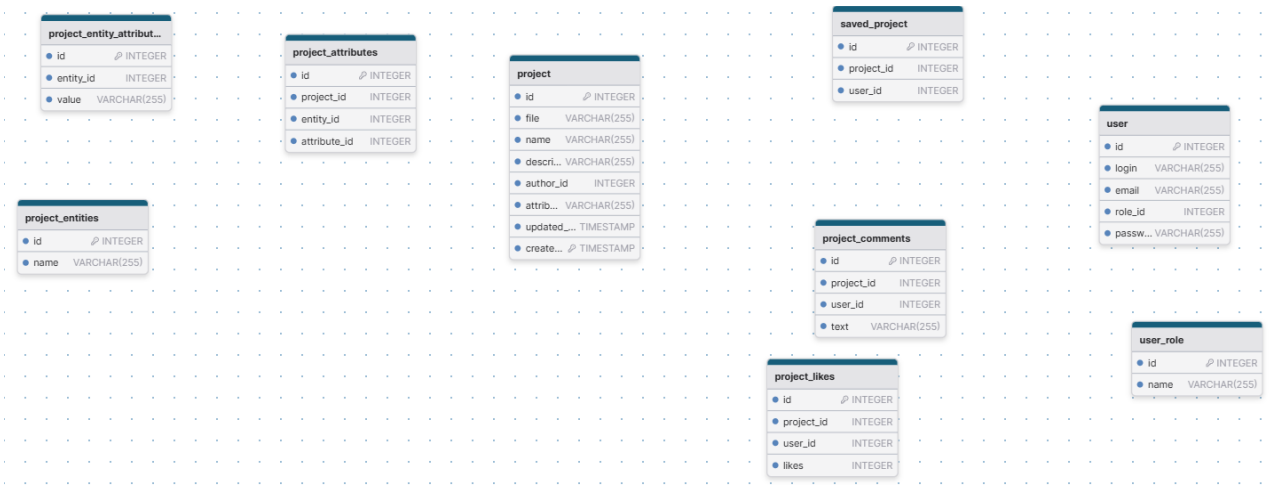


Рисунок 2.2.3 – База даних, що відповідає 2 нормальній формі

Після приведення бази даних до другої нормальної форми наступним кроком є її оптимізація відповідно до третьої нормальної форми. Третя нормальна форма вимагає, щоб база даних вже відповідала другій нормальній формі та щоб усі неключові атрибути були незалежними від інших неключових атрибутів у таблицях. Відповідно до вище описаної діаграми, база уже знаходиться у другій нормальній формі. Тепер необхідно встановити залежності між таблицями. Загалом існує 3 типи залежності: 1 до 1, 1 до n, та n до n. Сформовані взаємозв’язки між таблицями наведені на рисунку 2.2.4

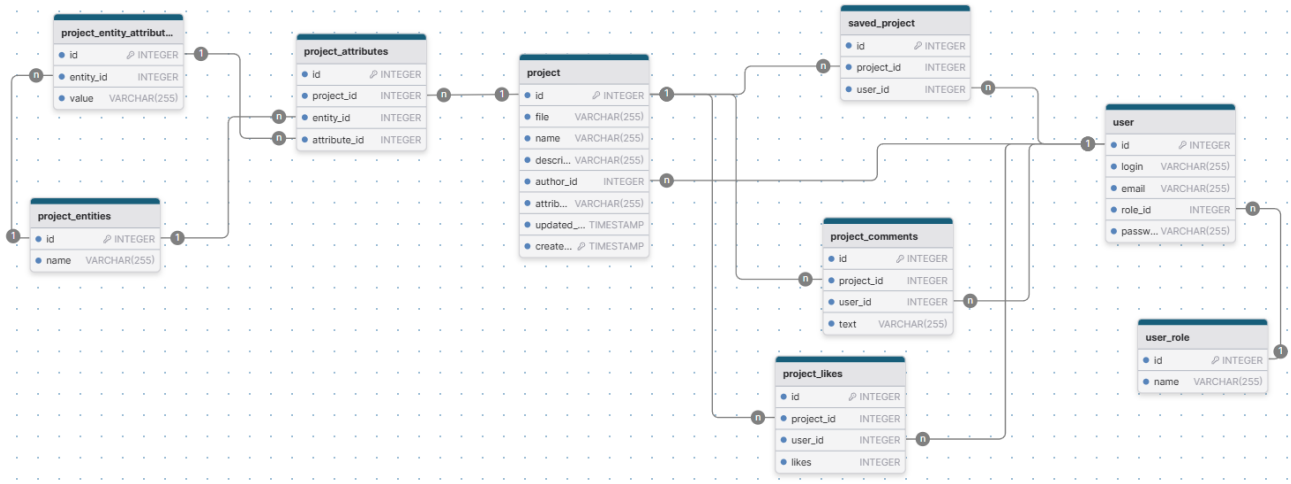


Рисунок 2.2.4 – Сформована діаграма бази даних

Ще однією із структур даних будуть дані, що передаються між клієнтською стороною та сервером. Всі вхідні запити будуть оброблені контролерами з відповідною логікою. На рисунку 2.2.5 зображена блок-схема з описом послідовності обробки запиту.

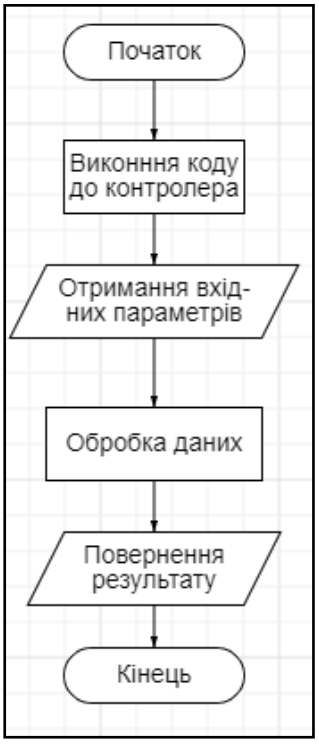


Рисунок 2.2.5 – Блок-схема послідовності обробки вхідних запитів

Контролери отримують глобальні масиви сформовані фреймворком нижчого рівня. Такі масиви мають назви _GET, _POST, _REQUEST, _SESSION. Дані користувача, що надіслав запит зберігаються у останньому масиві. В свою чергу, контролер після обробки логіки буде повертати результуючий масив, який буде використаний у blade файлах з розміткою. Саме такі файли підтримують інтеграцію PHP коду, що забезпечує зручному виводу необхідних даних включно з описом мовою розмітки. Також можна робити відлагодження коду у таких файлах, що полегшує процес розробки, дивлячись в runtime (реальний час виконання коду), які саме дані прилетіли, та як правильно вибрати та відобразити дані з масиву, що сформовані контролером як результуючі дані.

2.3 Проектування інтерфейсу

Правильно розроблений та продуманий інтерфейс відіграє критично важливу роль у залученні та утриманні користувачів. Саме зовнішній вигляд є першим елементом, з яким стикається користувач, і саме він формує початкове враження про всю систему загалом. Інтерфейс не лише передає стиль і характер продукту, а й демонструє рівень уваги до деталей, що безпосередньо впливає на довіру до сервісу. У сучасному цифровому середовищі, де конкуренція серед програмних продуктів надзвичайно висока, навіть функціонально потужний додаток може залишитися непоміченим або бути негативно сприйнятим, якщо його візуальна складова є застарілою, перевантаженою або незручною. Навпаки, навіть обмежений за функціоналом продукт із лаконічним, сучасним і зручним інтерфейсом здатен викликати довіру користувачів і отримати конкурентну перевагу.

У зв’язку з цим питання дизайну та оформлення інтерфейсу не варто розглядати як другорядне. Це один із ключових етапів створення цифрового продукту. Якісний дизайн суттєво впливає на загальне сприйняття і ефективність використання продукту**.** Проте розробка дизайну «з нуля» є трудомістким та ресурсоємним процесом, який вимагає залучення фахівців із UI/UX-дизайну, ретельного прототипування, тестування користувацького досвіду та неодноразового доопрацювання. В умовах обмежених ресурсів або стислих термінів значно раціональнішим рішенням є використання вже готових дизайнерських рішень, що створені професіоналами, перевірені на практиці та адаптовані до потреб різних типів проектів.

На сьогодні існує велика кількість платформ, які пропонують якісні адаптивні шаблони з продуманим користувацьким інтерфейсом. Однією з таких платформ є TemplateMonster - сервіс, що пропонує величезну кількість готових рішень для вебсайтів різного типу (рисунок 2.3.1). Використання таких шаблонів дозволяє не лише скоротити час на верстку та оформлення, але й забезпечити загальну візуальну цілісність, сучасний вигляд і кращу сумісність з різними пристроями.

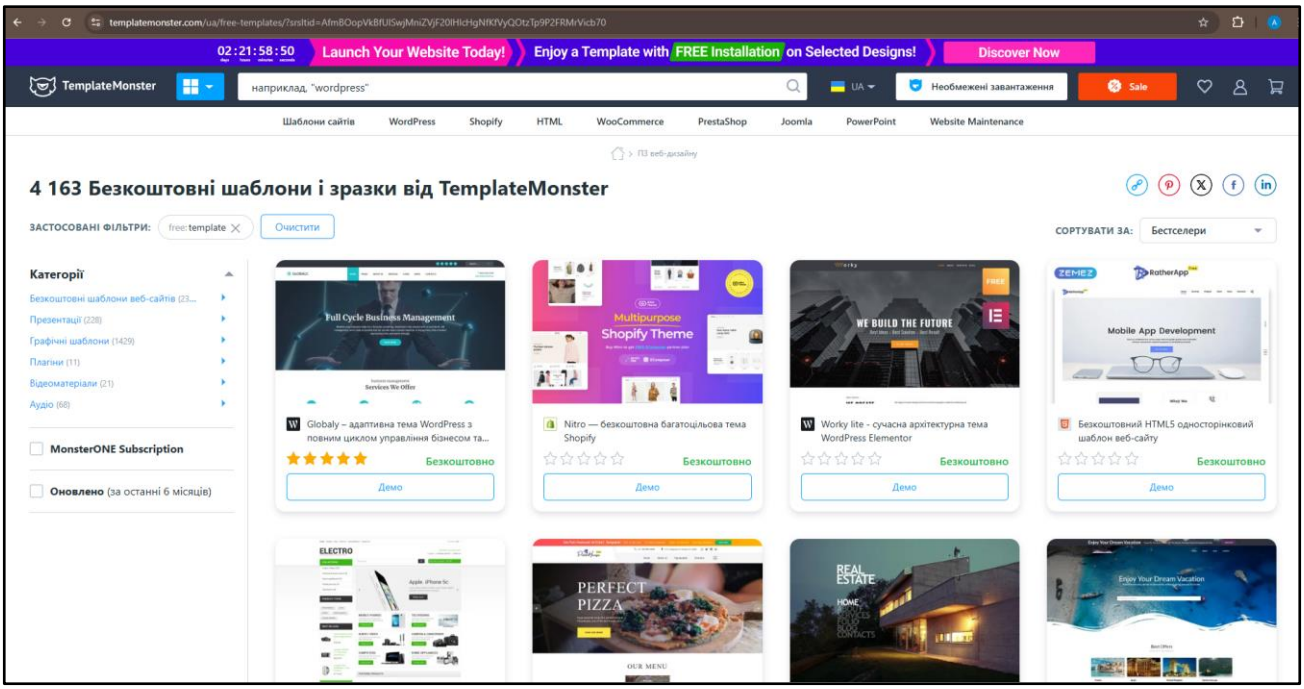


Рисунок 2.3.1- Сервіс TemplateMonster з безкоштовними шаблонами

У рамках даного проєкту було прийнято рішення використати шаблон InsurancePress (рисунок 2.3.2), який забезпечує оптимальний баланс між функціональністю, гнучкістю налаштувань та привабливим дизайном. Цей шаблон включає всі необхідні елементи, що відповідають сучасним тенденціям веб-дизайну: чітку сітку компонування, легку для сприйняття візуальну структуру, адаптивні компоненти, які коректно відображаються на мобільних пристроях. Крім того, шаблон відзначається хорошою оптимізацією продуктивності, швидким часом завантаження та високою сумісністю з популярними браузерами.

Використання професійного шаблону у поєднанні з сучасними технологіями розробки дає змогу створити зручний, привабливий та функціональний інтерфейс, який не потребує розробки з нуля. Це значно скорочує час і ресурси, необхідні для ручного верстання та налаштування, при цьому забезпечуючи високу якість кінцевого продукту. Крім того, використання перевірених шаблонів підвищує надійність інтерфейсу та позитивно впливає на загальний користувацький досвід.

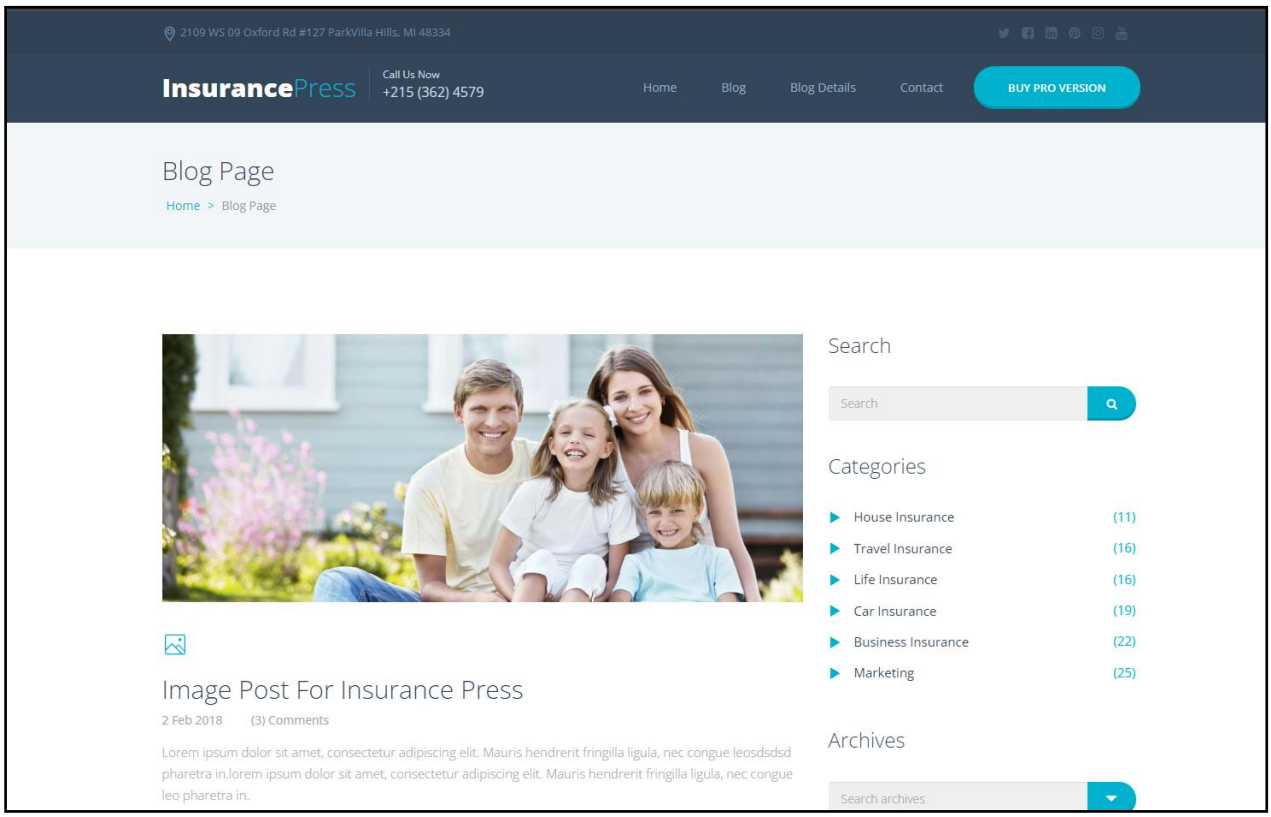


Рисунок 2.3.2 – Шаблонний дизайн для проекту

Як було згадано раніше, весь код верстки проекту зберігатиметься у каталозі resources/views. Тут будуть розташовані файли, які відповідають за відображення всього контенту користувацького інтерфейсу. Ця директорія є основною для роботи з HTML-розміткою, яка в Laravel реалізується за допомогою потужного механізму шаблонізації Blade.

Завдяки цьому механізму можна ефективно організувати структуру HTML-коду на основі принципу повторного використання. Це означає, що розробник може створити один базовий шаблон, у якому централізовано підключаються всі ключові елементи інтерфейсу: стилі оформлення, скрипти (JavaScript), мета-дані, заголовки сторінок, а також компоненти, які повторюються на кожній сторінці, наприклад верхнє меню (navbar), футер, повідомлення про успішні або помилкові дії, бокові панелі. Кожна окрема сторінка проєкту буде наслідувати цей головний шаблон і підставляти свій унікальний контент у відповідні секції, визначені за допомогою Blade-директив. На рисунку 2.3.2 зображено приклад організації шаблонів у структурі проєкту.

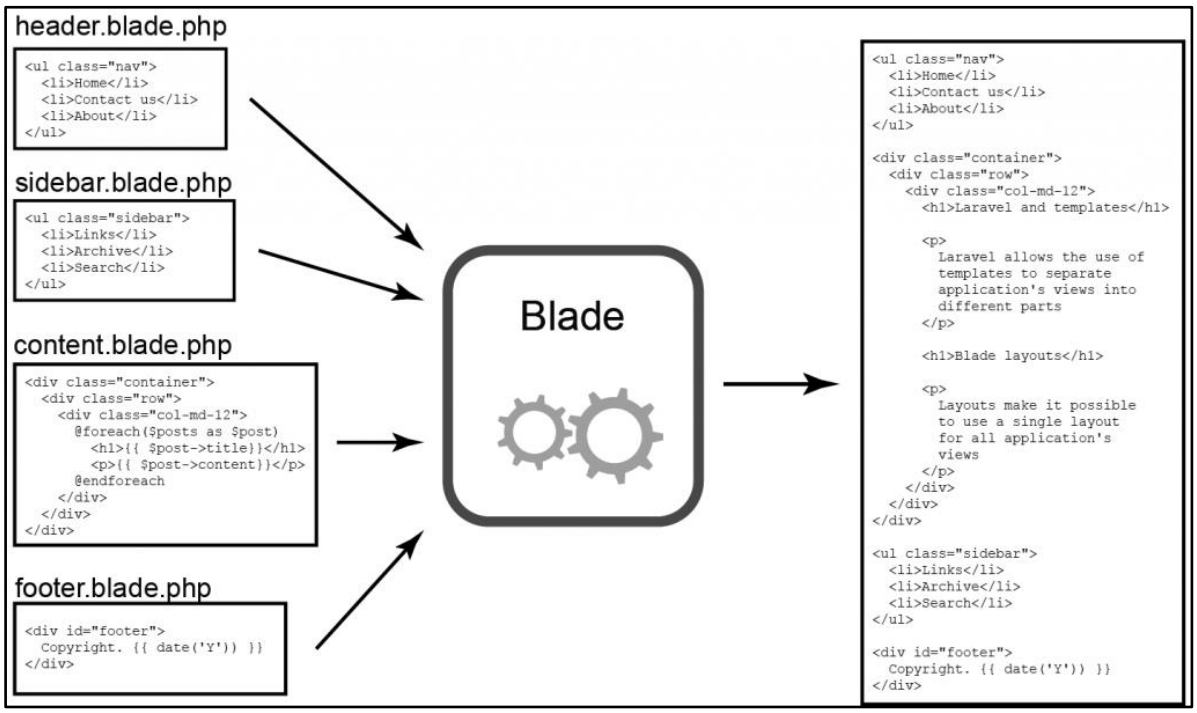


Рисунок 2.3.2 – Організація blade шаблонів

Такий підхід дозволяє суттєво зменшити обсяг дублювання коду, адже кожна нова сторінка (наприклад, сторінка авторизації, профілю користувача, списку проектів або адміністративної панелі) не створюється «з нуля», а успадковує базовий шаблон за допомогою інструкції @extends. У свою чергу, специфічні елементи сторінки (контент, форми, таблиці тощо) додаються через секції (@section) у відповідних підшаблонах, які автоматично підставляються у відповідні місця базового макета.

У контексті даного проекту буде реалізовано наслідування від головного шаблону, у якому, зокрема, буде чітко описано навігаційну панель – верхній блок сайту, що міститиме основні функціональні кнопки (вхід/реєстрація, вихід, додати проект, переглянути збережені об'єкти, доступ до профілю тощо). Така реалізація дозволить уникнути повторного написання однакового коду в кожному шаблоні сторінки, адже navbar буде автоматично відображатися при переході між усіма основними сторінками сайту, забезпечуючи послідовність та єдність візуального стилю. Це ж саме буде стосуватись секції із пагінацією, та футером, де можна розмістити контактну інформацію сайту.

Окрім покращення структури коду, даний підхід значно спрощує процес підтримки та масштабування інтерфейсу. У разі необхідності змінити елемент, що входить до базового шаблону (наприклад, додати нову кнопку або змінити стилі навігаційної панелі), достатньо внести зміни лише в одному файлі, і вони автоматично набудуть чинності для всіх сторінок про проекту. Це не тільки покращує гнучкість розробки, але й мінімізує ризик появи помилок унаслідок неузгоджених змін в інтерфейсі.

У підсумку, використання шаблонної системи Blade у поєднанні з наслідуванням базового шаблону дозволяє будувати структурований, гнучкий та підтримуваний код, що особливо важливо для середніх та великих проєктів з великою кількістю представлень (views).

Однак цей підхід має низку функціональних обмежень у контексті побудови сучасних інтерактивних інтерфейсів. Зокрема, оновлення інформації або взаємодія з динамічними компонентами інтерфейсу відбувається переважно шляхом повного або часткового перезавантаження вебсторінки, що може спричиняти втрату поточного стану, зниження швидкодії застосунку та погіршення загального користувацького досвіду.

Для вирішення цієї проблеми в межах проєктування та реалізації даного вебзастосунку було прийнято рішення про використання сучасного JavaScript-фреймворку Vue.js, який бездоганно інтегрується з Laravel. Завдяки реактивності Vue, розробник має можливість створювати інтерфейси з високим ступенем динаміки, в яких оновлення контенту відбувається без необхідності повного перезавантаження сторінки. Компонентна архітектура Vue дозволяє побудувати гнучку, масштабовану систему, в якій кожен елемент інтерфейсу є самодостатнім блоком із власною логікою та станом. Це значною мірою спрощує супровід та розширення функціоналу в майбутньому.

Окрему увагу в процесі проектування було приділено реалізації клієнт-серверної взаємодії. Для забезпечення обміну даними між інтерфейсом, реалізованим на Vue.js, та серверною частиною, побудованою з використанням Laravel, було обрано популярну бібліотеку Axios. Вона дозволяє надсилати

асинхронні HTTP-запити (GET, POST, PUT, DELETE) до API-серверу, отримувати відповіді та обробляти їх без необхідності оновлення вебсторінки. Це забезпечує плавний та безперебійний користувацький досвід, що є критично важливим у сучасних вебзастосунках, орієнтованих на високий рівень інтерфейсної взаємодії.

Таким чином, поєднання Laravel (на серверній стороні) з Vue.js (на клієнтській стороні) з Axios для комунікації забезпечує ефективну архітектуру вебзастосунку. Такий технологічний стек дозволяє розробляти сучасні, високопродуктивні, адаптивні вебінтерфейси, які відповідають актуальним вимогам до зручності користування, інтерактивності та масштабованості програмного забезпечення.

Для даного проекту необхідно визначити основні сторінки, які потребують розробки інтерфейсу. Однією з ключових є сторінка входу. Враховуючи, що для розширеного використання функціоналу користувач повинен пройти авторизацію або реєстрацію, ця сторінка є критично важливою. Її дизайн повинен бути мінімалістичним, без надлишкових елементів, що можуть відволікати від основного завдання – входу в систему. Простий та зручний інтерфейс дозволить користувачам швидко здійснювати реєстрацію та авторизацію без зайвих складнощів.

Головна сторінка повинна бути добре структурованою, без надмірної кількості графічних елементів, щоб не створювати зайвого інформаційного шуму. Інтерфейс має бути інтуїтивно зрозумілим, щоб користувач одразу розумів, як ним користуватися, та не викликав неоднозначної реакції.

Основний контент головної сторінки – список 3D проектів. Однак, для зручності користувачів та оптимізації завантаження сторінки, не варто відображати всі проекти одночасно. Замість цього необхідно реалізувати пагінацію, яка дозволить розбити контент на окремі сторінки та забезпечить швидке завантаження. Користувачі будуть мати змогу зручно перемикати сторінки та не завантажувати сервер запитамм, які повинні повертати усі проекти за один раз.

Щоб полегшити навігацію та пошук, варто передбачити фільтри та пошукову панель. Вона може бути розташована збоку та містити поля для пошуку за назвою та фільтрації.

Також необхідно продумати верхню навігаційну панель (header), де будуть розміщені основні кнопки для переходу між сторінками. Якщо користувач не авторизований, йому потрібно надати можливість швидкого доступу до сторінок входу та реєстрації. У разі, якщо користувач уже увійшов у систему, йому варто показувати додаткові кнопки для створення нового проекту, перегляду власних проектів та перегляду збережених об’єктів. Це зробить сайт зручнішим та логічнішим у використанні.

Адміністративна панель у межах даного вебзастосунку буде реалізована із використанням розширення Laravel Backpack, яке спеціально призначене для швидкої та зручної побудови адміністративних інтерфейсів у Laravel-додатках. Обрана технологія надає розробникам широкі можливості для конфігурування та персоналізації панелі керування без потреби у створенні складної логіки з нуля, що суттєво пришвидшує процес розробки й водночас гарантує високу якість кінцевого продукту.

На лівій боковій панелі буде розміщено навігаційне меню зі списком усіх основних сутностей системи, таких як користувачі, проекти, коментарі, лайки, ролі тощо. Кожна сутність буде представлена окремим пунктом меню, який відкриватиме відповідний CRUD-інтерфейс. Це дозволить адміністраторам швидко переходити до потрібного розділу та здійснювати необхідні дії над даними без зайвих переходів між сторінками.

Центральна частина інтерфейсу динамічно змінюватиметься відповідно до вибраної сутності. Тут буде відображено список записів з таблиці бази даних, представлених у вигляді зручної табличної структури з можливістю сортування, фільтрації та пошуку. Кожен запис буде мати кнопки керування (перегляд, редагування, видалення), що дасть змогу здійснювати всі необхідні адміністративні операції максимально ефективно та без додаткових навігаційних дій.

Більше того, Laravel Backpack дозволяє легко додавати додаткові елементи, такі як графіки, індикатори активності, віджети та розширені панелі фільтрації. Це створює умови для побудови розумного, адаптивного інтерфейсу, що зможе масштабуватись відповідно до зростання складності проекту та потреб адміністраторів.

У результаті, завдяки використанню Laravel Backpack, адміністративна частина системи буде зручною, інтуїтивно зрозумілою та придатною до масштабування. Вона ефективно виконає роль центру керування всіма об’єктами системи, забезпечуючи адміністраторам повний контроль над вмістом та користувачами застосунку.

Завдяки механізму розширення файлів можна створити основний (базовий) шаблон, у якому будуть підключені всі необхідні стилі, скрипти та загальні елементи інтерфейсу. При цьому верстку кожної окремої сторінки можна організувати у відповідних файлах, які будуть наслідувати цей шаблон.

У даному проекті буде реалізовано наслідування від базового файлу, в якому буде описано navbar – верхню панель з усіма функціональними кнопками. Це означає, що в кожному шаблоні сторінки не потрібно повторно створювати цей елемент, і він автоматично буде присутній при переході між сторінками. Такий підхід дозволить значно спростити підтримку коду та забезпечить уніфікований стиль інтерфейсу.

У даному розділі було здійснено повноцінне проектування програмного продукту, була визначена його архітектурна модель, структура бази даних, а також зовнішній вигляд користувацького інтерфейсу. Враховуючи особливості майбутньої системи та вимоги до її функціональності, було обрано клієнт-серверну архітектуру як найбільш оптимальну для реалізації цього проекту. Описана взаємодія клієнтської та серверної частин, які технології будуть використані та які інструменти допоможуть зробити розробку швидшою і зручнішою. Також спроектована база даних. Був обраний шаблон для інтерфейсу, що забезпечить зручне використання та розробку візуальних частин.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Програмна реалізація архітектури та бази даних

Першим етапом у розробці є створення базової структури даних, тобто організація бази даних. Для цього необхідно мати встановлену систему управління базами даних (СУБД). У даному випадку вибір зупинився на MySQL, яка є однією з найпопулярніших і надійних СУБД.

Наступним кроком є встановлення з'єднання з базою даних. Процес підключення відображений на рисунку 3.1.1. Саме через правильне під'єднання забезпечується можливість подальшого створення таблиць, збереження та обробки даних у системі.

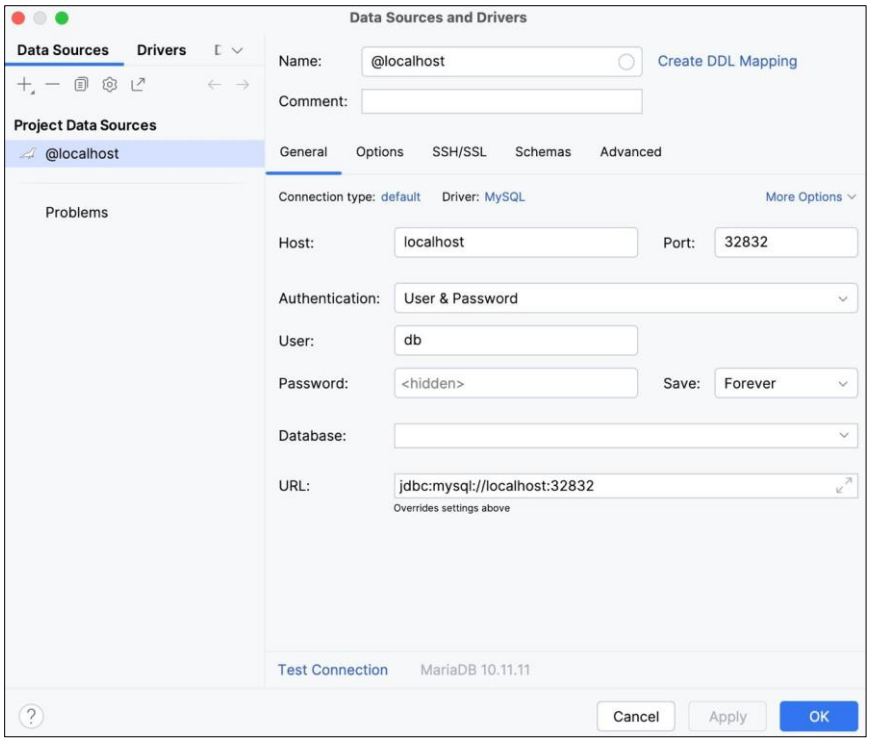


Рисунок 3.1.1 – Процес підключення до бази даних

Після створення власної бази даних наступним важливим кроком є інтеграція її з програмним проєктом. Для цього необхідно налаштувати параметри підключення до бази даних.

Як показано на рисунку 3.1.2, у кореневій директорії проекту потрібно відкрити файл конфігурації .env і внести відповідні дані для підключення. Зокрема, слід вказати назву бази даних, ім’я користувача, пароль, хост та порт. Це забезпечить коректну взаємодію між застосунком і базою даних на всіх етапах роботи.

```
23  DB_CONNECTION="mysql"
24  DB_HOST="db"
25  DB_PORT="3306"
26  DB_DATABASE="db"
27  DB_USERNAME="db"
28  DB_PASSWORD="db"
```

Рисунок 3.1.2 – Конфігурація для підключення проекту до бази даних

Після того як база даних була налаштована та підключена до проекту, наступним кроком є створення необхідних таблиць для зберігання даних.

У фреймворку Laravel процес створення таблиць здійснюється за допомогою міграцій. Для цього потрібно згенерувати нову міграцію, яка автоматично створює відповідний файл у директорії database/migrations.

У цьому файлі визначаються всі необхідні поля таблиці, їх типи, обмеження та залежності, зокрема налаштовуються зовнішні ключі для забезпечення цілісності даних.

На рисунку 3.1.3 продемонстровано приклад створення міграції для таблиці Project, де детально описані всі необхідні поля та їх властивості.

Аналогічним способом необхідно створити й усі інші міграції для решти таблиць бази даних, що будуть використовуватись у проєкті. Після підготовки всіх файлів міграцій потрібно виконати команду artisan migrate (рисунок 3.1.4). Вона застосує всі створені міграції та фізично сформує відповідні таблиці у базі даних.

```php
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    Oleksii Tsehelnyk
    public function up(): void
    {
        Schema::create( table: 'project', function (Blueprint $table) {
            $table->bigIncrements( column: 'id');
            $table->string( column: 'image')->default( value: null)->nullable();
            $table->string( column: 'file')->default( value: null)->nullable();
            $table->string( column: 'name');
            $table->longText( column: 'description')->default( value: null)->nullable();
            $table->unsignedBigInteger( column: 'editor_id')->default( value: null)->nullable();
            $table->timestamps();

            $table->foreign( columns: 'editor_id')
                ->references( columns: 'id')
                ->on( table: 'users')
                ->onUpdate( action: 'cascade');

        });
    }
```

Рисунок 3.1.3 – Створення міграції для таблиці Project

```
Terminal  Local  Local (2)  Local (3)
aleks@Oleksiis-Laptop Diploma % ddev artisan migrate

  INFO  Running migrations.

  0001_01_01_000001_create_cache_table ..... 33.25ms DONE
  0001_01_01_000002_create_jobs_table ..... 50.23ms DONE
  2025_04_25_132339_create_user_role_table ..... 13.96ms DONE
  2025_04_25_132340_create_users_table ..... 502.12ms DONE
  2025_04_25_132345_create_project_table ..... 137.38ms DONE
  2025_04_25_132402_create_saved_project_table ..... 513.81ms DONE
  2025_04_25_132427_create_project_comment_table ..... 102.15ms DONE
  2025_04_25_132446_create_project_like_table ..... 73.71ms DONE
  2025_04_25_132506_create_project_entities_table ..... 23.02ms DONE
  2025_04_25_132520_create_project_entity_attributes ..... 148.86ms DONE
  2025_04_25_132526_create_project_attributes_table ..... 401.67ms DONE

aleks@Oleksiis-Laptop Diploma % ddev artisan migrate --seed

  INFO  Nothing to migrate.

  INFO  Seeding database.

aleks@Oleksiis-Laptop Diploma %
```

Рисунок 3.1.4 – Міграція усіх таблиць та заповнення через seed

Після успішного створення таблиць наступним етапом є заповнення бази первинними (статичними) даними. Для цього у Laravel використовуються seeder. Ми створимо спеціальний клас, який автоматично додасть необхідні початкові записи. Зокрема, буде створено обліковий запис адміністратора системи.

Таким чином, завершуються основні етапи налаштування та створення бази даних для проекту, і можна переходити до подальшої роботи з даними.

Безпосередня робота з базою даних у фреймворку Laravel здійснюється за допомогою вбудованого механізму ORM (Object-Relational Mapping). Завдяки ORM можна взаємодіяти з таблицями бази даних через об'єкти, що значно спрощує синтаксис запитів та підвищує читабельність коду.

Для передачі отриманих із бази даних даних у шаблон (view), у Laravel часто використовується спеціальний метод compact(). Цей метод дозволяє передати змінні у представлення, обгортаючи їх у зручний асоційований масив. Передавати можна як окремі змінні, так і цілі масиви або об’єкти. У випадку, коли потрібно передати кілька змінних одночасно, їх імена перераховуються через кому у виклику compact().

Такий підхід дозволяє ефективно організувати обмін даними між контролерами та представленнями, роблячи код більш зрозумілим і легким для підтримки.

```
$projects = DB::table('project')->get()->all();
return view('create_project', compact('projects'));
```

Тепер розглянемо, яким чином форма на стороні в’ю отримує дані з бекенда, а також як забезпечується їх динамічне відображення у відповідних елементах інтерфейсу. Цей процес є ключовим для забезпечення інтерактивності та узгодженості даних між сервером і користувачем:

```
<div class="col-sm-8">
        @foreach($projects as $key => $project)
            <form method="get" action="/favourite-project-id-{{$project-
>id}}">
                <div class="single-post">
                    <div class="blog-img">
                        <a href="">
                            <img src="{{$project->image}}" class="img-
responsive imagerozmir">
                        </a>
                    </div>
                    <h2 class="blog-title">{{$project->name}}</h2>
                    <div class="blog-btn">
                        <button type="submit" class="btn-
default">Дізнатись більше</button>
                    </div>
                </div>
            </form>
        @endforeach
    @endif
</div>
```

Передача даних здійснюється через контролер, який надсилає змінні або об'єкти до шаблону. Завдяки цьому форма може автоматично підставляти актуальні значення, наприклад, заповнювати випадаючі списки, встановлювати початкові значення полів або динамічно генерувати елементи інтерфейсу на основі отриманої інформації.

3.2 Програмна реалізація інтерфейсу та логіки роботи проекту

Процес написання коду займає основну частину роботи над проектом. Щоб краще зрозуміти логіку функціонування системи, розглянемо приклад організації авторизації користувача.

Першим кроком необхідно додати новий маршрут у файл routes/web.php. Саме тут визначаються відповідні шляхи та обробники, які відповідатимуть на запити користувачів. У маршруті потрібно вказати клас-контролер, який буде відповідати за обробку запиту, а також конкретний метод цього класу, який буде виконуватись при зверненні за заданою адресою. Це дозволяє чітко розділяти логіку обробки різних запитів і підтримувати структуру коду зрозумілою.

```
Route::get('/login', [\App\Http\Controllers\Login\LoginController::class, 'getLoginPage']);
```

Наступним кроком необхідно описати логіку самого контролера, який обробляє запит, надісланий з форми. Контролер виконує роль посередника між клієнтським інтерфейсом і бізнес-логікою за стосунку. Він приймає вхідні дані, перевіряє їхню коректність, взаємодіє з моделями та передає результат назад до представлення.

```
public function getLoginPage()
{
    session_start();
    return view('login');
}
```

В даному прикладі спочатку іде запуск сесії, а далі повертаємо для користувача відповідну view форму, щоб юзер ввів свої дані. Нижче наведений код view:

```
      <div class="col-md-8 col-md-offset-2">
          <div class="form-box">
              <h2>Увійти</h2>
              <div class="form-content">
                  <form method="post" action="/api/login">
                      <input type="text" name="email" placeholder="Email"
required/>
                      <input type="password" name="pass" placeholder="Пароль"
required/>
                      <div class="text-center">
                          <input type="submit" class="btn-default"
value="Увійти" />
                      </div>
                  </form>
              </div>
          </div>
      </div>
```

На сторінці форми користувач має можливість ввести свої дані для авторизації, зазвичай це електронна пошта та пароль. Кожен тег input має атрибут name, який визначає ключ для передачі відповідного значення на сервер. Це забезпечує коректне зчитування полів на стороні бекенду.

Після натискання кнопки надсилання форма автоматично відправляється на сервер для подальшої обробки. У цей момент усі введені користувачем дані передаються методом POST за визначеним маршрутом /api/login. Цей маршрут приймає запит і передає його у відповідний контролер, який відповідає за логіку авторизації користувача.

Визначення самого маршруту здійснюється у файлі routes/api.php. Саме в цьому файлі прописується, який контролер і який саме метод цього контролера мають бути викликані для обробки запиту. Такий підхід дозволяє чітко структурувати код, забезпечити зручність у його супроводі та гарантувати безпечну й надійну взаємодію клієнта з сервером.

```
      Route::post('login',
[\App\Http\Controllers\Login\Post\LoginController::class, 'login']);
```

Код контролера, який приймає вхідний аргумент $request, відповідає за обробку надісланих із форми даних наведено нижче. У цьому методі відбувається витягування параметрів, переданих користувачем, таких як електронна пошта (або логін) та пароль. Після отримання цих значень контролер виконує перевірку: чи існує у базі даних користувач із вказаними обліковими даними, а також чи правильний введений пароль.

```
public function login(Request $request)
{
    try {
        $user = User::whereEmail($request['email'])->first();
        if ($user->exists){
            if (Hash::check($request['pass'], $user->password)){
                session_start();
                $_SESSION["is_auth"]= $user->id;
                return Redirect('/');
            }
        }
        throw new Exception('Неправильні дані входу!');
    } catch (Exception $exception){
        return Redirect('login');
    }
}
```

Після надходження запиту запускається сесія, яка дозволяє зберігати інформацію про стан користувача між запитами. За допомогою ORM виконується перевірка наявності користувача в базі даних. Якщо запис про користувача відсутній, його перенаправить назад на сторінку авторизації. У випадку, якщо користувача знайдено, з об’єкта $request витягується параметр pass. Цей пароль проходить обробку через механізм хешування, передбачений фреймворком Laravel. Отриманий хеш порівнюється з уже збереженим хешем пароля в базі даних. Якщо хешовані значення збігаються, у сесію записується інформація про те, що користувач успішно автентифікований, після чого його перенаправляють на головну сторінку сайту.

Тепер перейдемо до детального опису наших макетів. Як вже згадувалося раніше, у фреймворку Laravel реалізована можливість наслідування шаблонів однієї сторінки від іншої за допомогою Blade-шаблонізатора. Це значно полегшує процес розробки вебзастосунків, оскільки дозволяє уникнути дублювання коду, забезпечити централізоване керування структурою сторінок та швидко вносити зміни, що автоматично застосовуються до всіх дочірніх шаблонів.

У проекті використовується головний макет, який включає в себе навігаційну панель (navbar) з розташованими на ній кнопками для переходу між різними сторінками сайту. В центральній частині макету розміщено спеціальний фрагмент коду - @yield('content'). Ця конструкція позначає місце, куди буде підставлятися контент із шаблонів-нащадків. Для підключення шаблону-нащадка використовується наступний синтаксис:

```
@extends('index')
@section('content')
// унікальний вміст сторінки
@endsection
```

Тобто, за допомогою директиви @extends('index') ми вказуємо, що даний файл успадковує головний шаблон index.blade.php, а в секції @section('content') розміщуємо конкретний контент, який підставиться у визначене місце макету. Завдяки такому підходу можна змінювати лише зміст сторінок, залишаючи незмінними спільні елементи інтерфейсу, наприклад, навігаційну панель.

Крім того, Blade-шаблонізатор у Laravel підтримує зручні конструкції для роботи з даними. Наприклад, для організації циклів можна використовувати структуру:

```
@foreach ($items as $item)
    // Дії з кожним елементом масиву
@endforeach
```

Ця конструкція дозволяє перебирати масив даних, що був переданий з бекенду, і виконувати певні операції з кожним елементом окремо. Для умовних перевірок використовується синтаксис:

```
@if (умова)
    // Код, що виконується, якщо умова істинна
@endif
```

Синтаксис конструкцій Blade дуже схожий на звичайні PHP-оператори, що значно полегшує адаптацію для розробників, які вже мають досвід роботи з класичним PHP. Завдяки простоті та інтуїтивності Blade-шаблонізатора, створення динамічних інтерфейсів стає набагато швидшим і зручнішим.

Для виводу змінних або інших даних усередині шаблонів, як у циклах, так

і поза ними, використовуються подвійні фігурні дужки {{ ... }}. Вони автоматично екранують небезпечний HTML-код, що допомагає уникнути XSS-уразливостей без додаткових зусиль з боку розробника. Наприклад:

```
@foreach($projects as $key => $project)
    @if($key >= $minkey && $key <= $maxkey)
        <form method="get" action="project-id-{{$project->id}}">
            <div class="single-post">
                <div class="blog-img">
                    <a href="">
                        <img src="{{$project->image}}" class="img-
responsive imagerozmir">
                    </a>
                </div>
                <h2 class="blog-title">{{$project->name}}</h2>
                <div class="blog-btn">
                    <button type="submit" class="btn-default">Дізнатись
більше</button>
                </div>
            </div>
        </form>
        @endif
@endforeach
```

У цьому прикладі відбувається перебір усіх елементів масиву, який містить список проектів, після чого дані кожного з них виводяться у відповідних HTML-тегах. Це дає змогу формувати динамічний вміст сторінки на основі переданих з бекенду даних, не вдаючись до зайвого дублювання коду.

Таким чином, використовуючи можливості Blade-шаблонізатора, можна створювати гнучкі, масштабовані та зручні у підтримці шаблони для вебзастосунків. Це особливо важливо для великих проєктів із великою кількістю однотипних сторінок, адже дозволяє централізовано керувати їх структурою, стилем і вмістом.

Тепер перейдемо до розгляду реалізації компонентів на базі Vue.js. Компоненти у Vue дозволяють створювати багаторазові частини інтерфейсу, що значно спрощує структурування та використанні у коді, а також дає змогу динамічно працювати із виведеним контентом на сторінці браузера. Нижче, на

рисунку 3.2.1 наведено приклад базового фрагмента Vue-компонента для виведення секції оцінювання від 1 до 5 зірочок та динамічним надсиланням запиту для встановлення оцінки проекту:

```
<template>  Show component usages
    <div class="rating-area">
        <input type="radio" id="star-5" name="rating" value="5" @click="setLike($event)">
        <label for="star-5" title="Оцінка «5»"></label>

        <input type="radio" id="star-4" name="rating" value="4" @click="setLike($event)">
        <label for="star-4" title="Оцінка «4»"></label>

        <input type="radio" id="star-3" name="rating" value="3" @click="setLike($event)">
        <label for="star-3" title="Оцінка «3»"></label>

        <input type="radio" id="star-2" name="rating" value="2" @click="setLike($event)">
        <label for="star-2" title="Оцінка «2»"></label>

        <input type="radio" id="star-1" name="rating" value="1" @click="setLike($event)">
        <label for="star-1" title="Оцінка «1»"></label>
    </div>
</template>

<script>
import axios from "axios";

no usages  new *
export default {
    name: "setLikesComponent",
    props: ['user_id', 'likes', 'project_id', 'src'],
    mounted() {
        if (this.user_id == 0) {
            document.getElementById( elementId: 'star-5').disabled = true
            document.getElementById( elementId: 'star-4').disabled = true
            document.getElementById( elementId: 'star-3').disabled = true
            document.getElementById( elementId: 'star-2').disabled = true
            document.getElementById( elementId: 'star-1').disabled = true
        } else {
            switch (this.likes) {
                case '1': document.getElementById( elementId: 'star-1').checked = true
                    break;
                case '2': document.getElementById( elementId: 'star-2').checked = true
                    break;
                case '3': document.getElementById( elementId: 'star-3').checked = true
                    break;
                case '4': document.getElementById( elementId: 'star-4').checked = true
                    break;
                case '5': document.getElementById( elementId: 'star-5').checked = true
                    break;
            }
        }
    },
    methods: {
        async setLike(event) {
            await axios.post( url: 'api/set-likes',  data: {'user_id':this.user_id, 'id':this.project_id, 'likes': event.target.value})
        }
    }
}
</script>

<style scoped>

</style>
```

Рисунок 3.2.1 – Приклад Vue компонента setLikes

Компонент у Vue.js складається з трьох основних секцій: template, script та style. У блоці <template> описується структура майбутнього інтерфейсу,

використовуючи розмітку HTML. Саме тут створюються елементи, які будуть безпосередньо відображатися на сторінці. Цей шаблон забезпечує візуальне представлення даних, а також може містити директиви Vue для динамічного відображення інформації або взаємодії з користувачем.

У секції <script> визначається логіка роботи компонента. В даному випадку використовується бібліотека axios, яка відповідає за асинхронну передачу даних між фронтендом і бекендом через HTTP-запити. Також у цій частині коду задається ім'я компонента через властивість name, що дозволяє однозначно ідентифікувати його в проекті. Додатково в цій частині описується масив props. Цей об'єкт вказує, які властивості можуть бути передані в компонент зовні:

```
<set-likes user_id="{{$user_is_exists}}" likes="{{$likes}}" project_id="{{$id}}">
</set-likes>
```

Завдяки props можна налаштовувати поведінку та вміст компонента без необхідності змінювати його внутрішній код, що значно підвищує гнучкість і повторне використання компонентів.

Таким чином, поділ компонента на три окремі частини – розмітку, логіку та стилі – дозволяє зручно організувати код і спрощує розробку масштабованих веб-застосунків на базі Vue.js.

Щоб компонент успішно працював у проекті, його необхідно обов’язково зареєструвати у файлі resources/js/app.js. Саме в цьому файлі відбувається імпорт та глобальна реєстрація всіх Vue-компонентів, що дозволяє використовувати їх у різних частинах застосунку, зокрема безпосередньо у Blade-шаблонах або інших Vue-компонентах. Такий підхід забезпечує централізоване керування компонентами та їхню доступність по всьому проекту.

Приклад реєстрації компонентів наведено на рисунку 3.2.2. Наступні компоненти зареєстровані: додавання коментарів, лайків, динамічне додавання сутностей проектів та атрибутів, а також збереження проекту в улюблені.

```
require('./bootstrap');

window.Vue = require('vue').default;

/**
 * The following block of code may be used to automatically register your
 * Vue components. It will recursively scan this directory for the Vue
 * components and automatically register them with their "basename".
 *
 * Eg. ./components/ExampleComponent.vue -> <example-component></example-component>
 */

Vue.component('new-attribute-add', require('./components/AddAttributeComponent.vue').default);
Vue.component('new-entity-add', require('./components/AddEntityComponent.vue').default);
Vue.component('comments', require('./components/CommentsComponent.vue').default);
Vue.component('add-favourite', require('./components/AddToFavouriteComponent.vue').default);
Vue.component('set-likes', require('./components/setLikesComponent.vue').default);

/**
 * Next, we will create a fresh Vue application instance and attach it to
 * the page. Then, you may begin adding components to this application
 * or customize the JavaScript scaffolding to fit your unique needs.
 */

const app = new Vue({
    el: '#app',
});
```

Рисунок 3.2.2 – Реєстрація Vue компонентів

Тепер розглянемо процес створення адмін-панелі. Для того щоб мати можливість виводити різні сутності в адміністративній частині сайту, необхідно виконати кілька послідовних кроків. Перш за все, потрібно встановити Backpack пакет як додаткове розширення, командами:

```
composer require --dev laracasts/generators
composer require backpack/crud
php artisan backpack:install
```

Далі у всіх створених моделях у директорії app/Models потрібно створити модель відповідної сутності. Модель у Laravel виступає проміжною ланкою між базою даних та логікою програми, дозволяючи зручно взаємодіяти з таблицями та записами у базі. Також модель повинна містити використання CrudTrait, і прописати правильно залежності між сутностями. Нижче наведено опис взаємозалежностей сутностей юзера юзер ролі та проекту, використовуючи HasMany та BelongsTo. Модель користувача:

```
    /**
     * @return BelongsTo
     */
    public function userRole(): BelongsTo
    {
        return $this->belongsTo(UserRole::class, self::ROLE_ID);
    }
    /**
     * @return HasMany
     */
    public function projects(): HasMany
    {
        return $this->hasMany(Project::class);
    }
```

Тепер потрібно згенерувати CRUD для сутності. Це можливо використовуючи команду backpack:crud. На рисунку 3.2.3 – 3.2.4 зображено процес генерації усіх необхідних класів

```
php artisan backpack:crud entity
```

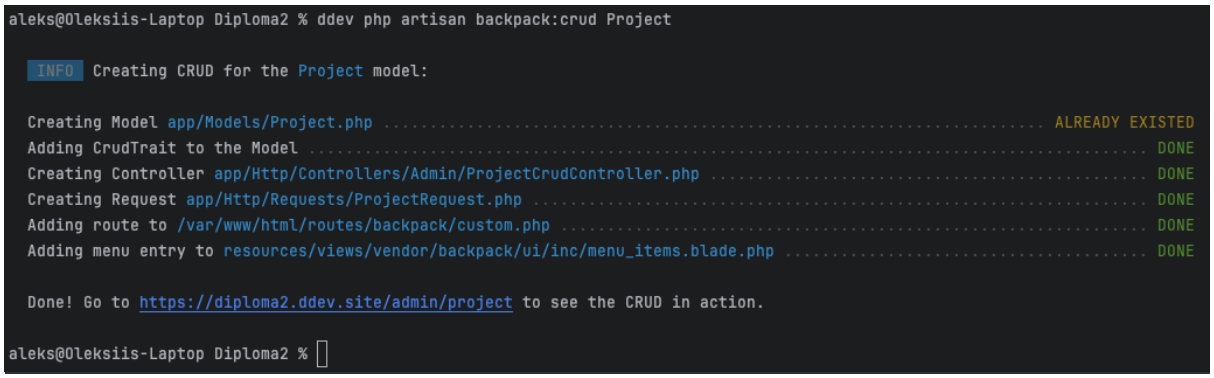


Рисунок 3.2.3 – Генерація CRUD для проектів

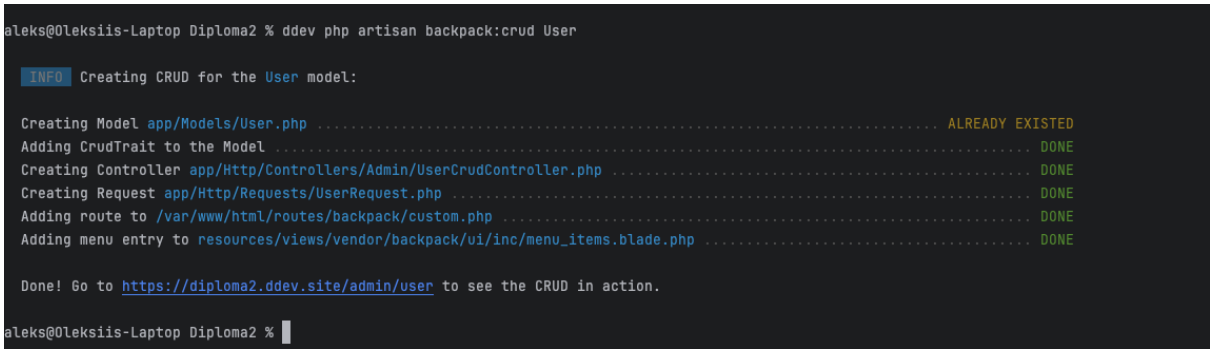


Рисунок 3.2.4 – Генерація CRUD для користувачів

Коли усі класи згенеровані, а саме route, blade, Controller та Request, можна описати UserCrudController, який лежить у директорії /app/Http/Controllers/Admin. В даному класі буде описано які поля потрібно витягувати з бази, які залежності від суміжних сутностей для правильного відображення даних:

```
public function setup()
{
CRUD::setModel(\App\Models\User::class);
CRUD::setRoute(config('backpack.base.route_prefix') . '/user');
CRUD::setEntityNameStrings('user', 'users');
}

/**
* @return void
*/
protected function setupListOperation()
{
CRUD::setFromDb(); // set columns from db columns.
$this->crud->addColumn('userRole');
}

/**
* @return void
*/
protected function setupCreateOperation()
{
CRUD::setValidation(UserRequest::class);
CRUD::setFromDb(); // set fields from db columns.
$this->crud->addField([
'name'       => 'role_id',
'label'      => 'Role',
'type'       => 'select',
'entity'     => 'userRole',
'model'      => \App\Models\UserRole::class,
'attribute'  => 'role',
'options'    => fn ($query) => $query->orderBy('role')->get(),
]);
}
```

```
    protected function setupShowOperation()
    {
        $this->setupListOperation();
    }
    /**
     * @return void
     */
    protected function setupUpdateOperation()
    {
        $this->setupCreateOperation();
        $this->crud->addField([
            'name'      => 'role_id',
            'label'     => 'Role',
            'type'      => 'select',
            'entity'    => 'userRole',
            'model'     => \App\Models\UserRole::class,
            'attribute' => 'role',
            'options'   => function ($query) {
                return $query->orderBy('role', 'ASC')->get();
            },
        ]);
    }
```

Після правильного налаштування всіх сутностей, при переході за адресою /admin відкриється адміністративний інтерфейс. У ньому будуть відображатися всі створені сутності у вигляді списку, а також доступні функціональні кнопки для здійснення основних CRUD-операцій (створення, читання, оновлення, видалення).

Таким чином, адмін-панель на базі Laravel Backpack надає потужні інструменти для зручного керування даними без необхідності додаткового написання великої кількості коду. На цьому етапі можна вважати процес розробки даного програмного продукту завершеним.

3.3 Тестування

Тепер перейдемо до етапу тестування додатку. Важливо зазначити, що модульне та інтеграційне тестування є невід’ємною частиною процесу розробки програмного забезпечення. Вони дозволяють переконатися у правильності

роботи окремих функцій та їх невеликих об'єднань ще до переходу до фінальної стадії розгортання системи.

Модульне тестування (unit testing) спрямоване на перевірку роботи окремих функціональних частин коду в ізольованому середовищі. Це дозволяє оперативно виявляти та виправляти помилки на ранніх стадіях розробки. Інтеграційне тестування (integration testing) у свою чергу перевіряє взаємодію між різними модулями системи, що забезпечує впевненість у їхній коректній спільній роботі.

Завдяки регулярному тестуванню забезпечується висока якість коду, мінімізується кількість помилок у продуктивному середовищі та підвищується загальна надійність програмного продукту.

На рисунках 3.3.1 – 3.3.2 наведено приклад тестування вхідних параметрів при створенні нового проекту. Це тестування дозволяє переконатися, що дані, які передаються у систему, відповідають необхідним вимогам валідації, і що додавання нової сутності відбувається коректно.

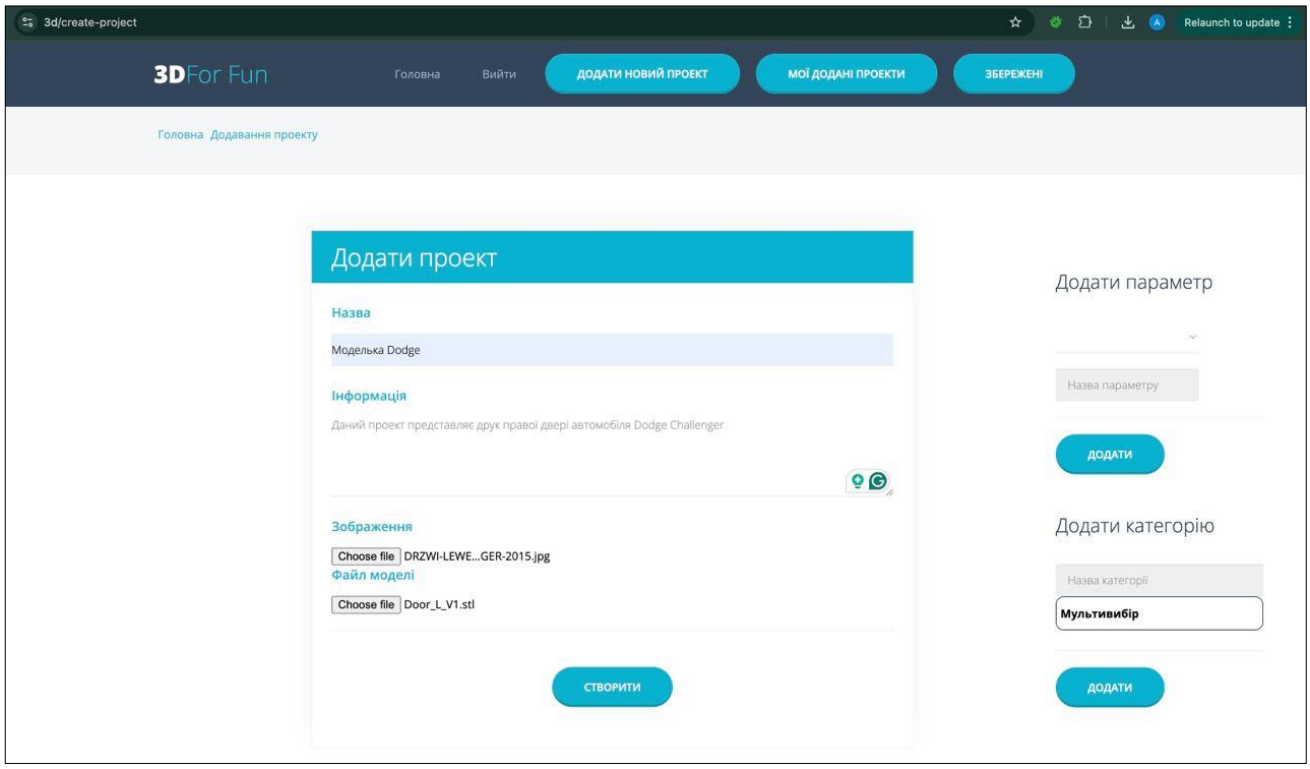


Рисунок 3.3.1 – Додавання нового проекту

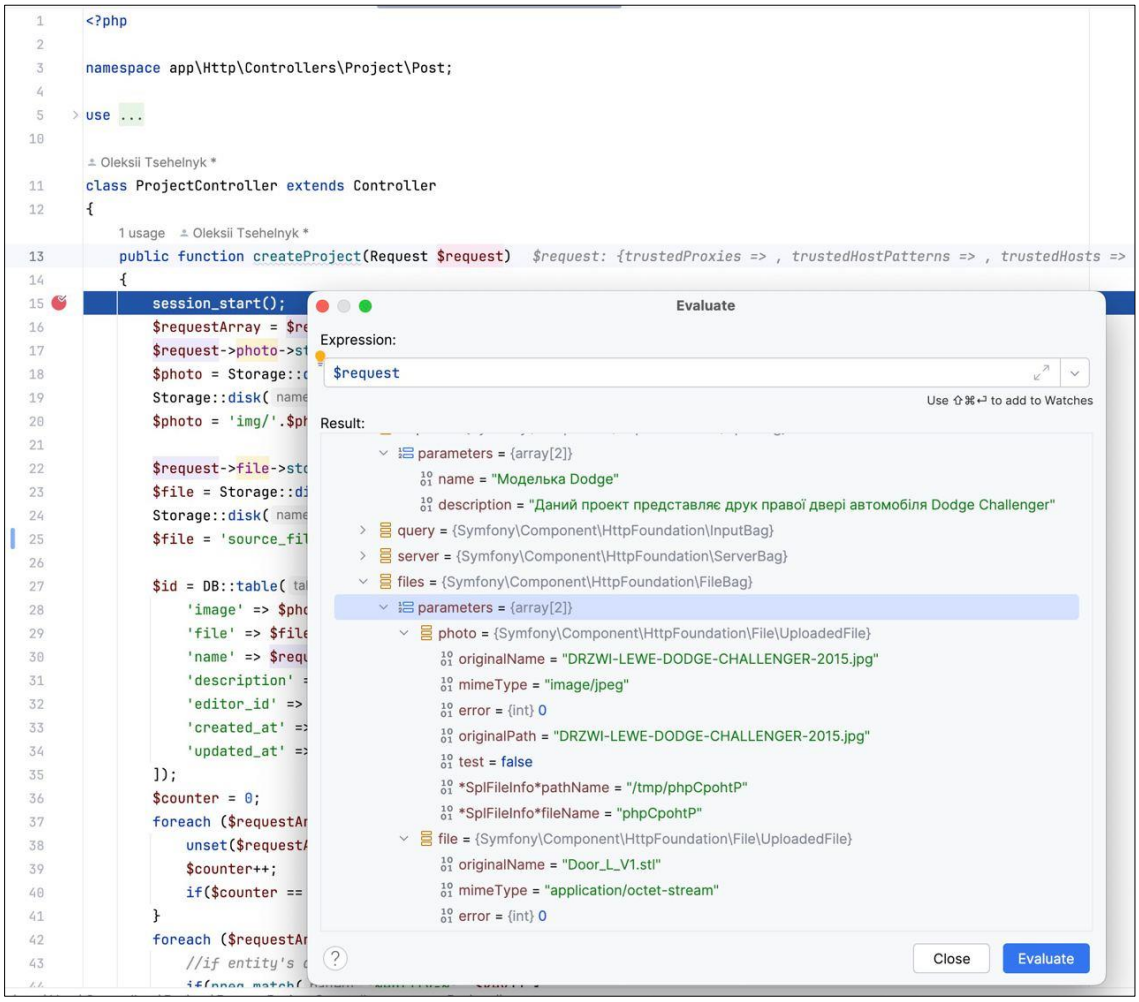


Рисунок 3.3.2 – Модульне тестування контролера

Аналогічним чином було проведено тестування всіх контролерів, що відповідають за обробку логіки додатку, з метою перевірки правильності виконання відповідних дій. Це дало змогу переконатися, що кожен окремий запит до контролера обробляється належним чином та повертає очікуваний результат.

Крім модульного тестування, було також застосоване інтеграційне тестування, яке полягало у перевірці взаємодії між різними модулями системи. Зокрема, тестувалася передача даних із Vue-компонента на серверну частину застосунку для їхнього подальшого збереження у базі даних. Результати одного з таких тестових запитів надсилання даних відображені на рисунку 3.3.3-3.3.4, де можна побачити правильне прийняття даних сервером та успішне збереження.

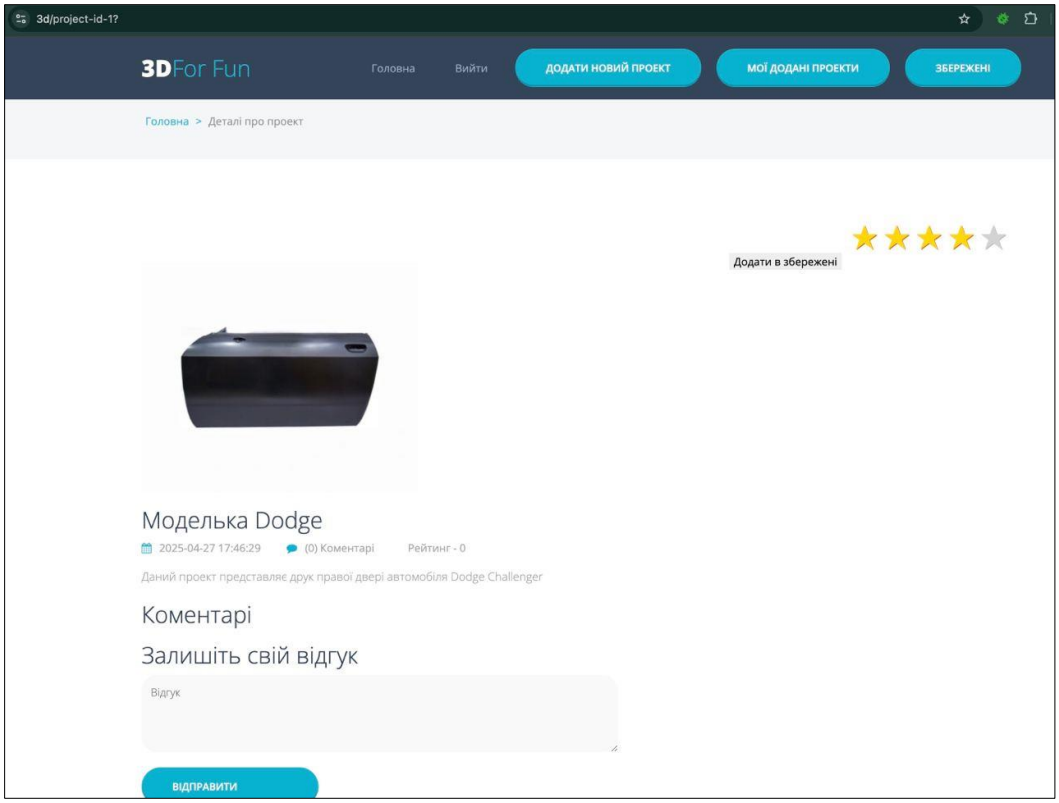


Рисунок 3.3.3 –Тестування Vue компонента з фронтової частини

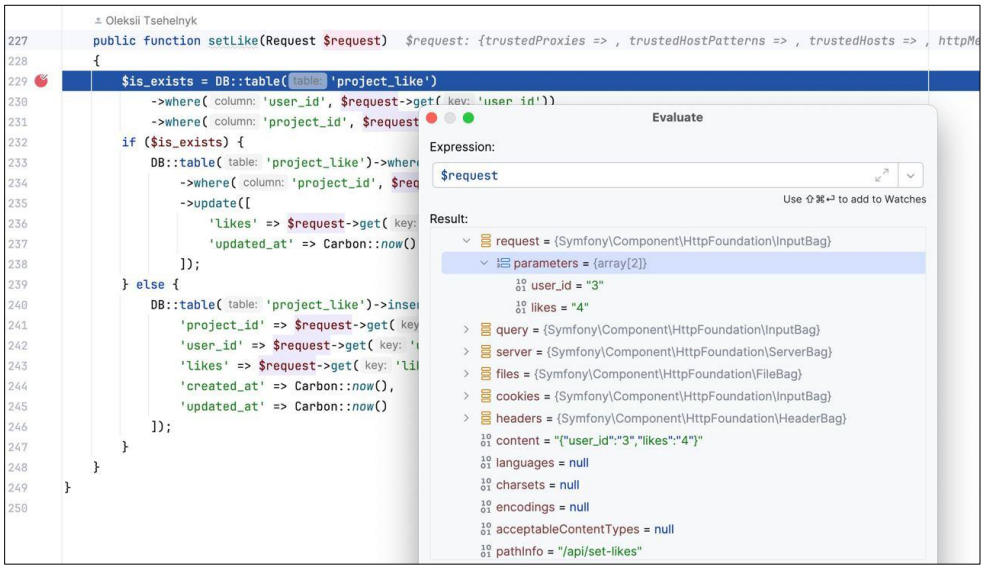


Рисунок 3.3.4 –Тестування Vue компонента з бекенд частини

Одним із важливих тестів було тестування адмін панелі, що відповідає за виконання CRUD дій над усіма сутностями додатку. Результати тестування наведені на рисунках 3.3.5 – 3.3.7.

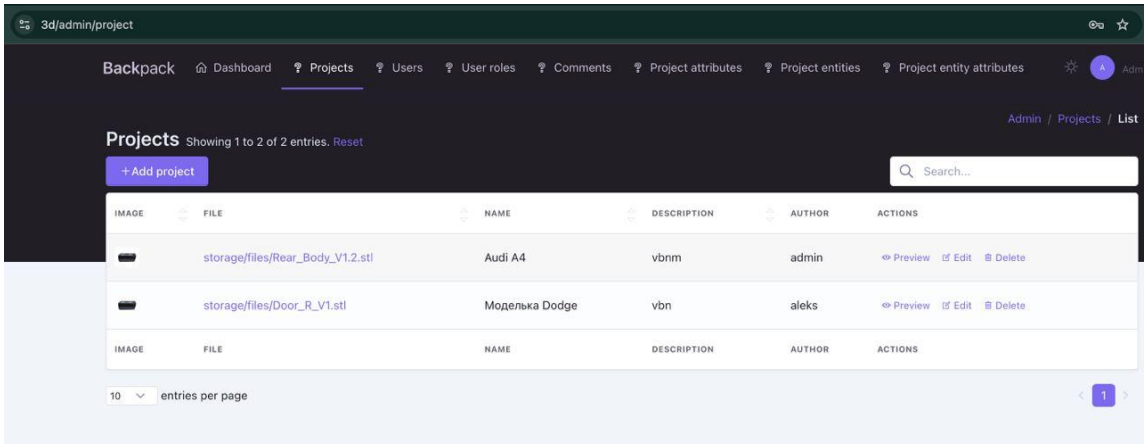


Рисунок 3.3.5 – Тестування Read даних

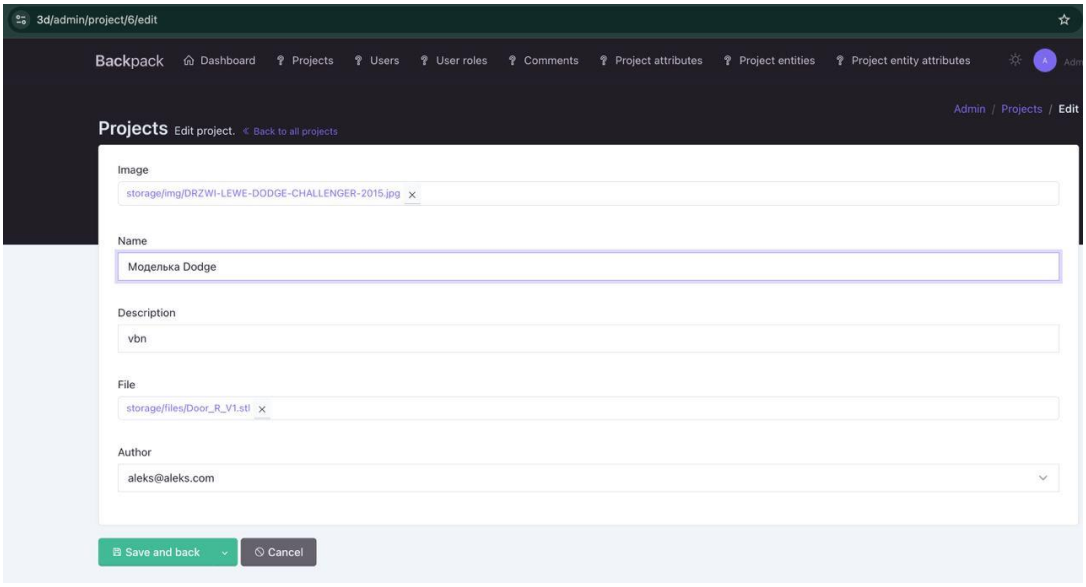


Рисунок 3.3.6 – Тестування Update даних

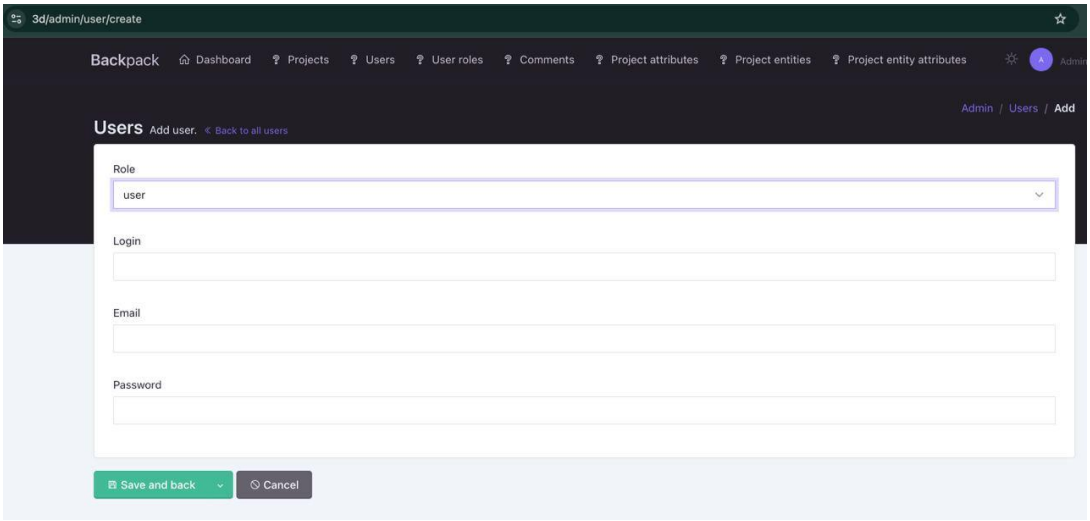


Рисунок 3.3.6 – Тестування Create даних

Крім функціонального тестування, важливим є оцінка ефективності роботи додатку. Одним із базових тестів продуктивності є вимірювання часу завантаження сторінки. Для цього можна скористатися інструментом браузера «Інспектором елементів». На вкладці «Network» можна детально побачити, скільки часу займає завантаження кожного ресурсу сторінки, а також загальний час, необхідний для повного завантаження веб-сторінки разом з усіма даними та ресурсами (зображеннями, скриптами, стилями тощо). На рисунку 3.3.5 представлені результати тестування часу завантаження сторінки.

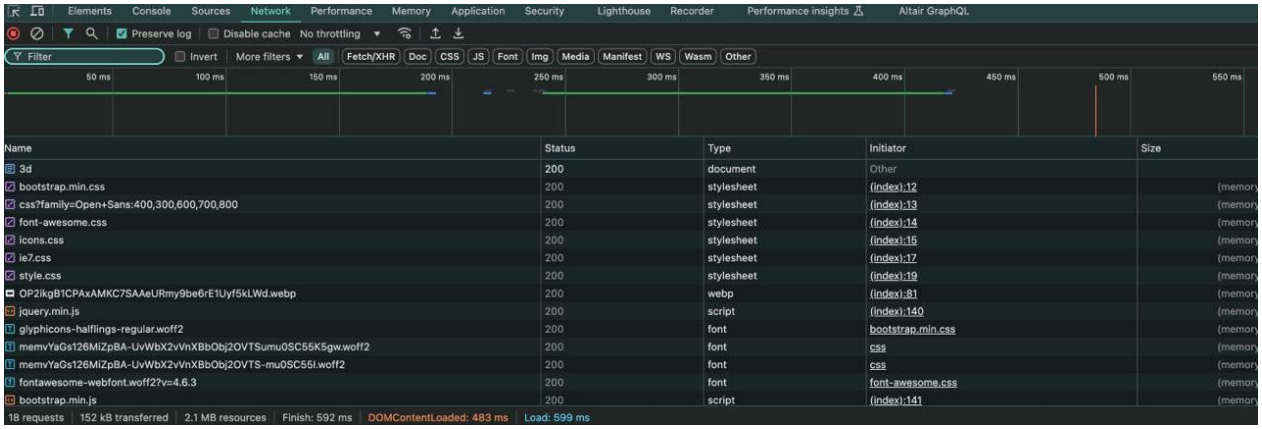


Рисунок 3.3.5 – Тестування ефективності

Таким чином, завдяки комплексному підходу до тестування, вдалося забезпечити надійність роботи додатку як на рівні окремих модулів, так і під час їхньої взаємодії, а також підвищити ефективність виконання запитів.

У цьому розділі було створено структуру бази даних за допомогою міграцій, що забезпечує зручність у подальшому оновленні схеми. Проведено зв’язок між таблицями, реалізовано прокидування та збереження даних. Оформлено користувацький інтерфейс за допомогою Laravel-blade шаблонів та Vue.js компонентів, що дозволяють динамічно взаємодіяти з даними без перезавантаження сторінки. Для адміністрування сутностей використано Laravel Backpack, який значно спрощує керування даними через готову адмінпанель. Наприкінці проведено тестування основного функціоналу, щоб переконатися у правильності роботи додатку.

ВИСНОВКИ

Метою виконаної роботи було створення ефективної платформи для публікацій 3D проектів, який би відповідав сучасним вимогам щодо функціональності та зручності користування. Застосунок повинен задовольняти потреби щодо взаємодії користувачів із проектами та забезпечувати простий, зрозумілий інтерфейс для користувачів різних рівнів досвіченості.

У рамках завдання було проведено аналіз потреб кінцевих користувачів, здійснено проектування архітектури та інтерфейсу системи, реалізовано програмну частину та протестовано основні функції застосунку, а також впроваджено заходи щодо підвищення безпеки й оптимізації продуктивності.

Робота охопила повний цикл створення застосунку: від дослідження предметної області до розробки та підведення підсумків. Зокрема, було виконано такі етапи:

– зібрано та проаналізовано інформацію щодо існуючих рішень у сфері вебзастосунків для публікацій 3D-проектів, визначено вимоги до майбутнього програмного продукту, сформовано технічне завдання;

– обрано найбільш відповідну архітектурну модель для реалізації вимог, побудовано структуру бази даних, розроблено прототип інтерфейсу користувача, а також визначено оптимальні інструменти та середовище для подальшої реалізації;

– здійснено розробку ключових модулів, створено базу даних, додано інструкції користувача і описано технічні вимоги. Проведено тестування системи з використанням різних підходів для виявлення можливих помилок і перевірки відповідності функціональних можливостей початковим вимогам.

У підсумку було створено повноцінний вебзастосунок з інтуїтивно зрозумілим та зручним інтерфейсом, який дозволяє користувачам ефективно керувати 3D-проектами: додавати нові моделі, редагувати наявні, залишати коментарі, оцінювати роботи інших користувачів, а також додавати динамічні атрибути до проєктів залежно від потреб. Уся реалізована функціональність

повністю відповідає поставленим завданням і технічним вимогам, що робить цей застосунок практичним рішенням у сфері 3D-друку та спільної роботи над моделями.

Даний програмний продукт був реалізований у середовищі розробки PHPStorm, що забезпечує зручні інструменти для написання, налагодження та організації коду. Основною мовою програмування обрано PHP. Як базовий фреймворк було використано Laravel – інструмент, який значно спрощує реалізацію логіки вебзастосунку завдяки вбудованим механізмам маршрутизації, ORM, шаблонізації, валідації, авторизації тощо.

Для збереження та обробки даних у проекті використовується реляційна база даних MySQL, яка забезпечує надійне зберігання інформації, масштабованість та ефективну роботу з великим обсягом структурованих даних.

Для реалізації динамічного інтерфейсу, було використано фреймворк Vue.js. Завдяки йому вдалося створити реактивні компоненти, що швидко реагують на дії користувача, покращуючи взаємодію та загальну зручність користування. Для обміну даними між клієнтською частиною і сервером застосовується Axios – бібліотека для асинхронної передачі HTTP-запитів, яка забезпечує зручну роботу з API та обробку відповіді без перезавантаження сторінки.

Особливу увагу було приділено розробці адміністративної панелі, яка реалізована за допомогою Backpack for Laravel – розширення для створення інтерфейсів адміністрування. Панель підтримує повноцінний функціонал CRUD, що надає можливість легко керувати користувачами, 3D-проєктами, коментарями та динамічними атрибутами.

Результати розробки підтвердили працездатність та ефективність обраної архітектури рішення. Проект виконує всі поставлені завдання, демонструє стабільну роботу та зручність в користуванні. Крім того, структура коду та модульна організація функціоналу дають хорошу основу для подальшого масштабування, розширення можливостей системи, інтеграції нових сервісів і вдосконалення користувацького досвіду.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Advantages and Disadvantages of 3D-printing. URL: https://3d4u.com.ua/uk/blog/post/101-advantages-and-disadvantages-of-3d-printing (дата звернення: 15.02.2025).
2. 34 Best Sites for Free 3D Downloading. URL: https://www.3dprinter.ua/34-luchshih-sajta-dlja-besplatnogo-skachivanija-stl-i-3d-fajlov (дата звернення: 18.02.2025).
3. 3D Printing in Education Building Schools of the Future. URL: https://www.creality.com/blog/3d-printing-in-education-building-schools-of-the-future (дата звернення: 20.02.2025).
4. The Challenges of 3D-printing in Education. URL: https://weareprintlab.com/blog/the-challenges-of-3d-printing-in-education (дата звернення: 21.02.2025).
5. Opportunities and Challenges of 3D Printing Integration into Engineering Education in Developing Countries. URL: https://www.researchgate.net/publication/380543921_Opportunities_and_Challenges_of_3D_Printing_Integration_into_Engineering_Education_in_Developing_Countries (дата звернення: 21.02.2025).
6. 6 Overlooked Benefits of 3D-printing for Your Supply Chain. URL: https://ultimaker.com/learn/6-overlooked-benefits-of-3d-printing-for-your-supply-chain (дата звернення: 22.02.2025).
7. 3D Printing Education. URL: https://www.3ds.com/make/solutions/industries/3d-printing-education (дата звернення: 25.02.2025).
8. 3D-printing Technology Teaching Learning. URL: https://www.sedi.psu.edu/news-archive/2019/3D-printing-technology-teaching-learning.aspx (дата звернення: 26.02.2025).

9. Improving Education for all Students With 3D-printing. URL: https://benetech.org/blog/improving-education-for-all-students-with-3d-printing (дата звернення: 27.02.2025).

10. Client-Server Architectures. URL: https://www.cs.sjsu.edu/~pearce/oom/ood/distArch/server.htm (дата звернення: 10.03.2025).

11. Architecture of program software. URL: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya (дата звернення: 11.03.2025).

12. Top 12 Database Design Principles in 2023. URL: https://vertabelo.com/blog/database-design-principles (дата звернення: 12.03.2025).

13. SQL or NoSQL, What is the Difference. URL: https://alternativescience.net/programming/242-sql-chy-nosql-os-v-chomu-pytannya (дата звернення: 13.03.2025).

14. Difference between Oracle and MySQL. URL: https://ua.sawakinome.com/articles/software/difference-between-oracle-and-mysql-3.html (дата звернення: 14.03.2025).

15. Three Tier Architecture. URL: https://uk.theastrologypage.com/three-tier-architecture (дата звернення: 15.03.2025).

16. Free Design Templates. URL: https://www.templatemonster.com (дата звернення: 18.03.2025).

17. Turi, J.A.; Khwaja, M.G.; Tariq, F.; Hameed, A. The role of big data analytics and organizational agility in improving organizational performance of business processing organizations. Bus. Process. Manag. J. 2023, 29, 2081-2106/

18. Web Resource module. URL: https://developer.atlassian.com/server/confluence/web-resource-module (дата звернення: 27.03.2025).

19. Best Laravel Admin Templates. URL: https://www.dronahq.com/best-laravel-admin-templates (дата звернення 15.04.2025).

20. Backpack for Laravel. URL: https://wpwebinfotech.com/blog/backpack-for-laravel (дата звернення: 16.04.2025).

21. Build Custom Admin Panels With BackPack. URL: https://laravel-news.com/laravel-backpack (17.04.2025).

22. Axios in JavaScript. URL: https://blog.logrocket.com/axios-javascript (дата звернення: 20.04.2025).

23. Making HTTP requests with Axios. URL: https://circleci.com/blog/making-http-requests-with-axios (дата звернення: 22.04.2025)

24. Vue JS With Laravel. URL: https://vueschool.io/articles/vuejs-tutorials/the-ultimate-guide-for-using-vue-js-with-laravel (дата звернення: 21.04.2025).

25. Документація VueJs. URL: https://vuejs.org (дата звернення: 21.04.2025).

26. Developers Forum. URL: https://stackoverflow.com (дата звернення: 28.04.2025).

27. Кваліфікаційна робота: Методичні настанови для здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 121 «Інженерія програмного забезпечення» / Л.П. Бедратюк, Г.І. Радельчук. Хмельницький: ХНУ, 2023. - 60 с.

28. Положення про атестацію здобувачів вищої освіти та наукових ступенів у Хмельницькому національному університеті. URL: https://khmnu.edu.ua/wp-content/uploads/normatyvni-dokumenty/polozhennya/pro-atestacziyu.pdf (дата звернення: 29.04.2025).

29. Положення про контроль і оцінювання результатів навчання здобувачів вищої освіти в Хмельницькому національному університеті. URL: https://khmnu.edu.ua/wp-content/uploads/normatyvni-dokumenty/polozhennya/pro-kontrol-i-oczinyuvannya-rezultativ-navchannya.pdf (дата звернення: 05.05.2025).

30. Документація Laravel. URL: https://laravel.com (дата звернення: 10.05.2025)

31. CSS docx. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 11.05.2025).

32. MDN Web Docs – JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 12.05.2025).

33. NGINX – What is NGINX? URL: https://www.nginx.com/resources/glossary/nginx/ (дата звернення: 12.05.2025).

34. Dependency Manager for PHP. URL: https://getcomposer.org/doc/00-intro.md (дата звернення: 12.05.2025).

35. Postman – API Development Explained. URL: https://www.postman.com/api-platform/ (дата звернення: 12.05.2025).

36. Browser Fingerprinting Techniques. URL: https://fingerprint.com/blog/browser-fingerprinting-techniques/ (дата звернення: 13.05.2025).

37. PHP script execution. URL: https://www.geeksforgeeks.org/php/how-to-execute-php-script-in-website-using-xampp-webserver (дата звернення: 13.05.2025).

38. Unit tests. URL: https://aws.amazon.com/what-is/unit-testing/?nc1=h_ls (дата звернення: 14.05.2025).

39. Deploy Laravel app. URL: https://docs.railway.com/guides/laravel (дата звернення: 14.05.2025).

40. How to connect to the remote server. URL: https://www.digitalocean.com/community/tutorials/how-to-use-ssh-to-connect-to-a-remote-server (дата звернення: 14.05.2025).

ДОДАТОК А
(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

1.Введення

Вебзастосунок було роблено для швидкого та зручного замовлення годинників.

Підставою для розробки є «Завдання до кваліфікаційної роботи».

Найменування розробки: «Вебзастосунок для публікації 3D-друкованих проєктів».

2.Користувачі ресурсу:

– адміністратор, який повинен контролювати наповнення (контент, моделі), а також адмініструвати користувачів;

– користувач, які заходять на платформу, та авторизовуються. Завдяки авторизації у вебзастосунку, користувачі будуть мати більші можливості, такі як публікації власних моделей, коментування та вподобання проектів інших користувачів, а також збереження моделей в розділ «улюблені».

– гість, не зареєстровані чи авторизовані відвідувачі веб застосунку. Для такої ролі буде доступно лише реєстрація, переглядання існуючих опублікованих моделей, а також використання простої групи фільтрів.

3.Функціональні вимоги:

– авторизація (зареєстрованих) та окремо реєстрація користувачів у веб за стосунку;

– вивід усіх наявних моделей;

– пагінація – розділення виводу даних на декілька сторінок;

– деталізований вивід кожної моделі з можливістю коментування та вподобання;

– створення окремих вподобаних списків для користувачів;

– можливість пошуку моделі по назві чи ключових словах;

– надання створення нових гнучких атрибутів при публікації нової моделі;

– надання створення нових гнучких значень атрибутів при публікації нової моделі.

4.Нефункціональні вимоги:

– мінімальний час відповіді ресурсу на запити;

– надійність та продуманість від майбутніх помилок, а також швидкість виправлення помилок, якщо такі стануться;

– пропускна здатність, а саме можливість обробляти велику кількість запитів за певний проміжок часу, або ж балансування вхідного трафіку;

– доступність для усіх користувачів з різних пристроїв.

5.Технічні характеристики:

Програмна реалізація вебзастосунку буде здійснюватися із застосуванням фреймворку Laravel із використанням мови програмування PHP, яка дозволить забезпечити високу швидкість роботи вебсайту для публікації 3D-друкованих проектів. В якості сервера бази даних передбачено використання серверу MySQL. Для динамічного відображення елементів буде використовуватись VueJS з бібліотекою Axios. Адмін панель буде створена завдяки розширенню Backpack.

6.Вимоги до інтерфейсу користувача:

– простий та зрозумілий для користувача без надлишкового навантаження;

– відображені елементи керування повинні бути відображені відповідно до наданої ролі;

– сторінки, на які звичайні користувачі не будуть мати доступу повинні мати валідацію та повертати на головну сторінку.

7.Вимоги до контенту:

Основна структура виводу даних публікованих моделей буде мати наступний вигляд

- відображення детального опису усіх характеристик проекту;
- секція для завантаження матеріалів у відповідному форматі;
- секція для з усіма динамічними атрибутами для проекту, що надасть більше динамічної інформації;
- секція з коментарями та окремим полем для авторизованих користувачів для додавання даних нових коментарів;
- секція з кількістю вподобань моделі за відображення кнопок для оцінки проекту;
- на головній сторінці для користувачів повинен бути доступний рядок пошуку по ключових словах моделей;
- повинна бути окрема секція, де будуть зберігатись усі проекти, які користувач уже зберіг;
- якщо користувач авторизований, він буде мати змогу залишати коментарі та оцінювати публіковані моделі іншими авторами за 5-ти бальною шкалою (5 зірочок);
- при створенні нової моделі, користувач зможе динамічно додавати нові атрибути, якщо існуючий набір атрибутів не містить необхідних. В подальшому, всі створенні динамічно атрибути будуть доступні для усіх інших моделей.

8.Визначення порядку і термінів для розробки:

Термін розробки в рамках роботи над кваліфікаційною роботою бакалавра:

- дослідження предметної області із виділенням меж;
- проведення аналізу вже існуючих;
- здійснення проектування інтерфейсу користувача, бази даних, архітектури концепції;
- реалізація проекту;
- проведення тестування та наповнення ресурсу.

Додаток Б

(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Рисунок Б.1 – Титульний слайд

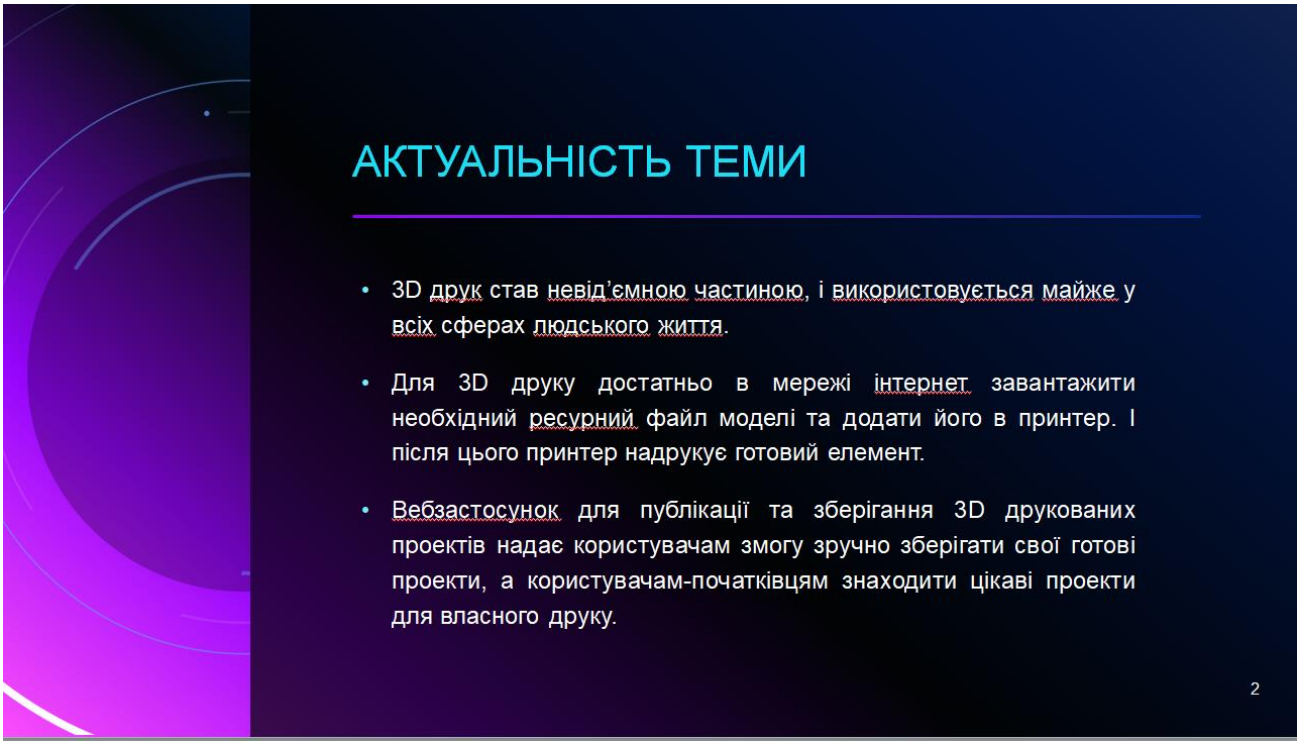


Рисунок Б.2 – Актуальність теми

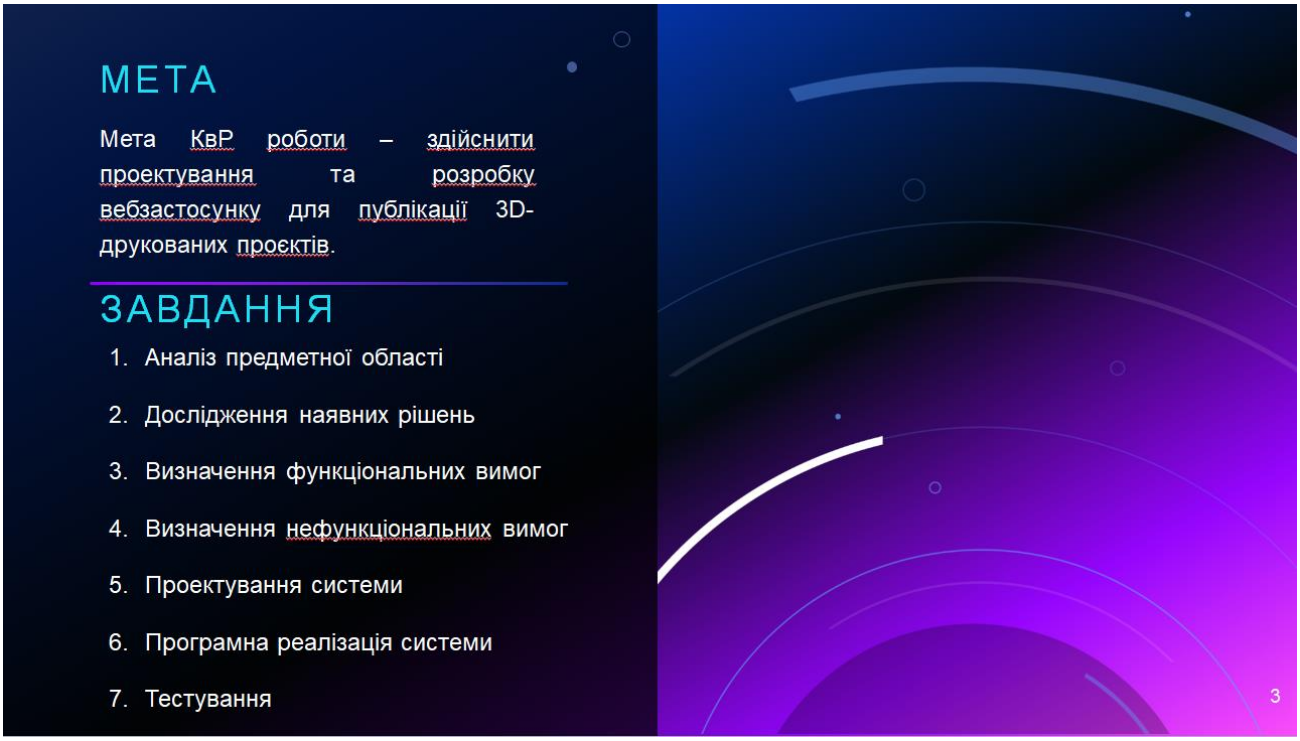


Рисунок Б.3 – Мета і завдання

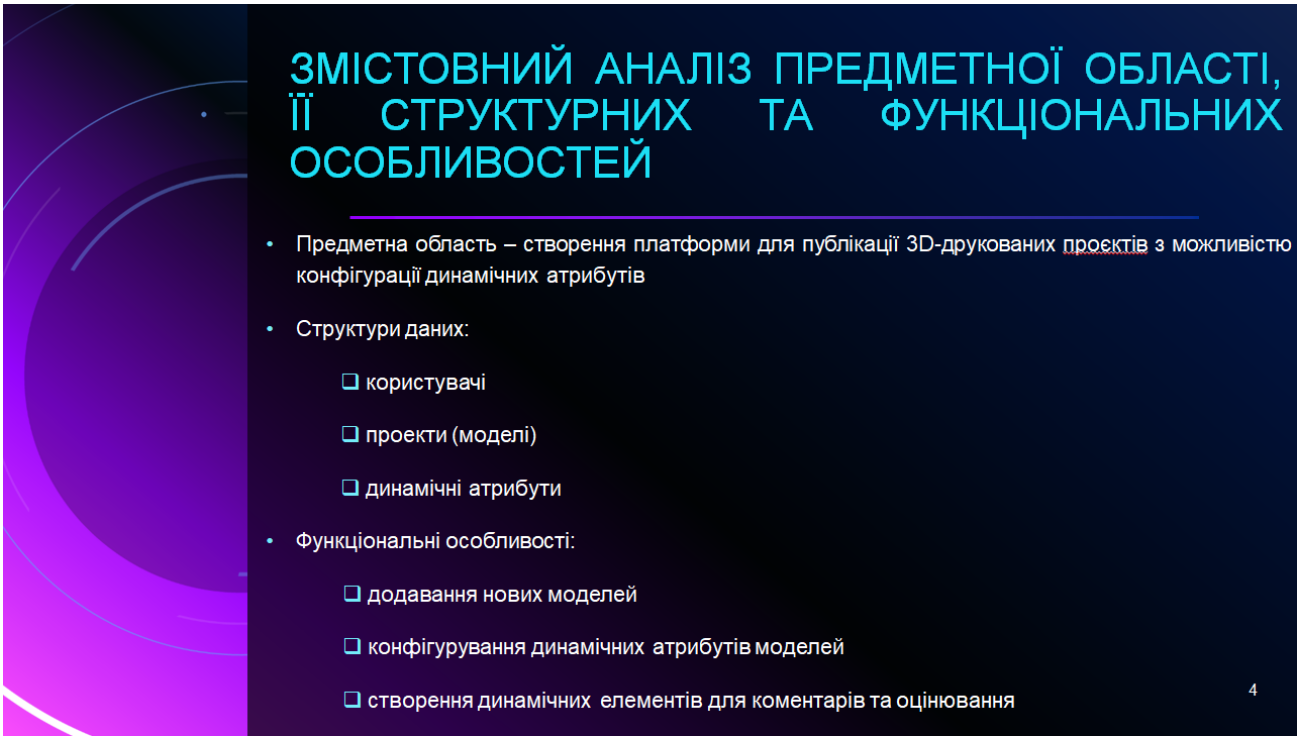


Рисунок Б.4 – Аналіз предметної області

Рисунок Б.5 – Аналіз наявного ПЗ

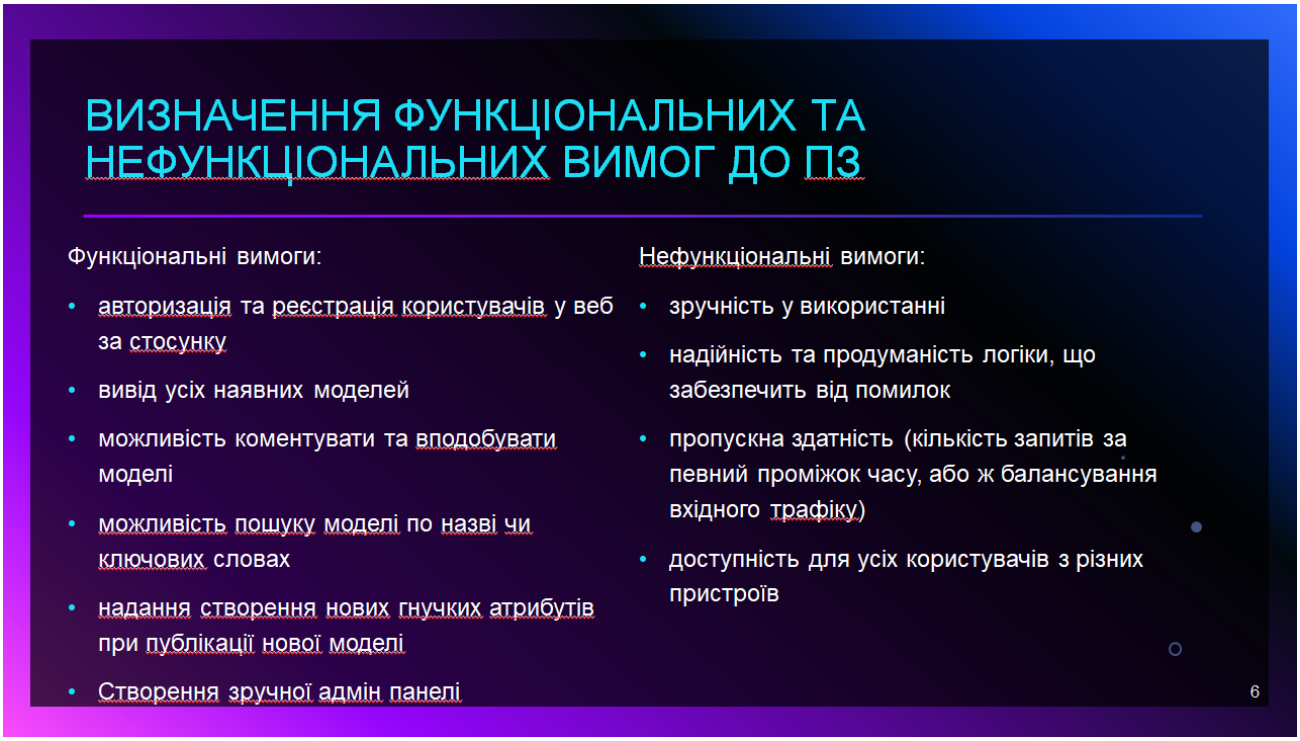


Рисунок Б.6 – Визначення вимог

Рисунок Б.7 – Проектування архітектури

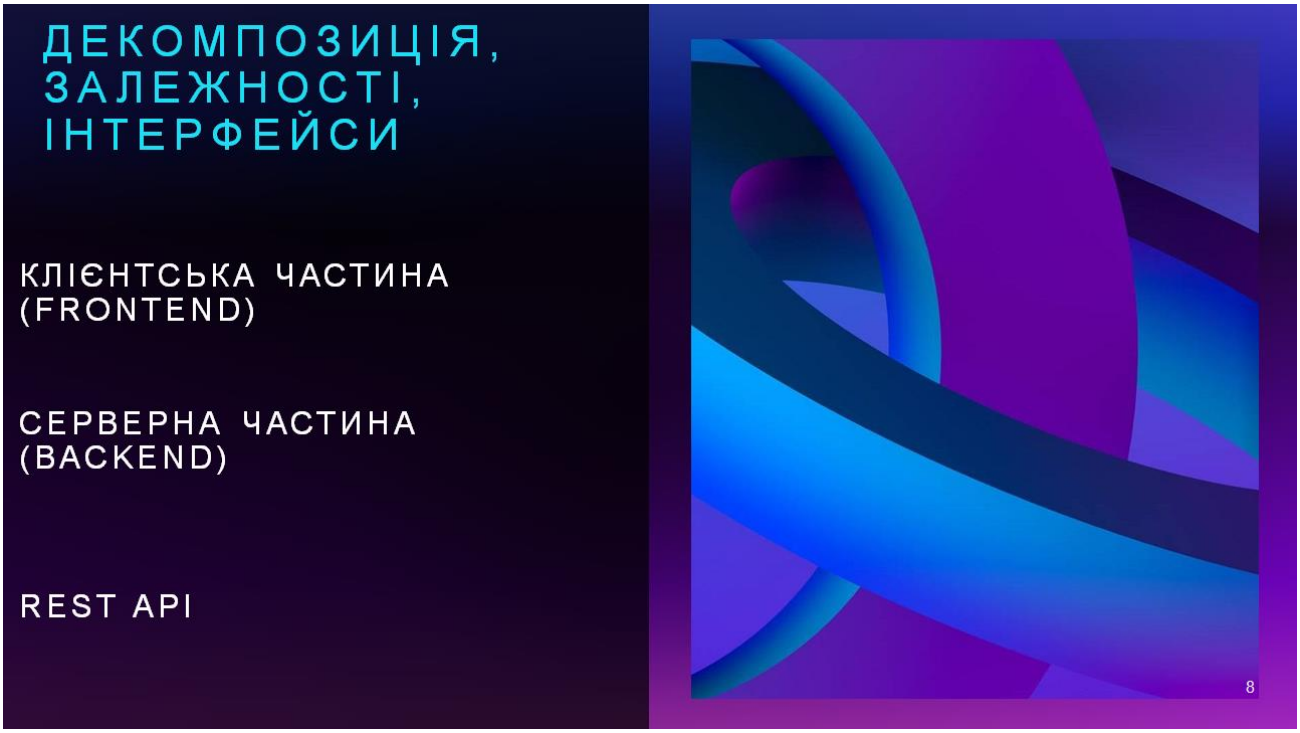


Рисунок Б.8 – Декомпозиція компонентів

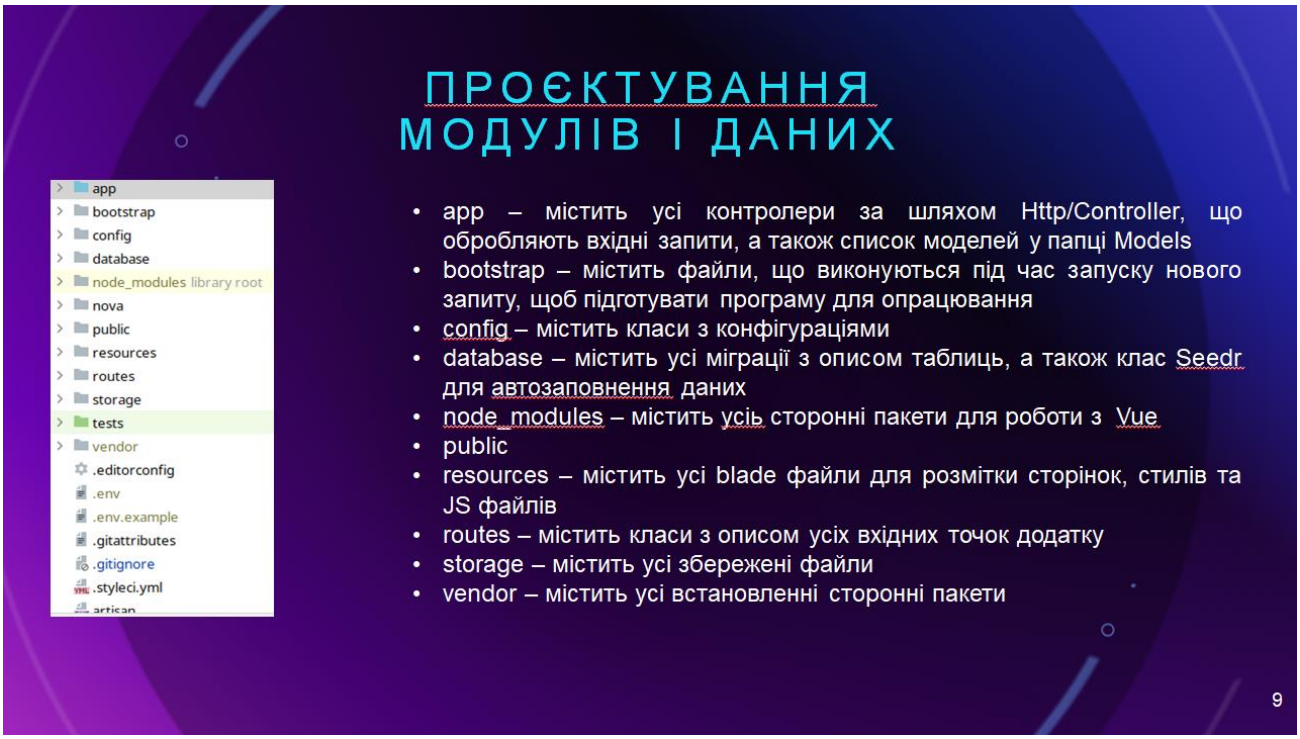


Рисунок Б.9 – Проектування модулів

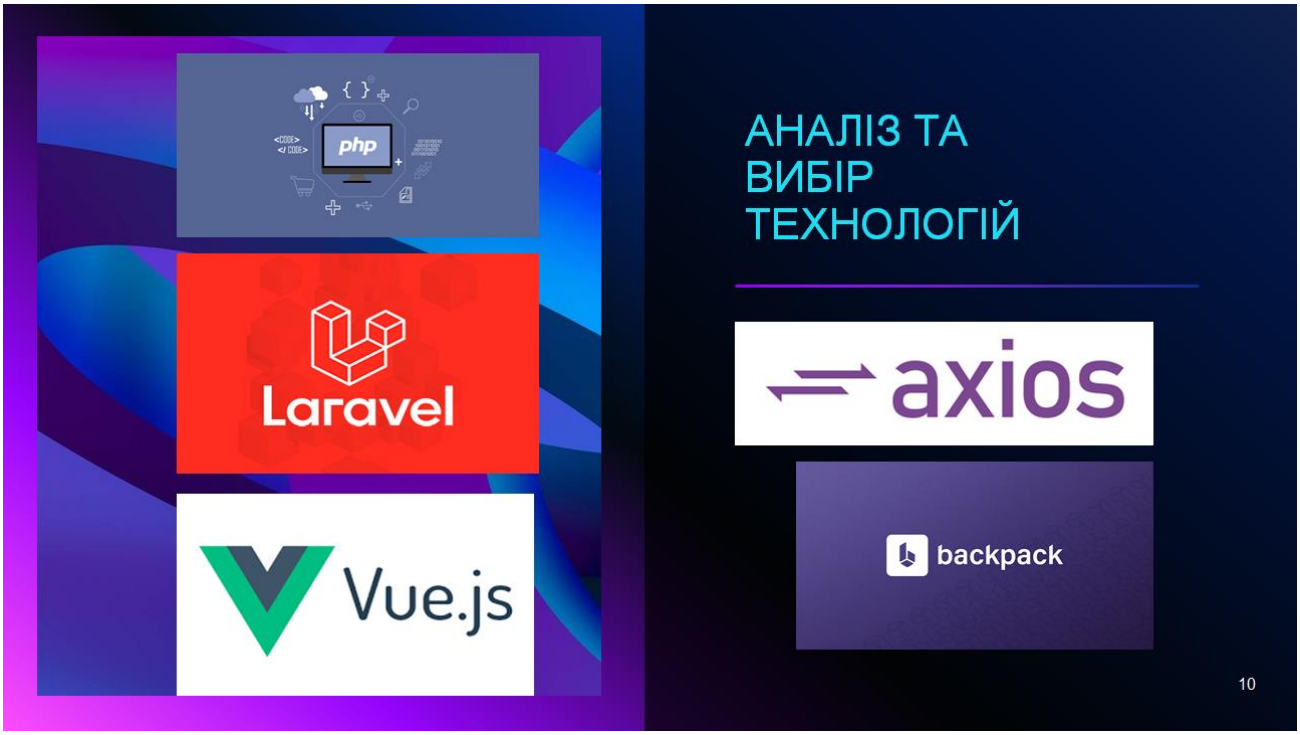


Рисунок Б.10 – Аналіз вибору технологій

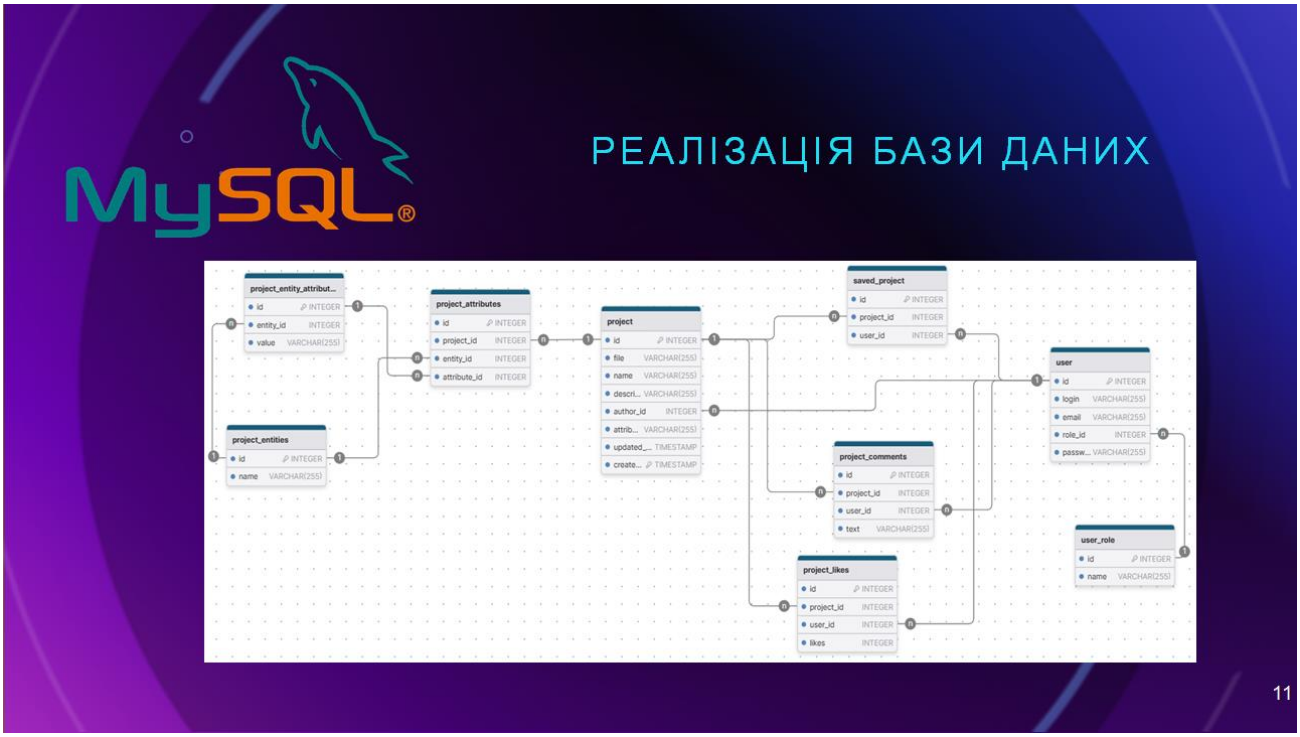


Рисунок Б.11 – Реалізація бази даних

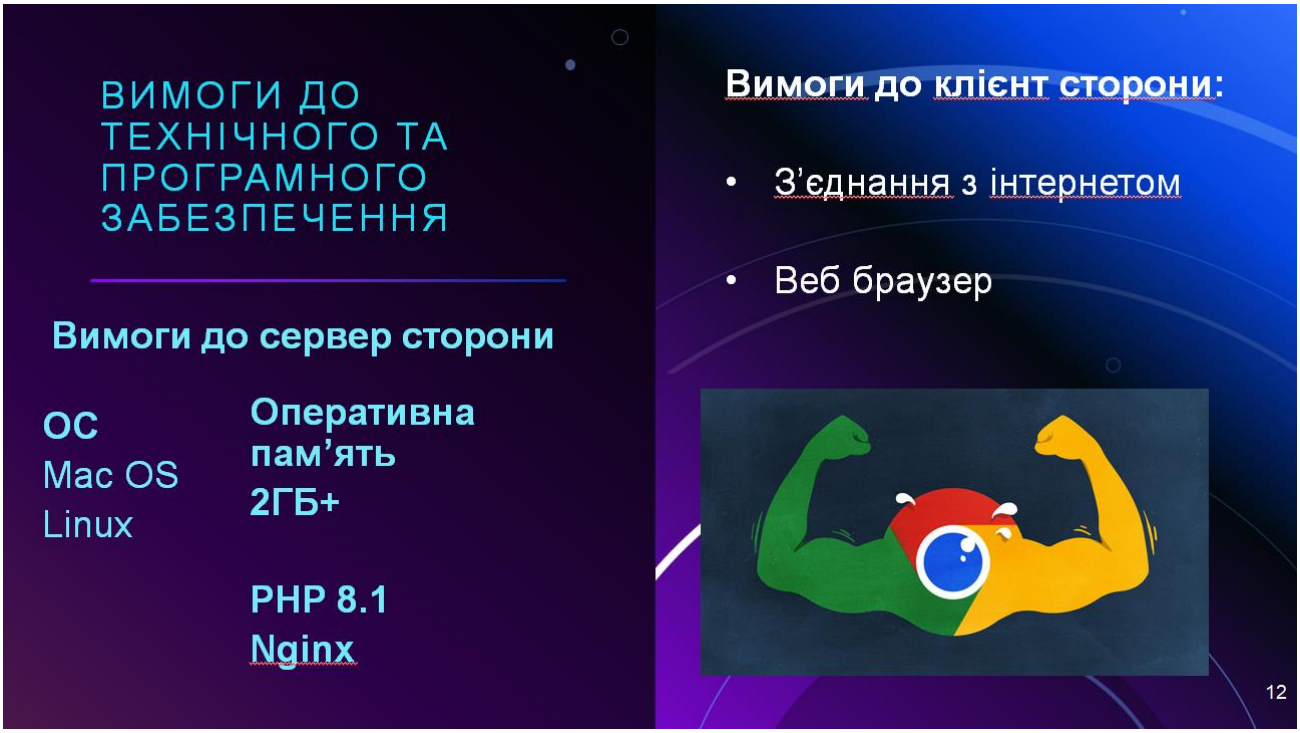


Рисунок Б.12 – Вимоги до ПЗ

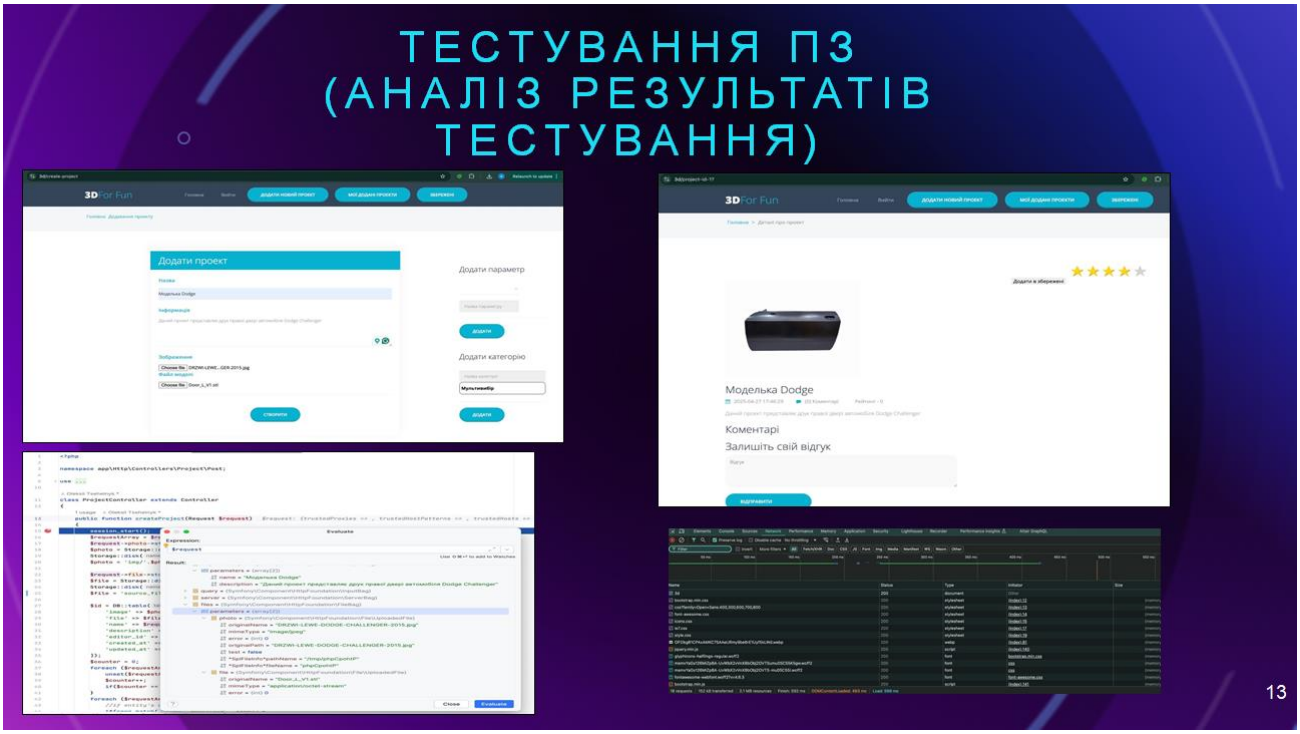


Рисунок Б.13 – Тестування ПЗ

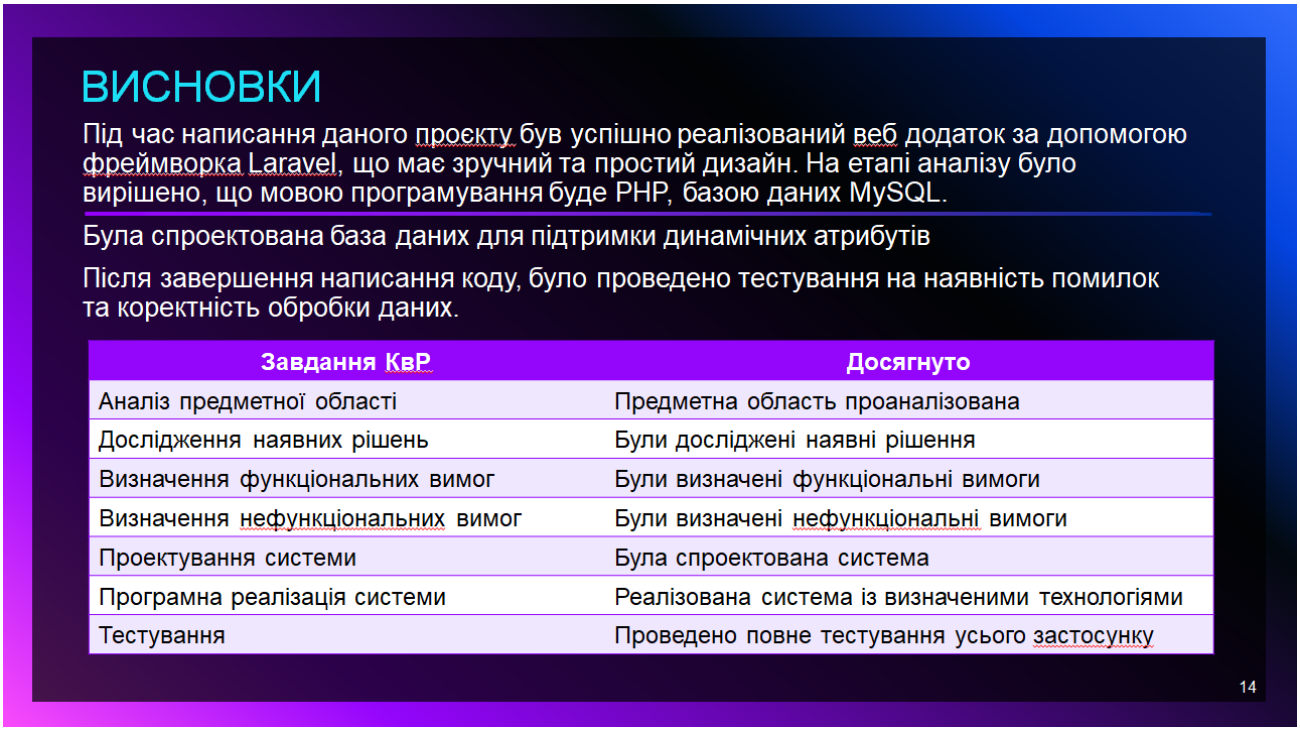


Рисунок Б.14 – Висновки

ДОДАТОК В
(обов'язковий)

КОД (ЛІСТИНГ) ЗАСТОСУНКУ

routes/web.php

```
<?php
use App\Http\Controllers\Login\Post\LoginController as LoginPostController;
use App\Http\Controllers\Project\Post\ProjectController as
ProjectPostController;
use Illuminate\Support\Facades\Route;
Route::post('login', [LoginPostController::class, 'login']);
Route::post('register', [LoginPostController::class, 'register']);
Route::post('logout', [LoginPostController::class, 'register']);
Route::post('create-project', [ProjectPostController::class, 'createProject']);
Route::post('edit-project-id-{id}', [ProjectPostController::class,
'editProject']);
Route::post('add-to-favourites', [ProjectPostController::class,
'addFavourites']);
Route::post('remove-from-favourites', [ProjectPostController::class,
'removeFavouritesProject']);
Route::post('search-projects', [ProjectPostController::class, 'searchProject']);
Route::post('change-project-admin', [ProjectPostController::class, 'delete']);
Route::post('delete-project-user', [ProjectPostController::class,
'deleteProject']);
Route::post('add-attribute', [ProjectPostController::class, 'addAttribute']);
Route::post('add-entity', [ProjectPostController::class, 'addEntity']);
Route::post('add-comment', [ProjectPostController::class, 'leaveMessage']);
Route::post('set-likes', [ProjectPostController::class, 'setLike']);
```

app/Http/Controllers/Project/Post/ProjectController.php

```
<?php
namespace app\Http\Controllers\Project\Post;
use App\Http\Controllers\Controller;
use Carbon\Carbon;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Storage;
use Symfony\Component\HttpFoundation\Request;
class ProjectController extends Controller
{
    public function createProject(Request $request)
    {
        session_start();
        $requestArray = $request->request->all();
        $photo = $request->file('photo')->storeAs(
            '', $request->file('photo')->getClientOriginalName(),
'project_photo'
        );
        $photoPath = 'storage/img/' . $photo;
        $file = $request->file('file')->storeAs(
            '', $request->file('file')->getClientOriginalName(), 'project_file'
        );
        $filePath = 'storage/files/' . $file;
        $id = DB::table('project')->insertGetId([
            'image' => $photoPath,
            'file' => $filePath,
            'name' => $request['name'],
            'description' => $request['description'],
            'editor_id' => $_SESSION["is_auth"],
```

```
            'created_at' => Carbon::now(),
            'updated_at' => Carbon::now()
        ]);
        $counter = 0;
        foreach ($requestArray as $key=>$item) {
            unset($requestArray[$key]);
            $counter++;
            if($counter == 9) break;
        }
        foreach ($requestArray as $key=>$attribute) {
            //if entity's attribute has no multi choice
            if(preg match('~entity-~', $key)) {
                DB::table('project_attributes')->insert([
                    'project_id' => $id,
                    'entity_id' => substr($key, 7),
                    'attribute_id' =>  $attribute
                ]);
            } else {
                $entity_id = DB::table('project_entity_attributes')
                    ->select('entity_id')
                    ->where('id',$key)
                    ->get()->all()[0];
                DB::table('project_attributes')->insert([
                    'project_id' => $id,
                    'entity_id' => $entity_id->entity_id,
                    'attribute_id' =>  $key
                ]);
            }
        }
        return Redirect('/');
    }
    public function editProject(Request $request)
    {
        session_start();
        DB::table('project')->where('id', $request['id'])->update([
            'name' => $request['name'],
            'description' => $request['description'],
            'editor_id' => $_SESSION['is_auth'],
            'updated_at' => Carbon::now()
        ]);
        $requestArray = $request->request->all();
        if($request->exists('photo'))
        {
            $request->photo->store('/', 'project_photo');
            $photo = Storage::disk('users')->put('/', $request->file('photo'));
            Storage::disk('users')->put('/', $request->file('photo'));
            $photo = 'img/'.$photo;
            DB::table('project')->where('id', $request['id'])->update([
                'image' => $photo,
                'updated_at' => Carbon::now()
            ]);
        }
        if($request->exists('file'))
        {
            $request->file->store('/', 'project_file');
            $file = Storage::disk('users')->put('/', $request->file('file'));
            Storage::disk('users')->put('/', $request->file('file'));
            $file = 'img/file/'.$file;
            DB::table('project')->where('id', $request['id'])->update([
                'file' => $file,
                'updated_at' => Carbon::now()
            ]);
```

```
        }
        $counter = 0;
        foreach ($requestArray as $key=>$item) {
            unset($requestArray[$key]);
            $counter++;
            if($counter == 10) break;//тут треба дебаг
        }
        $all_attributes = DB::table('project_attributes')-
>select('attribute_id')
            ->where('project_id', $request['id'])
            ->get()->all();
        $applied attributes = [];
        foreach ($requestArray as $key=>$attribute) {
            if(preg_match('~entity-~', $key)) {
                if(
                    DB::table('project_attributes')
                        ->where('entity_id', substr($key, 7))
                        ->where('project_id', $request['id'])->exists()
                ) {
                    DB::table('project_attributes')
                        ->where('project_id', $request['id'])
                        ->where('entity_id', substr($key, 7))
                        ->update([
                            'attribute_id' => $attribute
                        ]);
                } else {
                    DB::table('project_attributes')->insert([
                        'project_id' => $request['id'],
                        'entity_id' => substr($key, 7),
                        'attribute_id' => $attribute
                    ]);
                }
                $applied_attributes[] = (int)$attribute;
            } else {
                if($attribute == 0) {
                    $entity_id = DB::table('project_entity_attributes')
                        ->select('entity_id')
                        ->where('id', $key)->get()->all()[0];
                    DB::table('project_attributes')->insert([
                        'project_id' => $request['id'],
                        'entity_id' => $entity_id->entity_id,
                        'attribute_id' => $key
                    ]);
                }
                $applied_attributes[] = $key;
            }
        }
        foreach ($all_attributes as $attribute) {
            if(!in_array($attribute->attribute_id, $applied_attributes)) {
                DB::table('project_attributes')
                    ->where('project_id', $request['id'])
                    ->where('attribute_id', $attribute->attribute_id)
                    ->delete();
            }
        }
        return Redirect('/');
    }
    public function addFavourites(Request $request)
    {
        $is_exists = DB::table('saved_project')->where('user_id',$request-
>get('user_id'))
            ->where('project_id',$request->get('id'))->exists();
```

```
        if(!$is_exists) {
            DB::table('saved_project')->insert([
                'user_id' => $request->get('user_id'),
                'project_id' => $request->get('id'),
                'created_at' => Carbon::now(),
                'updated_at' => Carbon::now(),
            ]);
        }
    }
    public function removeFavouritesProject(Request $request)
    {
        session start();
        DB::table('saved_project')->where('user_id',$_SESSION['is_auth'])-
>where('project_id',$request['id'])->delete();
        return Redirect('/my_favourites');
    }
    public function searchProject(Request $request)
    {
        return Redirect('/search-projects?param='.$request['search']);
    }
    public function deleteProject(Request $request)
    {
        if(isset($request['id'])) {
            DB::table('project')->where('id', $request['id'])->delete();
        }
        return Redirect('all-projects');
    }
    public function addAttribute(Request $request)
    {
        $id = DB::table('project_entity_attributes')
            ->insertGetId(['entity_id' => $request->get('entity'), 'value' =>
$request->get('attribute')]);
        return json_encode($id);
    }
    public function addEntity(Request $request)
    {
        $id = DB::table('project_entities')
            ->insertGetId(['entity' => $request->get('entity'),
'is_multi_choice' => $request->get('is_multiple')]);
        $lastEntity = DB::table('project_entities')->where('id', $id)->get()[0];
        return json_encode($lastEntity);
    }
    public function leaveMessage(Request $request)
    {
        $user_id = $request->get('user_id');
        $project_id = $request->get('id');
        $text = $request->get('text');
        DB::table('project_comment')->insert([
            'text'=> $text,
            'project_id' => $project_id,
            'author_id' => $user_id,
            'created_at' => Carbon::now(),
            'updated_at' =>  Carbon::now()
        ]);
    }
    public function setLike(Request $request)
    {
        $is_exists = DB::table('project_like')
            ->where('user_id', $request->get('user_id'))
            ->where('project_id', $request->get('id'))->exists();
        if ($is_exists) {
```

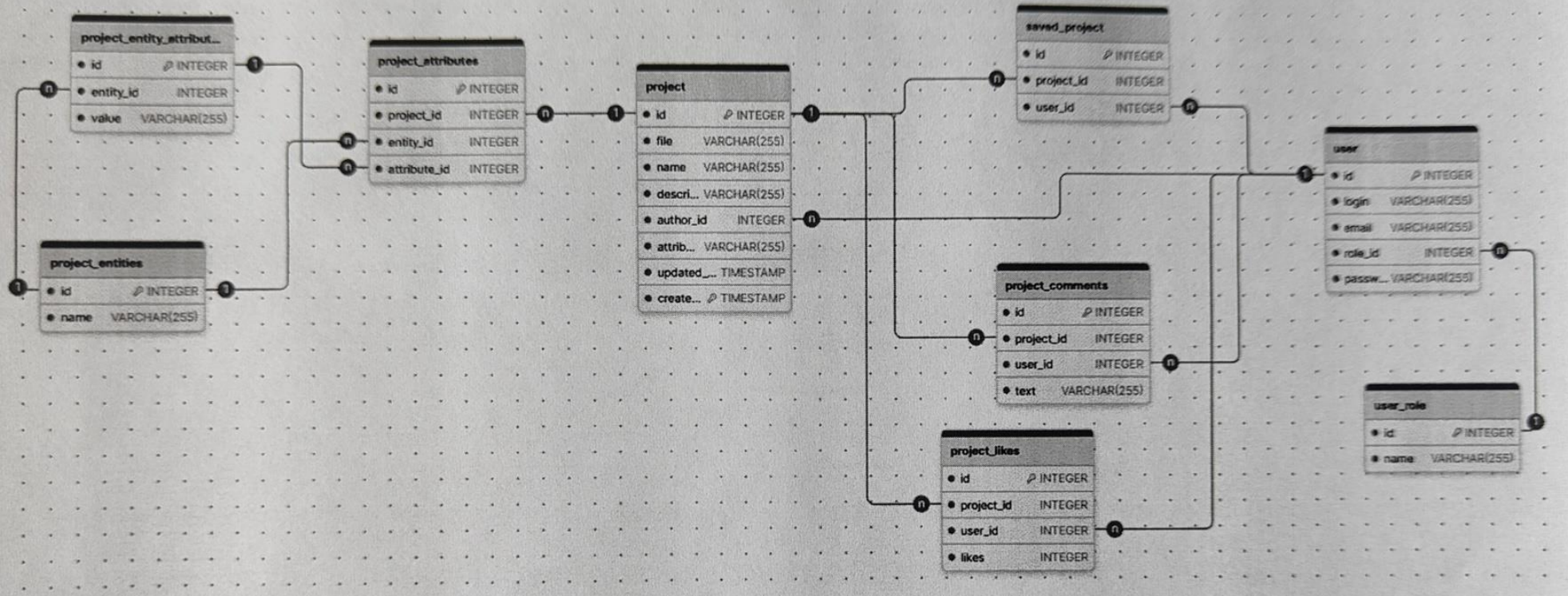
```
            DB::table('project_like')->where('user_id', $request-
>get('user_id'))
                ->where('project_id', $request->get('id'))
                ->update([
                    'likes' => $request->get('likes'),
                    'updated_at' => Carbon::now()
                ]);
        } else {
            DB::table('project_like')->insert([
                'project_id' => $request->get('id'),
                'user_id' => $request->get('user_id'),
                'likes' => $request->get('likes'),
                'created_at' => Carbon::now(),
                'updated_at' => Carbon::now()
            ]);
        }
    }
}
```

resources/components/AddAttributeComponent.vue

```
<template>
    <div>
        <div class="col-sm-2 staticpol">
            <div class="blog-post">
                <!-- Search -->
                <h2>Додати параметр</h2>
                <div class="input-group search-text">
                    <select id="attribute">
                        <option v-for="entity in entitiesArray"
:value="entity.id">{{entity.entity}}</option>
                    </select>
                    <input type="text" class="form-control search"
id="attributeName" placeholder="Назва параметру">
                </div>
                <hr>
                <button class="btn btn-default btn-search"
@click="addAttribute">Додати</button>
            </div>
            <new-entity-add></new-entity-add>
        </div>
    </div>
</template>
<script>
import axios from 'axios';
export default {
    name: "AddAttributeComponent.vue",
    props: ['entities'],
    data() {
        return {
            entitiesArray: []
        }
    },
    mounted() {
        this.entitiesArray = JSON.parse(this.entities)
    },
    methods: {
        async addAttribute() {
            let entity = document.getElementById('attribute').value
            let attribute = document.getElementById('attributeName').value
            let id = 0
            await axios.post('api/add-attribute', {'entity':entity,
'attribute':attribute})
```

```
                .then(response => id = response.data)
            let newElem = document.createElement("div")
            newElem.className = 'innerCheckbox'
            let newInput = document.createElement("input")
            newInput.className = 'checkStyle'
            newInput.id = 'checkbox'+id
            this.entitiesArray.forEach(key => {
                if (key.id == entity) {
                    if (key.is_multi_choice == 0) {
                        newInput.name = "entity-" + entity
                        newInput.type = "radio"
                        newInput.value = id
                    } else {
                        newInput.name = id
                        newInput.type = "checkbox"
                        newInput.value = "0"
                    }
                }
            })
            let newLabel = document.createElement("label")
            newLabel.className = 'checkmark'
            newLabel.htmlFor = 'checkbox'+id
            newLabel.textContent = attribute
            newElem.appendChild(newInput)
            newElem.appendChild(newLabel)
            let lastElem = document.getElementById('block-entity_id-' + entity)
            let oldChild = document.getElementById('hr-id-' + entity)
            lastElem.replaceChild(newElem, oldChild)
            lastElem.appendChild(oldChild)
            document.getElementById("attributeName").value = ''
        }
    }
}
</script>
<style scoped>
</style>
```

ГРАФІЧНА ЧАСТИНА



Змін.	Аркуш	№ докум.	Підпис	Дата	КвРПЗ.2201127.01.06.Е8		
Виконав		Цегельник О.В.		01.06	Схема бази даних	Літ	Аркуш 1 / Аркушів 1
Керівник		Бойко В.О.		01.06			
Н. контр.		Яшина О.М.		01.06		ХНУ, ІПЗс-22-1	
Зав. каф.		Бедратюк Л.П.		01.06			

СУПРОВІДНІ ДОКУМЕНТИ

Завідувачу кафедри інженерії програмного
забезпечення проф. Леоніду БЕДРАТЮКУ
здобувача вищої освіти
Цегельника Олексія Володимировича
факультет ІТ, ІІІ курс, група ІПЗс-22-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності в Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозиторії Хмельницького національного університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-обчислювального комплексу StrikePlagiarism та/або програмно-технічного засобу AntiPlagiarism і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення текстових збігів у роботах.

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

09.06.2025
дата

підпис

Mon Jun 09 21:52:38 EEST 2025, Форкун Юрій Вікторович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 3.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 10%**

	Document		Sum coincidence on the DB	
ID: 244351 Title: БКР_Вебзастосунок для публікації 3D друкованих проєктів Added in a DB: 2025-06-09 Authors: Цегельник Олексій Heads: Бойко Вячеслав Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	76133	1130	5465 (7%)	66 (6%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Цегельник Олексій

Співавтор:

Назва: БКР_Вебзастосунок для публікації 3D друкованих проєктів

Експерт:

Підрозділ: Кафедра інженерії програмного забезпечення

Коефіцієнт подібності 1:4%

Коефіцієнт подібності 2:0.8%

Мікропробіли: 0

Заміна букв: 1

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-07 13:13:32.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

10.06.2025
Дата

експерт

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Цегельник Олексій Володимирович

Тема Вебзастосунок для публікації 3D-друкованих проєктів

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень _____; кількість сторінок записки 85

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі досліджено предметну область 3D-друку та платформи для публікації проєктів. Проведено аналіз існуючих рішень, їхні переваги та недоліки, що підтвердило актуальність створення нового програмного забезпечення. Визначено функціональні та нефункціональні вимоги до вебзастосунку. Розглянуто варіанти архітектури та інструменти реалізації, з яких обрано найбільш доцільні. Розроблений застосунок протестовано на відповідність поставленим вимогам..

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання, містить у собі детальний опис дослідження предметної області, проектування, кодування та тестування. В результаті створено вебзастосунок, що відповідає усім вимогам.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі доведено актуальність теми, поставлено мету та завдання. У першому розділі проаналізовано предметну області, розглянуто існуючі рішення та визначені функціональні і нефункціональні вимоги до вебзастосунку для публікації 3D-друкованих проєктів. У другому розділі досліджено архітектури побудови вебзастосунку, аргументовано вибір клієнт-серверної архітектури, спроектована база даних. У третьому розділі виконано кодування. Були описані моделі, реалізовано взаємозв'язок з базою даних, створені нові динамічні Vue компоненти. Опісля проведено тестування на відповідність та правильну роботу застосунку.

4. Позитивні сторони роботи Тематика кваліфікаційної роботи є актуальною, оскільки 3D друк стрімко розвивається і необхідність в сервісах для публфкації проектів зберігається по цей день. Наявні існуючі програмні забезпечення не надають можливості для створення динамічних атрибутів. У проекті розроблена зручна адмін-панель, що дає змогу легко керувати наповненням вебзастосунку.

5. Негативні сторони роботи У роботі не реалізовано пошук за динамічними атрибутами, та при створенні нових проектів, користувачі обмежені у виборі лише одного виду ресурсних матеріалів, а саме .STL.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Виконання кваліфікаційної роботи демонструє належний рівень підготовки. Робота має чітку структуру, змістовна та свідчить про глибоке опрацювання предметної області. Технічне рішення відповідає актуальним стандартам розробки програмного забезпечення у вигляді вебзастосунку.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «добре».

РЕЦЕНЗЕНТ Юрій Клюк к.т.н., зав. кафедри кібербезпеки, ХНЕУ

"9" 06 2025 р. _____
(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів перевірки роботи, продукованими програмно-технічним засобом (ами), на наявність текстових збігів.
Назва кваліфікаційної роботи: «Вебзастосунок для публікації 3D-друкованих проєктів»
Автор: Цегельник Олексій Володимирович
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Спеціальність: 121 – Інженерія програмного забезпечення
Науковий керівник: Бойко В'ячеслав Олександрович, асистент
Після аналізу звіту/звітів зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається до захисту.	**відповідає**
2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована.	
3	Виявлені запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Виявлені запозичення частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнуті. Робота може бути допущена до захисту після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:
Запозичення, виявлені у роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системою перевірки на плагіат Anti-Plagiarism виявлено схожість з деякими документами у частині загальновживаних обов'язкових словосполучень у стандартних бланках, у структурі змісту, назвах назвах розділів/підрозділів, рамках форм, у назвах та URL-адресах публікацій переліку джерел посилання;

2) запозичення, виявлені у тексті роботи, є фрагментарними.

Максимальний обсяг запозичень, визначений системою Anti-Plagiarism, складає 3.0% з одного джерела. Загальна сумарна подібність у базі даних складає 7% за символами та 6% за лексемами. Крім того, за результатами додаткового аналізу коефіцієнт подібності 1 становить 0.3%, коефіцієнт подібності 2 - 0%. Не виявлено мікропробілів, зайвих білих знаків або маніпуляцій з інтервалами. З урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 10.06.25

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ В'ячеслав БОЙКО