МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___»_____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

«ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ 3D-ВІЗУАЛІЗАЦІЇ ДВОВИМІРНИХ ОБ'ЄКТІВ»

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____Олександр ГОЛОВКО
підпис
«_» _____20__р.

Керівник д-р техн. наук, проф. _____ Альона ШВЕД
підпис
«_» _____20__р.

Миколаїв – 2025

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Другий (магістерський)
Освітній ступень	Магістр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії
програмного забезпечення
_____Євген ДАВИДЕНКО
«___»_____2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

_____**Головко Олександра Володимировича**_____
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
___Програмне забезпечення 3D-візуалізації двовимірних об'єктів
_____

Затверджена наказом ректора ЧНУ ім. Петра Могили від «02» _липня_ 2025 р. № 182

2. Строк представлення кваліфікаційної роботи «____» _____ 2025 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Очікуваним результатом є програмне забезпечення 3D-візуалізації двовимірних об'єктів.

4. Перелік питань, що підлягають розробці
− дослідження предметної галузі;

− специфікація вимог до системи, що розробляється;

− моделювання, проєктування та програмна реалізація системи для 3D-візуалізації двовимірних об’єктів;

− визначення практичного застосування системи;

− тестування системи для 3D-візуалізації двовимірних об’єктів.

5. Перелік графічних матеріалів

Презентація.

6. Завдання до спеціальної частини

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Дата видачі завдання « ____»_____20_____р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Програмне забезпечення 3D-візуалізації двовимірних об’єктів

№ з/п	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівника кваліфікаційної роботи на здобуття освітнього ступеня магістра (КМР)	01.05.2025	02.05.2025	Виконано
2	Отримання завдання на виконання КМР	09.09.2025	14.09.2025	Виконано
3	Складання календарного плану роботи на весь період виконання КМР	18.09.2025	20.09.2025	Виконано
4	Отримання завдання на передатестаційну практику	21.09.2025	01.10.2025	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до КМР	03.10.2025	04.10.2025	Виконано
6	Розробка звіту з переддипломної практики	05.10.2025	10.10.2025	Виконано
7	Виконання КМР: проєктування та моделювання системи, розробка системи	15.10.2025	19.11.2025	Виконано
8	Попередній захист КМР на засіданні комісії кафедри	24.11.2025	24.11.2025	Виконано
9	Доробка та остаточне оформлення КМР	10.12.2025	10.12.2025	Виконано
10	Подання КМР рецензенту	15.12.2025	15.12.2025	Виконано
11	Подання КМР, її електронної копії та інших документів (відгуку, рецензії) до захисту	16.12.2025	16.12.2025	Виконано
12	Захист КМР перед екзаменаційною комісією (ЕК)	22.12.2025	22.12.2025	Виконано

Керівник роботи _____ _Альона ШВЕД_
Особистий підпис *Власне ім’я ПРІЗВИЩЕ*

Здобувач _____ _Олександр ГОЛОВКО_
Особистий підпис *Власне ім’я ПРІЗВИЩЕ*

«_____» _____20_____р.

АНОТАЦІЯ

до кваліфікаційної роботи
на здобуття освітнього ступеня магістра
«Програмне забезпечення 3D-візуалізації двовимірних об’єктів»
Здобувач гр. 608: Головко Олександр
Керівник: д-р техн. наук, проф. Швед Альона

Актуальність дослідження обумовлена потребою у швидких і наочних засобах представлення двовимірних даних у тривимірному просторі для задач вебдизайну, електронної комерції, поліграфії, технічної візуалізації та освіти. Інтерактивна 3D-візуалізація підвищує зрозумілість даних, скорочує цикл прийняття рішень і забезпечує користувачам інструменти для просторового аналізу без встановлення спеціалізованого ПЗ.

Об'єктом дослідження є процес побудови віртуальної 3D-візуалізації 2D-об’єктів.

Предметом дослідження є методи та інструменти створення 3D-візуалізації 2D-об’єктів у браузері з використанням сучасних вебтехнологій: WebGL/Three.js, React, WordPress, PHP та REST API.

Метою роботи є генерація фотореалістичних віртуальних 2D-об'єктів, шляхом розроблення системи, що забезпечує перетворення та інтерактивну 3D-візуалізацію 2D-об’єктів у вебсередовищі, інтегрованої з контент-менеджментом на базі WordPress і взаємодією через REST API.

Кваліфікаційна робота складається із вступу, 4 розділів, висновків та переліку джерел посилання.

У вступі визначено актуальність, сформульовано мету та задачі дослідження, а також об'єкт і предмет дослідження.

Перший розділ подає огляд технологій вебвізуалізації та підходів до обробки 2D-зображень.

Другий розділ описує архітектуру системи: React-фронтенд, WordPress/PHP-бекенд, REST-взаємодію, інтерфейс CMS.

Третій розділ зосереджується на алгоритмах перетворення 2D у 3D і рендерингу сцени, включно з налаштуваннями серверу для використання ML моделей.

Четвертий розділ присвячено реалізації, інтеграції, тестуванню продуктивності та оцінюванню зручності використання.

Результатом роботи є програмне забезпечення, що поєднує React-компоненти для 3D-інтерфейсу з бекендом WordPress/PHP та REST API, забезпечуючи завантаження, обробку й інтерактивну візуалізацію двовимірних об’єктів у 3D.

Кваліфікаційна робота викладена на 74 сторінках (без додатків), містить 4 розділи, 24 рис., 21 джерело посилання і 1 додаток.

Ключові слова: *3D-візуалізація, ML, Canvas, React, WordPress, PHP, REST API, SVG, рендеринг.*

ABSTRACT

to the qualifying master's thesis

Software for 3D visualization of two-dimensional objects

Student of 608 group: Holovko Oleksandr

Supervisor: Dr. Tech. Sciences, Prof. Shved Alyona

The relevance of the study is due to the need for fast and visual means of presenting two-dimensional data in three-dimensional space for the tasks of web design, e-commerce, printing, technical visualization and education. Interactive 3D visualization increases the clarity of data, shortens the decision-making cycle and provides users with tools for spatial analysis without installing specialized software.

The object of the study is the process of building a virtual 3D visualization of 2D objects.

The subject of the research is methods and tools for creating 3D visualization of 2D objects in a browser using modern web technologies: WebGL/Three.js, React, WordPress, PHP and REST API.

The aim of the work is to generate photorealistic virtual 2D objects by developing a system that provides conversion and interactive 3D visualization of 2D objects in a web environment, integrated with content management based on WordPress and interaction via REST API.

The qualification work consists of an introduction, 4 sections, conclusions and a list of references.

The introduction defines the relevance, formulates the goal and objectives of the research, as well as the object and subject of the research.

The first section provides an overview of web visualization technologies and approaches to processing 2D images.

The second section describes the system architecture: React frontend, WordPress/PHP backend, REST interaction, CMS interface.

The third section focuses on 2D to 3D conversion algorithms and scene rendering, including server settings for using ML models.

The fourth section is devoted to implementation, integration, performance testing and usability evaluation.

The result of the work is software that combines React components for a 3D interface with a WordPress/PHP backend and REST API, providing loading, processing and interactive visualization of two-dimensional objects in 3D.

The qualification work is presented on 74 pages (without annexes), contains 4 sections, 24 pictures, 21 sources of references and 1 annex.

Keywords*: 3D visualization, ML, Canvas, React, WordPress, PHP, REST API, SVG, rendering.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

КМР – кваліфікаційна магістрерська робота

ОС – операційна система

ПЗ – програмне забезпечення

AES – Advanced Encryption Standard

API – Application Programming Interface

CDN – Content Delivery Network

CSS – Cascading Style Sheets

CMS – Content Management System

IDE – Integrated Development Environment

MVC – Model-View-Controller

ORM – Object-Relational Mapping

REST – Representational State Transfer

SSL – Secure Sockets Layer

TLS – Transport Layer Security

UI – User Interface

UX – User Experience

ВСТУП

Актуальність теми зумовлена зростаючою потребою у швидких і наочних засобах перетворення двовимірних даних у тривимірні представлення для завдань вебдизайну, електронної комерції, поліграфії, технічної візуалізації та освіти. Інтерактивна 3D-візуалізація підвищує зрозумілість інформації, скорочує цикл прийняття рішень і надає користувачам інструменти просторового аналізу без встановлення спеціалізованого програмного забезпечення. Для систем електронної комерції особливо актуальним є створення 3D-подань товарів на основі наявних 2D-ресурсів (зображень, SVG-графіки), що підвищує залученість користувачів і якість візуальної комунікації з продуктом.

Об'єктом дослідження є процес побудови віртуальної 3D-візуалізації 2D-об’єктів.

Предметом дослідження є методи та інструменти створення 3D-візуалізації 2D-об’єктів у браузері з використанням сучасних вебтехнологій: WebGL/Three.js, React, WordPress, PHP та REST API.

Метою кваліфікаційної роботи є генерація фотореалістичних віртуальних 2D-об'єктів, шляхом розроблення системи, що забезпечує перетворення та інтерактивну 3D-візуалізацію 2D-об’єктів у вебсередовищі, інтегрованої з контент-менеджментом на базі WordPress і взаємодією через REST API.

Для досягнення поставленої мети передбачено комплекс взаємопов’язаних кроків. Насамперед буде досліджено методи та формати перетворення двовимірних даних у тривимірні представлення — від оброблення растрових зображень до інтерпретації векторних форматів на кшталт SVG. Далі буде здійснено критичний аналіз придатності й порівняльних переваг WebGL/Three.js, React, WordPress, PHP та REST API як технологічної основи інтерактивної системи. На цій підставі спроєктовано

архітектуру з React-фронтендом, WordPress/PHP-бекендом, REST-взаємодією та інтерфейсом керування даними в CMS. Передбачено реалізацію зручного користувацького інтерфейсу на React із підтримкою завантаження й керування 2D-ресурсами через WordPress REST API та розроблення рендерингу сцени. Інтегруються серверні модулі попередньої обробки та перетворення, у тому числі з використанням ML-моделей. Фінальний етап охоплює тестування продуктивності, доступності й кросбраузерності (Lighthouse, GTmetrix), підготовку рекомендацій щодо розгортання, безпеки та подальшого супроводу, а також оцінювання юзабіліті та визначення практичних сценаріїв застосування в E-Commerce.

Методи дослідження передбачають аналітичний огляд літератури й вебтехнологій, проєктування та прототипування, експериментальне вимірювання показників продуктивності й доступності, а також юзабіліті-оцінювання інтерфейсу користувача.

Практична цінність полягає у створенні вебрішення, що автоматизує перетворення 2D-ресурсів у 3D-представлення та забезпечує інтеграцію з CMS через REST API для типових бізнес-процесів E-Commerce (поповнення каталогу, прев’ю товарів, інтерактивні конфігуратори). Це сприяє скороченню часу візуальної підготовки контенту, покращенню користувацького досвіду й підвищенню конверсій.

1 ДОСЛІДЖЕННЯ ПРОГРАМНО-ТЕХНІЧНИХ АСПЕКТІВ СИСТЕМ УПРАВЛІННЯ КОНТЕНТОМ

1.1 Структура систем управління контентом

Системи управління контентом (CMS) є складними платформами, що поєднують функціональність створення, зберігання, публікації та доставки цифрового контенту, підтримку редакційних процесів, керування правами доступу та надання аналітики в режимі близькому до реального часу. Дослідження їхніх програмно-технічних аспектів дає змогу виявити, як такі платформи забезпечують надійність, безпеку й ефективність життєвого циклу контенту в умовах високих навантажень і різнорідних каналів доставки.

Ключові технічні аспекти CMS охоплюють архітектуру системи (монолітну, модульну чи мікросервісну), що визначає взаємодію компонентів і можливості масштабування. Важливими є протоколи обміну даними (REST, GraphQL, Webhooks), які впливають на швидкість обробки запитів та затримки між сервісами. До базових аспектів належать безпека (аутентифікація, авторизація на основі ролей, шифрування, аудит), керування даними (схеми БД, індексація, міграції), механізми продуктивності (кешування, CDN, черги повідомлень) і розширюваність через плагіни та API. Запровадження надійних підходів до масштабованості (горизонтальне масштабування, контейнеризація, оркестрація) дозволяє системам витримувати пікові навантаження, зберігаючи узгодженість і доступність контенту.

Розуміння й аналіз зазначених компонентів є необхідними для підвищення ефективності CMS і забезпечення користувачам стабільного, безпечного та зручного середовища керування контентом. Дослідження програмно-технічних аспектів також дає змогу окреслити шляхи еволюції таких систем - зокрема через інтеграцію безголової (headless) архітектури,

CI/CD-конвеєрів і автоматизованого тестування, з метою підвищення швидкодії, спостережуваності та якості доставки цифрового досвіду.

Структура систем управління контентом (CMS) складається з кількох ключових компонентів, кожен з яких виконує специфічні функції для забезпечення цілісності та ефективності всієї платформи. Основні елементи структури CMS включають презентаційний рівень (фронтенд), прикладну логіку (бекенд), сховище контенту (репозиторій/БД), модулі безпеки та аналітичні інструменти для вимірювання взаємодії користувачів з контентом, а саме:

- **фронтенд** — користувацький інтерфейс, через який редактори та відвідувачі переглядають, створюють і публікують матеріали, керують медіа та налаштовують відображення. Цей компонент є критично важливим для зручності роботи та доступності цифрового досвіду;
- **бекенд** — серверна частина, що обробляє редакційні робочі процеси, керує контентними моделями, ролями й правами доступу, а також координує взаємодію з іншими сервісами системи. Бекенд виступає центральним вузлом керування життєвим циклом контенту, тому його продуктивність і надійність безпосередньо визначають якість функціонування платформи;
- **база даних/репозиторій контенту** – система зберігання структурованих записів, медіафайлів, метаданих, журналів змін і версій. У CMS особлива увага приділяється узгодженості даних, швидкості індексації та пошуку, оскільки це впливає на час публікації та стабільність доставки матеріалів.

Ретельно спроєктована структура CMS забезпечує безперебійну роботу, масштабованість, стабільність і безпеку, що є визначальними умовами для ефективного керування контентом у багатоканальному середовищі (веб, мобільні застосунки, інтеграції з зовнішніми системами).

Системи управління контентом функціонують у контексті, де швидкість публікації, надійність зберігання та безпека доступу до даних є критично важливими факторами. Однією з технологічних основ їх структури є розподілена архітектура з використанням кешування, CDN і черг повідомлень. Такий підхід дає змогу розподіляти навантаження між серверами, мінімізувати затримки під час віддачі сторінок і медіа, а також підтримувати стабільну роботу в пікові періоди.

Іншим важливим компонентом є інтерфейс програмування застосунків (англ. Application Programming Interface, API), який забезпечує взаємодію фронтенду, бекенду та сторонніх сервісів. За допомогою API (зокрема REST або GraphQL) редакційні інструменти й інтеграції можуть автоматизувати публікаційні процеси, синхронізувати контент з іншими платформами та отримувати актуальні дані про стан матеріалів. Це відкриває можливості для розробки сторонніх застосунків і «безголових» (headless) клієнтів, що допомагають організаціям оптимізувати контентні потоки та приймати обґрунтовані управлінські рішення [1,2].

1.2 Захист даних у системах управління контентом

Захист даних є одним із найважливіших аспектів функціонування систем управління контентом, оскільки такі платформи обробляють значні обсяги персональної інформації користувачів, редакційні артефакти та службові метадані. Ефективна система безпеки повинна протидіяти зовнішнім атакам, несанкціонованому доступу, шахрайству й внутрішнім зловживанням; у контексті WordPress це включає захист облікових записів адміністраторів і редакторів, конфіденційних налаштувань сайту та цілісності публікацій [3,4].

CMS активно застосовують шифрування як під час передавання, так і зберігання даних. Для транспортного рівня використовується захищений протокол HTTPS на основі SSL/TLS, який забезпечує конфіденційність і

цілісність взаємодії між браузером, API та сервером. Для зберігання чутливих відомостей можуть застосовуватися алгоритми на кшталт AES (Advanced Encryption Standard) у межах СКБД або на рівні застосунків і модулів. У WordPress практикується шифрування конфігураційних секретів, використання унікальних «salts» і ключів автентифікації, а також обмеження прямого доступу до файлів і бази даних за допомогою політик сервера та інструментів керування доступом.

Для додаткового захисту користувачів і адміністраторів широко впроваджується багатофакторна автентифікація (2FA), що вимагає підтвердження особи одноразовими кодами або апаратними/програмними токенами під час входу чи виконання критичних операцій. У середовищі WordPress 2FA реалізується через відповідні модулі, а також через інтеграцію з провайдерами ідентифікації; це суттєво знижує ризик несанкціонованого доступу у разі компрометації пароля та підсилює модель ролей і прав.

Щоб протидіяти розподіленим атакам типу DDoS, які здатні порушити доступність вебресурсу, у CMS застосовуються хмарні служби пом’якшення, зокрема мережі доставки контенту (CDN) і балансувальники навантаження. Такі рішення зменшують тиск на оригінальні сервери, фільтрують шкідливий трафік і підтримують стабільну роботу платформи в пікові періоди. Для WordPress це часто доповнюється веб-аплікаційними міжмережевими екранами (WAF), які блокують типові експлойти рівня застосунку (SQLi, XSS) ще до того, як запит досягне ядра CMS.

Крім того, у системах управління контентом використовуються механізми моніторингу для відстеження активності та виявлення аномалій. Аналітика подій безпеки, кореляція логів і алгоритми машинного навчання дозволяють виявляти нетипову поведінку в реальному часі та оперативно реагувати, запобігаючи інцидентам. У WordPress це доповнюється аудитом змін (журнали входів, модифікацій контенту та конфігурацій), контролем цілісності файлів ядра й плагінів, сегрегацією секретів, регулярними

офлайн-резервними копіями з перевіркою відновлення та застосуванням принципу найменших привілеїв для облікових записів і сервісів.

Сукупність наведених заходів дозволяє CMS, зокрема WordPress, забезпечувати високий рівень захисту даних, підтримуючи довіру користувачів і стійкість публікаційної інфраструктури. Ефективний комплексний підхід знижує ризик втрати конфіденційної інформації та порушення цілісності контенту, що є ключовим чинником надійності сучасних систем управління контентом.

1.3 Дослідження актуальних трендів у розширенні функціональності систем управління контентом

Модульний підхід у системах управління контентом, зокрема в екосистемі WordPress, є ключовою передумовою сталого розвитку платформи та гнучкого розширення її можливостей без порушення цілісності ядра. Технології, що підтримують плагінну архітектуру, постійно удосконалюються для задоволення зростаючих потреб редакційних команд і бізнесу. Серед провідних аргументів на користь модульності вирізняються можливість цілеспрямованого розширення функцій, сумісність з різними архітектурними стилями (у тому числі headless), підвищення зручності редакторського інтерфейсу та системне посилення безпеки. WordPress слугує показовим прикладом того, як плагіни перетворюють базову CMS на платформу з багатою екосистемою функцій.

Одним із найпереконливіших обґрунтувань модульності є автоматизація контентних процесів за допомогою плагінів. Розширення можуть додавати інтелектуальні підказки, генерацію чернеток, автотегування, планування та тригерні робочі процеси, не змінюючи ядро. Це забезпечує швидке й безперервне публікування матеріалів і дає змогу оперативно реагувати на зміни попиту. У WordPress відповідні плагіни для workflow-оркестрації, інтеграцій із сервісами ШІ та попереднього моделювання контенту

підвищують ефективність кампаній і знижують сукупну вартість змін (TCO), адже функціональність можна додавати або вилучати без ризику для стабільності системи.

Другим вагомим аргументом є підтримка headless- і JAMstack-сценаріїв завдяки чітким розширювальним точкам: REST/GraphQL-шарам, вебхукам і фільтрам/хукам WordPress. Плагіни дають змогу інтегрувати CMS з мікросервісами, CDN, чергами повідомлень і сторонніми API, розділяючи фронтенд і бекенд без форку ядра. Такий композиційний підхід полегшує доставку контенту до вебу, мобільних застосунків та IoT-пристроїв, усуває «монолітні» вузькі місця та спрощує еволюційне масштабування за рахунок локалізованих змін у вигляді окремих модулів.

Модульність безпосередньо покращує користувацький інтерфейс редакторів завдяки розширенням для Gutenberg і спеціалізованим плагінам для медіаменеджменту, аналітики й попереднього перегляду. Додавання нових блоків, панелей та інструментів відбувається ізольовано від ядра та інших плагінів, що зменшує ризики конфліктів і дозволяє командам без глибокої технічної підготовки конфігурувати потрібні сценарії роботи. У підсумку це підвищує швидкість прийняття рішень і якість цифрового досвіду без загальної перебудови системи.

Питання безпеки також набуває вигоди від модульного підходу: плагіни для багатофакторної автентифікації, WAF-інтеграцій, контролю цілісності, журналювання й моніторингу дозволяють адресно знижувати ризики доступу та втрати даних. Завдяки розділенню обов’язків (separation of concerns) безпекові функції реалізуються як автономні компоненти з окремими циклами оновлень, що спрощує аудит, пришвидшує виправлення вразливостей і знижує витрати на підтримку.

Отже, модульна архітектура WordPress забезпечує кероване, передбачуване й економічно доцільне розширення функціональності. Вона узгоджує вимоги до безпеки, зручності та ефективності, зберігаючи

стабільність ядра й сумісність екосистеми. Розуміння та практичне застосування цього підходу є критично важливими для стратегій впровадження та еволюції CMS-рішень на базі WordPress.

1.4 Огляд схожих рішень для систем управління контентом в управлінні 3D-об’єктами

У сфері систем управління контентом (CMS) для роботи з 3D-об’єктами існує чимало рішень і платформ, що надають інструменти для завантаження, зберігання, перегляду та конфігурування тривимірних моделей. Такі рішення забезпечують доступ до інтерактивних в’юверів, API-інтерфейсів, засобів оптимізації й, за потреби, інтеграцій з e-commerce, що дозволяє приймати обґрунтовані рішення щодо підбору технологічного стеку. Огляд аналогічних рішень допомагає виокремити найефективніші підходи та виявити недоліки, які можна поліпшити у розроблюваній системі керування 3D-контентом.

Mimeeq – рішення, що надає інструменти для побудови 3D-конфігураторів і відтворення моделей у браузері з акцентом на варіативність і зручність інтеграції. Платформа розрахована на швидке впровадження інтерактивних 3D-компонентів у контентні сторінки.

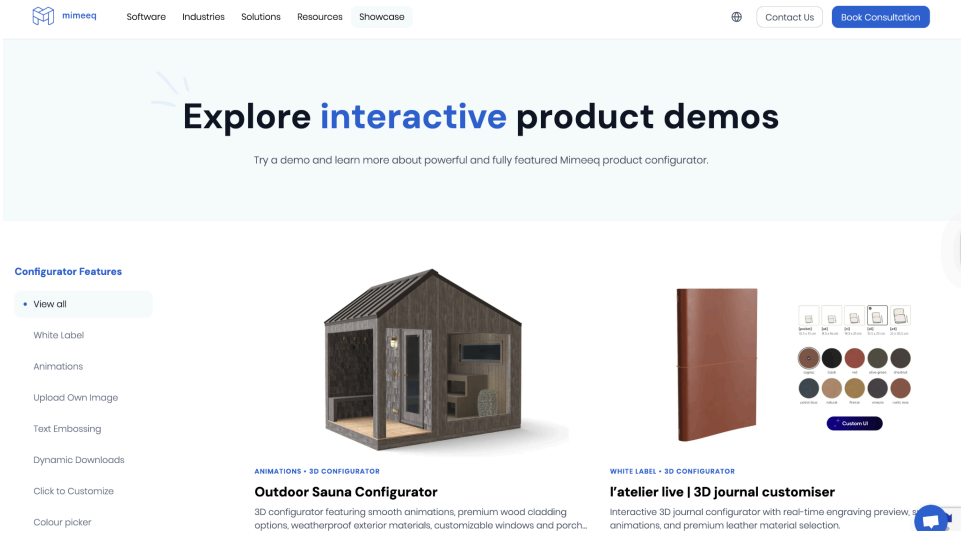

Рисунок 1.1 – Приклади 3D-візуалізацій на платформі Mimeeq

Переваги:

– гнучкий механізм конфігурації моделей із підтримкою різних параметрів і опцій;

– орієнтація на веб-вбудовування та підключення до існуючих публікаційних процесів CMS.

Недоліки:

– обмежений набір нативних інструментів для повного циклу редакційного контролю;

– залежність від якості підготовки вихідних 3D-асетів і оптимізації.

Sketchfab – хмарний сервіс для публікації та управління 3D-активами з веб-в’ювером і можливістю вбудовування в контентні сторінки. Підходить як зовнішній шар зберігання й відтворення моделей.

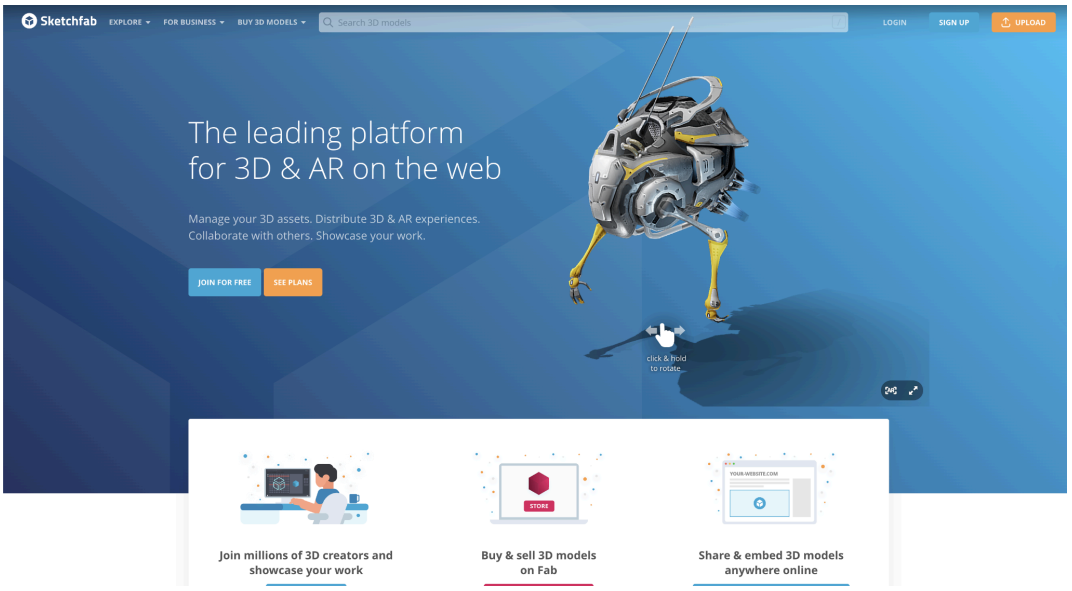

Рисунок 1.2 – Приклад 3D-візуалізації на платформі Sketchfab

Переваги:

– зрілий в’ювер і просте інтегрування у CMS-матеріали через вставки;

– базові можливості командної співпраці та організації бібліотеки моделей.

Недоліки:

– зовнішнє зберігання активів і залежність від стороннього сервісу;

– відсутність розширених інструментів для складних бізнес-правил конфігурації.

Threekit – платформа для побудови 3D/AR-конфігураторів і централізованого керування великою кількістю варіантів моделей, орієнтована на масштабування та пов’язані інтеграції.

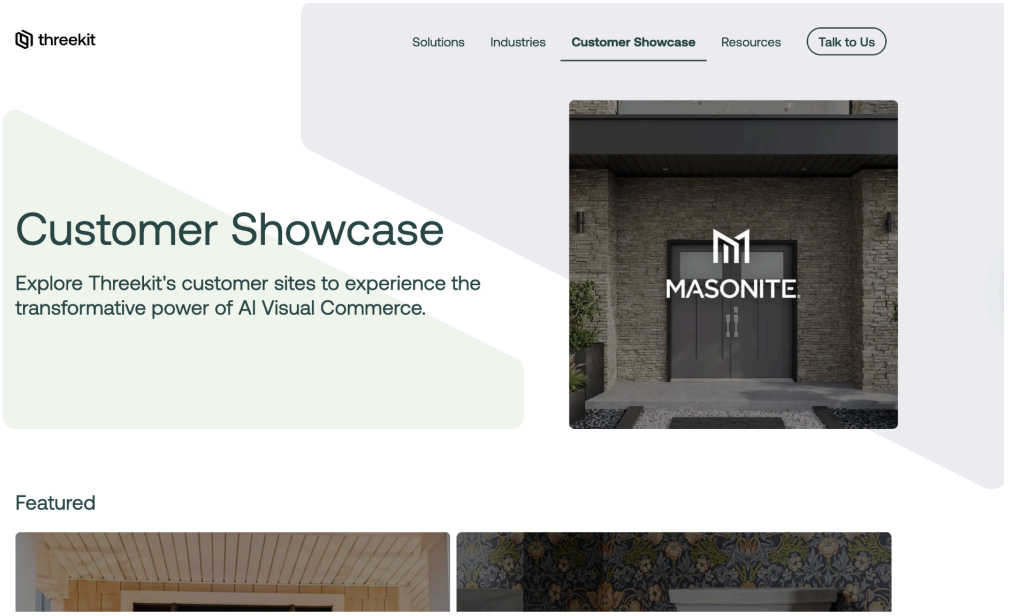

Рисунок 1.3 – Приклад 3D візуалізації на платформі Threekit

Переваги:

– інструменти для керування каталогами 3D-ресурсів і правилом варіантів;

– широкі можливості інтеграції з зовнішніми системами та процесами.

Недоліки:

– вища складність впровадження у невеликих проєктах;

– потреба у ретельній підготовці процесів і ресурсів для досягнення повної віддачі.

Кожна з розглянутих платформ – Mimeeq, Sketchfab і Threekit - має власні сильні сторони та цільові сценарії використання, що відповідають різним потребам користувачів і організацій: Mimeeq надає гнучкі інструменти

для швидкого додавання 3D-конфігураторів у контентні сторінки; Sketchfab виконує роль зовнішнього шару публікації та зберігання 3D-активів; Threekit орієнтовано на масштабні випадки з розгалуженими правилами варіантів. Вибір платформи залежить від вимог до глибини керування 3D-контентом, інтерактивності та інтеграцій у публікаційні процеси CMS.

Висновки до розділу 1

У цьому розділі було ґрунтовно розглянуто основні технічні та архітектурні аспекти сучасних систем управління контентом, з акцентом на їхню внутрішню структуру та принципи взаємодії компонентів. Детально проаналізовано класичну багаторівневу організацію CMS, що включає клієнтський рівень, серверну логіку, сховище контенту й базу даних, а також допоміжні підсистеми безпеки, моніторингу й аналітики. Окрему увагу приділено питанням захисту даних, зокрема використанню шифрування транспортного рівня, ролей і прав доступу, багатофакторної автентифікації, міжмережевих екранів вебзастосунків, а також засобів постійного моніторингу та аудиту подій. Зазначені підходи є критично важливими для забезпечення цілісності контенту, захисту персональних даних користувачів і стабільності редакційних процесів у середовищі CMS.

Значну частину розділу присвячено аналізу актуальних тенденцій розвитку систем управління контентом, серед яких домінують модульність, сервісна орієнтованість та перехід до headless- і JAMstack-архітектур. Показано, що відокремлення рівня представлення від бекенду дозволяє використовувати сучасні JavaScript-фреймворки, зокрема React, для створення гнучких і високопродуктивних інтерфейсів. У контексті WordPress це відображається в еволюції редактора Gutenberg та можливості створення повноцінних плагінів із власним React-інтерфейсом. Окремо наголошено на зростаючій ролі інтерактивного контенту, зокрема 3D-візуалізації, яка стає

важливим чинником підвищення привабливості й інформативності вебресурсів, особливо у сфері електронної комерції та цифрових каталогів.

Крім того, у розділі здійснено огляд суміжних програмних рішень для роботи з 3D-об’єктами в межах CMS, включно з платформами для конфігурації та візуалізації продукції, сервісами хостингу й публікації 3D-активів, а також комплексними комерційними системами, тісно інтегрованими з бізнес-процесами. Аналіз таких рішень дозволив виявити спільні архітектурні підходи та обмеження, а також обґрунтувати доцільність використання WP REST API як універсального інтерфейсу інтеграції плагінів з ядром WordPress і зовнішніми сервісами. Розглянуто можливості створення кастомних маршрутів, схем даних і механізмів контролю доступу, що забезпечують підтримку повного циклу операцій над контентними об’єктами та їх метаданими. Такий підхід сприяє уніфікації доступу до даних і спрощує інтеграцію розширеного функціоналу в існуючі редакційні процеси.

Як базову систему зберігання даних обґрунтовано вибір MySQL з механізмом InnoDB, що забезпечує транзакційність, надійність і повну сумісність з WordPress. У сукупності розглянуті підходи й технології формують міцну теоретичну та практичну основу для подальшої еволюції систем управління контентом і впровадження розширених можливостей, зокрема інтерактивної 3D-візуалізації у вебсередовищі.

2 МЕТОДИ ТА ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ СИСТЕМИ ТРАНСФОРМАЦІЇ 2D ОБ’ЄКТІВ

2.1 Вимоги до програмного забезпечення систем 3D-візуалізації

1) **Призначення та межі проєкту.**
   a) ПЗ призначене для завантаження двовимірних зображень, їх автоматизованого перетворення у тривимірні моделі, інтерактивного перегляду результатів у браузері та експорту артефактів.
   b) Використовується архітектура «WordPress (PHP, WP REST API) + React (адмін-UI) + зовнішній/вбудований модуль інференсу». Єдине джерело істини — серверна частина плагіна. Безпека базується на ролях і можливостях WordPress, нонс-токенах і HTTPS.
   c) У межах плагіна реалізуються: приймання файлів, постановка задач на перетворення, моніторинг станів, публікація GLB/прев’ю, перегляд і експорт. Поза межами: власні алгоритми 3D-генерації (використовується зовнішній/окремий сервіс), редактування 3D-сцен та повноцінний DAM.
2) **Загальний опис.**
   a) Сфера застосування — адміністрування товарного контенту у CMS, підготовка інтерактивних 3D-представлень для карток товарів, каталогів і конфігураторів.
   b) Адміністратори й контент-менеджери WordPress із базовими навичками роботи з медіа; технічні редактори, які керують параметрами якості.
   c) Загальна структура і склад системи – плагін WordPress (PHP) з REST-маршрутами; адмін-UI на React (SPA у межах wp-admin); сховище на MySQL; зовнішній сервіс для перетворень; файлове

сховище завантажень/артефактів.

d) Загальні обмеження – макс. розмір вхідного зображення та підтримувані MIME-типи (JPEG/PNG/SVG, за політикою інсталяції); браузери з підтримкою WebGL 2; робота через HTTPS; обмеження на кількість одночасних задач відповідно до виділених ресурсів.

3) Функції системи.

a) Доступ до екрана плагіна та REST-операцій за ролями WordPress; перевірка нонса.
   i) Вхід: сесія користувача, X-WP-Nonce. Вихід: логаут.
   ii) Інтерфейси доступні лише для користувачів із upload_files/manage_options; усі змінні, що впливають на безпеку, санітизуються; усі виклики через HTTPS.

b) Завантаження 2D-даних.
   i) Приймання зображення як attachment, валідація типу/розміру, збереження у каталозі завантажень.
   ii) Вхід: файл PNG/JPEG/SVG + параметри якості. Вихід: attachment у WP.

c) Постановка та виконання задачі перетворення.
   i) Створення запису задачі, передача параметрів у сервіс обчислень, зміна статусів (queued, running, succeeded, failed).
   ii) Вхід: jobId, src_url. Вихід: статуси та артефакти (GLB, прев’ю).

d) Перегляд і експорт результатів.
   i) Рендер GLB у WebGL-переглядачі, базові маніпуляції камерою, завантаження файлу моделі.
   ii) Вхід: URL GLB/прев’ю. Вихід: інтерактивне відтворення, файл GLB для збереження.

iii) Ініціалізація рендера лише після валідного URL; коректне вивільнення ресурсів.

4) **Вимоги до інформаційного забезпечення.**
   a) Джерела і зміст вхідної інформації (даних) — зображення у форматах PNG/JPEG/SVG;
   b) Нормативно-довідкова інформація — класифікатори MIME-типів; політики розміру завантажень WordPress; правила іменування файлів; налаштування якості, описані у внутрішньому довіднику плагіна.
5) **Вимоги до технічного забезпечення.**
   a) Сервер: x86-64, 2–4 CPU, 4–8 ГБ RAM, SSD; мережа з пропускною здатністю ≥100 Мбіт/с; дисковий простір відповідно до обсягу медіа; за наявності окремого обчислювального сервісу — GPU/виробничий воркер згідно з його специфікацією. Клієнт: сучасний браузер із підтримкою WebGL 2.
6) **Вимоги до програмного забезпечення.**
   a) WordPress 6.x; PHP 8.1+ з розширеннями json, mbstring, curl, gd/imagick; MySQL/MariaDB із підтримкою InnoDB та utf8mb4; веб-сервер Nginx/Apache з HTTPS. На фронтенді — React 18, Three.js (GLTFLoader); підтримка браузерів Chrome/Edge/Firefox/Safari поточних версій. Безпека: обов’язковий HTTPS, перевірка нонсів, контроль ролей, санітизація/ескейпинг даних, ліміти розміру/типу файлів, ведення журналів помилок. Продуктивність: ліниве підвантаження модулів візуалізації, вузькі REST-контракти, адаптивний полінг, кешування короткоживучих станів.

2.2 Сутність систем 3D-візуалізації та особливості їх архітектури

Сьогодні системи перетворення 2D-об’єктів у 3D-моделі набувають дедалі більшого значення для візуалізації та електронної комерції. Такі платформи мають бути не лише функціональними, а й ефективними, стабільними і безпечними, щоб відповідати високим вимогам ринку. Вивчення архітектури цих систем допомагає зрозуміти, як їхні підсистеми взаємодіють між собою, щоб забезпечити зручність роботи користувачів і стабільність виконання завдань.

Сучасні платформи перетворення від 2D до 3D поєднують кілька тісно інтегрованих компонентів. Це модель генерації Hunyuan3D, інфраструктура виконання і масштабування на базі сервісу RunPod, підсистеми зберігання і керування даними, а також інструменти перевірки результатів, де ComfyUI використовується для тестових сценаріїв. Ефективний аналіз охоплює бази даних і сховища, обчислювальну архітектуру, засоби захисту даних, конвеєри попередньої та післяобробки. Якісна система має відповідати двом ключовим критеріям. По-перше, потрібна цілісна архітектура, яка об’єднує підготовку датасетів, інференс Hunyuan3D, постобробку і API видачі результатів в один керований процес. По-друге, складність даних і операцій вимагає продуманих пайплайнів, що здатні опрацьовувати великі масиви зображень, маски, текстури і метадані як у пакетному режимі, так і в режимі, близькому до реального часу.

Система трансформації 2D-об’єктів має надавати розробникам плагінів, контент-редакторам і фахівцям з контролю якості актуальні 3D-результати з відтворюваною геометрією і матеріалами, щоб можна було приймати зважені рішення щодо публікації. На практиці не всі платформи однаково ефективні. Відсутність прозорої оркестрації інференсу, недостатня автоматизація перевірок і нестача масштабованості здатні погіршити користувацький досвід.

Система перетворення включає кілька ключових компонентів. Це інтерфейс для запуску і контролю процесів, серверна частина для обробки запитів, сховища даних для зображень і результатів, а також модулі захисту і інструменти контролю якості. У нашому випадку ядром генерації є модель Hunyuan3D, інфраструктуру для обчислень і масштабування надає сервіс RunPod, а для тестування працездатності і перевірки результатів використовується ComfyUI. Для дослідження такої системи важливо розуміти, як ці компоненти поєднуються і працюють разом.

Платформа має мати цілісну архітектуру, що забезпечує безперебійну роботу окремих підсистем. Йдеться про узгоджену взаємодію сховищ даних, модулів безпеки і конвеєрів обробки зображень. Правильне налаштування цих частин дає змогу отримувати стабільні результати, відстежувати стан завдань і швидко знаходити помилки. Це напряму впливає на надійність і передбачуваність роботи всієї системи.

– Масштабованість і продуктивність. Архітектура має передбачати можливість зростання навантаження, коли збільшується кількість користувачів і обсяг даних. Для цього потрібні механізми розширення обчислювальних ресурсів і черг завдань, розподіл обробки між кількома вузлами, кешування проміжних результатів і грамотна організація зберігання. Інфраструктура RunPod допомагає додавати графічні процесори за потреби, а це зменшує час очікування і підтримує стабільний час відгуку.

– Гнучкість для інтеграції нових функцій. Оскільки вимоги швидко змінюються, архітектура повинна дозволяти безпечне і швидке додавання нових етапів у конвеєр. Це може бути новий модуль попередньої обробки, інший спосіб оцінки якості або додатковий формат експорту. ComfyUI корисна для побудови і перевірки таких сценаріїв, а чіткі інтерфейси між компонентами спрощують інтеграцію нових елементів без переробки ядра системи.

Розуміння сутності систем і особливостей їх архітектури дає змогу створити більш ефективне, гнучке і стабільне рішення для трансформації 2D-об’єктів у 3D-моделі. Це допомагає розробникам враховувати актуальні вимоги, планувати масштабування і підвищувати якість результатів. Поєднання моделі Hunyuan3D, інфраструктури RunPod і середовища ComfyUI формує практичну основу для надійної роботи та зручної підтримки всієї платформи.

2.2.1 Сутність систем трансформації 2D-об’єктів

Системи трансформації 2D-об’єктів у 3D-моделі стають важливими інструментами для розробників, дизайнерів та контент-команд. Такі системи створені для швидкого перетворення зображень у придатні до використання 3D-результати, що допомагає приймати обґрунтовані рішення щодо публікації та подальшої обробки. Платформи цього типу поєднують кілька компонентів, які виконують основні функції. Це підсистема завантаження і підготовки даних. Модуль генерації, у нашому випадку модель Hunyuan3D. Засоби післяобробки для очищення геометрії і текстур. Сховища для збереження проміжних і фінальних артефактів. Інструменти звітування та контролю якості. Інфраструктурна частина базується на екосистемі RunPod, що дає змогу розгортати і масштабувати обчислення на графічних процесорах. Для тестування працездатності, швидких експериментів і перевірки графів обробки використовується ComfyUI.

2.2.2 Особливості архітектури систем трансформації 2D-об’єктів

Архітектура систем трансформації 2D-об’єктів у 3D-моделі повинна відповідати вимогам стабільності, продуктивності, масштабованості та безпеки, адже такі рішення працюють з великими обсягами зображень і метаданих та обслуговують багатьох користувачів одночасно. Головна мета архітектурних рішень полягає у забезпеченні злагодженої взаємодії між

основними компонентами системи. Йдеться про інтерфейс керування процесами, серверну частину для інференсу та оркестрації, бази й сховища даних для вхідних і вихідних артефактів, а також модулі контролю якості та безпеки. У нашому випадку генерацію забезпечує модель Hunyuan3D, інфраструктуру для обчислень і розгортання надає екосистема RunPod, а перевірку працездатності та тестування графів обробки виконує ComfyUI. Кожен із цих компонентів робить внесок у надійний і зручний доступ до функцій платформи, а також у захист даних і результатів.

Масштабованість є однією з головних характеристик архітектури, адже кількість запитів і обсяг зображень можуть різко зростати залежно від навантаження. Масштабування дає змогу системі стабільно працювати під час піків. Це досягається збільшенням обчислювальних ресурсів, додаванням нових вузлів обробки та розподілом черг завдань між окремими інстансами. RunPod допомагає гнучко під’єднувати додаткові графічні процесори, що скорочує час інференсу Hunyuan3D і зберігає швидку реакцію системи навіть за великої кількості одночасних операцій.

Ще одна важлива особливість архітектури полягає у гнучкості для інтеграції нових функцій. Потреби користувачів швидко змінюються, тому система має підтримувати безпечне і швидке додавання нових етапів у конвеєр. Це можуть бути модулі попередньої обробки зображень, інші методи оцінки якості, нові формати експорту або альтернативні сценарії постобробки. ComfyUI спрощує побудову і перевірку таких сценаріїв, а чіткі програмні інтерфейси між компонентами дають можливість розширювати функціональність без переробки ядра.

Забезпечення високого рівня безпеки є ключовим аспектом архітектури. Системи обробляють вихідні матеріали, користувацькі дані та результати, тому стають потенційними мішенями для атак. Архітектурні рішення повинні враховувати багаторівневі механізми захисту. Потрібне шифрування під час передавання і зберігання, контроль доступу за ролями, аудит подій, захист від

перевантаження та ізоляція середовищ інференсу. Важливим є надійне зберігання проміжних і фінальних артефактів, щоб запобігти втратам і зберегти довіру до платформи.

Отже, архітектура системи трансформації 2D-об’єктів має бути ретельно спланованою, щоб забезпечити стабільну, продуктивну і безпечну роботу. Такий підхід допомагає платформі залишатися конкурентоспроможною та ефективною в умовах зростання вимог. Поєднання моделі Hunyuan3D, інфраструктури RunPod та середовища ComfyUI підтримує зручний і надійний доступ до інструментів, пришвидшує інтеграцію нових можливостей і гарантує якість результатів для користувачів.

2.3 Огляд методів 3D-візуалізації

Огляд і аналіз методів та засобів для вирішення завдань КМР охоплює інструменти розробки, інфраструктуру виконання і засоби перевірки працездатності. PHPStorm є професійним кросплатформним середовищем розробки, яке надає розширені можливості для PHP, а також підтримує JavaScript, HTML, CSS і TypeScript. Це корисно для проєктів, де поєднуються різні технології під час створення складних рішень. У межах нашого проєкту PHPStorm забезпечує середовище для інтеграції React, Redux, Tailwind CSS і JSX, що допомагає створити зручний та продуктивний інтерфейс для модулів. Додатково планується використання екосистеми RunPod для розгортання інфраструктури та ComfyUI для тестування працездатності окремих сценаріїв.

Недоліки PhpStorm:

– вартість, оскільки це платний продукт і для частини розробників це може бути бар’єром;

– вимоги до ресурсів, адже IDE споживає помітний обсяг пам’яті та процесорного часу, що впливає на швидкодію на старших комп’ютерах;

– складність освоєння, бо середовище містить багато функцій і інструментів, тож новачкам потрібен час, щоб звикнути;

– навантаженість інтерфейсу, через що інколи важко швидко знайти потрібні опції.

Переваги PhpStorm:

– широка підтримка технологій, зокрема PHP, JavaScript, TypeScript, CSS і HTML, що полегшує побудову інтерфейсу і логіки KMP;

– інтеграція з Git, яка спрощує контроль версій і відстеження змін у коді при роботі з багатьма компонентами;

– розвинені інструменти аналізу та рефакторингу коду, підказки та автозавершення, що підвищують ефективність під час роботи з великими проєктами;

– вбудована підтримка роботи з базами даних, що важливо для операцій з клієнтськими записами, угодами і журналами подій, а також зручна робота з SQL прямо з IDE;

– інструменти для відладки, включно з відлагоджувачем PHP і можливостями налагодження фронтенду, що корисно для інтерактивних елементів інтерфейсу.

PHPStorm разом із RunPod і ComfyUI надає усе необхідне для реалізації проєкту з завдань KMP. Середовище дозволяє ефективно працювати з кількома мовами програмування, контролем версій, базами даних і відладкою коду. Таке поєднання підходить для створення складних і масштабованих систем, де важливі зручність, стабільність і можливість швидко перевіряти працездатність окремих сценаріїв [13].

2.4 Аналіз та підбір технологій для розробки системи

Розробка системи вимагає уважного підбору технологій, які забезпечать ефективність, зручність і масштабованість. Вибір має відповідати вимогам до обробки великої кількості даних, швидкої взаємодії з користувачем та

інтеграції з зовнішніми джерелами. У нашому випадку важливо підтримувати оновлення інформації майже в реальному часі, надійну роботу інтерфейсу і безпечний обмін даними. Обрані інструменти мають гарантувати стабільність роботи, просте розширення функцій і можливість підключення нових модулів без суттєвих змін у коді.

Окремої уваги потребує безпека. Система працюватиме з конфігураціями, користувацькими матеріалами і службовими метаданими, тому потрібні сучасні методи автентифікації, шифрування і контролю доступу. Це стосується як серверної частини на PHP, так і взаємодій через REST API. Додатково варто врахувати захист точок входу до адмін-панелі WordPress та перевірку прав на виконання операцій, щоб запобігти небажаним змінам і витокам даних.

Ще один критерій вибору технологій це інтеграція з зовнішніми сервісами. Система має легко звертатися до API, отримувати і віддавати дані у форматі JSON, а також підтримувати підключення інструментів для обробки і тестування. Гнучкість обраних рішень дозволить швидко реагувати на зміни вимог, додавати нові можливості і не перевантажувати кодову базу.

Загалом, правильний набір технологій визначає зручність, продуктивність і надійність системи. Обрана комбінація повинна підтримувати високе навантаження, стабільний час відгуку та простий розвиток. Це забезпечить швидкий доступ до інструментів і надійні механізми взаємодії між компонентами.

Основні технології для реалізації системи:

PHP і WordPress. PHP є мовою для серверної логіки, а WordPress надає перевірену основу для управління контентом і плагінів. Плагінна модель спрощує розширення функцій і підтримку ролей та прав.

Недоліки:

– залежність від конфігурації хостингу і модулів;
– потреба у регулярних оновленнях ядра та плагінів;

– можливі конфлікти розширень, якщо порушені кращі практики.

Переваги:

– велика спільнота і документація;

– готові інструменти для авторизації, локалізації, роботи з БД;

– швидкий запуск завдяки темам і плагінам, спрощене адміністрування.

React. React використовується для побудови адміністративних і користувацьких інтерфейсів з повторно використовуваних компонентів. Це корисно для панелей плагіна та інтерактивних віджетів.

Недоліки:

– швидкі зміни в екосистемі і потреба стежити за оновленнями;

– необхідність у додаткових бібліотеках для маршрутизації, стану і запитів;

– вищий поріг входу для початківців.

Переваги:

– висока продуктивність завдяки віртуальному DOM;

– компонентна структура, що полегшує підтримку;

– односторонній потік даних і передбачувана робота стану;

– велика кількість готових бібліотек для типових задач.

REST API. WP REST API забезпечує обмін даними між фронтендом і бекендом у форматі JSON. Це основа для інтеграції з іншими системами і сервісами.

Недоліки:

– потреба у чіткому версіонуванні маршрутів;

– додаткові перевірки безпеки і прав доступу;

– необхідність оптимізації запитів і кешування.

Переваги:

– уніфікований протокол взаємодії;

– простота інтеграції клієнтських застосунків;

– підтримка фільтрації, пагінації і розширюваних схем.

RunPod. Екосистема RunPod використовується для розгортання інфраструктури і підключення ресурсів під навантаженням. Підходить для окремих сервісів, що потребують виділених ресурсів.

Недоліки:

– додаткові витрати на інфраструктуру;
– потреба у моніторингу і стеженні за квотами;
– налаштування безпеки і мережевих політик.

Переваги:

– гнучке масштабування під час піків;
– ізольовані середовища для експериментів;
– швидке розгортання сервісів без складної підтримки заліза.

ComfyUI. ComfyUI застосовується для тестування працездатності сценаріїв, перевірки графів обробки і швидких експериментів з інтеграціями.

Недоліки:

– потреба у налаштуванні пайплайнів під конкретний проєкт;
– обмеження в автоматизації без додаткових скриптів;
– необхідність узгодження з політиками безпеки і доступів.

Переваги:

– наочне складання і перевірка обробки;
– швидке порівняння результатів і логів;
– зручність для демонстрацій і приймання змін.

Серверна частина відповідає за обробку запитів і обмін даними між клієнтом та базою. Використання PHP з WordPress дає контрольоване середовище з готовими механізмами авторизації і розмежування прав. Інтерактивний інтерфейс на React покращує взаємодію, дає зрозумілі компоненти і швидке оновлення стану. Стилізація може виконуватися стандартними засобами WordPress або будь-якими сумісними бібліотеками. Це знижує кількість власного коду і спрощує підтримку в майбутньому.

Система контролю версій на базі Git забезпечує командну роботу, історію змін і безпечне додавання нових функцій. Це підвищує надійність процесу розробки і зменшує ризик помилок під час інтеграції. Автоматичні перевірки, збірки і тестові розгортання на інфраструктурі RunPod дозволяють швидко бачити результат і оцінювати стабільність.

Підсумовуючи, обрані технології підтримують високе навантаження, постійну інтерактивність і захист даних. Поєднання PHP і WordPress для ядра, React для інтерфейсу, REST API для обміну, RunPod для інфраструктури і ComfyUI для тестування формує збалансовану систему. Вона дає швидкий доступ до інструментів, стабільну роботу і можливість розвиватися без зайвих витрат часу. Це відповідає вимогам до сучасних рішень, які мають бути зручними, продуктивними і безпечними.

2.5 Використання технологій для створення системи перетворення 2D-об’єктів

У сучасній системі перетворення 2D-об’єктів у 3D-моделі важливо забезпечити високу продуктивність, швидку обробку даних і зручність для користувачів. Для цього потрібно правильно підібрати та впровадити технології на кожному етапі створення платформи. Йдеться про інтерфейс користувача, серверну частину, модулі обробки зображень і зберігання результатів. Важливо також мати підтримку роботи майже в реальному часі, щоб швидко отримувати і публікувати 3D-моделі.

Інтерфейс користувача є одним з ключових компонентів, адже саме через нього відбувається завантаження зображень, налаштування параметрів і перегляд результатів. Для створення інтерактивного та динамічного інтерфейсу доречно використовувати сучасні бібліотеки. Вони допомагають оновлювати дані без перезавантаження сторінки, працювати з великими обсягами файлів та надавати зручні інструменти для стилізації. Завдяки

цьому користувач отримує зрозуміле і адаптивне середовище, яке відповідає очікуванням щодо швидкості та простоти.

Серверна частина виконує роль основного обробника запитів і координатора черг завдань. Вона приймає зображення, викликає модулі перетворення і повертає готові 3D-результати. Для обробки великої кількості запитів потрібні технології, які підтримують паралельне виконання і швидке реагування на зміни. Це дає можливість масштабувати систему без втрат продуктивності, що особливо важливо, коли одночасно працює багато користувачів або обробляється велика черга задач.

Для зберігання вхідних матеріалів і згенерованих моделей використовується надійна база даних та файлові сховища. Вони забезпечують швидкий доступ до інформації, збереження метаданих, історію запусків і версій результатів. Важливо, щоб сховище могло витримувати часті операції запису та читання, підтримувало резервне копіювання і швидке відновлення у разі збою. Це дозволяє не втрачати цінні дані і підтримувати безперервність роботи.

Система перетворення повинна надавати інструменти для обробки і контролю якості. Для цього впроваджуються модулі попередньої та післяобробки, що працюють з масками, сіткою та текстурами. Доречним є підключення зовнішніх сервісів і API, які покращують окремі етапи конвеєра. Також корисні графічні інструменти для візуальної перевірки результатів і порівняння різних налаштувань. Це допомагає швидко оцінити якість та прийняти рішення про публікацію або повторний запуск.

Безпека є критично важливою для платформ, які працюють з користувацькими матеріалами. Використані технології повинні забезпечувати автентифікацію, шифрування під час передавання і захист від зовнішніх загроз. Потрібен контроль доступу до завантажених файлів і результатів, аудит подій і моніторинг підозрілої активності. Такі заходи підтримують

довіру користувачів і допомагають зберігати стабільну роботу навіть під час підвищеного навантаження.

Контроль версій є невід’ємною частиною розробки такої системи. Він дозволяє відстежувати зміни в коді, швидко впроваджувати нові функції і виправляти помилки. Команді простіше перевіряти сумісність модулів, проводити огляди коду і повертатися до стабільних версій у разі потреби. Це підвищує надійність розробки і зменшує ризик збоїв під час інтеграції.

Отже, кожна з обраних технологій виконує свою роль у побудові цілісної та надійної системи перетворення 2D-об’єктів. Разом вони дають змогу користувачам швидко працювати з даними, отримувати якісні 3D-моделі та бути впевненими в безпеці матеріалів. Такий підхід забезпечує стабільність, високу продуктивність і зручність, що є ключовими вимогами для сучасних рішень у сфері комп’ютерної графіки.

Висновки до розділу 2

У цьому розділі детально розглянуто і проаналізовано сучасні методи та технології, потрібні для створення системи перетворення 2D-об’єктів у 3D-моделі. Правильний підбір інструментів є одним з ключових чинників успіху проєкту, адже від нього залежать швидкість обробки даних, зручність інтерфейсу, стабільність роботи та можливість подальшого масштабування.

Сучасні бібліотеки і фреймворки, обрані для розробки інтерфейсу, дають змогу побудувати гнучке середовище з інтерактивними елементами. Інтерфейс на React швидко відображає стан черг, прогрес інференсу і результати перетворення. Користувачі оперативно отримують оновлення без перезавантаження сторінок. Це важливо для систем, де швидкий доступ до інформації і простота використання мають пріоритет.

Серверна частина забезпечує стабільну роботу під навантаженням і підтримує паралельну обробку запитів. PHP і WordPress керують плагінами, маршрутами і чергами завдань, а WP REST API відповідає за обмін даними

між інтерфейсом і бекендом. Такий підхід дає змогу обслуговувати багатьох користувачів одночасно, підтримувати швидкий доступ до сховищ і служб перетворення. Контроль версій на базі Git спрощує командну роботу, дозволяє відстежувати зміни і зменшує ризик помилок під час інтеграції.

Безпека даних розглядається як критично важливий аспект. Застосовуються сучасні методи автентифікації, шифрування та контролю доступу. Захищаються завантаження, проміжні артефакти і фінальні моделі. Використовуються журнали подій і моніторинг. Ізольовані середовища в RunPod допомагають відокремити обчислення і зменшити ризики.

Підсумовуючи, у розділі обґрунтовано вибір технологій, які забезпечують потрібний рівень продуктивності, гнучкості і масштабованості. Ядро на PHP і WordPress, інтерфейс на React, обмін через WP REST API, інфраструктура на RunPod і тестування в ComfyUI разом утворюють надійну основу. Вони допомагають підтримувати стабільну роботу, зручне керування процесами та швидкий розвиток системи.

Дослідження також охопило роль зовнішніх і внутрішніх API у побудові ефективного конвеєра. Завдяки WP REST API плагін взаємодіє з бекендом і службами перетворення, передає параметри інференсу та отримує результати. Обчислювальні вузли в RunPod викликаються через REST запити, що дає змогу оперативно запускати обробку і повертати готові 3D-моделі для подальшої публікації.

3 АРХІТЕКТУРА, МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Діаграма прецедентів

Передусім необхідно спроєктувати архітектуру, яка охоплює всі рівні. Йдеться про клієнтську частину, серверну частину, базу даних та інтеграції із зовнішніми джерелами. Особлива увага приділяється створенню стійкої та масштабованої структури. Вона має обробляти великі обсяги даних у режимі, близькому до реального часу, та забезпечувати стабільну роботу навіть за значного навантаження. Розробка структури та взаємодії між компонентами орієнтується на зручність для розробників і користувачів. Система має легко піддаватися модифікаціям і тестуванню. Інтерфейс користувача проєктується так, щоб забезпечити швидкий доступ до необхідної інформації, мінімізувати кількість дій і підтримати аналітичні сценарії. Застосовуються принципи модульності, повторного використання та чіткої навігації.

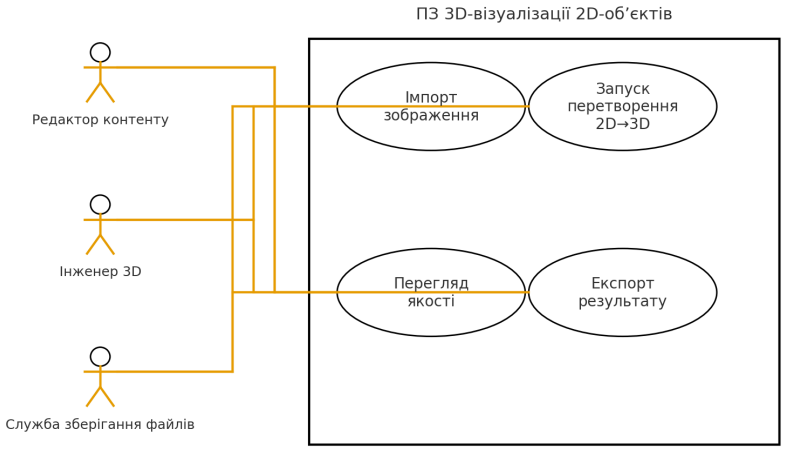

Рисунок 3.1 – Діаграма прецедентів

У цьому підрозділі зосереджено увагу на створенні моделей об’єкта та предмета дослідження, які відображають ключові аспекти системи, її архітектуру, основні процеси та взаємодію компонентів.

Для наочності і уточнення вимог буде підготовлено чотири UML діаграми. Вони допоможуть деталізувати роботу системи перетворення 2D-об’єктів у 3D-моделі й показати її структуру в межах дослідження.

Діаграма прецедентів, або діаграма варіантів використання, є засобом подання функціональних вимог і взаємодій із зовнішніми елементами та користувачами. Вона показує, як різні ролі співпрацюють із системою, щоб досягти потрібних цілей. Подана діаграма прецедентів формалізує функціональні вимоги до програмного забезпечення 3D-візуалізації двовимірних об’єктів для E-Commerce та демонструє взаємодію користувацьких і зовнішніх ролей із системою в межах чітко окресленої границі. У рамці системи розміщено чотири базові прецеденти, що утворюють мінімально достатній контур процесу: імпорт вхідного зображення, запуск перетворення 2D у 3D, перегляд показників якості та експорт отриманого 3D-результату. Така композиція забезпечує пряме відображення життєвого циклу даних — від надходження медіа до видачі готового артефакту для подальшого використання в електронній комерції.

Зовні щодо межі системи визначено три актори. Редактор контенту ініціює імпорт вихідних зображень, перевіряє отриману якість і здійснює експорт, що відповідає його ролі в публікаційних конвеєрах інтернет-крамниць. Інженер 3D відповідає за технічну стадію — запуск процесу перетворення та валідацію якості — забезпечуючи коректну реконструкцію геометрії й текстур. Служба зберігання файлів представлена як зовнішній сервіс і взаємодіє з системою в операціях імпорту та експорту, що відображає інтеграцію з корпоративною інфраструктурою зберігання контенту.

3.2 Діаграма послідовності

Діаграма послідовності виступає динамічною моделлю, що показує порядок взаємодій об’єктів під час виконання конкретного сценарію. Вона

фокусується на хронології обміну повідомленнями між компонентами і дозволяє оцінити узгодженість кроків.

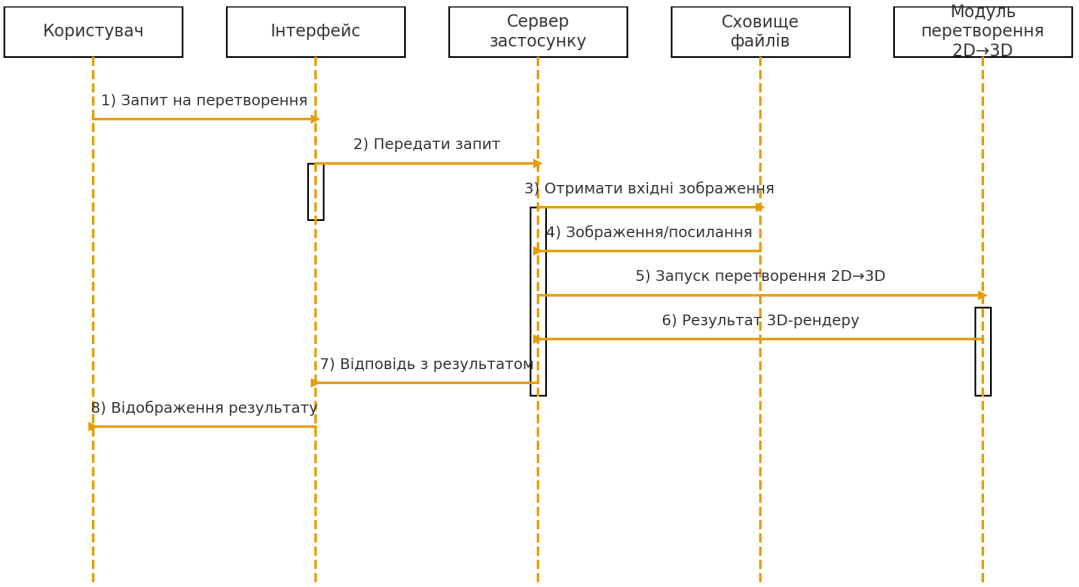

Рисунок 3.2 – Діаграма послідовності

Діаграма моделює динамічну поведінку програмного забезпечення 3D-візуалізації двовимірних об’єктів у межах окремого сценарію, фіксуючи хронологію обміну повідомленнями між компонентами системи. У діаграмі учасники взаємодії представлені як окремі об’єкти, а повідомлення відображають виклики функцій і передачу даних, впорядковані зверху вниз за часовою віссю. Розглянутий сценарій відтворює отримання результату перетворення, де користувач ініціює запит через інтерфейс, далі інтерфейс передає запит серверній частині, сервер звертається до сховища для доступу до вхідних зображень і ініціює обчислення у модулі перетворення 2D у 3D. Після завершення обробки сервер формує відповідь і повертає її в інтерфейс для відображення результату користувачеві. Така репрезентація наочно демонструє послідовність кроків, узгодженість взаємодій і точки інтеграції, необхідні для досягнення цільової мети — отримання коректного та своєчасного 3D-рендеру з 2D-зображень.

3.3 Діаграма розгортання

Діаграма розгортання є інструментом для відображення фізичного розміщення програмних компонентів на апаратних або віртуальних ресурсах. Вона уточнює інфраструктурні зв’язки і допомагає планувати розміщення сервісів.

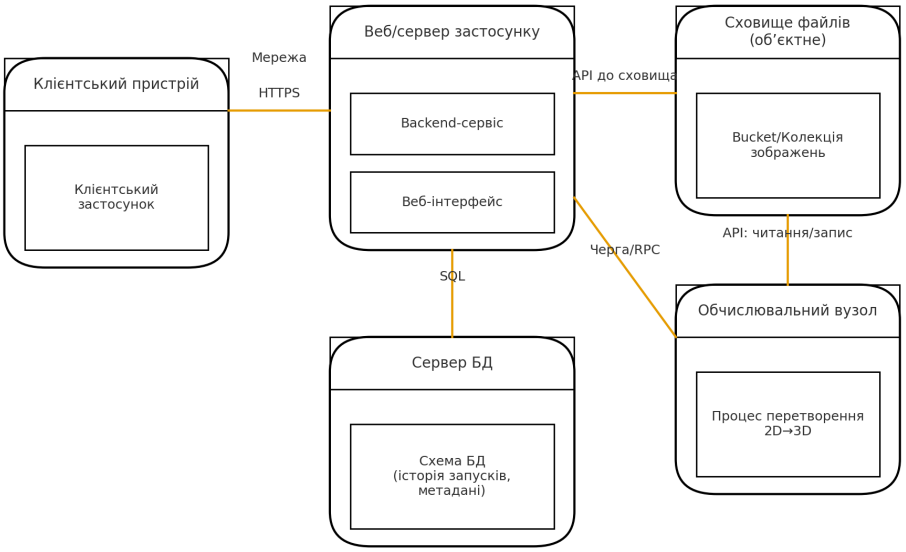

Рисунок 3.3 – Діаграма розгортання

Діаграма формалізує фізичне розміщення компонентів програмного забезпечення 3D-візуалізації двовимірних об’єктів на обчислювальних ресурсах і відображає канали їхньої взаємодії.

У межах моделі вузлами виступають клієнтський пристрій, веб/сервер застосунку, сервер бази даних, об’єктне сховище файлів і окремий обчислювальний вузол для перетворення. До кожного вузла прив’язано відповідні артефакти: на клієнті розміщується клієнтський застосунок; на веб/сервері — веб-інтерфейс і backend-сервіс; на сервері бази даних — схема БД з метаданими та історією запусків; у сховищі — колекція вхідних зображень і проміжних артефактів; на обчислювальному вузлі — процес перетворення 2D у 3D. Шляхи комунікації окреслюють мережеві протоколи та інтерфейси: клієнт взаємодіє із сервером через HTTPS; сервер застосунку

звертається до бази даних за допомогою SQL, до сховища — через програмний API, а обчислювальний вузол задіюється через чергу повідомлень або віддалений виклик процедур.

Розроблювана система матиме багаторівневу структуру, що складається з клієнтської частини, серверної частини, бази даних і зовнішніх сервісів. Така побудова забезпечує високий рівень інтерактивності, продуктивності та масштабованості, що важливо для роботи з великими обсягами даних і обробки запитів майже в реальному часі.

Серверна частина працюватиме у середовищі WordPress та виконуватиме обробку запитів від клієнтської частини, інтеграцію з базою даних і взаємодію з зовнішніми сервісами. Сервер відповідатиме за основну логіку системи, включаючи перевірку прав доступу, агрегацію даних, формування результатів і запис змін. Через WP REST API сервер надає маршрути для читання і запису, обробляє запити на отримання історичних даних з БД та актуальних від зовнішніх джерел, а потім повертає підготовлені відповіді клієнту. Такий підхід дозволяє працювати з великим обсягом інформації, не перевантажуючи інтерфейс, і зберігати стабільність під час пікових навантажень.

База даних є основним сховищем. У ній зберігатимуться історичні записи, профілі користувачів, результати обробки та звіти. Як СУБД використовується MySQL, оскільки вона забезпечує надійне зберігання, швидкий доступ і гнучкі запити. У базі зберігатимуться всі ключові дані, потрібні для подальшої аналітики, а також налаштування користувачів і службові метадані. Це дає змогу формувати персоналізовані подання та забезпечує швидкий пошук у великому масиві записів.

Зовнішні сервіси слугуватимуть джерелом актуальної інформації та виконуватимуть спеціалізовані задачі. Для цього використовується окремий модуль APIConnector, який отримує дані із зовнішніх джерел і передає їх серверній частині. Залежно від частоти оновлення, APIConnector регулярно

звертається до потрібних ресурсів і надає найсвіжішу інформацію. Дані можуть зберігатися в БД для подальшої обробки або одразу повертатися клієнту для відображення.

Окремий компонент прикладної логіки розташовано на сервері. Він відповідає за виконання розрахунків, підготовку агрегатів і перевірку якості даних. Результати обробки записуються в базу, щоб клієнтська частина мала до них швидкий і зручний доступ. Це спрощує повторне використання підготовлених матеріалів і зменшує затрати на повторні обчислення.

Уся система спирається на чітку взаємодію між клієнтською частиною, сервером, базою даних і зовнішніми сервісами. Клієнт звертається до сервера через WP REST API. Сервер працює з БД і зовнішніми джерелами, формує відповіді та повертає їх клієнту для відображення у зручному вигляді. Така взаємодія забезпечує швидку і ефективну роботу системи, надає повний набір даних і мінімізує затримки.

Отже, структура системи складається з узгоджених клієнтської частини, сервера, бази даних і зовнішніх сервісів. Кожна частина виконує свої чіткі функції. Це забезпечує гнучкість, масштабованість і високу швидкодію, потрібні для якісної роботи платформи.

Діаграма моделі системи показує ключові компоненти та їхню взаємодію. На ній видно, що клієнтська частина на React є інтерфейсом для користувача. Вона приймає дії користувача і надсилає запити через WP REST API на сервер. Серверна частина обробляє ці запити, звертається до MySQL для збережених даних і до зовнішніх сервісів для актуальної інформації. Після обробки сервер повертає підготовлені результати клієнтській частині, де вони відображаються у реальному часі. Така схема забезпечує швидкий обмін даними між усіма частинами системи, підтримує актуальність інформації та дає змогу виконувати глибоку аналітику без зайвих затримок.

3.4 Діаграма діяльності

Діаграма діяльності відображає операційний потік системи 3D-візуалізації двовимірних об’єктів і формалізує порядок виконання дій, точки прийняття рішень та паралельні гілки обробки.

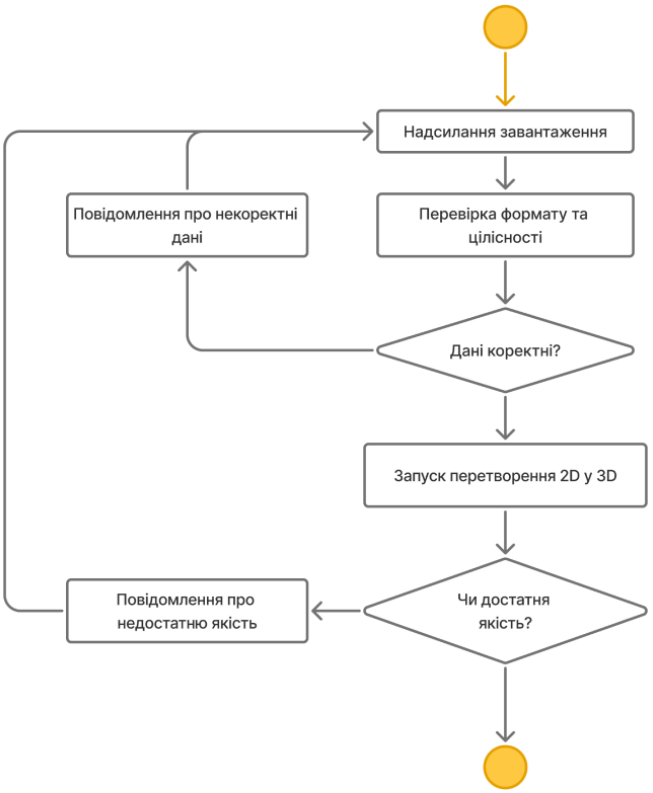

Рисунок 3.4 – Діаграма діяльності

Процес стартує з надсилання користувачем завантаження, після чого система виконує перевірку формату та цілісності вхідних даних. За позитивного результату відбувається підготовка даних. Після завершення ініціюється перетворення 2D у 3D. Далі система виконує оцінку якості проміжних і фінальних результатів. Якщо якість недостатня, здійснюється налаштування параметрів з повторним запуском обробки; якщо достатня, процес завершується відображенням результату та збереженням артефактів для подальшого використання в E-Commerce. Така репрезентація забезпечує прозорість послідовності кроків, унаочнює розподіл відповідальності між компонентами та полегшує верифікацію коректності сценаріїв.

3.5 Діаграма класів

Для розв’язання поставлених задач потрібно описати ключові компоненти системи та методи, які забезпечать виконання основних вимог проєкту. Мета підрозділу полягає в тому, щоб показати, як окремі частини платформи працюють разом і які підходи дають змогу досягти високої продуктивності, стабільності та точності. Такий опис допоможе узгодити очікування, спростить подальше проєктування і дасть чітке бачення шляхів реалізації.

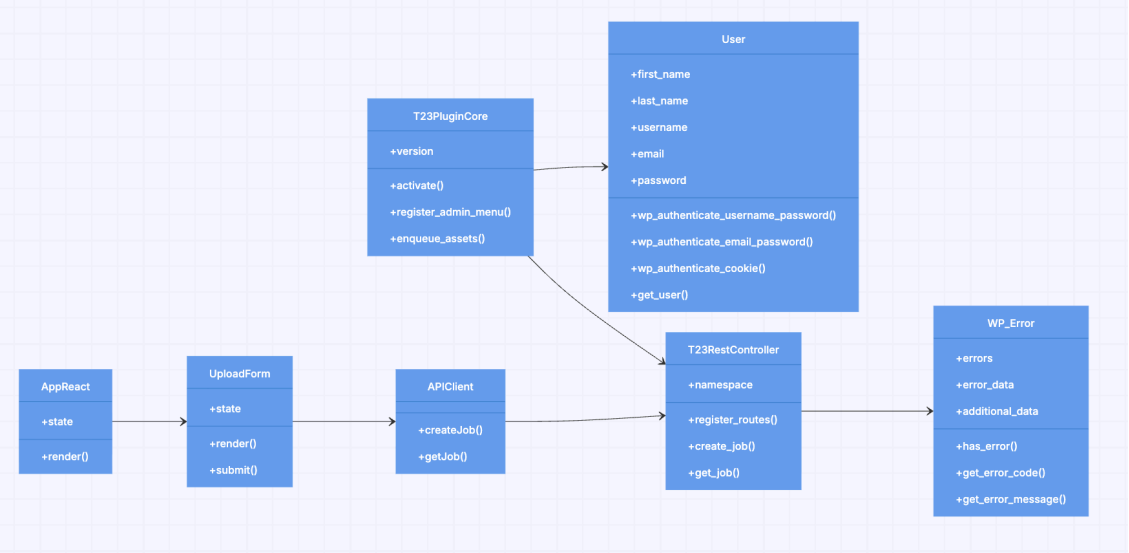

Рисунок 3.5 – Діаграма класів

Діаграма класів є структурною UML моделлю, що дозволяє наочно подати головні класи системи та їхні зв’язки. Вона допомагає упорядкувати дані, поведінку та взаємодії, потрібні для виконання задач системи. Діаграма дає змогу побачити, які об’єкти використовуються, як вони пов’язані і які атрибути та методи містить кожен клас. Це зменшує неоднозначності і підвищує керованість розробки.

До прикладу, у діаграму класів доцільно включити елементи, що відповідають предметній області системи. Атрибути облікового запису: ідентифікатор, ім’я, електронна пошта, пароль, роль користувача. Методи взаємодії: реєстрація, вхід, перегляд даних, формування звіту. Опис

транзакційної події або операції обробки. Клас для з’єднання з зовнішнім API. Клас для формування підсумкового звіту. Такий набір дає мінімально достатню основу для відображення доступів, дій і збереження результатів.

Умовна UML діаграма класів демонструє основні компоненти платформи, подаючи їх атрибути, методи та взаємозв’язки. На ній видно декілька ключових класів, кожен з яких закриває окрему ділянку функціоналу. Це дозволяє розподілити відповідальність, спростити тестування та забезпечити розвиток системи без зайвого дублювання коду.

Клас User подає користувача платформи. Він має атрибути на зразок ID, ім’я, електронна пошта, пароль і роль. Користувач взаємодіє із системою через методи реєстрації та входу, отримує доступ до перегляду даних і може ініціювати створення звітів. Кожен компонент містить власні властивості та методи, що відповідають за конкретні дії. Компонент для перегляду даних періодично оновлює стан на підставі відповідей сервера, тож інформація лишається актуальною без перезавантаження сторінки. Компоненти аналітики показують графіки і таблиці, які змінюються залежно від обраних параметрів, наприклад періоду або типу даних. Такий підхід поєднує швидкість, простоту взаємодії і наочність.

Керування станом на рівні плагіна реалізується серверною логікою на PHP і обміном через WP REST API. Плагін реєструє власний простір імен і маршрути, які приймають і повертають дані у форматі JSON. PHP контролери виконують перевірку прав, валідацію вхідних параметрів, звертаються до бази даних і повертають відповідь для інтерфейсу React. Кешування відповідей, пагінація і фільтри допомагають зменшити навантаження і пришвидшити відображення. Такий підхід зберігає єдине джерело істини на сервері, спрощує аудит дій і підвищує безпеку доступів.

Завдяки узгодженій роботі React, PHP і WP REST API вдається створити систему, яка швидко реагує на дії, тримає дані у контрольованому серверному середовищі і надає однаковий доступ усім компонентам

інтерфейсу. Користувач працює інтуїтивно та без зайвих затримок, а команда розробки отримує передбачувану основу для розвитку функціоналу. Така структура забезпечує високу швидкодію, гнучкість і стабільність всього проєкту, а також полегшує масштабування і супровід у майбутньому.

3.6 Інформаційне забезпечення системи

Інформаційне забезпечення розроблюваної системи ґрунтується на реляційній СУБД MySQL/MariaDB, у межах якої WordPress уже підтримує впорядкований набір базових сутностей для контенту, користувачів, метаданих і конфігурацій. Плагін «2D to 3D» інтегрується в цю модель без порушення сумісності, спираючись на усталені таблиці ядра і доповнюючи їх спеціалізованою структурою для керування життєвим циклом задач перетворення. Вхідні 2D-зображення зберігаються як вкладення у таблиці постів із типом запису attachment, при цьому файлові шляхи та технічні характеристики (розміри, варіанти прев’ю) фіксуються у метаданих поста. Облікові записи користувачів, їхні хешовані паролі та карта можливостей зберігаються у стандартних користувацьких таблицях, а саме керування доступом до адміністративних операцій плагіна здійснюється через існуючу модель ролей WordPress, що унеможливлює дублювання механізмів безпеки. Налаштування, параметри якості генерації та мережеві координати зовнішнього модуля інференсу зберігаються у таблиці опцій, що забезпечує централізований контроль конфігурацій і спрощує перенесення системи між середовищами.

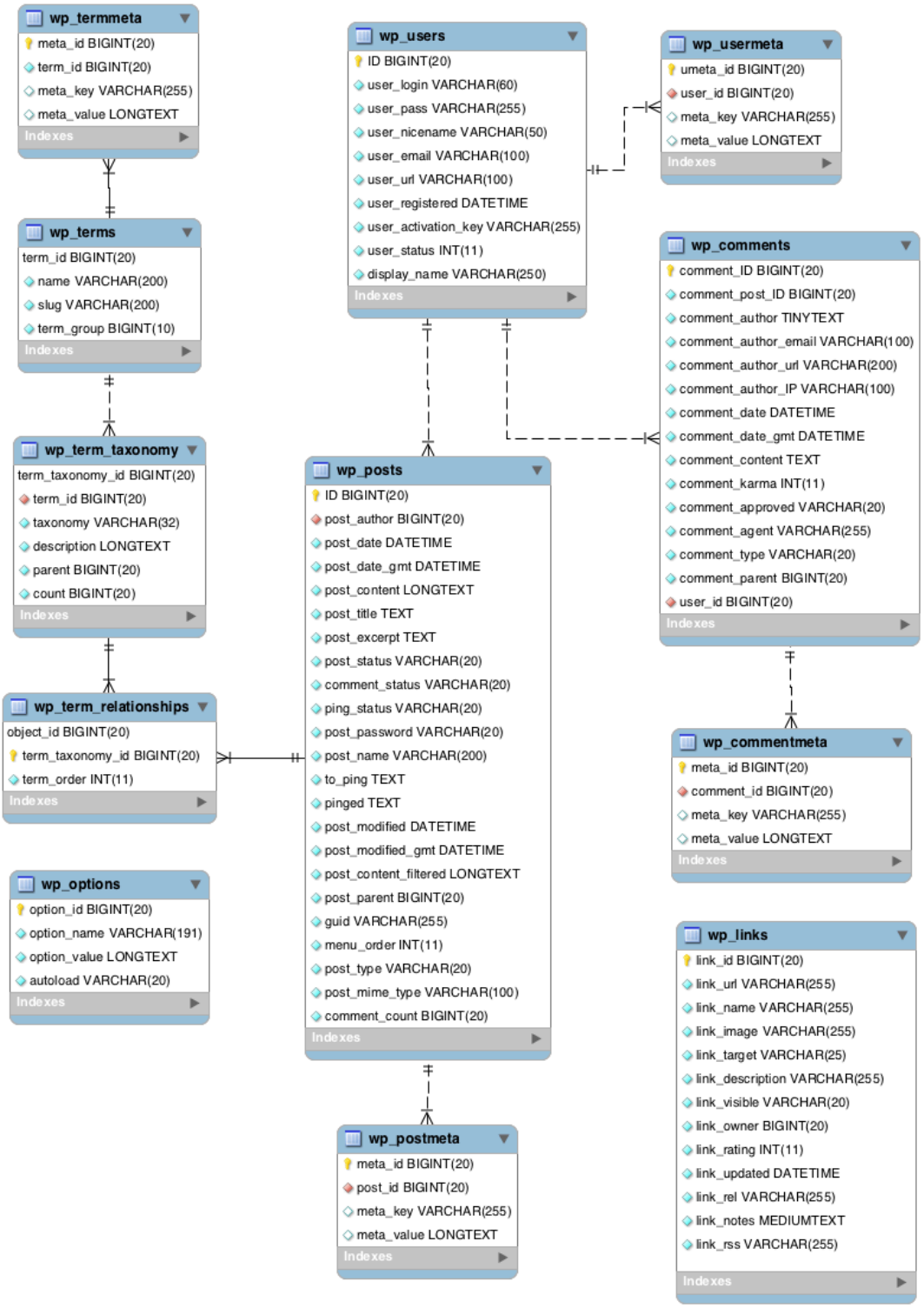

Рисунок 3.6 – Структура бази даних

Для прозорого відстеження конвеєра перетворення 2D у 3D вводиться окрема таблиця, в якій кожному завданню відповідає запис зі сталими ідентифікаційними ознаками, часовими мітками, параметрами перетворення та посиланнями на артефакти. Така таблиця логічно «склеює» вхідне

вкладення з результатами перетворення, не дублюючи можливостей ядра: з одного боку, вона утримує зв’язок із користувачем-ініціатором і джерельним зображенням через відповідні ідентифікатори, з іншого — описує власний стан життєвого циклу, включно з етапами очікування, виконання, успішного завершення чи відмови, та містить діагностичне повідомлення у разі помилки. Зберігання параметрів якості у форматі JSON надає гнучкість еволюції схеми без проведення важких міграцій, а індексація за статусом, датою створення, ідентифікатором користувача та джерельним вкладенням пришвидшує типові вибірки черги для планувальників і фонових воркерів. Водночас посилання на згенеровані GLB-моделі та прев’ю можуть дублюватися окремими записами типу attachment для їхньої появи в медіа-бібліотеці, що полегшує повторне використання контенту в межах CMS.

Дані проходять повний цикл обробки за таким принципом: після завантаження зображення формується запис-вкладення, а у спеціалізованій таблиці створюється нова задача зі станом «у черзі» та серіалізованими параметрами інференсу; після завершення зовнішнім чи локальним модулем обчислень контролер плагіна оновлює стан, фіксує посилання на результати та за потреби реєструє артефакти як медіа. Взаємодія клієнтського інтерфейсу з даними відбувається через WP REST API з чітко визначеним JSON-контрактом, який повертає ідентифікатор завдання, поточний статус і опис артефактів; таким чином зберігається єдине джерело істини на сервері, а React-компоненти працюють із контрольованими, узгодженими представленнями стану. Для зниження навантаження під час полінгу станів використовується об’єктний кеш або механізм транзієнтів із часовим обмеженням дії, що скорочує кількість звернень до бази та стабілізує латентність інтерфейсу.

Питання цілісності та безпеки розв’язуються у кількох шарах. Перед створенням або оновленням записів перевіряються роль та можливості користувача, валідність nonce і коректність вхідного файлу; операції побудовано ідемпотентно, що дозволяє безконфліктно повторювати запити у разі мережевих збоїв. Схема спеціалізованої таблиці може версіонуватись

засобом dbDelta, що гарантуватиме безпечне оновлення плагіна без ручних маніпуляцій із БД. Резервні копії охоплюють як стандартні таблиці WordPress, так і спеціалізовану таблицю завдань, а журналювання ключових подій забезпечує аудит дій користувачів і відтворення причин збоїв. Додаткові індекси впроваджуються за результатами профілювання, що дозволяє еластично масштабувати інформаційний шар відповідно до зростання номенклатури товарів і обсягів конвертацій. Суцільно така організація даних забезпечує узгоджений, безпечний і ефективний обіг інформації між плагіном WordPress, підсистемою інференсу та інтерфейсом React, створюючи передбачувану основу для промислової експлуатації та подальшого розвитку системи.

Висновки до розділу 3

У третьому розділі результати архітектурного опрацювання зафіксовані у взаємодоповнювальних діаграмах, що забезпечують однозначне тлумачення вимог і механіки платформи перетворення 2D у 3D. Діаграма прецедентів структурує взаємодію зацікавлених ролей із системою, окреслюючи кордони продукту, ключові сценарії доступу до функцій і очікувані інваріанти безпеки. Діаграма послідовності формалізує протоколи обміну між інтерфейсом, плагіном WordPress і сервісом обчислень, демонструючи часову координацію викликів під час створення завдання, полінгу статусів і публікації артефактів. Діаграма розгортання фіксує топологію середовищ, канали зв’язку, обмеження мережевої доступності та розподіл відповідальності між вузлами інфраструктури, що прямо впливає на пропускну здатність і надійність. Діаграма діяльності описує потокові стани й гілкування у бізнес-процесі — від завантаження зображення та валідації до постпроцесу моделі та повідомлення користувача — завдяки чому усунуто неоднозначності у правилах переходів і помилкових ситуаціях.

Сукупно ці представлення утворюють повний, взаємно перевірний комплект проєктних артефактів: від «що робити» і «хто взаємодіє» до «як саме викликається» і «де це розгорнено», — що дозволило знизити ризики інтеграції, підвищити передбачуваність продуктивності й гарантувати масштабованість без порушення цілісності платформи.

4 СТВОРЕННЯ ТА ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Розробка та впровадження сервісів для системи

Перший підрозділ зосереджується на побудові архітектурного каркаса та реалізації ключових компонентів: WordPress/PHP-бекенда з REST-маршрутами для приймання 2D-даних і керування конвеєром перетворення; підсистеми зберігання проміжних і фінальних артефактів; React-фронтенда з візуалізаційним шаром на WebGL/Three.js для інтерактивного перегляду результатів у браузері. Така композиція компонентів відповідає обраному технологічному стеку (React, WordPress, PHP, REST API, WebGL/Three.js) і забезпечує узгоджену взаємодію CMS та клієнтської частини. На рівні функціоналу реалізовано повний користувацький цикл: автентифікація, завантаження 2D-об’єктів, автоматизоване перетворення, інтерактивний рендеринг на canvas, експорт 3D-моделі.

```php
<?php

if ( ! defined( constant_name: 'ABSPATH' ) ) exit;

register_activation_hook( file: __FILE__, function() {
    global $wpdb;
    $table = $wpdb->prefix . 'three_jobs';
    $charset_collate = $wpdb->get_charset_collate();
    $sql = "CREATE TABLE IF NOT EXISTS $table (
      id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
      status VARCHAR(32) NOT NULL DEFAULT 'queued',
      src_url TEXT NOT NULL,
      model_url TEXT NULL,
      preview_url TEXT NULL,
      params JSON NULL,
      created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
      updated_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
    ) $charset_collate;";
    require_once ABSPATH . 'wp-admin/includes/upgrade.php';
    dbDelta($sql);
});

add_action('admin_menu', function() {
    add_menu_page(
        page_title: '2D→3D Converter',
        menu_title: '2D→3D Converter',
        capability: 'manage_options',
        menu_slug: 'two-to-three',
        callback: 'two_to_three_admin_app',
        icon_url: 'dashicons-media-code',
        position: 58
    );
});

1 usage  new *
function two_to_three_admin_app() {
    echo '<div id="two-to-three-root"></div>';
}
```

Рисунок 4.1 – Структура та ініціалізація плагіну

У цьому підрозділі наведено проєктні рішення та приклади реалізації сервісів, що забезпечують повний цикл перетворення 2D у 3D у межах екосистеми WordPress: серверна частина на PHP (плагін, REST-маршрути, керування артефактами), а також фронтенд-інтерфейс адміністративного UI плагіну на React, який взаємодіє з REST API та відображає результат у WebGL (Three.js). Така декомпозиція відповідає вимогам E-Commerce-сценаріїв: інтегрованість у CMS, відтворюваність конвеєра та низькі затримки взаємодії.

Поданий фрагмент коду реалізує бутстрап плагіна WordPress, який формує базову серверну інфраструктуру системи. На етапі активації створюється спеціалізована таблиця БД ($wpdb->prefix . 'three_jobs') для менеджменту життєвого циклу задач конвертації: фіксуються стан, джерело вхідних даних, посилання на результати (GLB/прев’ю), параметри оброблення та часові мітки. Ця таблиця є «єдиним джерелом істини» (Single Source of Truth) для сервісів, що взаємодіють із конвеєром перетворення: UI, REST API, фонові процеси та потенційні зовнішні модулі інференсу.

Далі плагін інтегрується у панель адміністратора через пункт меню верхнього рівня і готує контейнер DOM (#two-to-three-root) для вмонтованого React-інтерфейсу. Під час завантаження адміністративної сторінки підключаються зібрані фронтенд-ресурси (JS/CSS), а також здійснюється передавання конфігурації в UI (адреса REST-шлюзу, nonce для авторизованих запитів, шляхи до сховища завантажень). Таке відокремлення відповідальностей — PHP відповідає за безпеку, доступ до БД та життєвий цикл плагіна, тоді як React забезпечує інтерактивність — спрощує еволюцію інтерфейсу без змін у внутрішніх контрактах.

З позиції безпеки і керованості рішення використовує нативні механізми WordPress: dbDelta для міграцій, manage_options/upload_files для контролю доступу, wp_localize_script для безпечного прокидання конфігурації у середовище браузера. Як результат реалізовано достатній каркас плагіна,

який забезпечує узгоджене завантаження UI, підготовку БД та формує базис для подальшої реалізації REST-маршрутів і фонових задач.

```php
<?php
add_action('rest_api_init', function () {
    register_rest_route( route_namespace: 't23/v1', route: '/jobs', [
        'methods'  => 'POST',
        'callback' => 't23_create_job',
        'permission_callback' => function() { return current_user_can( capability: 'upload_files'); }
    ]);

    register_rest_route( route_namespace: 't23/v1', route: '/jobs/(?P<id>\d+)', [
        'methods'  => 'GET',
        'callback' => 't23_get_job',
        'permission_callback' => function() { return current_user_can( capability: 'upload_files'); }
    ]);
});
```

Рисунок 4.2 – Реєстрація REST маршрутів

Фрагмент коду реєструє REST-маршрути простору імен t23/v1 і реалізує два ключові сценарії:

– приймання вхідного 2D-зображення та постановку задачі конвертації (POST /jobs);

– опитування стану задачі та отримання посилань на артефакти (GET /jobs/{id});

На рівні створення задачі здійснюється валідація та безпечне завантаження файлів через wp_handle_upload, після чого у таблиці плагіна реєструється новий запис зі станом queued і серіалізованими параметрами конвертації (тип сітки, розмір текстури, політика UV та склад післяоброблення). Відповідь із кодом 202 інформує UI про прийняття задачі та повертає її ідентифікатор для подальшого моніторингу.

Другий маршрут реалізує читання стану конвеєра від імені авторизованого користувача: повертає поточний статус (queued/run/succeeded/failed) та, за наявності, структуру артефактів (GLB і прев’ю). Така уніфікація представлення результатів спрощує інтеграцію з клієнтською частиною та зовнішніми системами (наприклад, автоматична

публікація у каталозі товарів). При цьому захист доступу покладається на ролеву модель WordPress і перевірку nonce, що забезпечує належний рівень безпеки для адміністративних операцій.

4.2 Інтерфейс користувача

Користувацький інтерфейс розробленої системи виконує роль ключового посередника між бізнес-процесами E-Commerce та конвеєром перетворення 2D у 3D. Основною метою інтерфейсу є забезпечення інтуїтивної та відтворюваної взаємодії з повним життєвим циклом перетворення: від завантаження двовимірних зображень і параметризації якості до моніторингу виконання та інтерактивного перегляду 3D-результату.

```tsx
import "./global.css";

import { Toaster } from "@/components/ui/toaster";
import { createRoot } from "react-dom/client";
import { Toaster as Sonner } from "@/components/ui/sonner";
import { TooltipProvider } from "@/components/ui/tooltip";
import { QueryClient, QueryClientProvider } from "@tanstack/react-query";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Index from "./pages/Index";
import NotFound from "./pages/NotFound";
import { Header } from "@/components/Header";

const queryClient : QueryClient  = new QueryClient();

1+ usages  new *
const App = () => (
  <QueryClientProvider client={queryClient}>
    <TooltipProvider>
      <Toaster />
      <Sonner />
      <BrowserRouter>
        <Header />
        <main>
          <Routes>
            <Route path="/" element={<Index />} />
            <Route path="*" element={<NotFound />} />
          </Routes>
        </main>
      </BrowserRouter>
    </TooltipProvider>
  </QueryClientProvider>
);

const root : HTMLElement  = document.getElementById( elementId: "root");
if (root) {
  createRoot(root).render(<App />);
}
```

Рисунок 4.3 – Структура UI плагіна

Проєктування UI здійснено за принципами task-oriented design: усі сценарії згруповано у логічні секції («Завантаження», «Параметри», «Черга задач», «Перегляд та експорт»), що мінімізує когнітивне навантаження й скорочує час до отримання корисного результату.

Інтерфейс реалізовано як адміністративний UI плагіну WordPress на основі React із використанням WebGL/Three.js для рендерингу GLB/GLTF у браузері. Така інтеграція забезпечує безшовну взаємодію з REST-маршрутами плагіну (приймання файлів, створення/моніторинг задач, отримання артефактів) і водночас дозволяє застосувати сучасні підходи до побудови SPA: динамічне оновлення стану без перезавантаження сторінки, локальні й глобальні стани компонентів, оптимістичні оновлення інтерфейсу. В основу покладено модульність: окремі компоненти відповідають за завантаження даних, панель прогресу/статусів та 3D-переглядач із базовими маніпуляціями (обертання, масштабування, панорамування).

Керування станом організовано через поєднання локального стану React і уніфікованого доступу до даних REST (наприклад, через власні хуки/apiFetch). Такий підхід забезпечує єдине джерело правди для критичних сутностей (ідентифікатор задачі, статуси queued/running/succeeded/failed, посилання на артефакти), знижує кількість дубльованих запитів і спрощує повторне використання логіки у різних частинах інтерфейсу. Для підвищення чутливості UI застосовано періодичний полінг із експоненційним back-off, індикатори прогресу, а також оброблення винятків із пояснювальними повідомленнями й можливістю повторного надсилання запиту. Це особливо важливо за умов непостійного часу виконання та потенційною чергою задач.

Окрему увагу приділено продуктивності й доступності. На фронтенді використано lazy-завантаження важких підсистем (GLTFLoader та пов’язані модулі Three.js) та мемоізацію розрахунково дорогих операцій. У 3D-переглядачі застосовано стратегічні оптимізації: адаптивний розмір полотна, динамічне масштабування пристроями з високою щільністю пікселів, керовану частоту кадрів у фоновому режимі.

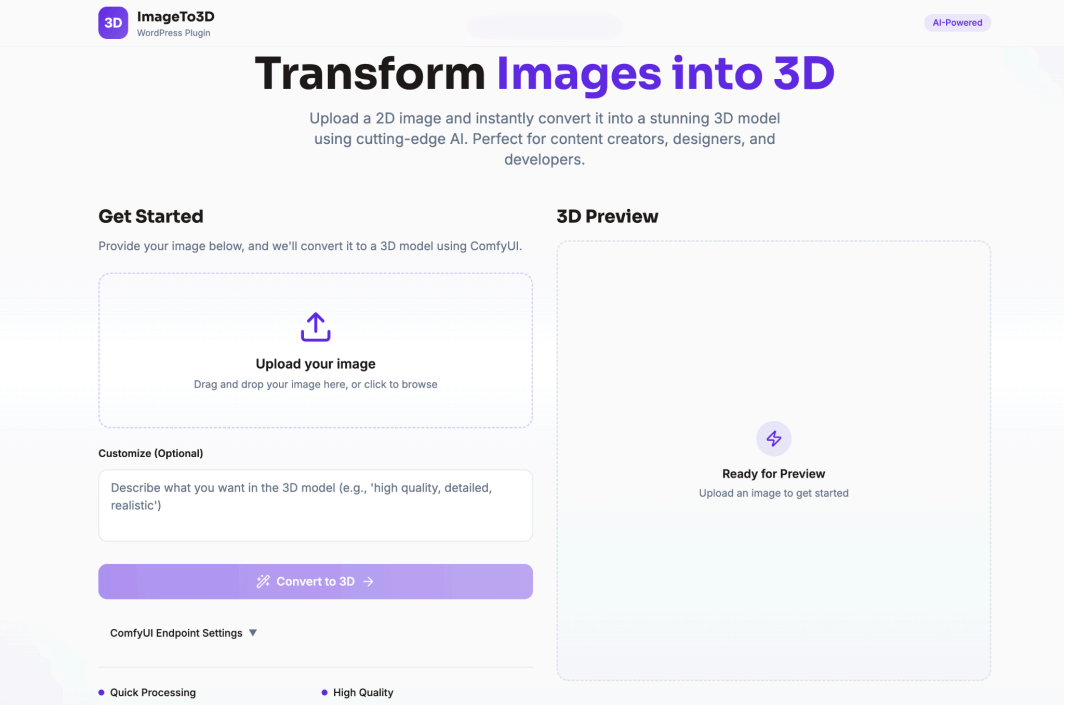

Рисунок 4.4 – Інтерфейс системи перетворення 2D у 3D

Верхній заголовок «Transform Images into 3D» виконує роль семантичного маркера сценарію та одночасно формує інформаційну ієрархію сторінки. Підзаголовок уточнює призначення сервісу — перетворення 2D-зображення на 3D-модель із використанням сучасних методів ШІ — і задає контекст цільової аудиторії (контент-креатори, дизайнери, розробники). Такий підхід знижує когнітивні витрати користувача на первинне орієнтування та корелює з принципом «self-describing UI».

Компонент «Upload your image» є вхідною точкою конвеєра і підтримує вибір файлу через провідник, що узгоджується з вимогами до ергономіки взаємодії. Блок виконує валідацію типів і розміру файла на клієнті, формує об’єкт FormData та ініціює передачу даних до серверного REST-ендпоїнта плагіну; у разі помилки відображаються повідомлення з рекомендаціями (повторне завантаження, зміна формату, зменшення роздільності).

Текстове поле призначене для декларативного налаштування цільових властивостей 3D-моделі (наприклад «high quality, detailed, realistic»). Значення цього поля серіалізується до параметрів задачі і може

інтерпретуватися модулем інференсу як підказка (prompt) для підбору режимів ремешингу, масштабу текстури чи профілю матеріалів. Наявність поля позначена як необов’язкова, що зберігає «короткий шлях» до конверсії та, водночас, не обмежує експертні сценарії.

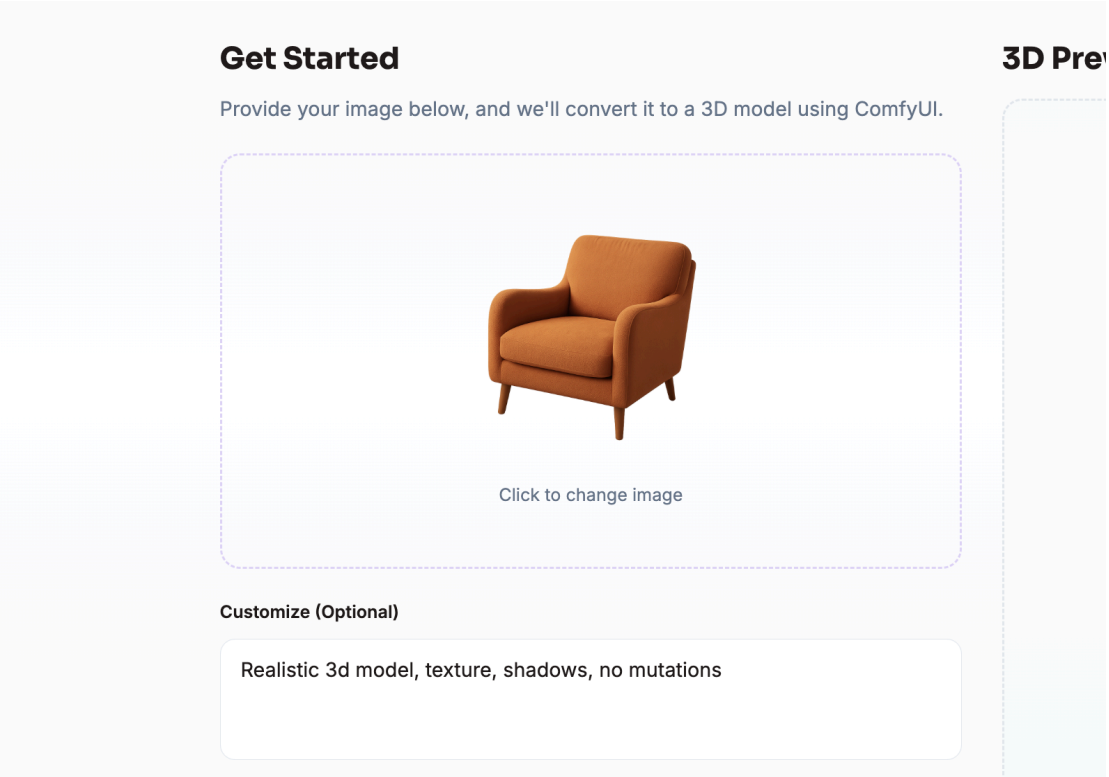

Рисунок 4.5 – Приклад запиту для перетворення 2D у 3D

Центральна кнопка є тригером ініціалізації процесу: після кліку виконується відправлення форми, створення задачі на сервері та перехід інтерфейсу у стан «в обробці». Кнопка має керовану доступність (disabled, поки не вибрано файл), що мінімізує помилки користувача, а також супроводжується індикаторами стану (спінер/текстове повідомлення) для відгуку системи у реальному часі.

ComfyUI Endpoint Settings відповідає за параметризацію з’єднання з бекенд-ендпоїнтом інференсу (URL сервісу). Архітектурно секція відокремлює операційні налаштування від базового сценарію завантаження, що дає змогу адміністратору коригувати середовище виконання без змін у користувацькому потоці. Колапсований стан за замовчуванням зменшує

візуальне навантаження й унеможливлює випадкову модифікацію критичних параметрів.

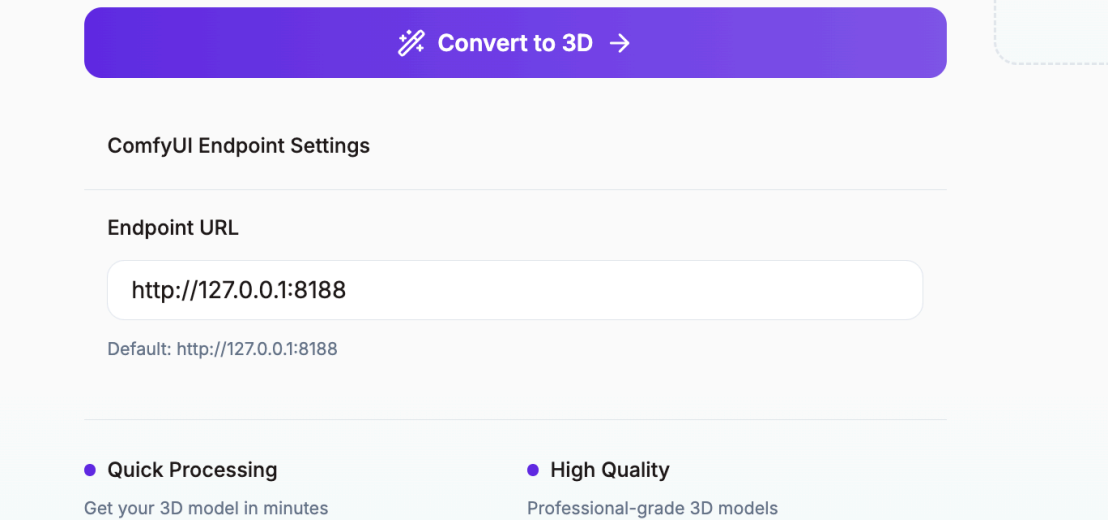

Рисунок 4.6 – Розгорнутий стан секції налаштувань ComfyUI

Інформаційні елементи нижнього рівня артикулюють ключові властивості сервісу: низький час очікування (оперативна генерація 3D) і професійний рівень якості результатів (структура сітки, UV-розгортка, текстури). Їхня функція — формувати очікування щодо SLA та вихідних артефактів, підтримуючи довіру користувача до системи.

Панель «3D Preview» виступає контейнером для WebGL-полотна (canvas), на якому після завершення обробки відтворюється GLB/GLTF-модель. Стан «Ready for Preview» сигналізує про готовність прийняти дані, а після появи моделі ініціюються компоненти сцени (камера, освітлення, маніпулятори) для інтерактивної візуалізації (обертання, масштабування, панорамування). Таким чином, користувач отримує негайний зворотний зв’язок про якість сітки й матеріалів до моменту експорту чи публікації в каталозі.

Ліва колонка відповідає за формування та відправлення «контракту задачі» (вхідний файл + параметри), тоді як права — за візуалізацію результату й контроль станів. Комунікація реалізується через REST-звернення: після натискання «Convert to 3D» створюється задача, UI

запускає полінг статусу, і за подією «succeeded» у правій панелі завантажується та рендериться модель. Таке розмежування обов’язків зберігає модульність, полегшує тестування і масштабування, а також відповідає принципам побудови продуктивних SPA-інтерфейсів для E-Commerce-процесів.

Безпека та цілісність даних забезпечуються вбудованими механізмами WordPress: перевіркою ролей і можливостей користувача для доступу до адміністративних операцій, використанням nonce у всіх інтеракціях з REST, валідацією типів і розміру завантажуваних файлів, а також централізованим відображенням помилок мережі або сервера. З боку UX це супроводжується прозорими станами інтерфейсу («очікування», «в обробці», «успіх», «помилка») та рекомендаціями щодо виправлення (наприклад, зниження роздільної здатності вхідних зображень або зміна параметрів післяобробки).

У підсумку інтерфейс системи є структурованим інструментом управління конвеєром перетворення 2D у 3D, який поєднує інтегрованість у CMS, продуктивність SPA-підходів і спеціалізовану 3D-візуалізацію. Така організація дозволяє контент-командам E-Commerce масштабовано створювати та перевіряти 3D-рендери товарів, скорочуючи операційні витрати й підвищуючи якість представлення асортименту в онлайн-каталогах.

Екосистема WordPress застосовує надійні механізми автентифікації та контролю доступу, що видно вже зі стандартної форми входу: облікові дані перевіряються на сервері, паролі зберігаються у вигляді хешів (із використанням сучасних алгоритмів і «солей»), а автентифіковані сесії підтримуються захищеними cookie. Для протидії підробці запитів (CSRF) у адміністративній частині та REST-інтерфейсі використовуються одноразові токени (nonce), а модель ролей і можливостей (roles & capabilities) забезпечує принцип мінімально необхідних привілеїв для кожного користувача.

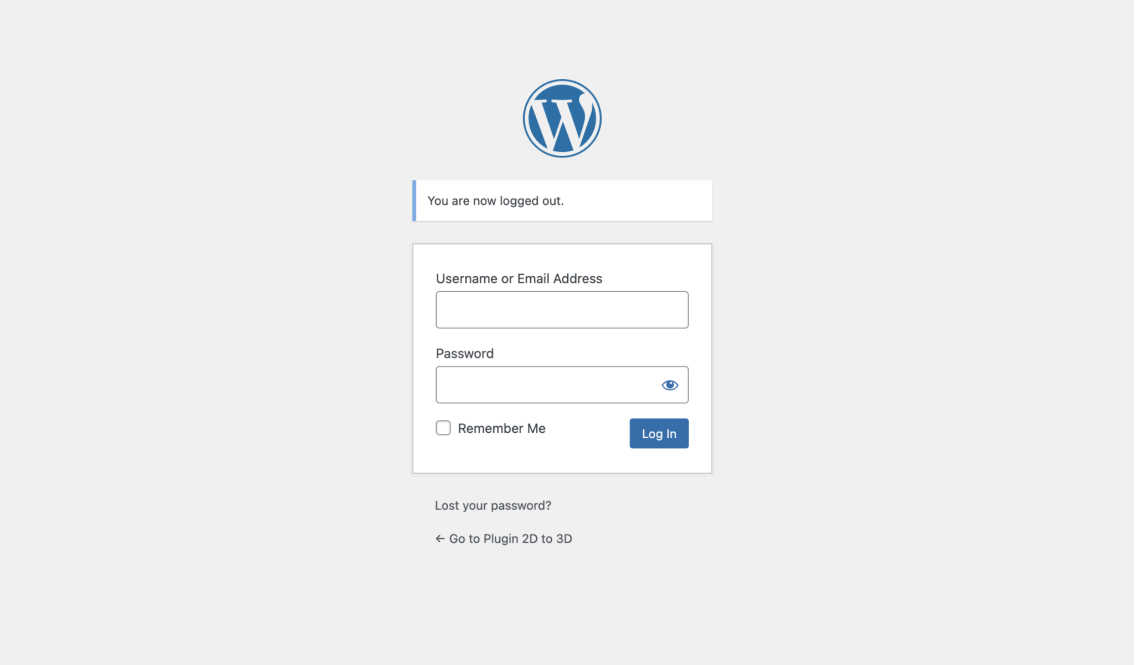

Рисунок 4.7 – Інтерфейс аутентифікації WordPress

Для нашого плагіна «2D to 3D» це означає, що доступ до адміністративного інтерфейсу та REST-маршрутів обмежується авторизованими користувачами з відповідними правами (наприклад, upload_files чи manage_options). Використання nonce у запитах плагіна та вимога HTTPS для панелі керування додатково підвищують стійкість до атак перехоплення трафіку й повторного надсилання запитів. За потреби платформа легко розширюється політиками складності паролів, лімітами на спроби входу та двофакторною автентифікацією через штатні плагіни, зберігаючи сумісність із базовими механізмами безпеки WordPress.

Плагін «2D to 3D» успішно інтегровано в інсталяцію WordPress: в адмін-панелі на лівій бічній панелі з’явився однойменний пункт головного меню з іконкою (Dashicons). Наявність цього елемента свідчить про коректну реєстрацію плагіна в системі, підключення його хуків та готовність до взаємодії з інтерфейсом керування.

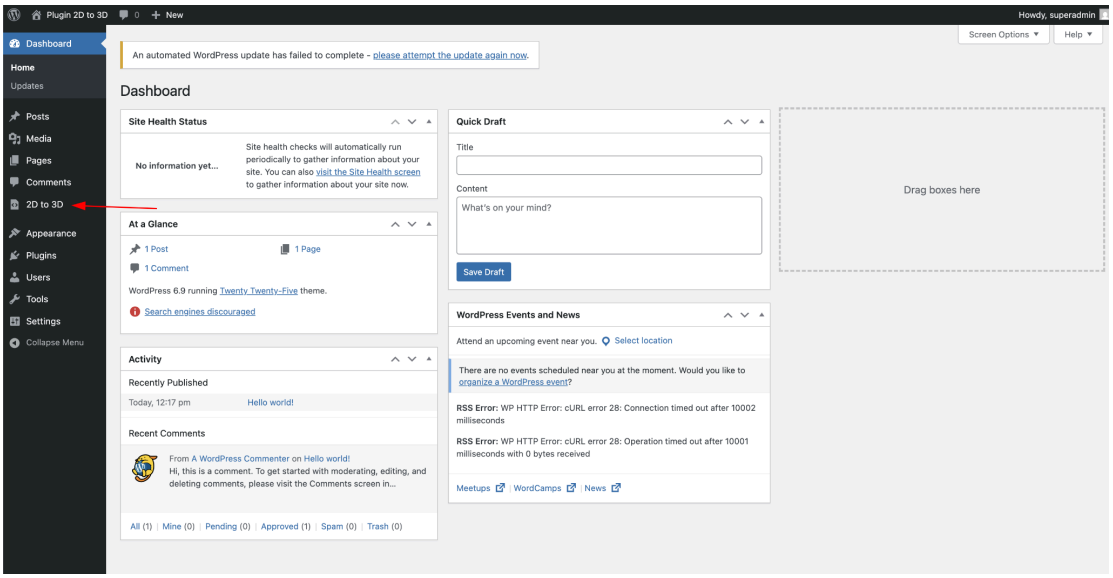

Рисунок 4.8 – Інтерфейс аутентифікації WordPress

Для доступу до функціоналу достатньо обрати пункт «2D to 3D» у боковому меню. Перехід відкриває адміністративну сторінку плагіна з контейнером для подальшого UI (React), де надалі виконуватимуться операції завантаження 2D-зображень, конфігурування параметрів перетворення та перегляд згенерованих 3D-моделей.

4.3 Тестування

Перевірка функціональності створеної системи перетворення двовимірних зображень у тривимірні представлення є завершальним і водночас критичним етапом життєвого циклу програмного забезпечення. У межах цього етапу здійснюється валідація коректності реалізованих сценаріїв, перевіряється узгодженість компонентів і підтверджується спроможність рішення працювати стабільно за реальних умов експлуатації в E-Commerce-середовищі. Під час тестування увага розподілялася між функціональною придатністю серверної частини плагіна WordPress, що реалізує REST-взаємодію та керує чергами конвертації, інтерактивним інтерфейсом на React із доступом до мережевих ресурсів і механізмами автентифікації WordPress, а також візуалізаційним шаром WebGL/Three.js,

який відповідає за відтворення GLB/GLTF-артефактів у браузері. Така цілісна перспектива дозволила охопити як «щасливі» сценарії роботи, так і навмисно спровоковані збої — від відсутності або некоректного формату вхідних зображень до таймаутів інференсу та відмов мережевої інфраструктури.

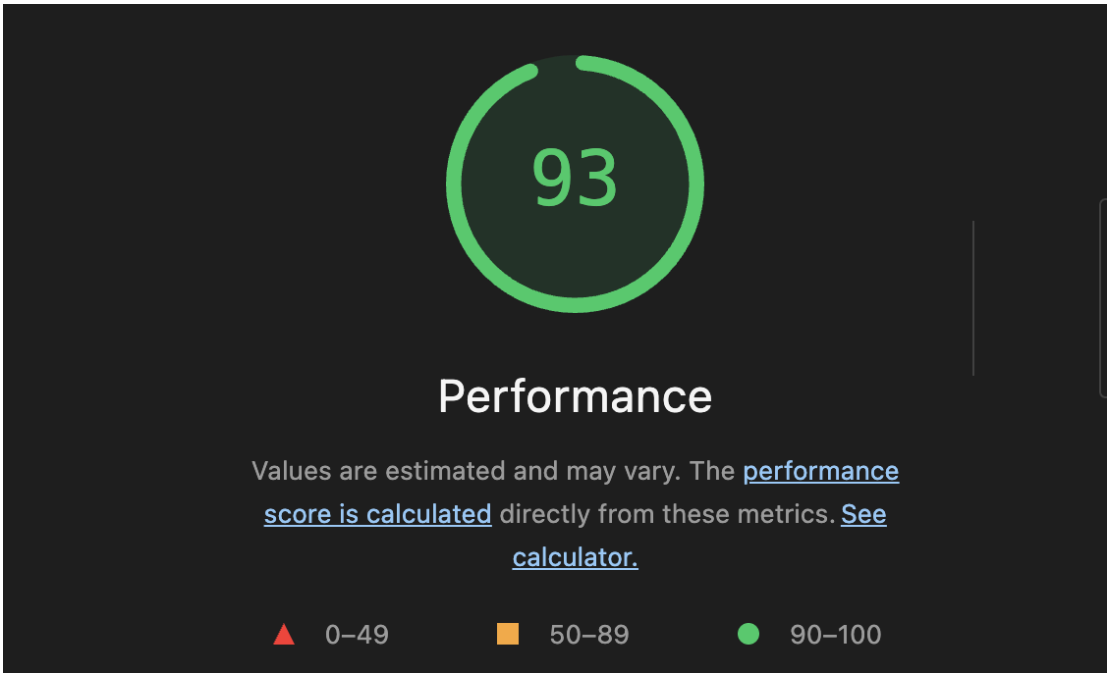

Рисунок 4.9 – Результати тесту LightHouse

Особливе місце у верифікації посідали інструменти аудиту продуктивності, що дають відтворювані числові оцінки та деталізовані рекомендації для подальшої оптимізації. Застосування Lighthouse продемонструвало високий рівень ефективності клієнтської частини: сумарний показник продуктивності становив 93 зі 100, що відповідає «зеленій» зоні й підтверджує якісну організацію ланцюга критичного рендерингу. Зафіксовані час першого відмальовування вмісту на рівні 1,3 секунди та індекс швидкості з таким самим значенням свідчать, що користувачі отримують перший візуальний відгук інтерфейсу без відчутних затримок, а композиція макета залишається стабільною впродовж завантаження завдяки відкладеному підключенню важких модулів на кшталт GLTFLoader.

Навантажувальні властивості оцінювалися окремо засобом Locust на рівні REST-інтерфейсу плагіна. Мета випробувань полягала у перевірці стійкості черги конвертацій та часових характеристик маршруту створення задачі і маршруту читання статусу за умов конкурентних звернень. Імітувалися типові користувацькі дії: завантаження вхідного 2D-файла, реєстрація задачі з параметрами якості, періодичне опитування стану доти, доки система не поверне «succeeded» або «failed». Вимірювалися латентності, частка помилок і відгук при збільшенні кількості одночасних користувачів; приймальними критеріями вважалися відсутність деградації до 5xx-відповідей, стабільність часу створення задачі у межах прийнятного інтервалу та збереження детермінованості статусів. Для відтворюваності нижче наведено компактний скрипт навантажувального профілю.

```python
from locust import HttpUser, task, between
import os, time

SAMPLE_IMAGE = os.getenv("T23_SAMPLE", "sample.png")

class T23User(HttpUser):
    wait_time = between(0.5, 2.0)
    headers = {"X-WP-Nonce": os.getenv("T23_NONCE", "")}
    base = os.getenv("T23_BASE", "/wp-json/t23/v1")

    @task
    def convert_and_poll(self):
        with open(SAMPLE_IMAGE, "rb") as f:
            files = {"image": (os.path.basename(SAMPLE_IMAGE), f, "image/png")}
            data  = {"meshType": "tri", "textureSize": "1024"}
            with self.client.post(f"{self.base}/jobs", files=files, data=data,
                                  headers=self.headers, name="POST /jobs",
                                  catch_response=True) as resp:
                if resp.status_code not in (200, 202):
                    resp.failure(f"Unexpected status {resp.status_code}")
                    return
                job_id = resp.json().get("jobId")

        deadline = time.time() + 60
        while time.time() < deadline:
            r = self.client.get(f"{self.base}/jobs/{job_id}", headers=self.headers, name="GET /jobs/{id}")
            if r.status_code != 200:
                break
            status = r.json().get("status")
            if status in ("succeeded", "failed"):
                break
            time.sleep(1.0)
```

Рисунок 4.10 – Конфігурація тесту Locust

Чеклісти оформлено у вигляді текстових переліків, щоби чітко фіксувати готовність кожного шару рішення. Для користувацького інтерфейсу

перевіркою охоплено видимість і доступність пункту «2D to 3D» у бічному меню, коректне завантаження сторінки плагіна, наявність контейнера для React-компонента, стабільність макета при зміні розміру екрана, локалізацію повідомлень та поведінку кнопки «Convert to 3D», яка має бути неактивною до вибору файла і переходити у стан завантаження після відправлення форми. Для REST-інтерфейсу зафіксовано вимоги до належних кодів відповіді при створенні задачі та читанні стану, перевірено коректну обробку порожнього запиту, некоректного MIME-типу і надмірного розміру файла, а також підтверджено наявність X-WP-Nonce та перевірку повноважень користувача. Для безпеки встановлено обов’язковість HTTPS у продакшн-середовищі, контроль прав доступу на рівні manage_options або upload_files, фільтрацію імен файлів і видалення тимчасових ресурсів; для 3D-візуалізації перевірено ініціалізацію сцени лише після появи дійсного GLB, коректне обертання і масштабування, відсутність витоків пам’яті при повторних завантаженнях, а також відпрацювання втрати контексту WebGL із подальшим відновленням.

Розділ	Перевірка
3.1 Функціональність	Після активації плагіна в адмін-меню з’являється «2D to 3D», сторінка відкривається без помил
3.1 Функціональність	На сторінці є контейнер UI (root) для React; підключення JS/CSS без помилок консолі.
3.1 Функціональність	Кнопка «Convert to 3D» неактивна до вибору файла; після вибору стає активною.
3.1 Функціональність	Завантаження коректних PNG/JPEG завершується 202; інтерфейс показує jobId і стани queued/r
3.1 Функціональність	Після завершення обробки статус «succeeded», є посилання на GLB/прев’ю; модель рендериться
3.1 Функціональність	Невалідні файли (тип/розмір) відхиляються; локалізоване повідомлення з поясненням.
3.2 REST і безпека	POST /wp-json/t23/v1/jobs доступний лише користувачам з правами upload_files; перевіряється
3.2 REST і безпека	GET /wp-json/t23/v1/jobs/{id} повертає статус і артефакти; для неіснуючого id — 404.
3.2 REST і безпека	Перевірка MIME-типу і граничного розміру; нормалізація імен файлів.
3.2 REST і безпека	Коректні коди статусів (2xx/4xx/5xx) і структуровані помилки.
3.2 REST і безпека	У продакшні використовується HTTPS; cookies мають secure; вимкнено індексацію системних д
3.3 Інтерфейс та A11y	Семантична розмітка елементів; aria-label/aria-describedby для полів і кнопок.
3.3 Інтерфейс та A11y	Повна навігація клавіатурою; видимі фокусні стани; логічний порядок фокусу.
3.3 Інтерфейс та A11y	Повідомлення про успіх/помилки в live-region; локалізація відповідає мові інтерфейсу.
3.3 Інтерфейс та A11y	Адаптивність для основних брейкпоінтів; критичні елементи не перекриваються.
3.4 WebGL переглядач	Ініціалізація рендера лише після дійсного URL GLB; повторні завантаження без витоків пам’яті.
3.4 WebGL переглядач	Підтримка обертання/масштабування/панорамування; коректний ресайз і aspect ratio.
3.4 WebGL переглядач	Відпрацювання втрати контексту WebGL і відновлення без перезавантаження сторінки.
3.4 WebGL переглядач	Обмеження розміру текстур на мобільних; стабільний FPS.
3.5 Lighthouse	Mobile і Desktop профілі без критичних зауважень; інтегральний бал ≥ 90/100.
3.5 Lighthouse	FCP і LCP у «зеленій» зоні; CLS мінімальний завдяки стабільному макету.
3.5 Lighthouse	Відсутні блокувальні ресурси; Three.js завантажується ліниво.
3.6 Locust	За 20–50 одночасних користувачів відсоток 5xx = 0%; медіана латентності стабільна.
3.6 Locust	POST /jobs не є «гарячою точкою»; пропускна здатність зростає лінійно до цільового навантаж
3.6 Locust	Полінг GET /jobs/{id} з back-off не перевантажує API; частота запитів помірна.

Рисунок 4.11 – Оформлення чеклистів

Тест-кейси подано у формі розгорнутих сценаріїв. Базовий позитивний сценарій описує завантаження припустимого PNG або JPEG, подання форми з типовими параметрами, отримання коду 202 із ідентифікатором задачі, перехід інтерфейсу до спостереження за прогресом, зміну статусу на «succeeded» і поява прев’ю разом зі списком артефактів, після чого виконується візуальний контроль у переглядачі та перевірка метаданих моделі. Негативний сценарій моделює помилку валідації під час спроби завантажити файл недопустимого типу; очікуваною поведінкою є повернення коду 400 на REST-рівні і відображення локалізованого повідомлення в UI з підказкою щодо підтримуваних форматів. Окремий інтеграційний сценарій перевіряє поведінку системи при затримці інференсу: після ініціації завдання полінг виконується із збільшенням інтервалу між запитами, інтерфейс демонструє проміжні повідомлення без дублювання запитів і продовжує роботу до появи фінального статусу або спливу нормативного тайм-ауту. Ще один приклад описує відмову мережі під час полінгу: інтерфейс має повідомити про тимчасову недоступність, зберегти контекст задачі і запропонувати повтор після відновлення з’єднання.

A	B	C	D
Назва	**Передумови**	**Кроки**	**Очікуваний результат**
Успішна конвертація коректного зображення	Активований плагін; користувач із правами upload_files; доступ до адмін-панелі.	1) Відкрити розділ «2D to 3D». 2) Обрати валідний PNG/JPEG (до 5 МБ). 3) Натиснути «Convert to 3D». 4) Дочекатися завершення обробки.	Сервер повертає 202 та jobId; статус «succeeded»; посилання на GLB/прев’ю; модель рендериться.
Відмова при завантаженні невалідного типу файла	Активований плагін; користувач авторизований.	1) Обрати файл недопустимого типу (наприклад, .exe або .txt). 2) Натиснути «Convert to 3D».	REST повертає 400; інтерфейс показує локалізоване повідомлення з переліком підтримуваних форматів; задача не створена.
Відпрацювання таймауту інференсу	Доступний сервер із штучною затримкою або симуляцією таймауту.	1) Ініціювати конвертацію валідного зображення. 2) Очікувати перевищення ліміту часу. 3) Спостерігати поведінку інтерфейсу.	Полінг сповільнюється за back-off; після таймауту інформативне повідомлення, можливість повтору; система не зависає.
Захищеність маршрутів REST нонсом та правами	Користувачі Administrator і Subscriber; валідний/невалідний nonce.	1) POST /jobs із валідним nonce від імені адміністратора. 2) Повторити без nonce. 3) Повторити від імені Subscriber.	Перший запит успішний (202); другий і третій завершуються помилками авторизації/дозволів.
Клавіатурна доступність інтерфейсу	Відкритий екран плагіна у сучасному браузері.	1) Навігувати між елементами клавішею Tab. 2) Активувати кнопки Enter/Space. 3) Перевірити видимість фокусу.	Усі елементи досяжні з клавіатури; логічний порядок фокусу; дії виконуються без миші; фокусні індикатори видимі.
Стабільність рендерингу та очищення ресурсів	GLB модель доступна за валідним URL.	1) Завантажити модель у переглядач. 2) Перезавантажити модель тричі. 3) Виміряти споживання пам’яті вкладки.	Витоків пам’яті немає; FPS стабільний; відсутні помилки WebGL у консолі.
Аудит продуктивності Lighthouse (Desktop/Mobile)	Chrome DevTools; режим інкогніто; холодний кеш.	1) Запустити Lighthouse для сторінки плагіна. 2) Проаналізувати Performance, Accessibility, Best Practices, SEO.	Бал Performance ≥ 90; FCP/LCP у «зеленій» зоні; немає критичних блокувальних ресурсів; рекомендації зафіксовані.
Навантажувальний профіль Locust: створення задач і полінг	Розгорнутий Locust; доступ до REST; наявні nonce і тестове зображення.	1) Запустити Locust із 20–50 одночасними користувачами. 2) Зібрати метрики медіани/95-го перцентиля латентності та % помилок.	Помилки 5xx = 0; латентність стабільна; API масштабується без деградації.

Рисунок 4.12 – Оформлення тесткейсів

Методики тестування обрано так, щоби поєднати глибину охоплення із практичною відтворюваністю. Функціональне тестування з позицій «чорної скриньки» застосовано для перевірки поведінки інтерфейсу та REST-контрактів без занурення у внутрішню реалізацію, що дозволило

фокусуватися на сприйнятті системи кінцевим користувачем. Інтеграційний підхід використано для підтвердження коректності зв’язків між WordPress-плагіном, чергою конвертацій і візуалізатором, включно з часовою координацією подій і відпрацюванням помилок. Регресійна складова забезпечила сталість ключових характеристик після внесення змін до фронтенд-збірки та серверних обробників. Накладання цих методик із кількісним Lighthouse-аудитом і навантажувальним профілем Locust створило всебічну картину готовності рішення: система демонструє швидку появу вмісту, стабільну взаємодію під час обчислювально затратних операцій, чітку діагностику помилок і відсутність деградації під робочим навантаженням, що є необхідним для її експлуатації у виробничих процесах електронної комерції.

Сукупність наведених спостережень і числових індикаторів дозволяє зробити виважений висновок про готовність системи до промислового використання. Вона коректно виконує повний цикл перетворення 2D у 3D у межах WordPress, забезпечує швидку віддачу інтерфейсу та стабільну інтерактивну візуалізацію, а також демонструє стійкість до типових відмов і мережевих обмежень. Практичний ефект для E-Commerce полягає у скороченні часу публікації високоякісних 3D-рендерів товарів і зменшенні операційних витрат завдяки автоматизованому та відтворюваному конвеєру, який пройшов повноцінну перевірку якості за сучасними методиками.

Висновки до розділу 4

У четвертому розділі було послідовно реалізовано та апробовано веборієнтовану систему перетворення двовимірних зображень у тривимірні представлення для E-Commerce, побудовану як плагін WordPress із адміністративним інтерфейсом на React та візуалізатором на базі WebGL/Three.js.

Перевірка якості системи була сфокусована на відтворюваних метриках продуктивності та реалістичних сценаріях експлуатації. Аудит Lighthouse засвідчив відповідність інтерфейсу вимогам ефективного критичного шляху

рендерингу, стабільність макета та відсутність блокувальних ресурсів, що безпосередньо відображається на швидкому візуальному відгуку та зручності користування.

Перевірка якості системи була сфокусована на відтворюваних метриках продуктивності та реалістичних сценаріях експлуатації. Аудит Lighthouse засвідчив відповідність інтерфейсу вимогам ефективного критичного шляху рендерингу, стабільність макета та відсутність блокувальних ресурсів, що безпосередньо відображається на швидкому візуальному відгуку та зручності користування. Система коректно масштабується без деградації функціональності та інтерфейсу.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено комплексну систему перетворення 2D-об’єктів у 3D-моделі. Протягом роботи було детально опрацьовано всі аспекти створення вебсистеми, починаючи від теоретичних основ і аналізу суміжних рішень до архітектури програмного забезпечення, розробки компонентів плагіна WordPress, тестування й оптимізації конвеєра перетворення.

На початковому етапі було виконано аналіз існуючих методів і платформ для 3D-візуалізації та генерації, включаючи засоби управління 3D-активами і конфігураторами. Це дозволило визначити основні вимоги до системи та сформулювати цілі проєкту. Було розглянуто технології, необхідні для створення ефективного інтерфейсу та надійної роботи системи в умовах зростаючого навантаження. Особливу увагу приділено питанням безпеки даних, швидкості відгуку інтерфейсу плагіна і коректного відображення 3D-результатів у режимі, наближеному до реального часу.

Розробка системи охоплювала процеси архітектурного проєктування та моделювання основних компонентів. Використання React у складі плагіна WordPress і WP REST API дозволило створити сучасний, зручний і динамічний інтерфейс, що забезпечує просту взаємодію користувачів із системою. Розроблена система включає розділи для завантаження зображень, запуску перетворення за допомогою моделі Hunyuan3D, перегляду та порівняння результатів, а також управління історією запусків і метаданими. Використання структурованих REST маршрутів забезпечило оптимізацію обміну даними та прискорило завантаження панелей керування.

Проведене тестування системи показало її стабільність та високу продуктивність. За допомогою ComfyUI було відпрацьовано графи обробки і виявлено потенційні вузькі місця конвеєра, а засоби профілювання інтерфейсу допомогли покращити швидкодію і узгодженість реакції клієнтської частини. Це дало змогу досягти показників високої продуктивності, низької затримки при видачі результатів і стабільності інтерфейсу. Окрім цього, забезпечено зручний доступ до системи з різних

пристроїв, включаючи мобільні, що робить рішення універсальним і зручним у використанні.

Завдяки проведеному тестуванню було досягнуто оптимального рівня продуктивності, що дозволяє системі впевнено працювати під значними навантаженнями та забезпечувати стабільний користувацький досвід навіть у пікові моменти. Було також передбачено масштабування інфраструктури на базі екосистеми RunPod для майбутнього розширення функціоналу або збільшення кількості одночасних завдань, що робить систему гнучкою і готовою до розвитку.

Таким чином, розроблена система перетворення 2D-об’єктів у 3D-моделі досягла своєї основної мети. Вона надає користувачам зручний інструмент для запуску генерації, перевірки якості та подальшої публікації результатів, забезпечуючи точність, надійність і швидкість. Виконаний проєкт демонструє застосування сучасних вебтехнологій у поєднанні з належними практиками безпеки, продуктивності та доступності, створюючи якісний продукт, що відповідає високим вимогам користувачів. Реалізовані функції дозволяють командам ефективно організувати конвеєр 2D до 3D, приймати зважені рішення щодо публікації і подальшої обробки, що робить систему важливим інструментом для розробників, редакторів контенту та інженерів 3D.

Було також проведено дослідження робочих сценаріїв і показників системи, які дозволили визначити найефективніші підходи до підготовки даних, конфігурування Hunyuan3D і організації зберігання результатів. Застосування інтеграції з зовнішніми сервісами через WP REST API забезпечило актуальність і точність обміну даними, що важливо для користувачів і адміністраторів під час контролю якості та прийняття рішень.

Робота є прикладом комплексного підходу до веброзробки та створення інформаційних систем, що відповідають сучасним вимогам до керування мультимедійним контентом і 3D-візуалізацією.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Algharabat R. S. Effects of Visual Control and Graphical Characteristics of 3D Product Presentations on Perceived Trust in Electronic Shopping.

2. Analysis of big data in computer graphics / O. N. Romanyuk et al. Optoelectronic Information-Power Technologies. 2024. Vol. 47.

3. Artificial Intelligence Based Generation and Analysis of 3D Models : patent US10621779B1 United States : 15 / 990,212. Published on 14.04.2020. 46 p.

4. Enhanced Three Dimensional Visualization Using Artificial Intelligence : patent US20230102949A1 United States : 17/487,985. Published on 30.03.2023. 15 p.

5. Interference-aware geometric modeling / D. Harmon et al. ACM Transactions on Graphics. 2011. Vol. 30, no. 6. P. 1–10 DOI: 10.1145/2070781.2024171.

6. Lehkyi M., Zhuravchak L. Virtual Online Garment Fitting Using Augmented Reality. Інформаційні системи та мережі. 2024. Vol. 15. P. 184–199.

7. Lin Y. The Review of Modeling and Rendering 3D Environment: A Survey. Highlights in Science, Engineering and Technology. 2024.

8. Machine-Learning for 3D Object Detection : patent US20220189070A1 United States : 17/553,403. Published on 16.01.2022. 28 p.

9. Personalized Product Assortment with Real-time 3D Perception and Bayesian Payoff Estimation / P. Jenkins et al. KDD '24: The 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Barcelona Spain.

10. Product Design, Configuration and Decision System Using Machine Learning : patent US20220138383A1 United States. No. 17 / 574,160 ; published on 05.05.2022, Bulletin no. 28.

11. Systems and Methods for Visualizing Machine Intelligence : patent US20240078163A1 United States : 18/506,415. Applied on 10.11.2023 ; published on 07.03.2024. 48 p.

12. Transformer guided geometry model for flow-based unsupervised visual odometry / X. Li et al. Neural Computing and Applications. 2021. DOI: 10.1007/s00521-020-05545-8.

13. Архітектурна 3D візуалізація: що це, ким створюється і де використовується. cgischool. URL: https://cgischool.ua/arkhitekturna-3d-vizualizatciya-vyznachennya/ (дата звернення: 30.04.2025).

14. Богом’я В., Гудзь А. Штучний інтелект: сучасний стан і перспективи застосування. Сучасні інформаційні технології у сфері безпеки та оборони.

15. Островка Д. В. Інформаційна технологія синтезу тривимірного зображення користувача для мобільних систем доповненої реальності : Дисертація. Львів, 2023. 131 с.

16. Приймич Я. Б., Войцеховська О. В. Візуалізації 3D моделей зі збереженням деталізації карт нормалей. Вінницький національний технічний університет. 2023.

17. Проскурніна Н. Штучний інтелект у маркетинговій діяльності. Зовнішня торгівля: економіка, фінанси, право. Серія. Економічні науки. 2020. № 4 (111). С. 129–140.

18. Що Таке 3D-Рендеринг? 6 Основних Принципів 3D-візуалізації. Гудвіл Дизайн сервіс. URL: https://gudvil.com.ua/ua/blog/scho-take-3d-rendering-6-grundlegende-prinzipien/ (дата звернення: 30.04.2025).

19. Як навчитися 3D візуалізації швидко: 5 лайфхаків. CGI School. URL: https://cgischool.ua/yak-navchytysya-3d-vizualizaciyi-shvydko/ (дата звернення: 30.04.2025).

20. Ясінська Т., Федотова Н., Ващенко С. Веб-ресурс візуалізації 3D-моделей. Наука та виробництво. 2020. № 23. С. 266–273.

21. Головко О. В., Швед А. В., Програмне забезпечення 3D-візуалізації двовимірних об’єктів. Інтелектуальні інформаційні системи 2025.

ДОДАТОК

Лістинг коду системи

```
import { useState, useEffect } from "react";
import { ConversionForm } from "@/components/ConversionForm";
import { Canvas3D } from "@/components/Canvas3D";
import { comfyUIService } from "@/services/comfyui";
import { AlertCircle, CheckCircle2, Zap } from "lucide-react";

export default function Index() {
  const [currentStep, setCurrentStep] = useState<
    "upload" | "processing" | "success" | "error"
  >("upload");
  const [modelUrl, setModelUrl] = useState<string | null>(null);
  const [errorMessage, setErrorMessage] = useState<string | null>(null);
  const [comfyEndpoint, setComfyEndpoint] = useState(
    "http://127.0.0.1:8188"
  );

  const handleConversion = async (
    imageFile: File,
    _imagePreview: string,
    prompt: string
  ) => {
    setCurrentStep("processing");
    setErrorMessage(null);

    const reader = new FileReader();
    reader.onload = async (event) => {
      if (!event.target?.result) return;

      const imageBase64 = event.target.result as string;

      const result = await comfyUIService.convert({
        imageUrl: imageBase64,
        prompt,
      });

      if (result.status === "success" && result.modelUrl) {
        setModelUrl(result.modelUrl);
        setCurrentStep("success");
      } else {
        setErrorMessage(
          result.message ||
            "Failed to convert image. Please check your ComfyUI endpoint."
        );
        setCurrentStep("error");
```

```
    }
   };
   reader.readAsDataURL(imageFile);
  };

  const resetForm = () => {
   setCurrentStep("upload");
   setModelUrl(null);
   setErrorMessage(null);
  };

  useEffect(() => {
   if (comfyEndpoint) {
    comfyUIService.setEndpoint(comfyEndpoint);
   }
  }, [comfyEndpoint]);

  return (
   <div className="min-h-screen bg-gradient-to-b from-slate-50 via-background to-slate-50">
    <div className="relative">
     <div className="absolute inset-0 overflow-hidden">
      <div className="absolute -top-40 -right-40 w-80 h-80 bg-primary/5 rounded-full
blur-3xl" />
      <div className="absolute -bottom-40 -left-40 w-80 h-80 bg-primary/5 rounded-full
blur-3xl" />
     </div>

     <div className="relative container max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-12">
      <div className="text-center space-y-4 mb-12">
       <div className="inline-block">
        <div className="flex items-center gap-2 px-4 py-2 rounded-full bg-primary/10 border
border-primary/20">
         <Zap className="w-4 h-4 text-primary" />
         <span className="text-sm font-medium text-primary">
          AI-Powered 3D Generation
         </span>
        </div>
       </div>

       <h1 className="text-4xl sm:text-5xl md:text-6xl font-bold text-foreground
leading-tight">
        Transform <span className="text-primary">Images into 3D</span>
       </h1>

       <p className="text-lg sm:text-xl text-muted-foreground max-w-2xl mx-auto
leading-relaxed">
        Upload a 2D image and instantly convert it into a stunning 3D
        model using cutting-edge AI. Perfect for content creators,
        designers, and developers.
       </p>
```

```
        </div>

        <div className="grid lg:grid-cols-2 gap-8 items-start">
          <div className="space-y-6">
            {currentStep === "upload" && (
              <div className="space-y-6">
                <div className="space-y-3">
                  <h2 className="text-2xl font-bold text-foreground">
                    Get Started
                  </h2>
                  <p className="text-muted-foreground">
                    Provide your image below, and we'll convert it to a
                    3D model using ComfyUI.
                  </p>
                </div>
                <ConversionForm onSubmit={handleConversion} />

                <details className="group">
                  <summary className="cursor-pointer flex items-center gap-2 px-4 py-3 rounded-lg
hover:bg-secondary/30 transition-colors">
                    <span className="text-sm font-medium text-foreground">
                      ComfyUI Endpoint Settings
                    </span>
                    <span className="text-muted-foreground text-xs group-open:hidden">
                      ▼
                    </span>
                  </summary>
                  <div className="px-4 py-4 space-y-3 mt-2 border-t border-border">
                    <label className="block text-sm font-medium text-foreground">
                      Endpoint URL
                    </label>
                    <input
                      type="text"
                      value={comfyEndpoint}
                      onChange={(e) => setComfyEndpoint(e.target.value)}
                      placeholder="http://127.0.0.1:8188"
                      className="w-full px-4 py-2 rounded-lg border border-border bg-background
text-foreground placeholder:text-muted-foreground focus:outline-none focus:ring-2
focus:ring-primary/50"
                    />
                    <p className="text-xs text-muted-foreground">
                      Default: http://127.0.0.1:8188
                    </p>
                  </div>
                </details>
              </div>
            )}

            {currentStep === "processing" && (
```

```
          <div className="space-y-6 p-8 rounded-2xl bg-primary/5 border
border-primary/20">
            <div className="flex justify-center">
              <div className="relative w-16 h-16">
                <div className="absolute inset-0 rounded-full border-4 border-primary/20" />
                <div className="absolute inset-0 rounded-full border-4 border-transparent
border-t-primary animate-spin" />
              </div>
            </div>
            <div className="space-y-2 text-center">
              <h3 className="text-lg font-semibold text-foreground">
                Converting Your Image
              </h3>
              <p className="text-muted-foreground">
                Please wait while we generate your 3D model... This may
                take a minute or two.
              </p>
            </div>
          </div>
        )}

        {currentStep === "success" && (
          <div className="space-y-6">
            <div className="p-6 rounded-2xl bg-green-50 border border-green-200 space-y-3">
              <div className="flex items-start gap-3">
                <CheckCircle2 className="w-6 h-6 text-green-600 mt-0.5 flex-shrink-0" />
                <div>
                  <h3 className="font-semibold text-green-900">
                    Conversion Complete!
                  </h3>
                  <p className="text-sm text-green-800 mt-1">
                    Your 3D model has been generated successfully. You
                    can view it in the canvas on the right.
                  </p>
                </div>
              </div>
            </div>

            <button
              onClick={resetForm}
              className="w-full px-6 py-3 rounded-xl font-semibold bg-primary
text-primary-foreground hover:shadow-lg hover:shadow-primary/30 transition-all duration-200"
            >
              Convert Another Image
            </button>
          </div>
        )}

        {currentStep === "error" && (
          <div className="space-y-6">
```

```
            <div className="p-6 rounded-2xl bg-red-50 border border-red-200 space-y-3">
              <div className="flex items-start gap-3">
                <AlertCircle className="w-6 h-6 text-red-600 mt-0.5 flex-shrink-0" />
                <div>
                  <h3 className="font-semibold text-red-900">
                    Conversion Failed
                  </h3>
                  <p className="text-sm text-red-800 mt-1">
                    {errorMessage}
                  </p>
                </div>
              </div>
            </div>

            <button
              onClick={resetForm}
              className="w-full px-6 py-3 rounded-xl font-semibold bg-primary
text-primary-foreground hover:shadow-lg hover:shadow-primary/30 transition-all duration-200"
            >
              Try Again
            </button>
          </div>
        )}

        <div className="grid grid-cols-2 gap-4 pt-6 border-t border-border">
          <div className="space-y-2">
            <div className="flex items-center gap-2">
              <div className="w-2 h-2 rounded-full bg-primary" />
              <span className="text-sm font-medium text-foreground">
                Quick Processing
              </span>
            </div>
            <p className="text-xs text-muted-foreground">
              Get your 3D model in minutes
            </p>
          </div>
          <div className="space-y-2">
            <div className="flex items-center gap-2">
              <div className="w-2 h-2 rounded-full bg-primary" />
              <span className="text-sm font-medium text-foreground">
                High Quality
              </span>
            </div>
            <p className="text-xs text-muted-foreground">
              Professional-grade 3D models
            </p>
          </div>
        </div>
      </div>
```

```
          <div className="space-y-4">
            <div className="sticky top-24">
              <h2 className="text-2xl font-bold text-foreground mb-4">
                3D Preview
              </h2>

              {currentStep === "success" && modelUrl ? (
                <Canvas3D modelUrl={modelUrl} className="h-[600px]" />
              ) : (
                <div className="w-full h-[600px] rounded-2xl border-2 border-dashed
border-border bg-muted/30 flex flex-col items-center justify-center space-y-3">
                  <div className="w-12 h-12 rounded-full bg-primary/10 flex items-center
justify-center">
                    <Zap className="w-6 h-6 text-primary" />
                  </div>
                  <div className="text-center space-y-1">
                    <p className="font-semibold text-foreground">
                      {currentStep === "processing"
                        ? "Generating..."
                        : "Ready for Preview"}
                    </p>
                    <p className="text-sm text-muted-foreground">
                      {currentStep === "processing"
                        ? "Your 3D model will appear here"
                        : "Upload an image to get started"}
                    </p>
                  </div>
                </div>
              )}
            </div>
          </div>
        </div>
      </div>
    </div>
  );
}

import "./global.css";


const queryClient = new QueryClient();


const App = () => (
  <QueryClientProvider client={queryClient}>
    <TooltipProvider>
```

```
      <Toaster />
      <Sonner />
      <BrowserRouter>
        <Header />
        <main>
          <Routes>
            <Route path="/" element={<Index />} />
            <Route path="*" element={<NotFound />} />
          </Routes>
        </main>
      </BrowserRouter>
    </TooltipProvider>
  </QueryClientProvider>
);

const root = document.getElementById("root");
if (root) {
  createRoot(root).render(<App />);
}

<?php

if ( ! defined('ABSPATH') ) exit;

register_activation_hook(__FILE__, function() {
    global $wpdb;
    $table = $wpdb->prefix . 'three_jobs';
    $charset_collate = $wpdb->get_charset_collate();
    $sql = "CREATE TABLE IF NOT EXISTS $table (
      id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
```

```
    status VARCHAR(32) NOT NULL DEFAULT 'queued',
    src_url TEXT NOT NULL,
    model_url TEXT NULL,
    preview_url TEXT NULL,
    params JSON NULL,
    created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
  ) $charset_collate;";
  require_once ABSPATH . 'wp-admin/includes/upgrade.php';
  dbDelta($sql);
});

add_action('admin_menu', function() {
  add_menu_page(
    '2D→3D Converter',
    '2D→3D Converter',
    'manage_options',
    'two-to-three',
    'two_to_three_admin_app',
    'dashicons-media-code',
    58
  );
});

add_action('admin_enqueue_scripts', function($hook) {
  if ( $hook !== 'toplevel_page_two-to-three' ) return;
  // Зібраний React-бандл (див. підрозділ 4.1.3)
  wp_enqueue_script('two-to-three-app', plugins_url('build/index.js', __FILE__), ['wp-api-fetch'], '1.0.0', true);
  wp_enqueue_style('two-to-three-style', plugins_url('build/index.css', __FILE__), [], '1.0.0');
```

```
    wp_localize_script('two-to-three-app', 'T23', [
        'rest'  => esc_url_raw( rest_url('t23/v1/') ),
        'nonce' => wp_create_nonce('wp_rest'),
        'uploads' => wp_get_upload_dir(),
    ]);
});

import { useLocation } from "react-router-dom";
import { useEffect } from "react";
const NotFound = () => {
  const location = useLocation();
  useEffect(() => {
    console.error(
      "404 Error: User attempted to access non-existent route:",
      location.pathname,
    );
  }, [location.pathname]);
  return (
    <div className="min-h-screen flex items-center justify-center bg-gray-100">
      <div className="text-center">
        <h1 className="text-4xl font-bold mb-4">404</h1>
        <p className="text-xl text-gray-600 mb-4">Oops! Page not found</p>
      </div>
    </div>
  );
};

export default NotFound;
```