МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ОНЛАЙН ВІЗУАЛІЗАТОРА ГЕОМЕТРИЧНИХ ОБ'ЄКТІВ»**

Виконав: студент 4 курсу, групи 6.1211-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
Д.Є. Грішанов
(ініціали та прізвище)

Керівник завідувач кафедри програмної інженерії, доцент, к.ф.-м.н. Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Решевська К.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Грішанову Данилу Євгеновичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка онлайн візуалізатора геометричних об’єктів

керівник роботи Лісняк Андрій Олександрович, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 02.06.2025 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз інструментів, існуючих рішень та постановка вимог до застосунку.
2. Проєктування системи.
3. Програмна реалізація проєкту.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 26.12.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	24.01.2025	
2.	Збір вихідних даних.	14.02.2025	
3.	Обробка методичних та теоретичних джерел.	04.03.2025	
4.	Розробка першого та другого розділу.	04.04.2025	
5.	Розробка третього розділу.	19.05.2025	
6.	Оформлення та нормоконтроль	26.05.2025	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	16.06.2025	

Студент _____ Д.Є. Грішанов
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.О. Лісняк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка онлайн візуалізатора геометричних об’єктів»: 52 с., 43 рис., 2 табл., 23 джерел, 1 додаток.

БАЗА ДАНИХ, ВІЗУАЛІЗАЦІЯ, МОДЕЛІ, ОБ’ЄКТИ, 3D, JS, NODE.JS, THREE.JS.

Об’єкт дослідження – процес розробки додатку для взаємодії та візуалізації моделей.

Мета роботи: проєктування та розробка додатку для взаємодії з 3D-моделями та геометричними об’єктами.

Метод дослідження – моделювання, проєктування, програмний та аналітичний методи.

У роботі розглянуто процес проєктування та розробки вебдодатку для візуалізації геометричних об’єктів з використанням Node.js, Three.js, Express.js, Vue.js та бази даних MongoDB. Використано UML для демонстрації можливостей системи, зокрема було використано діаграми варіантів використання, послідовності, діяльності та розгортання.

Система надає можливості завантажувати, редагувати, переглядати, видаляти моделі. Реалізовано систему реєстрації та автентифікації, зберігання даних про модель у MongoDB.

Саме так, розроблений вебзастосунок забезпечує зручний та зрозумілий інтерфейс для перегляду, керування та поширення власних 3D-моделей. Використання Node.js, Three.js, Express.js, Vue.js та бази даних MongoDB мало вплив на створення ефективної системи.

SUMMARY

Bachelor’s qualifying paper “Development of an Online Visualizer of Geometric Objects”: 52 pages, 43 figures, 2 tables, 23 references, 1 supplement.

3D, DATA BASE, JS, MODELS, NODE.JS, OBJECTS, THREE.JS., VISUALIZATION.

The object of the study is the process of developing an application for interaction and visualization of models.

The aim of the study is designing and developing of an application for interaction with 3D models and geometric objects.

The methods of research are modeling, design, software and analytical methods.

The work examines the process of designing and developing a web application for visualizing geometric objects using Node.js, Three.js, Express.js, Vue.js, and the MongoDB database. UML was used to demonstrate the capabilities of the system, in particular, use case, sequence, activity and deployment diagrams were used.

The system provides the ability to upload, edit, view, delete models. A registration and authentication system has been implemented, and model data is stored in MongoDB.

This is how the developed web application provides a convenient and intuitive interface for viewing, managing, and sharing your own 3D models. The use of Node.js, Three.js, Express.js, Vue.js and the MongoDB database contributed to the creation of an efficient system.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Вступ 8
1 Аналіз інструментів, існуючих рішень та постановка вимог до застосунку 10
1.1 Поточний стан і розвиток 3D-візуалізації у вебсередовищі 10
1.2 Огляд технологій 3D-візуалізації 10
1.2.1 WebGL 10
1.2.2 Babylon.js 11
1.2.3 Three.js 11
1.2.4 Порівняння WebGL, Babylon.js та Three.js 12
1.3 Огляд форматів 3D-моделей 12
1.4 Огляд та аналіз існуючих рішень 13
1.5 Постановка вимог до візуалізатора 14
1.6 Висновки до першого розділу 15
2 Проєктування вебдодатку 16
2.1 Використання UML-під час розробки системи 16
2.2 Діаграма варіантів використання 16
2.2.1 Опис варіантів використання 20
2.3 Діаграма діяльності 25
2.4 Діаграма послідовності 29
2.5 Діаграма розгортання 30
2.6 Висновки до другого розділу 32
3 Програмна реалізація проєкту 33
3.1 Інструменти розробки застосунку 33
3.1.1 Node.js 33
3.1.2 Vue.js 33
3.1.3 Express.js 34

3.2 Основний функціонал системи 34
3.3 Підключення MongoDB 35
3.4 Взаємодія користувача з системою 36
3.4.1. Навігація користувача 36
3.4.2 Реєстрація в системі 37
3.4.3 Вхід та вихід до/з системи 39
3.4.4 Завантаження моделей 41
3.4.5 Керування моделями. 43
3.5 Висновки до третього розділу 48
Висновки 49
Перелік посилань 50
Додаток А Посилання на Git 52

ВСТУП

У сучасному світі тривимірна графіка має ключову роль у багатьох галузях: від інженерного проєктування, архітектури, кінематографу до розробки ігор, медичної сфери та освіти. Паралельно тому, як поширювалися вебтехнології, зросла потреба у зручних, доступних та інтерактивних інструментах для перегляду 3D-об’єктів, які не потребують встановлення спеціалізованого програмного забезпечення. Саме за цієї причини, актуальною задачею є створення вебзастосунків, які здатні надати змогу звичайному користувачу працювати з 3D-об’єктами безпосередньо в межах браузеру.

Вебвізуалізатори об’єктів надають можливості для створення все можливих інтерактивних інструментів, такі як: інтерактивні презентації, онлайн-магазини, освітні платформи та технічні документації. Користувачі подібних візуалізаторів мають змогу перегляди складні просторові об’єкти в режимі реального часу та взаємодіяти з ними (змінювати ракурс, масштаб об’єктів та переглядати їх деталі).

З огляду на це, було прийнято рішення розробити додаток, який буде одночасно простим у користуванні, безпечним та функціональним.

Метою даної роботи є проєктування та розробку додатку для взаємодії з 3D-моделями та геометричними об’єктами.

Для досягнення поставленої мети необхідно виконати такі задачі:

– оглянути існуючі рішення;
– сформувати технічні вимоги до додатку;
– спроєктувати та реалізувати додаток;
– протестувати додаток.

Об’єктом дослідження є процес розробки додатку для взаємодії та візуалізації з 3D-моделями.

Предметом дослідження є бібліотеки Vue.js, Three.js, Express.js, Node.js та база даних MongoDB.

Методами дослідження є моделювання, проєктування, програмний та аналітичний.

У першому розділі розглянуто технології вебвізуалізації та їх порівняльна характеристика, формати 3D-моделей, аналоги розроблюваного додатку, їх порівняння та буде постановлено функціональні та нефункціональні вимоги до застосунку.

Другий розділ присвячено проєктуванню системи вебзастосунку. Буде продемонстровано діаграми варіантів використання, діяльності, послідовності та розгортання. Також буде наведено опис прецедентів з діаграми варіантів використання.

У третьому розділі наведено опис використаних інструментів, реалізованого функціоналу додатку та налаштування підключення застосунку до бази даних MongoDB. Також буде наведено приклади взаємодії користувача з розробленим додатком.

1 АНАЛІЗ ІНСТРУМЕНТІВ, ІСНУЮЧИХ РІШЕНЬ ТА ПОСТАНОВКА ВИМОГ ДО ЗАСТОСУНКУ

1.1 Поточний стан і розвиток 3D-візуалізації у вебсередовищі

3D-візуалізація, у веброзробці, набула широкого застосування в період останнього десятиліття завдяки розвитку апаратного прискорення та появи стандарту WebGL (Web Graphics Library). У часи початку розвитку інтернет браузерів перегляд тривимірних об’єктів потребував встановлення додаткових плагінів (Unity Web player або Flash player), однак, сьогодення, сучасні браузери мають можливість нативної підтримки відображення 3D-графіки.

Вебвізуалізація, у сучасному світі, є надзвичайно актуальною в таких галузях, як: електронна комерція (перегляд 3D-моделей товарів), архітектура(презентація макетів), промисловий дизайн (перегляд макетів деталей), медицина та освіта.

Саме розвиток інструментів та фреймворків, що базуються на WebGL, дав змогу будувати складні 3D-сцени навіть на мобільних пристроях.

1.2 Огляд технологій 3D-візуалізації

1.2.1 WebGL

WebGL – це кросплатформений API для рендернигу 2D- і 3D-графіки в межах браузеру без використання сторонніх плагінів.

Перевагами WebGL є: можливість прямого доступу до GPU для апаратного прискорення візуалізації об’єктів; рендеринг сцен з великою кількістю полігонів; повна підтримка в усіх основних браузерах (Chrome, Firefox, Edge, Safari).

Основним недоліком WebGL є висока складність реалізації та використання: для створення сцени з об’єктом, камерою, освітлення матеріалами потрібно велика кількість низькорівневого коду. Саме за цієї причини почали з’являтися численні фреймворки-обгортки [1, 2].

1.2.2 Babylon.js

Babylon.js – це потужний фреймворк для побудови 3D-додатків. Він має вбудовану підтримку фізики, анімацій, постобробки та шейдерів. Основною орієнтацією цієї бібліотеки є ігрова розробка, ніж візуалізація.

Перевагами Babylon.js є: повноцінна ігрова фізика; вбудований редактор сцен; розширені можливості камери та контролерів.

Недоліками цієї бібліотеки є складніше опанування; більший розмір бібліотеки та, в нашому випадку, орієнтованість цієї бібліотеки на ігрову розробку [3, 4].

1.2.3 Three.js

Three.js – це JavaScript-бібліотека з відкритим кодом, яка спрощує розробку 3D-сцен. Вона абстрагує складність WebGL, надаючи готові об’єкти для камер, світла, геометрії, матеріалів, завантаження моделей, анімацій, тощо [5, 8].

Основними компонентами Three.js є:

– Scene: контейнер для всіх 3D-елементів;
– Camera: точка зору;
– Renderer: механізм рендерингу;
– Mesh: поєднання геометрії та матеріалу;
– Loader: модулі для завантаження моделей різних форматів (GLB, STL, OBJ, тощо).

Перевагами цієї бібліотеки є її простота, активна спільнота розробників, доступна документація, підтримка великої кількості форматів моделей, сумісність з багатьма сучасними браузерами, можливість інтеграції з іншими фреймворками, регулярні оновлення, підтримка від розробників та приклади на офіційному сайті [5-8].

1.2.4 Порівняння WebGL, Babylon.js та Three.js

Порівняльна характеристика WebGL, Babylon.js та Three.js продемонстрована у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика технологій для роботи з 3D-графікою

Критерій	**WebGL**	**Three.js**	**Babylon.js**
Тип	API низького рівня	Високорівнева бібліотека	Повноцінний ігровий рушій
Рівень складності	Високий	Середній	Низький
Мова програмування	JavaScript	JavaScript/TypeScript	JavaScript/TypeScript
Призначення	Керування GPU	3D графіка, візуалізація сцен та моделей	Ігри, інтерактивні сцени, VR/AR
Підтримка сцен	Сцени моделюються вручну	Автоматична модуляція	Високорівнева модуляція
Документація	Технічна	Широка документація з прикладами	Дуже детальна документація з офіційною пісочницею
Використання	Професійні застосування, симуляції	Вебвізуалізації, 3D-сайти	Ігри, VR/AR-додатки, великі 3D-проєкти

1.3 Огляд форматів 3D-моделей

Формат 3D-моделі є важливим, оскільки вибір формату впливає на такі параметри, як якість відображення, сумісність із програмним забезпеченням,

швидкість завантаження та можливості подальшого використання моделі – у веб, 3D-друці чи віртуальній реальності. Саме тому є важливим розглянути популярні формати 3D-моделей.

Формат .GLB є сучасним форматом для передачі 3D-моделей через всесвітню мережу Інтернет розроблений Khronos Group. Має підтримку текстур, анімацій та PBR-матеріалів, що у поєднанні робить цей формат ідеальним для вебдодатків та мобільних платформ. Зберігається у вигляді єдиного GLB-файлу. Основними перевагами є компактність, швидкість завантаження та активна підтримка спільноти. Недоліком є обмеженість підтримки у застарілому програмному забезпечені [9].

Формат .OBJ – популярний формат збереження геометрії 3D-моделей. Присутня підтримка вершин, нормалей, координат текстур та матеріалів, які описуються окремим .mtl файлом. Цей формат є простим та сумісним із більшістю програм, але не має підтримки анімацій та має більший розмір файлів [9].

Формат .STL є базовим форматом для 3D-друку, який описує модель як сітку трикутників. Не має підтримки кольорів, текстур чи матеріалів. Є ідеальним для створення фізичних моделей, але непридатний для повноцінних візуалізацій. Перевагою цього формату є простота та підтримка в усіх програмах для 3D-друку [9].

1.4 Огляд та аналіз існуючих рішень

Перед тим як постановити вимоги до застосунку, є важливим врахувати існуючі рішення, їхні можливості та обмеження. У таблиці 1.2 наведено порівняльну характеристику таких рішень, як Sketchfab, Three.js Editor та Babylon.js Viewer.

Sketchfab – це онлайн платформа для перегляду, обміну та публікації 3D-моделей. Ця платформа підтримує велику кількість форматів (наприклад GLTF, FBX, OBJ). Вона має розвинену систему матеріалів, а також підтримку

анімацій та VR/AR перегляду. Інтерфейс орієнтовано на кінцевого користувача з великим вибором моделей та обміном моделей між користувачами [10].

Three.js Editor – це браузерний редактор, який було засновано на основі бібліотеки Three.js, призначений для створення, редагування та перегляду 3D-сцен. Він має можливості гнучкого керування світлом, матеріалами, камерою та має підтримку анімацій та експорту сцен. В основному орієнтований на розробників та дизайнерів [11].

Babylon.js Viewer є легким та настроюваним візуалізатором можливості якого дозволяють легко вбудовувати 3D-моделі на вебсторінки з мінімальною кількістю строк коду. Має підтримку багатьох форматів, режими VR/AR, освітлення та фізику [12].

Таблиця 1.2 – Порівняльна характеристика існуючих рішень 3D-вебвізуалізаторів

Характеристика	**Sketchfab**	**Three.js Editor**	**Babylon.js Viewer**
Підтримка форматів	Широка підтримка форматів	Широка підтримка форматів	Підтримка популярних форматів
Базові трансформації	Присутні	Присутні	Присутні
Налаштування відображення	Підтримка матеріалів, текстур, сітки	Підтримка матеріалів	Розширена підтримка освітлення
Галерея моделей	Велика-онлайн галерея	Відсутня	Відсутня
Автентифікація	Відсутня власна автентифікація	Відсутня власна автентифікація	Відсутня власна автентифікація
Анімації	Підтримуються	Підтримуються	Підтримуються
VR/AR	Повна підтримка	Відсутня підтримка	Повна підтримка

1.5 Постановка вимог до візуалізатора

На основі аналізу існуючих рішень визначено функціональні та нефункціональні вимоги до розроблюваного вебдодатку.

Функціональними вимогами до застосунку є:

– підтримка форматів файлів GLB, OBJ, STL;
– можливість завантаження моделей як з локальної системи так і з галереї;
– перегляд моделей з різноманітних ракурсів, що включає в себе масштабування, обертання та переміщення моделі;
– налаштування відображення сцени, що поєднує в собі налаштування сітки, осей координат та зміну кольору фону;
– інтеграція з бекендом для збереження та завантаження моделей.

Нефункціональними вимогами є:

– валідація типів файлів при завантаженні моделей;
– обробка помилок при завантаженні моделей;
– інтуїтивно зрозумілий інтерфейс;
– оптимізація розмірів моделі під розмір сцени та її автоматичне масштабування.

1.6 Висновки до першого розділу

У першому розділі було проведено аналіз поточного стану 3D-візуалізації у вебсередовищі. Також було розглянуто, проаналізовано та порівняно три популярні технології 3D-візуалізації – WebGL, Babylon.js та Three.js з метою визначення їх переваг, недоліків та сумісність з потребами та обмеженнями проєкту. У доповнені до цього, було розглянуто формати 3D-моделей, які підтримують сучасні візуалізатори. Крім того, було проаналізовано та порівняно вже існуючі на ринку аналоги, на основі якого було визначено вимоги до функціоналу візуалізатора та стек технологій для реалізації проєкту.

2 ПРОЄКТУВАННЯ ВЕБДОДАТКУ

2.1 Використання UML-під час розробки системи

Unified Modeling Language (UML) – це стандартна мова моделювання, яка використовується для візуалізації, специфікації, конструювання та документації програмних систем. Використання UML під час розробки системи має критичний вплив на процес розробки програмного забезпечення, так як дозволяє візуалізувати та задокументувати функціональні вимоги до розроблюваної системи, що забезпечує спільне розуміння між замовниками, аналітиками та розробниками. Також, UML надає широкий перелік діаграм для моделювання архітектури системи. Поведінкові діаграми UML, такі як діаграми послідовності, стану та зв’язку, дозволяють краще розуміти динамічну поведінку системи, послідовність взаємодій між об’єктами системи та послідовність переходів між станами [13-15].

Використання UML заохочує до повторного використання коду, оскільки дозволяє не тільки моделювати компоненти та їх взаємозв’язки, а й полегшує виявлення можливостей для повторного використання коду та проєктування модульною архітектури.

Діаграми UML – стандартизовані, тому вони полегшують розуміння для всіх учасників проєкту, що також полегшує створення документації та передачу інформації про систему.

2.2 Діаграма варіантів використання

Діаграма варіантів використання (use case diagram) – діаграма, яка використовується для візуалізації та документування функціональних вимог системи з точки взаємодії акторів з системою. На ній зображуються актори (внутрішні та зовнішні сутності, які взаємодіють з системою) та варіанти

використання (дії або їх послідовність, які, при виконанні системою, надають певний функціонал) [14].

Діаграма варіантів використання дозволяє виконувати завдання, такі як визначення меж системи, опис функціональних вимог з точки зору користувача, ідентифікація акторів та їх ролі, візуалізація взаємодій між варіантами використання.

Під час створення діаграми варіантів використання є важливим дотримуватися основних принципів та правил для її створення, таких як: використання простої та зрозумілої термінології, чітке визначення меж системи, уникнення надмірної деталізації або надмірної абстракції, відображення основних потоків подій та альтернативних сценаріїв [13].

На рисунку 2.1 зображено діаграму варіантів використання 3D-вебвізуалізатору.

На діаграмі представлено акторів «Гість» та «Зареєстрований користувач», кожен з яких може взаємодіяти з системою. Також присутній актор «Користувач», який включає спільні функціональні можливості гостя та зареєстрованого користувача, як користувачів системи.

Основним варіантом використання є «Додати модель». Після того, як користувач до власного кабінету до пункту меню «Додати модель». Система відображає інтерфейс завантаження нової моделі. Користувач задає назву та опис.

Користувач може додати модель натиснувши на кнопку «Обрати файл», система відкриє огляд пристрою користувача, з якого він зможе обрати необхідну модель. Таким самим чином користувач задає мініатюру моделі.

Після вибору моделі, її назви, опису та мініатюри, користувач натискає на кнопку «Опублікувати модель, система завантажує модель в MongoDB та зберігає дані на сервері.

Після успішного завантаження модель можна переглянути або в кабінеті користувача, або в галереї, в залежності від налаштувань приватності при завантажені моделі.

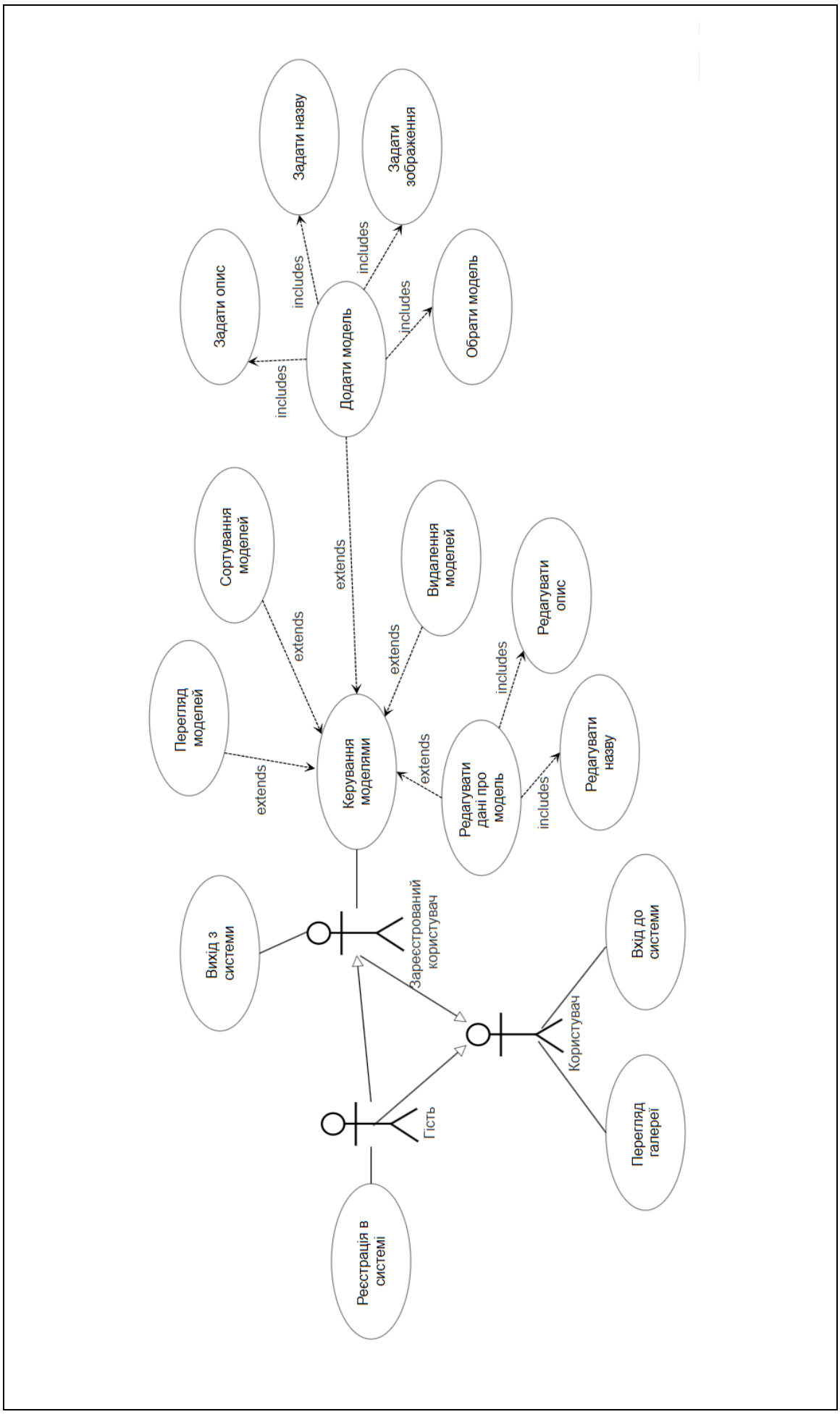


Рисунок 2.1 – Діаграма варіантів використання

Варіанти використання визначають функціональні можливості системи та засоби їх використання користувачами чи зовнішніми акторами. Кожен варіант використання представляє послідовність дій, яку може виконати актор для того щоб досягти певної мети або результату в межах взаємодії з конкретною системою. Вони є потужним інструментом для документації потреб до функціональних можливостей розроблюваної системи. Вони описують різноманітні способи взаємодії користувачів або сторонніх сутностей з системою для досягнення певних цілей чи результатів (див. рис. 2.2 – 2.3).

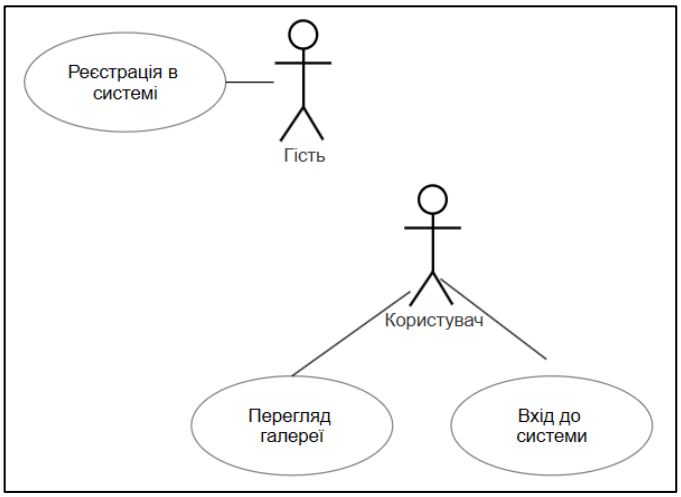


Рисунок 2.2 – Діаграма варіантів використання «Гість» та «Користувач»

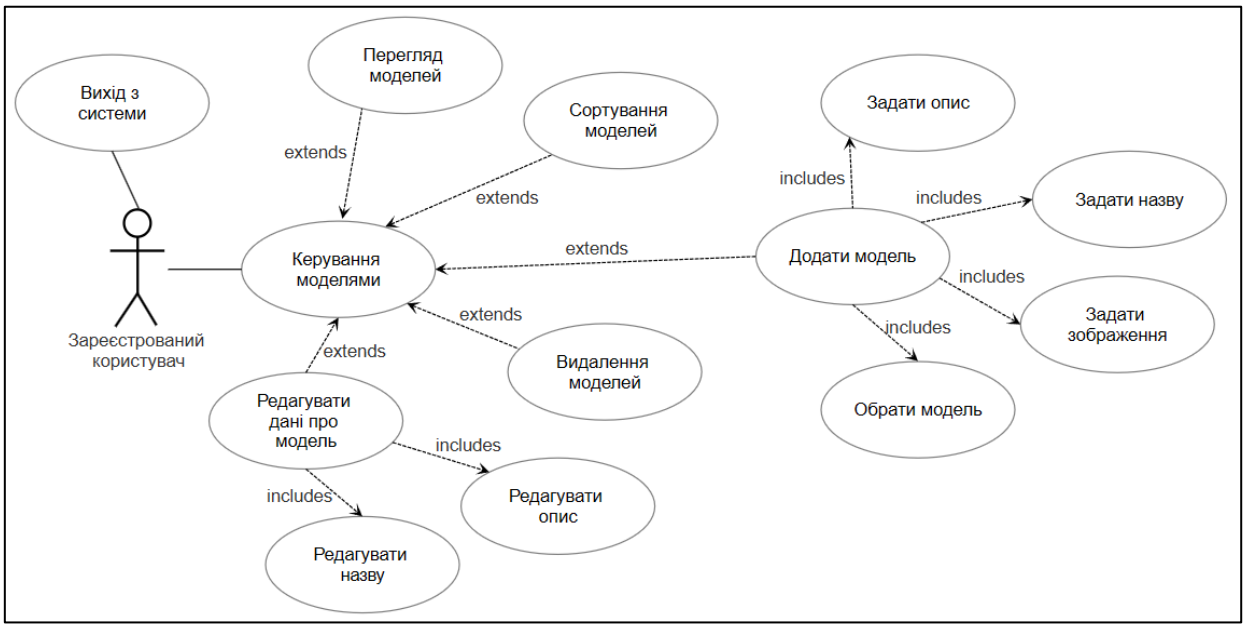


Рисунок 2.3 – Діаграма варіантів використання «Зареєстрований користувач»

2.2.1 Опис варіантів використання

Прецедент «Реєстрація в системі».

Призначення: надає дозвіл гостю створити новий обліковий запис для доступу до основної частини функцій додатку.

Основний потік подій:

– гість на головній сторінці натискає кнопку «Почати», або на навігаційній панелі натискає кнопку «Зареєструватися»;
– введення необхідних даних (логін, email та пароль);
– натиск на кнопку «Зареєструватися»;
– система перевіряє дані та створює обліковий запис;
– гість отримує підтвердження про успішну реєстрацію.

Виняткові ситуації:

– введені дані є некоректними або ці дані вже зайняті іншим користувачем – вивід повідомлення про помилку;
– проблеми з сервером – вивід повідомлення про технічну помилку.

Прецедент «Вхід до системи».

Призначення: надає дозвіл користувачу авторизуватися для доступу до основної частини функцій додатку.

Основний потік подій:

– користувач відкриває форму входу;
– введення пошти та паролю;
– натиск на кнопку «Увійти»;
– система перевіряє дані та авторизує користувача.

Виняткові ситуації:

– невірно введена пошта або пароль – виведення повідомлення про помилку;
– проблема з сервером – вивід повідомлення про технічну помилку.

Прецедент «Вихід з системи».

Призначення: дозволяє користувачу завершити сесію та вийти з облікового запису.

Основний потік подій:

– користувач натискає кнопку «Вийти» на навігаційній панелі;
– система завершує сесію та змінює статус зареєстрованого користувача на статус гостя.

Прецедент «Керування моделями».

Призначення: дозволяє зареєстрованому користувачу додавати, редагувати, переглядати та видаляти свої 3D-моделі.

Основний потік подій:

– користувач переходить у власний кабінет;
– з власного кабінету користувач має можливість додавати, редагувати, переглядати та видаляти власні моделі.

Виняткова ситуація: проблема з сервером – повідомлення про технічну помилку.

Прецедент «Додати модель».

Призначення: надає можливість зареєстрованому користувачу завантажити власну 3D-модель у систему.

Основний потік подій:

– користувач відкриває форму додавання моделі у власному кабінеті;
– вказує назву та опис, завантажує файли зображення та моделі;
– натискає на кнопку «Опублікувати модель»;
– система перевіряє введені дані та зберігає модель.

Виняткові ситуації:

– некоректний формат моделі – відповідне повідомлення про помилку;
– некоректний розмір моделі – відповідне повідомлення про помилку;
– некоректні дані, деякі поля не заповнені або відсутні файли зображення або моделі – відповідне повідомлення про помилку;
– проблеми з сервером – повідомлення про технічну помилку.

Прецедент «Задати опис».

Призначення: надає дозвіл зареєстрованому користувачу задати опис при додаванні моделі.

Основний потік подій:

– під час прецеденту «Додати модель» у полі «опис» ввести опис моделі;
– збереження опису.

Виняткова ситуація: порожній опис – відповідне повідомлення про помилку.

Прецедент «Задати назву».

Призначення: дає змогу зареєстрованому користувачу задати назву при додаванні моделі.

Основний потік подій:

– під час прецеденту «Додати модель» у полі «Назва» ввести назву моделі;
– збереження назви.

Виняткова ситуація: порожня назва – відповідне повідомлення про помилку.

Прецедент «Задати зображення».

Призначення: дає змогу зареєстрованому користувачу завантажити зображення-мініатюру для моделі.

Основний потік подій:

– під час прецеденту «Додати модель» у полі вибору зображення натиснути на кнопку «Вибір файлу»;
– обрати зображення з пристрою;
– система перевіряє формат файлу;
– зображення додається до форми додання моделі.

Виняткова ситуація: некоректний формат файлу – відповідне повідомлення про помилку.

Прецедент «Обрати модель».

Призначення: дозволяє користувачу вибрати файл 3D-моделі для завантаження.

Основний потік подій:

– під час прецеденту «Додати модель» у полі вибору моделі натиснути на кнопку «Вибір файлу»;
– обрати файл моделі коректного формату (.glb, .obj, .stl);
– система перевіряє формат та розмір файлу;
– файл моделі додається до форми.

Виняткові ситуації:

– некоректний формат файлу – відповідне повідомлення про помилку;
– некоректний розмір файлу – відповідне повідомлення про помилку.

Прецедент «Перегляд моделей».

Призначення: дозволяє користувачу переглядати свої моделі та їх список.

Основний потік подій:

– у своєму кабінеті користувач відкриває сторінку «Мої моделі»;
– система відображає список моделей користувача;
– на картці моделі натиснути кнопку «Переглянути»;
– відкривається вікно перегляду моделі з кнопкою «Перейти до вьювера», що дозволяє перейти у вікно перегляду моделі з подальшими налаштуваннями сцени.

Виняткові ситуації:

– проблема з сервером – повідомлення про технічну помилку;
– проблема завантаження моделі для перегляду – відповідне повідомлення про помилку.

Прецедент «Сортування моделей».

Призначення: дає можливість користувачу сортувати моделі за категоріями.

Основний потік подій:

– у власному кабінеті або у галереї користувач обирає розгортаємий список з усіма категоріями;
– система оновлює список моделей згідно обраною категорії.

Виняткова ситуація: проблема з сервером – повідомлення про технічну помилку.

Прецедент «Видалення моделей».

Призначення: дає можливість зареєстрованому користувачу видалити свою модель із системи.

Основний потік подій:

– у власному кабінеті на картці моделі користувач натискає кнопку «Видалити»;
– система запитує підтвердження;
– після підтвердження модель видаляється.

Виняткова ситуація: проблема з сервером – повідомлення про технічну помилку.

Прецедент «Перегляд галереї».

Призначення: надає змогу користувачу переглядати галерею з загальнодоступними моделями.

Основний потік подій:

– користувач відкриває сторінку галереї;
– система відображає список моделей;
– в залежності від того чи зареєстрований користувач, він може переглядати та сортувати моделі.

Виняткові ситуації:

– користувач незареєстрований, але намагається переглянути модель – повідомлення про те що користувач незареєстрований. Перенаправлення на сторінку реєстрації;
– проблеми з сервером – повідомлення про технічну помилку.

Прецедент «Редагувати дані про модель».

Призначення: надає змогу зареєстрованому користувачу змінити інформацію про модель.

Основний потік подій:

– у власному кабінеті на картці моделі натиснути на кнопку «Редагувати»;
– ввести зміни (опис, назва);
– збереження змін.

Виняткова ситуація: некоректні дані або порожнє поле – відповідне повідомлення про помилку.

Прецедент «Редагувати опис».

Призначення: надає змогу зареєстрованому користувачу змінити опис моделі.

Основний потік подій:

– під час прецеденту «Редагувати дані про модель» змінити опис моделі;

– зберегти зміни.

Виняткова ситуація: некоректні дані або порожнє поле – відповідне повідомлення про помилку.

Прецедент «Редагувати назву».

Призначення: надає змогу зареєстрованому користувачу змінити назву моделі.

Основний потік подій:

– під час прецеденту «Редагувати дані про модель» змінити назву моделі;

– зберегти зміни.

Виняткова ситуація: некоректні дані або порожнє поле – відповідне повідомлення про помилку.

2.3 Діаграма діяльності

Діаграма діяльності (activity diagram) є важливою частиною UML, яка допомагає візуалізувати робочі процеси або дії в системі. Вона зображає, як різні дії пов’язані між собою та як система переходить з одного стану в інший. Надаючи чітке уявлення як простих, так і складних робочих процесів, діаграми діяльності полегшують розробникам та іншим зацікавленим сторонам розуміння того, як різні елементи взаємодіють в системі.

В діаграмах діяльності нотації схожі на візуальні символи, які дають змогу представити різні елементи та дії простим способом [13].

На рисунку 2.4 наведено частину діаграму діяльності прецеденту «Додати модель», яка зосереджена на взаємодії користувача з системою у процесі додавання нової моделі.

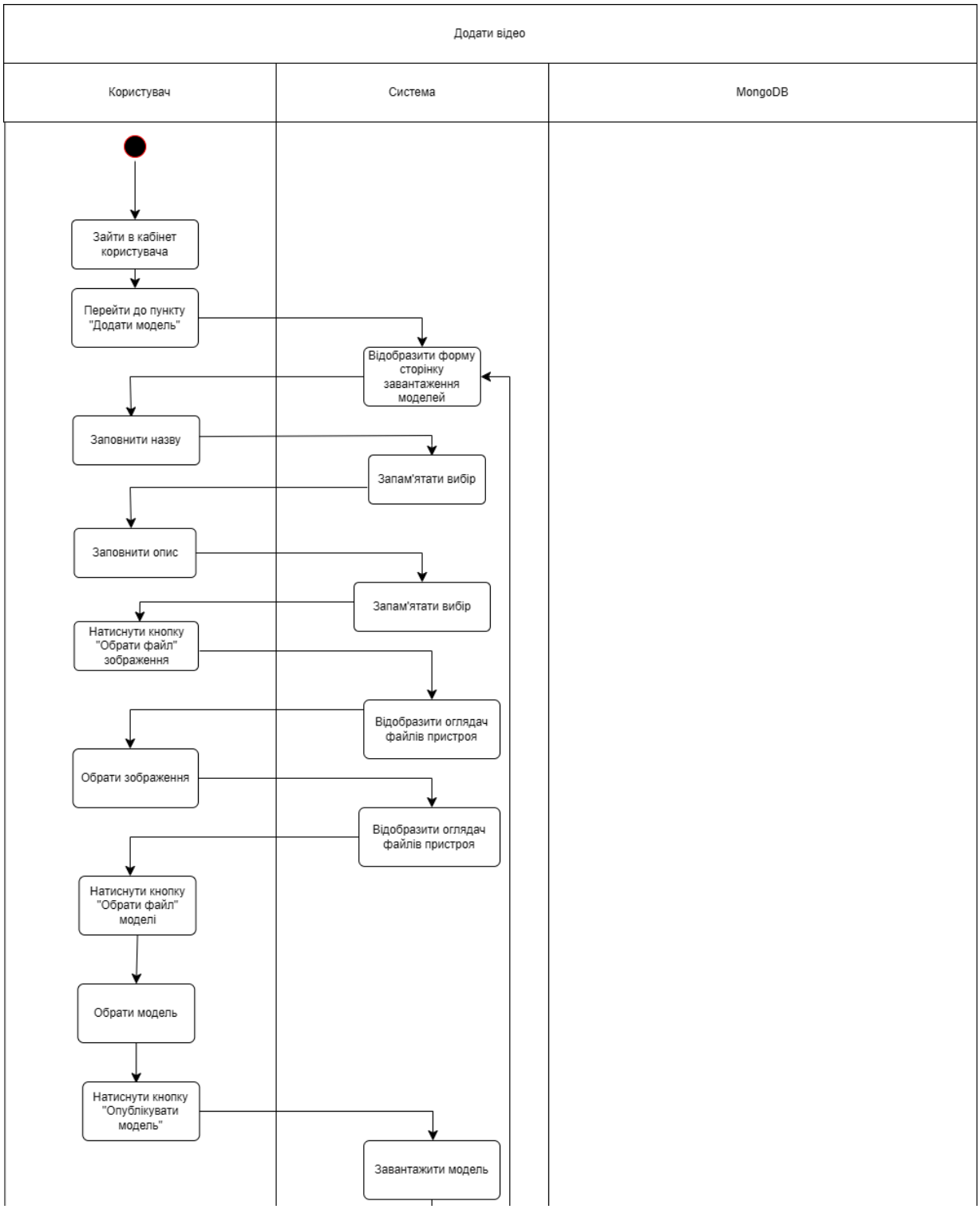


Рисунок 2.4 – Діаграма діяльності (1 частина)

Розглянемо послідовність дій з рисунку 2.4:

– користувач:
    1) зайти в кабінет користувача (ініціалізація);
    2) перейти до пункту «Додати модель»;
– система:
    3) відображення форми завантаження моделі (форма для введення інформації та завантаження);
– користувач:
    4) заповнити назву моделі;
– система:
    5) запам’ятати вибір (тимчасове збереження введених даних);
– користувач:
    6) заповнити опис моделі;
– система:
    7) запам’ятати вибір (тимчасове збереження введених даних);
– користувач:
    8) натиснути «Обрати файл» у полі зображення;
– система:
    9) відобразити огляд файлів пристрою;
– користувач:
    10) обрати файл зображення;
    11) натиснути «Обрати файл» у полі моделі;
– система:
    12) відобразити огляд файлів пристрою;
– користувач:
    13) обрати файл моделі;
    14) натиснути «Опублікувати модель»;
– система:
    15) завантажити модель (початок процесу передачі даних до БД).

На рисунку 2.5 зображено частину діаграми діяльності прецеденту «Додати модель», яка зосереджена на взаємодії системи з базою даних MongoDB у процесі додання моделей.

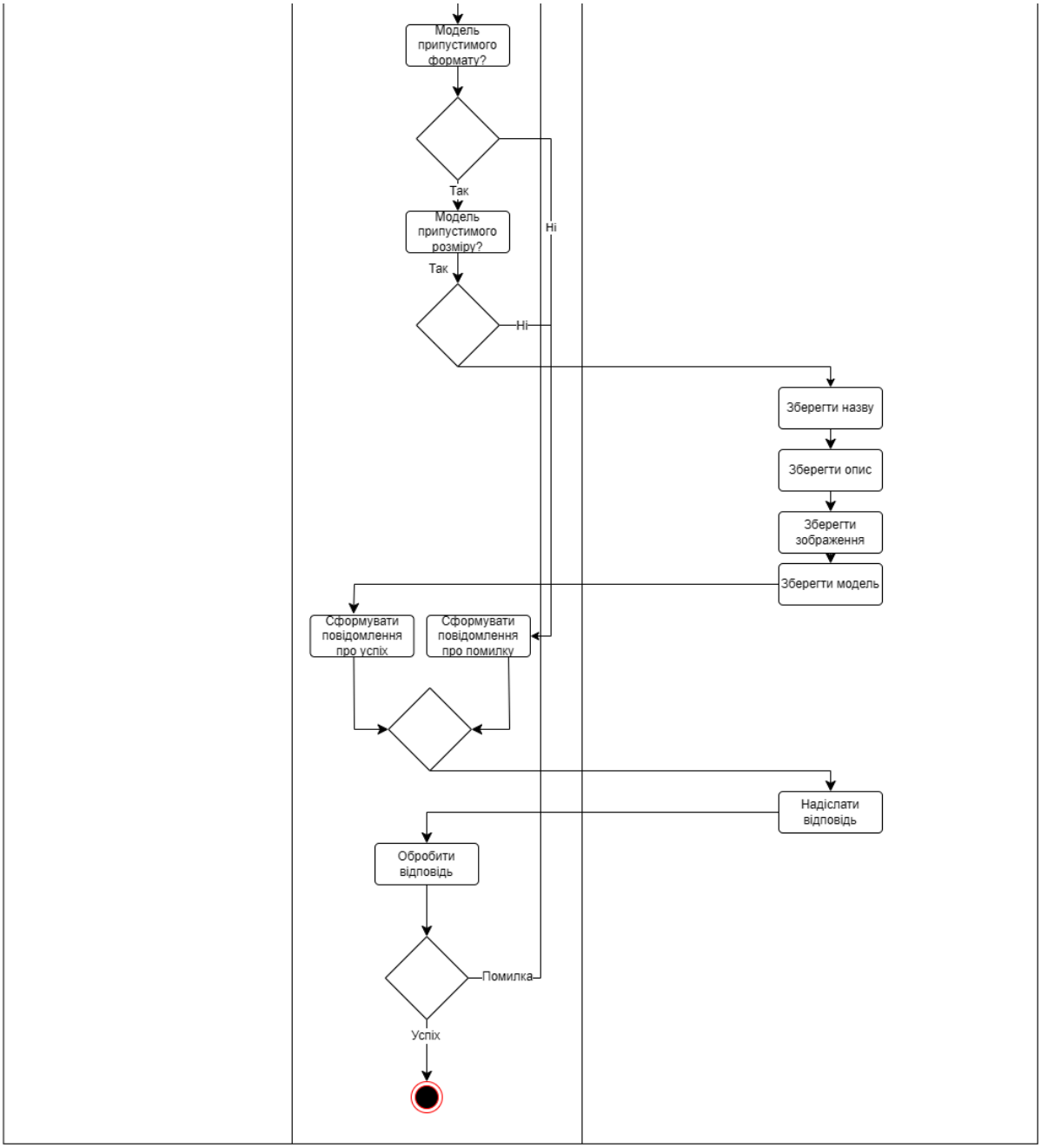


Рисунок 2.5 – Діаграма діяльності (2 частина)

Розглянемо послідовність дій з рисунку 2.5:

– система:

16) перевірка формату моделі: так – наступний крок; ні – сформувати повідомлення про помилку;

17) перевірка розміру моделі: так – наступний крок; ні – сформувати повідомлення про помилку;

– mongoDB:

18) збереження даних: назва моделі, опис моделі, зображення, файл моделі;

– система:

19) сформувати повідомлення про успіх;

– mongoDB:

20) надіслати відповідь про успіх;

– система:

21) обробити відповідь;

22) перевірка результату обробки: успіх – процес завершено; помилка – сформувати повідомлення про помилку та відобразити форму завантаження моделей.

2.4 Діаграма послідовності

Діаграма послідовності (Sequence diagram) – тип UML діаграми яка візуально демонструє взаємодію між об’єктами або компонентами з часом в системі. Вона фокусує на порядку та моментах надходження повідомлень або подіях, що обмінюються між різними елементами системи. Діаграма відображає, як об’єкти взаємодіють один з одним через серію сповіщень, забезпечуючи точне уявлення про послідовність операцій та процесів.

Діаграми послідовностей використовуються, оскільки вони пропонують чіткий та детальний огляд взаємодій між об’єктами або компонентами в системі, зосереджуючись на порядку та часі цих взаємодій [14].

Наведемо опис діаграми послідовності.

Актори: користувач, Frontend, Backend, MongoDB.

Послідовність взаємодій:

1) користувач → Frontend: відкриває сторінку «Додати модель»;

2) користувач → Frontend: заповнює форму та обирає файли;
3) користувач → Frontend: натискає «Опублікувати модель»;
4) FRONTEND → Backend: HTTP POST /api/models з формою та файлами;
5) Backend: валідує дані та перевіряє токен;
6) Backend: зберігає файли на сервері;
7) Backend → MongoDB: додає запис про модель;
8) MongoDB → Backend: підтвердження збереження;
9) Backend → Frontend: відповідь 201 Created;
10) Frontend: відображення повідомлення про успіх;
11) Frontend: оновлює список моделей.

На рисунку 2.6 зображено діаграму послідовності прецеденту «Додати модель».

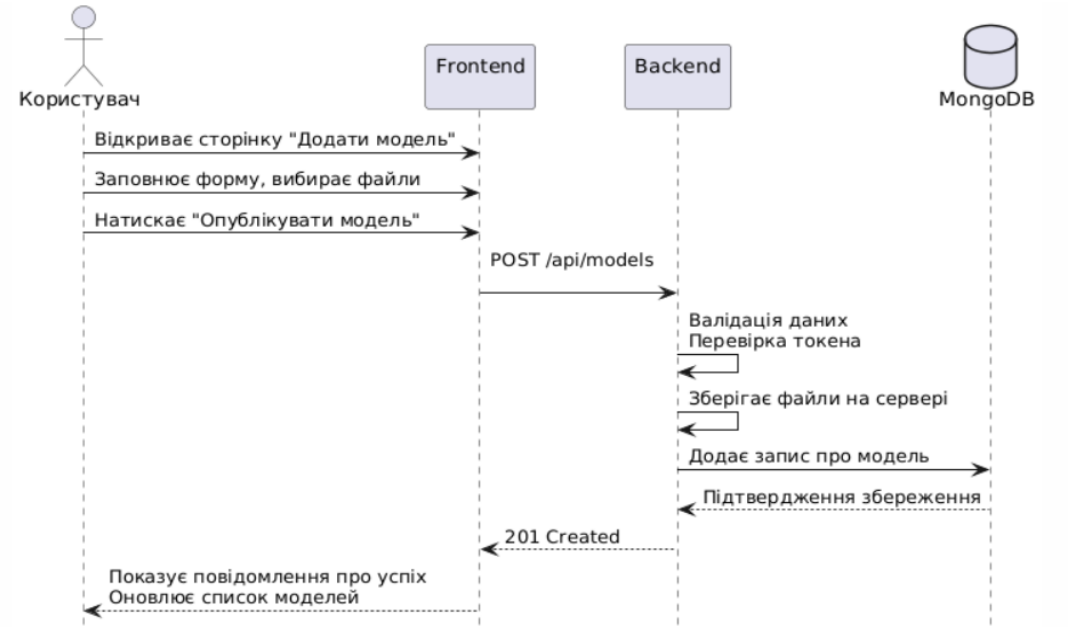


Рисунок 2.6 – Діаграма послідовності

2.5 Діаграма розгортання

Діаграма розгортання (Deployment Diagram) використовується для представлення фізичної архітектури системи та візуалізації розміщення

програмних компонентів які знаходяться на різних вузлах або пристроях. Ця діаграма демонструє зв’язки між артефактами програмного забезпечення, такими як файли, бібліотеки та компоненти, та фізичний вузол, на якому вони розміщені [13].

На рисунку 2.7 продемонстровано діаграму розгортання.

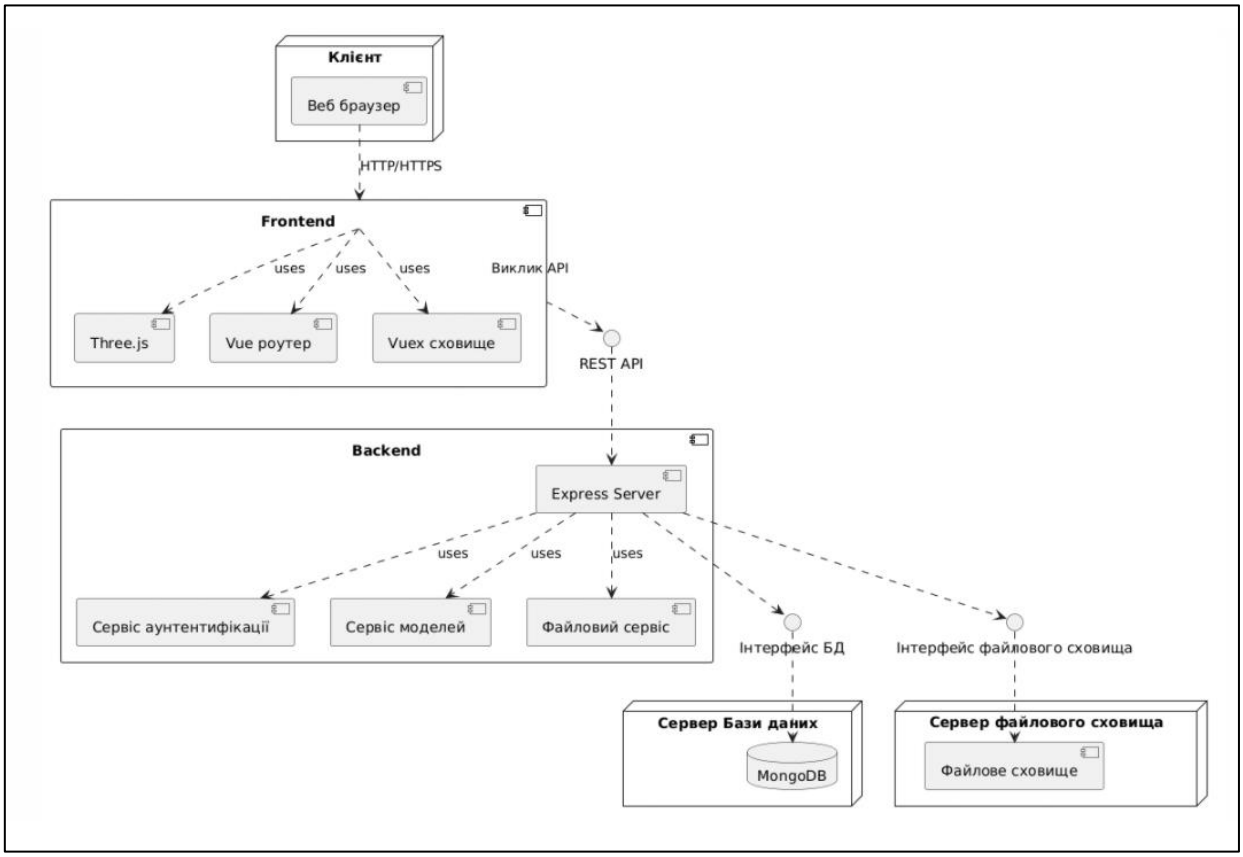


Рисунок 2.7 – Діаграма розгортання

Наведемо опис діаграми розгортання для розроблюваної системи.

Клієнтський пристрій (фізичний вузол) може бути як комп’ютер, так і смартфон користувача. В якості програмного забезпечення для взаємодії з застосунком використовується веббраузер. Основними функціями вузла є взаємодія з користувачем, відправка HTTP-запитів на сервер, отримання даних про моделі та завантаження моделей через форму у кабінеті користувача.

Сервер застосунку є логічним або фізичним вузлом, на якому працює серверна частина, який реалізовано за допомогою Node.js та Express.js. Основними функціями є обробка HTTP-запитів клієнта,

авторизація/автентифікація користувача, збереження файлів у сховище, збереження метаданих у БД та повернення відповідей клієнту.

База даних – MongoDB. Основними функціями БД є зберігання структурованих даних, якими є інформація про моделі та дані користувача, та обслуговування запитів на читання, запис та оновлення.

Файлове сховище може бути як локальним, так і хмарним. Основним функціями є зберігання 3D-моделей, надання доступу до моделей та отримання файлів від сервера.

1) зв’язки між вузлами;
2) клієнт ←→ Backend: відповідає за обмін HTTP-запити;
3) Backend ←→ MongoDB: відповідає за запити до БД;
4) Backend ←→ File Storage: відповідає за завантаження та зчитування 3D-моделей.

2.6 Висновки до другого розділу

У другому розділі було спроєктовано вебдодаток, описано та візуалізовано діаграму варіантів використання, діяльності, послідовності та розгортання. Також описано детально описано кожен прецедент зображений на діаграмі використання.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЄКТУ

3.1 Інструменти розробки застосунку

3.1.1 Node.js

Node.js є середовищем виконання JavaScript поза браузером. Воно дає змогу писати серверну логіку на мові клієнтської частини – JavaScript. Основними особливостями є:

– можливість обслуговувати велику кількість запитів одночасно без втрати продуктивності;
– швидкість виконання коду;
– доступ до великої кількості бібліотек, які спрощують розробку.

Node.js використовується для створення REST API, застосунків в реальному часі або сервісів для обробки файлів [17, 18].

3.1.2 Vue.js

Vue.js – JavaScript-фреймворк для побудови інтерфейсу користувача основними особливостями якого є:

– інтерфейс складається з окремих компонентів, які можливо повторно використовувати;
– легке поєднання з іншими бібліотеками;
– можливість створення динамічних вебзастосунків без перевантаження сторінки.

Vue.js підходить для розробки малих додатків та має зручну документацію [19, 20].

3.1.3 Express.js

Express.js – фреймворк для створення вебсерверів та API на Node.js. Одними з основних особливостей є:

– легке налаштування сервера та обробки маршрутів;
– зручне створення REST API з методами GET, POST, PUT, DELETE;
– можливість обробляти запити до або після маршруту;
– гнучка структура архітектури;
– легка інтеграціями з іншими модулями Node.js.

Express.js є одним з найпростіших фреймворків для побудови API на Node.js [21-23].

3.2 Основний функціонал системи

Застосунок, який було розроблено, виконує функції системи візуалізації 3D-моделей з підтримкою завантаження файлів, керування моделями, авторизації/автентифікації та взаємодії з користувачем. Так як головним прецедентом розробленої системи є «Додати модель», тоді основними класами будуть ті, які дають можливість взаємодіяти з моделями.

Клас **User** (клас користувача) – клас який представляє користувача в системі. Зберігає дані користувача, такі як ім’я, електронна адреса, пароль та дата створення.

Клас **Item** (клас моделей) – клас який представляє модель у базі даних. Містить інформацію про назву, опис, зображення, модель, категорію, статус публічності, дату завантаження та користувача який завантажив цю модель.

Клас **ModelViewer** є основним компонентом для відображення моделей. Відповідає за візуалізацію, керуванням та налаштуванням перегляду та відображення моделей.

Клас **AuthService** є сервісом автентифікації користувачів. Відповідає за вхід, реєстрацію, перевірку та оновлення токенів.

Клас **UserCabinet** являє собою особистий кабінет користувача. Відображає моделі користувача, сторінки перегляду моделей та форму додання нових моделей.

Клас **ItemService** – сервіс який керує 3D-моделями. Керує створенням, оновленням, видаленням моделей та завантаженням файлів.

Клас **FileHandler** відповідає за валідацію файлів, обробку зображень і 3D-моделей.

Зазначені класи стосуються різних етапів додавання моделей. Деякі з цих класів відповідають за взаємодію з моделями, інші – за зв’язок з БД, автентифікацію користувача, обробку файлів.

Детальніше ознайомитися з кодом можна в додатку А.

3.3 Підключення MongoDB

Для збереження даних, файлів та процесу реєстрації/автентифікації використовується база даних MongoDB.

Опишемо процес налаштування з’єднання з БД.

Для взаємодії застосунку з базою даних використовується бібліотека Mongoose, яка дозволяє працювати з MongoDB за допомогою моделей та схем. Адреса підключення до БД зберігається у файлі оточення .env в каталозі /backend розроблюваної системи. На рисунку 3.1 продемонстровано загальний вигляд файлу налаштування підключення до БД.

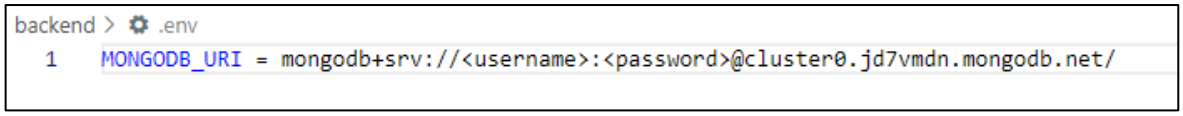


Рисунок 3.1 – Приклад файлу налаштування підключення до БД

У розробленому застосунку реалізовано механізми обробки помилок під час підключення до БД, відстеження подій з’єднання з БД та перевірки наявності з’єднання. Механізми зображено на рис. 3.2 – 3.4 відповідно.

```
mongoose.connection.on('error', (err) => {
  console.error('MongoDB connection error:', {
    error: err.message,
    name: err.name,
    stack: err.stack
  });
});
```

Рисунок 3.2 – Механізм обробки помилок

```
console.log('MongoDB connected successfully:', {
  host: conn.connection.host,
  port: conn.connection.port,
  name: conn.connection.name,
  state: conn.connection.readyState
});
```

Рисунок 3.3 – Механізм відстеження подій з’єднання

```
  await mongoose.connection.db.admin().ping();
  console.log('MongoDB ping successful');
```

Рисунок 3.4 – Механізм перевірки наявності з’єднання

Реалізовані механізми гарантують стабільне підключення до БД та стабільну роботу застосунку. Детальніше ознайомитися з кодом можна в додатку А.

3.4 Взаємодія користувача з системою

3.4.1. Навігація користувача

Після входу в систему користувач бачить головну сторінку застосунку та навігаційну панель (див. рис. 3.5 – 3.6 відповідно). На початковій сторінці знаходиться демонстраційний варіант візуалізатору моделей з передзавантаженою моделлю. Над ним знаходиться вікно з привітаннями та кнопкою «Почати» яка перекидає на сторінку реєстрації. На навігаційній

панелі знаходяться посилання на GitHub репозиторій проєкту, назва застосунку, кнопки переходу до головної сторінки, галереї та реєстрації.

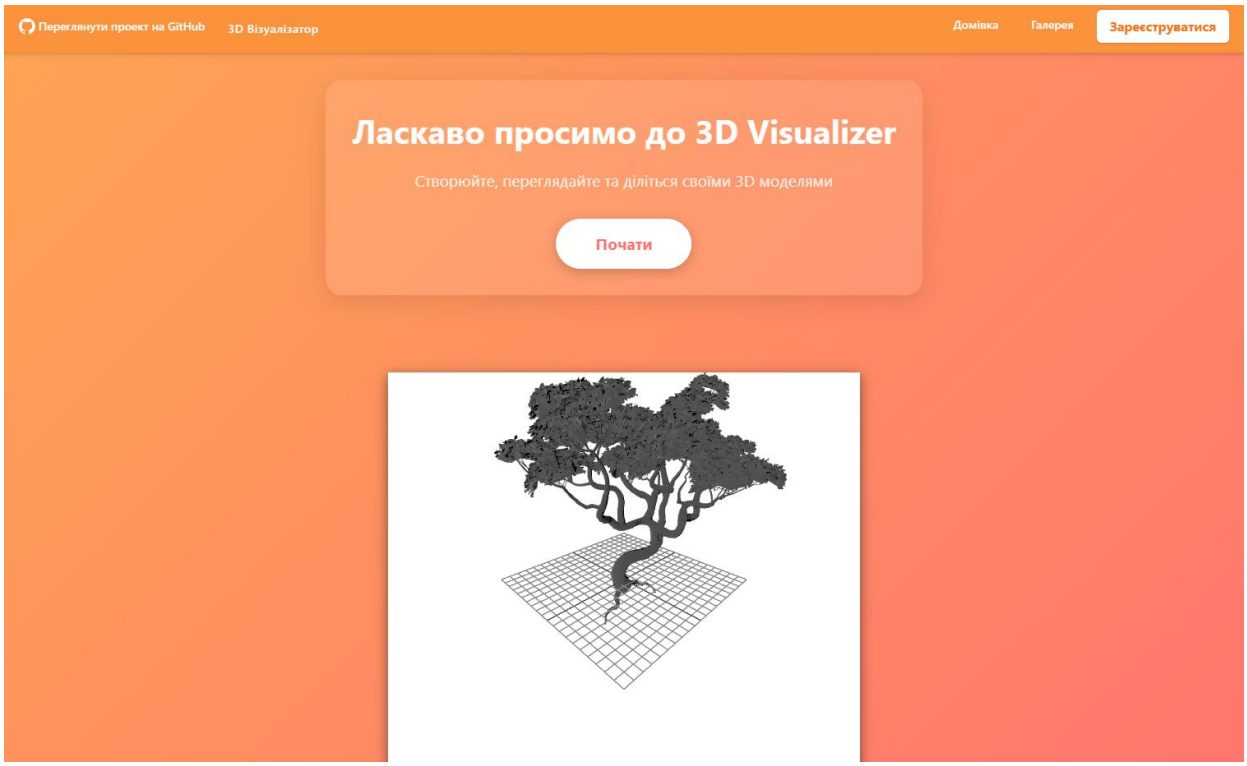


Рисунок 3.5 – Початкова сторінка застосунку



Рисунок 3.6 – Навігаційна панель застосунку

3.4.2 Реєстрація в системі

Перед тим як користувач зможе використовувати весь можливий функціонал додатку, користувачу необхідно провести реєстрацію в системі.

Для цього на початковій сторінці сайту користувач може натиснути на кнопку «Почати» або на навігаційній панелі кнопку «Зареєструватися» (рис. 3.7 – 3.8).

Після цього система переносить на сторінку форми реєстрації та відображає її (рис. 3.9).

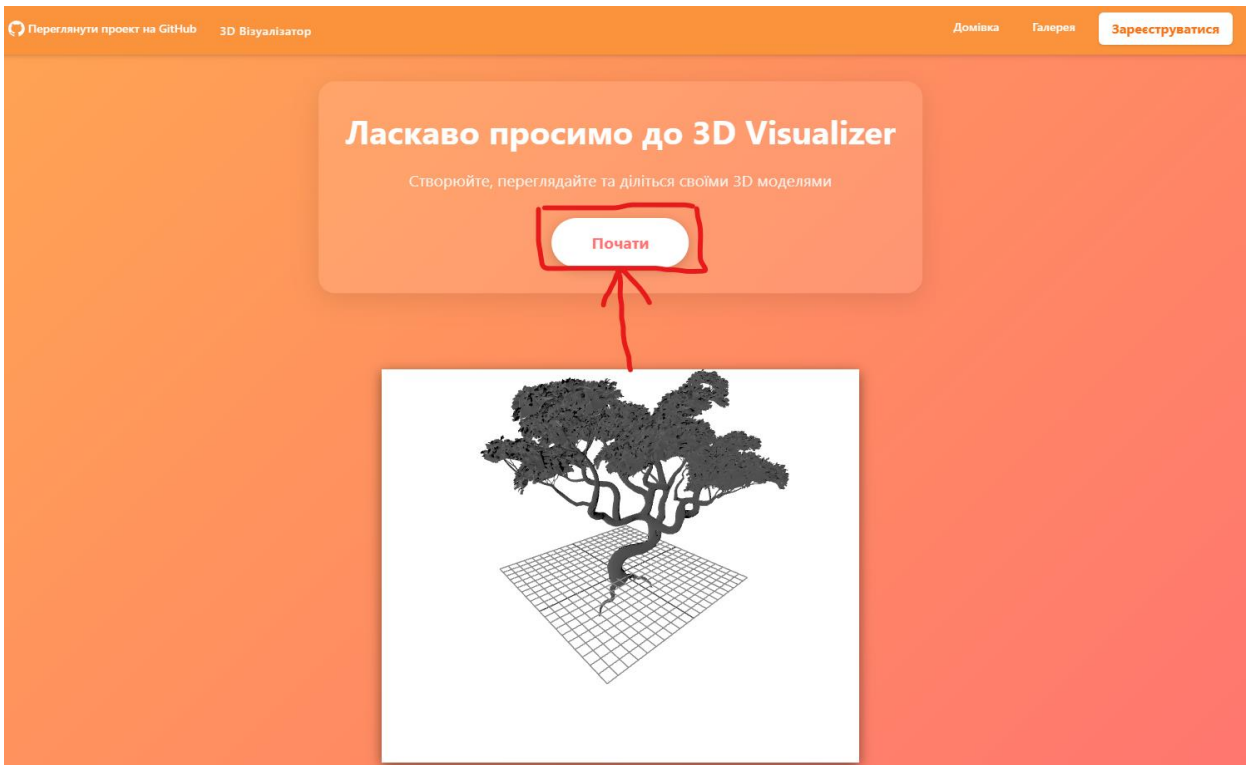


Рисунок 3.7 – Кнопка «Почати» на початковій сторінці



Рисунок 3.8 – Кнопка «Зареєструватися» на навігаційній панелі

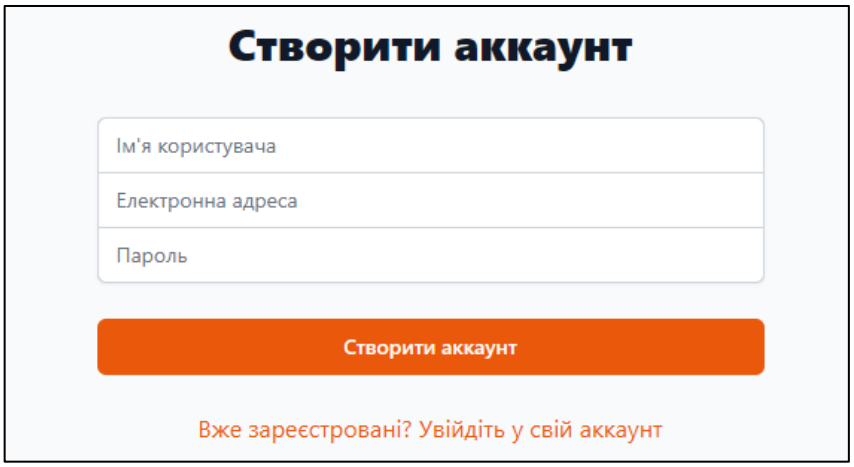


Рисунок 3.9 – Форма реєстрації

Після відображення форми, користувач має можливість заповнити всі поля відповідними даними, якщо користувач вводить невалідні дані при заповнені, то форма сповіщає про це у вигляді підказки під полем де введено некоректні дані (рис. 3.10 – 3.12).

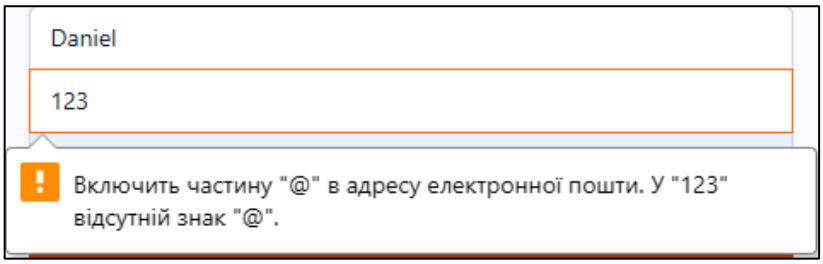


Рисунок 3.10 – Невалідний електронна пошта

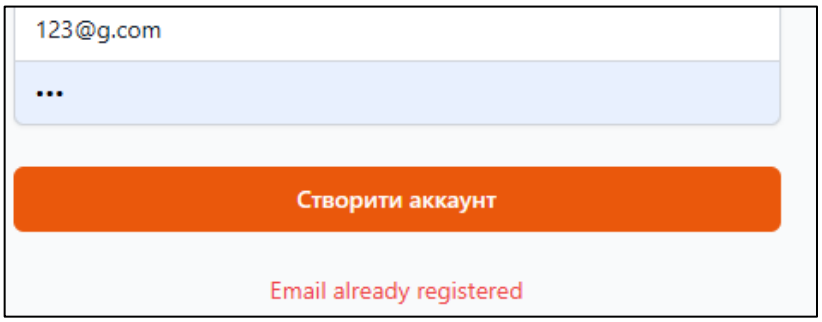


Рисунок 3.11 – Електронна пошта вже використовується

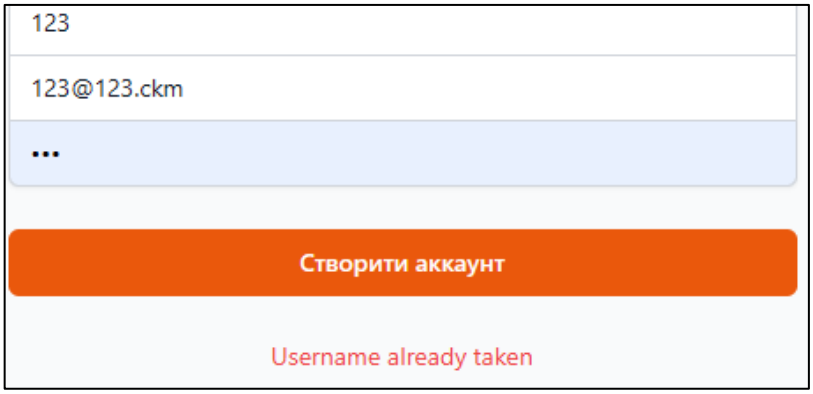


Рисунок 3.12 – Ім’я користувача вже використовується

Після успішної реєстрації система переадресовує на початкову сторінку та відображає навігаційну панель з додатковими пунктами меню.

3.4.3 Вхід та вихід до/з системи

Якщо користувач вже зареєстрований у формі реєстрації потрібно натиснути на «Вже зареєстровані? Увійдіть у свій аккаунт». Після цього відкривається форма входу (див. рис. 3.13).

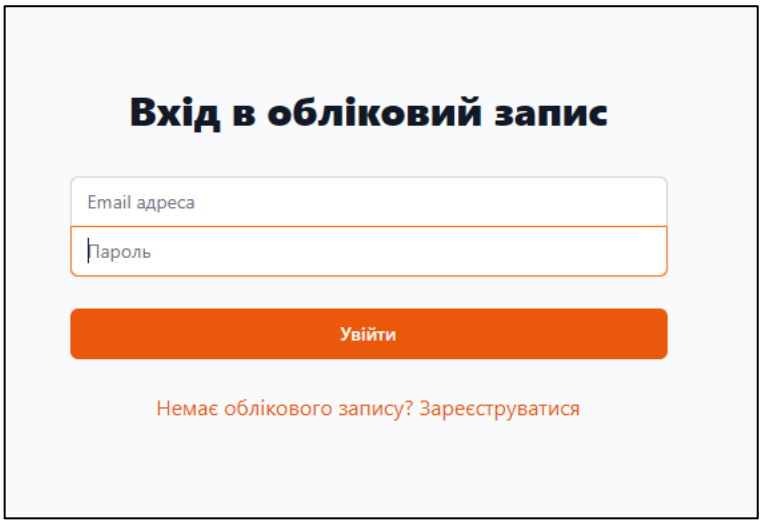


Рисунок 3.13 – Форма входу

Якщо при спробі увійти в аккаунт, користувач увів помилкові дані або немає відповіді сервера відображаються відповідні повідомлення (див. рис. 3.14 – 3.15).

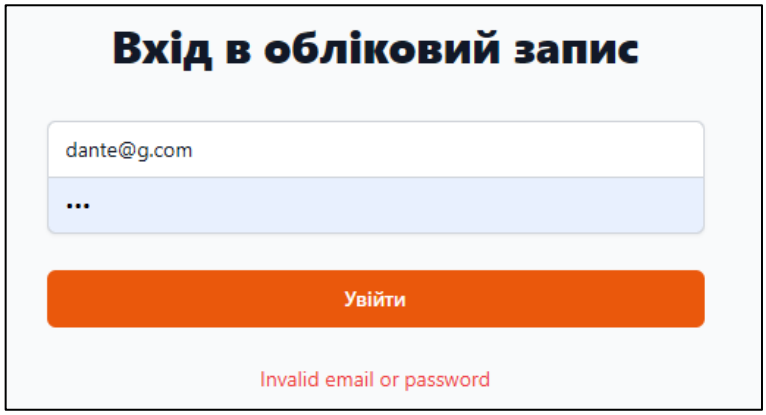


Рисунок 3.14 – Введено невірні дані

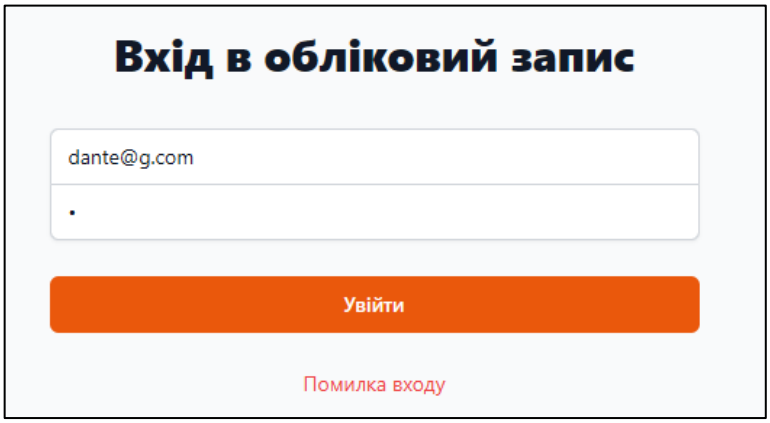


Рисунок 3.15 – Помилка входу (сервер не відповідає)

Після успішного входу, система відображає початкову сторінку та навігаційну панель на якій з’являються додаткові пункти такі як «Мій кабінет» та кнопка «Вийти».

Щоб вийти з аккаунту потрібно натиснути кнопку «Вийти» на навігаційній панелі (рис. 3.16).



Рисунок 3.16 – Кнопка вийти на навігаційній панелі

3.4.4 Завантаження моделей

Для того щоб завантажити модель спочатку на навігаційній панелі потрібно обрати пункт «Мій кабінет» (рис. 3.17).



Рисунок 3.17 – Пункт «Мій кабінет» на навігаційній панелі

Після цього система відображає інтерфейс кабінету. В власному кабінеті потрібно обрати пункт «Додати модель» (рис. 3.18). Система відображає форму яка дозволяє додати модель до системи (рис. 3.19).

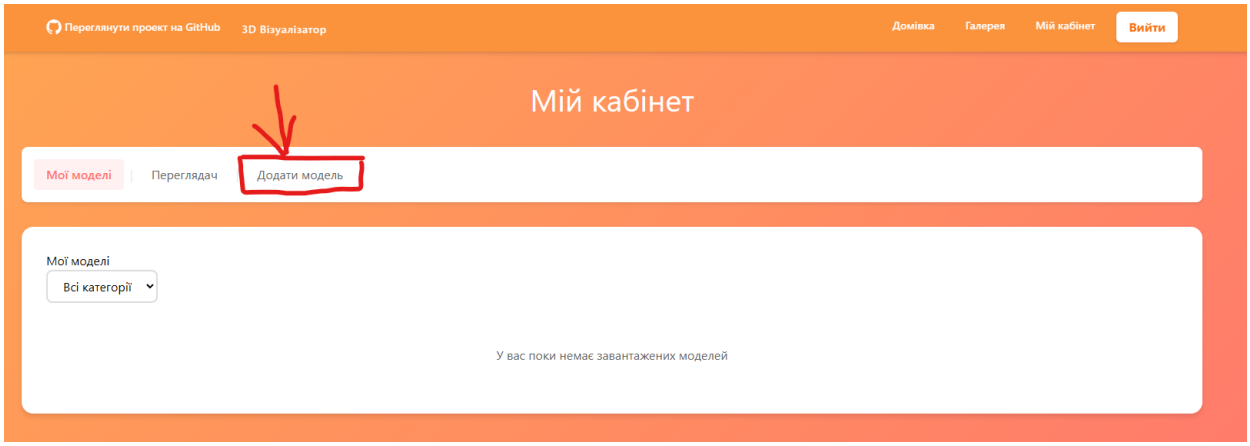


Рисунок 3.18 – Пункт «Додати модель» у власному кабінеті

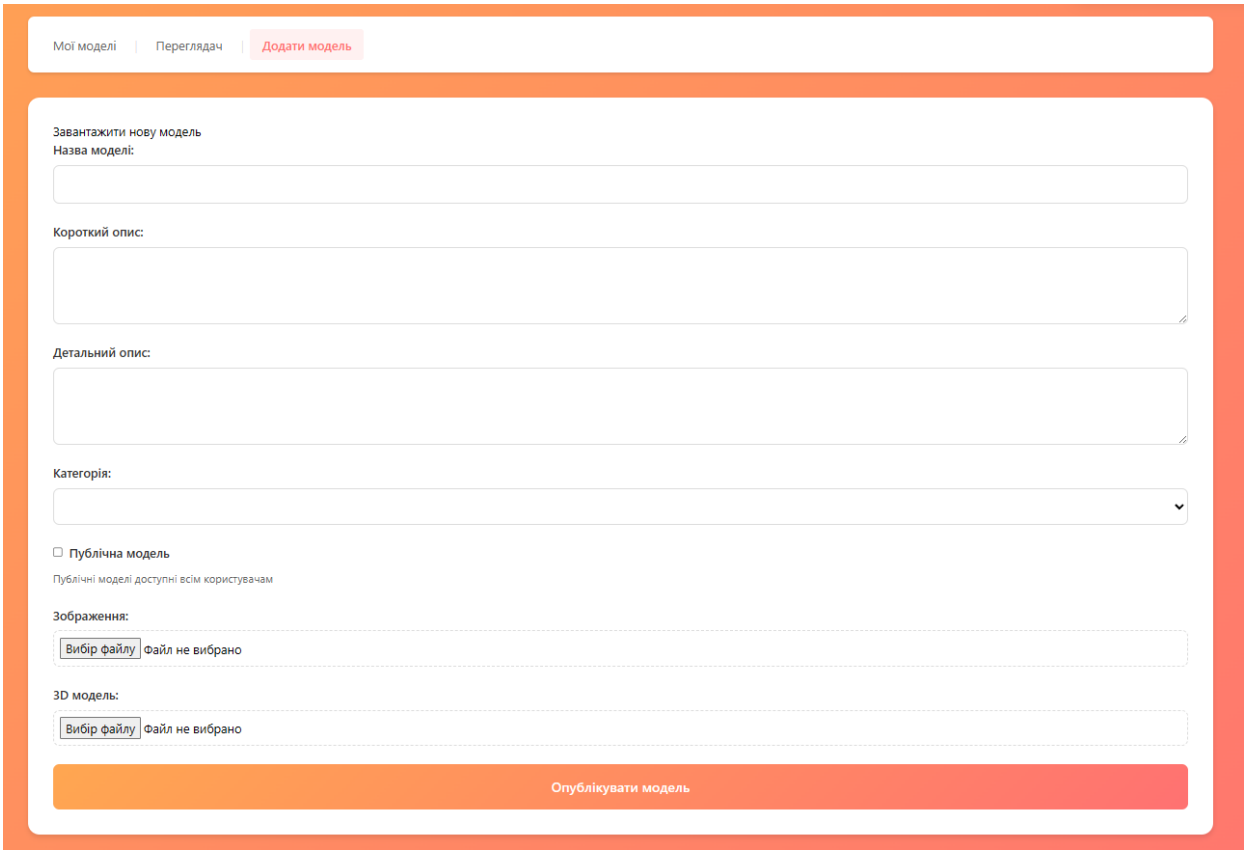


Рисунок 3.19 – Форма завантаження 3D-моделей

При спробі опублікувати модель без заповнення полів назви та описів виводиться відповідне повідомлення (рис. 3.20).

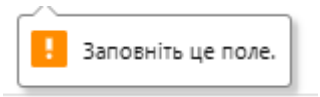


Рисунок 3.20 – Повідомлення про необхідність заповнення поля

При спробі опублікувати модель без вибору категорії виводиться відповідне повідомлення (рис 3.21).

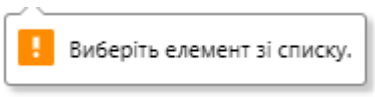


Рисунок 3.21 – Повідомлення про необхідність вибору елементу зі списку

При спробі опублікувати модель без вибору зображення або/та моделі виводиться відповідне повідомлення (рис. 3.22).

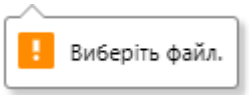


Рисунок 3.22 – Повідомлення про необхідність вибору файлу

При спробі обрати файл моделі більше за 50МБ виводиться повідомлення про помилку завантаження (рис. 3.23).

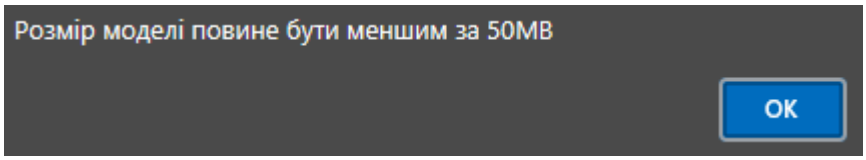


Рисунок 3.23 – Повідомлення про некоректний розмір файлу

При помилці зв’язку з сервером повертається повідомлення про помилку (рис. 3.24).

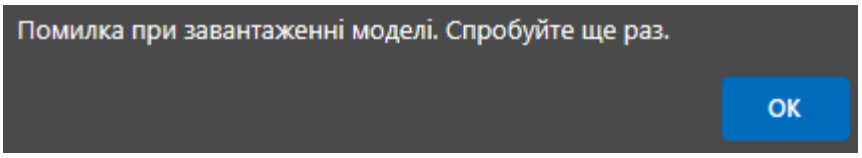


Рисунок 3.24 – Повідомлення про помилку завантаження

При успішному завантажені моделі виводить повідомлення про успіх (рис. 3.25).

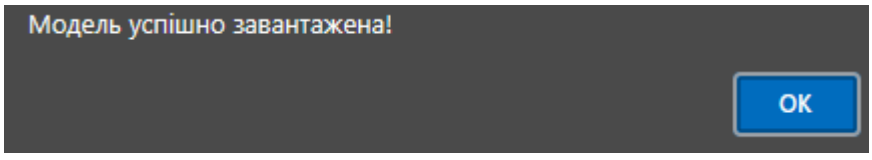


Рисунок 3.25 – Повідомлення про успіх

3.4.5 Керування моделями

Застосунок надає можливість редагувати дані моделей, переглядати та видаляти власні моделі. Щоб редагувати дані моделей зареєстрованому

користувачу потрібно у власному кабінеті відкрити пункт «Мої моделі», після чого можна подивитися усі завантажені моделі (рис. 3.26). Після цього потрібно натиснути кнопку редагувати на картці моделі після чого відкриється форма редагування даних про модель (рис. 3.27).

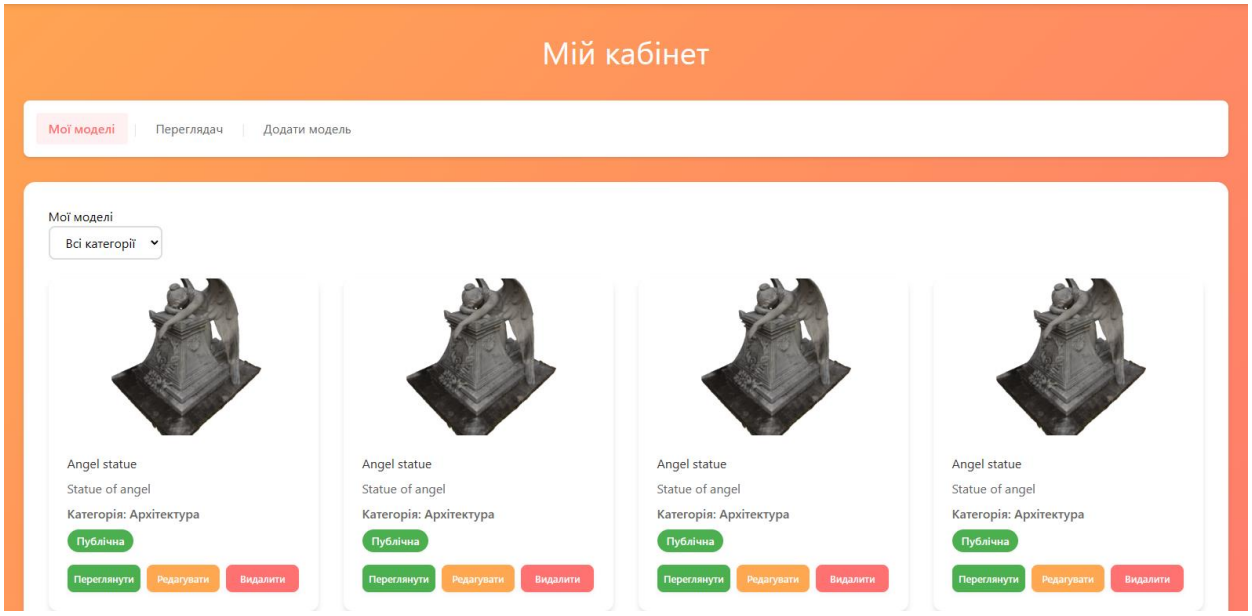


Рисунок 3.26 – Всі завантажені моделі у кабінеті користувача

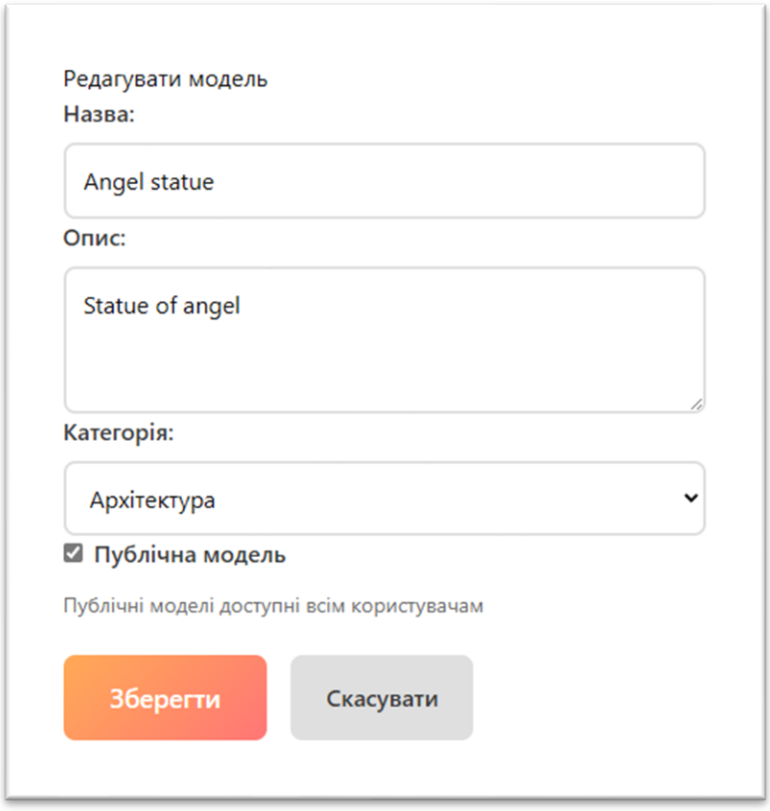


Рисунок 3.27 – Форма зміни даних моделі

При успішній зміні даних виводиться відповідне повідомлення (рис. 3.28).



Рисунок 3.28 – Повідомлення про успіх оновлення даних моделі

Щоб переглянути модель в кабінеті на картці моделі потрібно натиснути кнопку «Переглянути». Після цього відкривається вікно з передоглядом моделі (рис. 3.29).

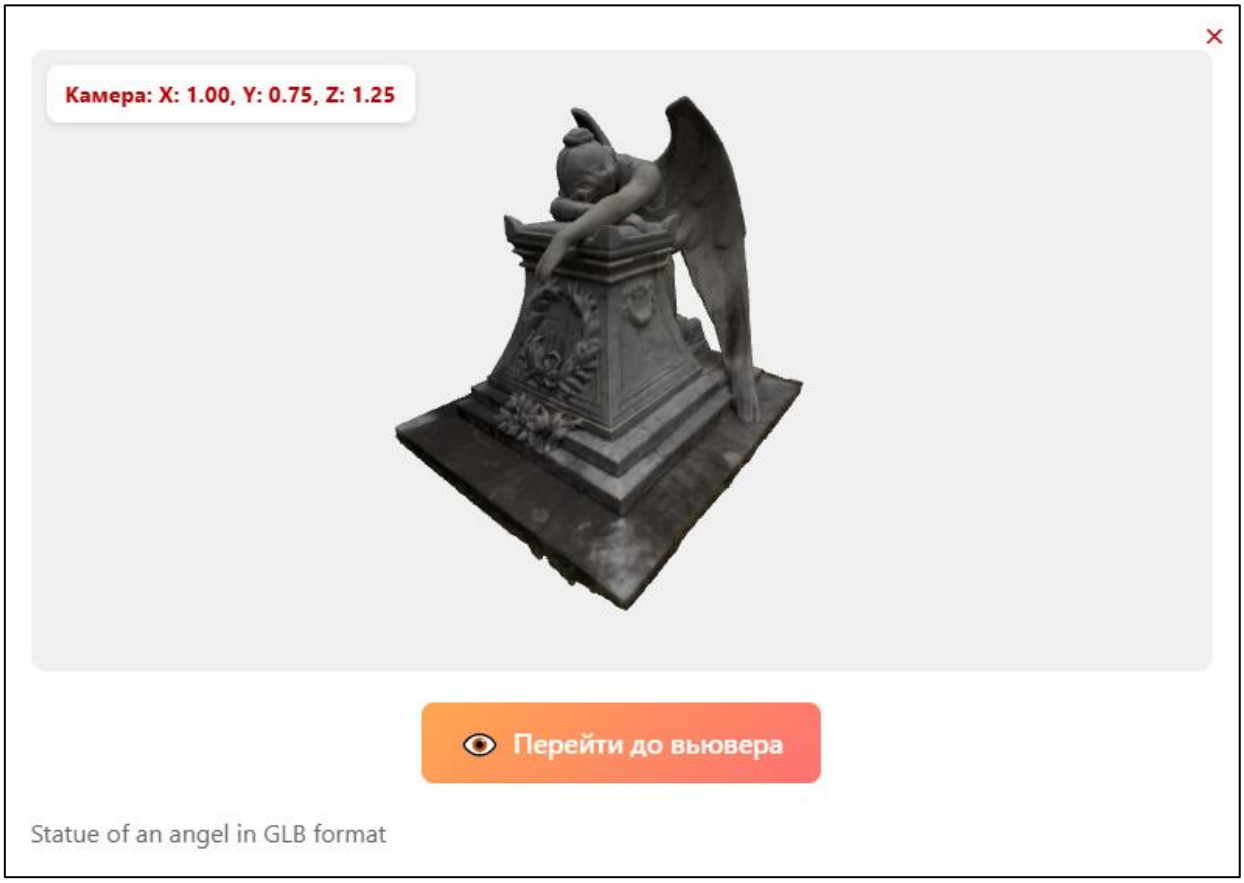


Рисунок 3.29 – Передогляд моделі

В цьому вікні є кнопка «Перейти до вьювера» яка перенесе до повноцінного переглядача з налаштуваннями моделі та сцени (рис. 3.30 – 3.31).

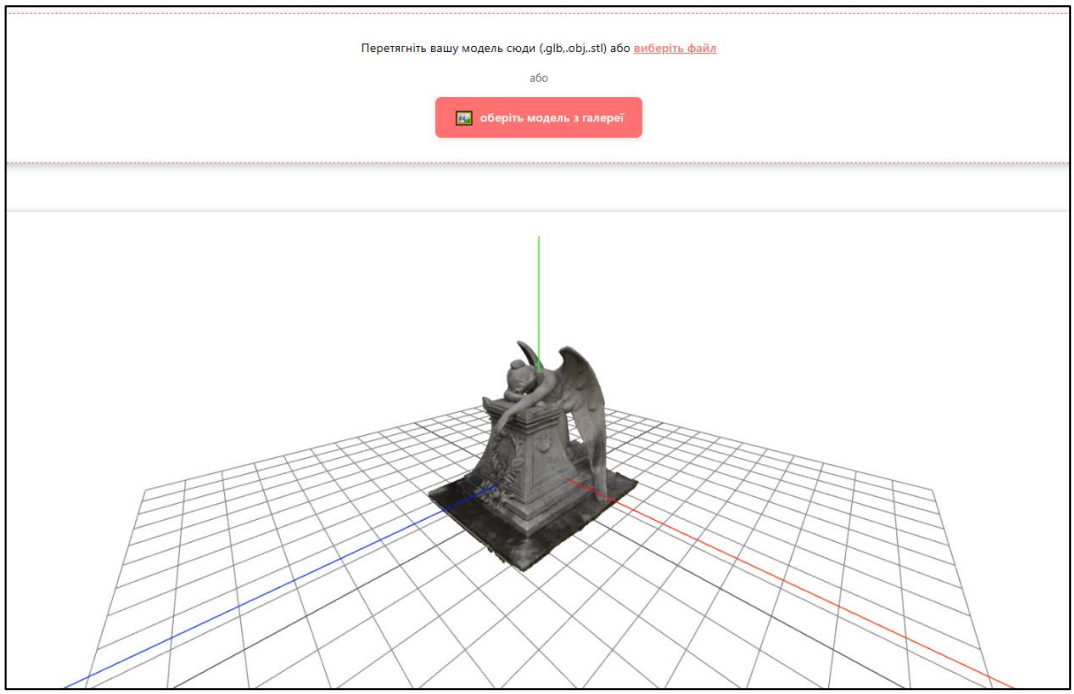


Рисунок 3.30 – Повноцінний переглядач моделей

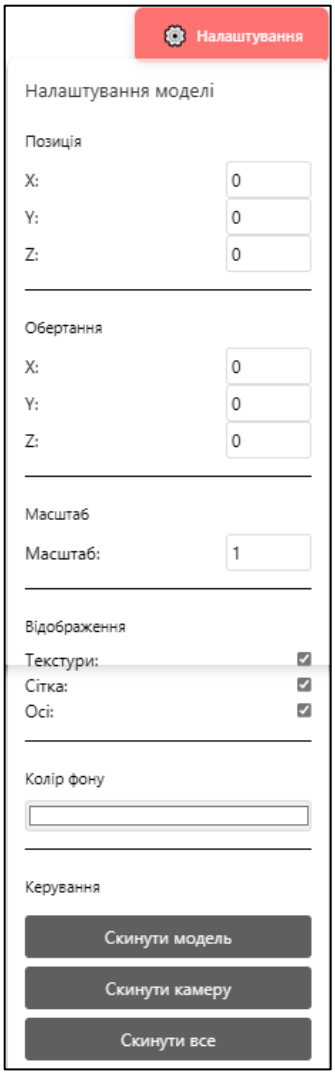


Рисунок 3.31 – Налаштування сцени та моделей

Щоб видалити модель в кабінеті на картці моделі потрібно натиснути кнопку «Видалити». Після цього з’являється повідомлення про підтвердження дії (рис. 3.32). Якщо виникла помилка виводиться відповідне повідомлення (рис. 3.33). При успішному видалені виводиться повідомлення про успіх (рис. 3.34).

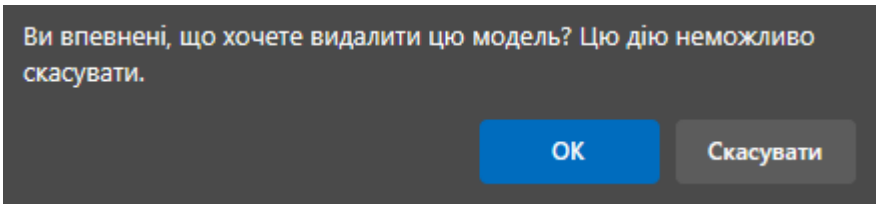


Рисунок 3.32 – Повідомлення з підтвердженням видалення

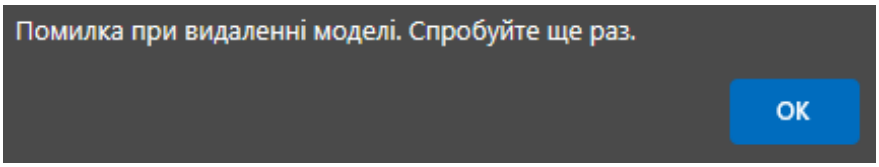


Рисунок 3.33 – Повідомлення про помилку видалення



Рисунок 3.34 – Повідомлення про успіх

Також користувач має змогу сортувати моделі за категоріями. На рис. 3.35 та 3.36 продемонстровано застосування фільтру за категоріями.

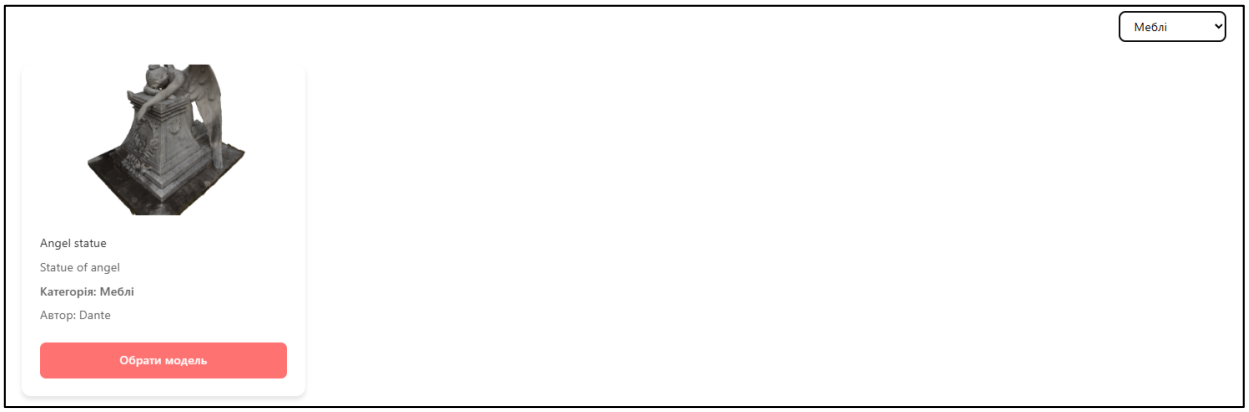


Рисунок 3.35 – Вигляд галереї моделі після застосування фільтру «Меблі»

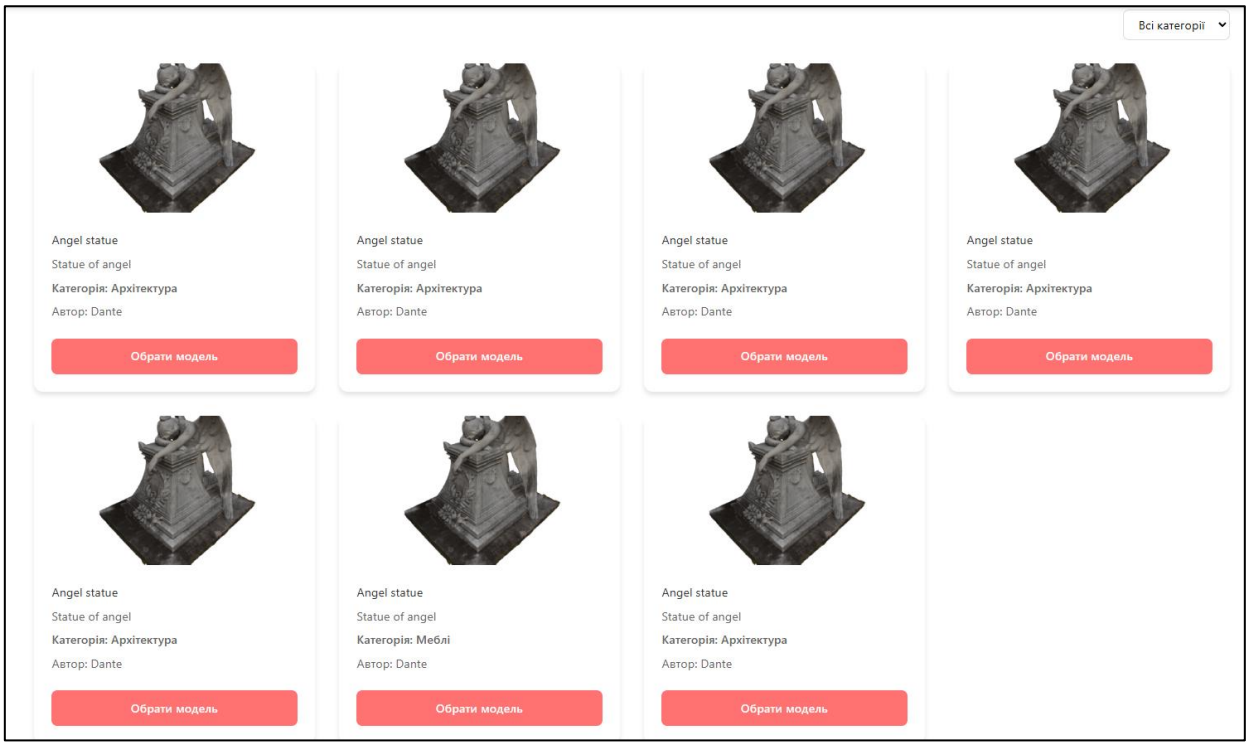


Рисунок 3.36 – Вигляд галереї моделей до застосування фільтру

Цей процес можливий тому, що додаток має вбудовану публічну галерею моделей. На сторінці галереї зареєстрований користувач може переглядати публічні моделі інших користувачів.

Детальніше ознайомитися з кодом можна в додатку А.

3.5 Висновки до третього розділу

У третьому розділі було детально описано розроблений вебдодаток (інструменти для його розробки, основні класи застосунку та підключення до бази даних MongoDB). Також було протестовано та продемонстровано можливості розробленого додатку.

ВИСНОВКИ

В даній роботі було розроблено вебдодаток «3D-візуалізатор геометричних об’єктів» з використанням таких інструментів як Node.js, Three.js, Express.js, Vue.js та бази даних MongoDB.

В першому розділі було розглянуто технології та бібліотеки для роботи з 3D-об’єктами такі як: WebGL, Babylon.js та Three.js та наведено їх порівняльну характеристику. Було проаналізовано популярні формати 3D-об’єктів та наведено порівняння вже існуючих аналогів розробленого вебзастосунку. Також було постановлено функціональні та нефункціональні вимоги до візуалізатору.

У другому розділі було описано проєктування вебдодатку. Було детально описано діаграму варіантів використання та присутні прецеденти, діаграми діяльності та послідовності для прецеденту «Додати модель» та діаграму розгортання вебзастосунку.

У третьому розділі було наведено опис використаних інструментів, реалізований функціонал додатку та налаштування підключення застосунку до бази даних MongoDB. Також було наведено приклади взаємодії користувача з розробленим додатком.

Використання наданих інструментів є доцільним для створення різноманітних подібних вебдодатків. Наприклад на сайтах інтернет магазинів для демонстрації 3D-моделі товару, сайтах з науковими статтями для візуалізації об’єктів дослідження конкретних наукових статей, тощо.

Результати даної роботи можуть бути використані при подальшій роботі у сфері розробки програмного забезпечення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Matsuda K., Lea R. WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL. Addison Wesley Professional, 2013. 552 p.

2. WebGL: 2D and 3D graphics for the web – Web APIs | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API (дата звернення: 05.03.2025).

3. Babylon.js Essentials. Packt Publishing, Limited, 2016. 200 p.

4. Babylon.js docs. *Babylon.js docs*. URL: https://doc.babylonjs.com/ (дата звернення: 10.03.2025).

5. Learn Three.js: Program 3D Animations and Visualizations for the Web with JavaScript and WebGL. Packt Publishing, Limited, 2023. 554 p.

6. Three.js – JavaScript 3D Library. *Three.js – JavaScript 3D Library*. URL: https://threejs.org/ (дата звернення: 13.03.2025).

7. Three js – Glossary of web terms | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Glossary/Three_js (дата звернення: 13.03.2025).

8. Conquering JavaScript: Three. Js. CRC Press LLC, 2023. 194 p.

9. Exploring 3D Model Formats: A Comprehensive Guide. *VividWorks*. URL: https://www.vividworks.com/blog/3d-model-formats-guide (дата звернення: 20.03.2025).

10. Sketchfab – The best 3D viewer on the web. *Sketchfab*. URL: https://sketchfab.com/ (дата звернення: 21.03.2025).

11. three.js editor. *Three.js – JavaScript 3D Library*. URL: https://threejs.org/editor/ (дата звернення: 21.03.2025).

12. Babylon.js Sandbox – View glTF, glb, obj and babylon files. *Babylon.js*. URL: https://sandbox.babylonjs.com (дата звернення: 21.03.2025).

13. Unhelkar B. Software Engineering with UML. Boca Raton : Auerbach Publications, 2020. 426 p.

14. Bennett S., McRobb S., Farmer R. Object-oriented Systems Analysis and Design Using UML : 3rd ed. McGraw Hill Higher Education, 2007. 624 p.

15. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Pearson Education, Limited, 2018. 208 p.

16. UML. *KDE Documentation*. URL: https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-basics.html (дата звернення: 30.03.2025).

17. Casciaro M. Node. js Design Patterns. Packt Publishing, Limited, 2014. 454 p.

18. Node.js – Run JavaScript Everywhere. *Node.js – Run JavaScript Everywhere*. URL: https://nodejs.org/en (дата звернення: 05.04.2025).

19. Fullstack Vue: The Complete Guide to Vue.js. CreateSpace Independent Publishing Platform, 2018. 442 p.

20. Vue.js. *Vue.js – The Progressive JavaScript Framework | Vue.js*. URL: https://vuejs.org/ (дата звернення: 07.04.2025).

21. Mardan A. Express.js Deep API Reference. Apress L. P., 2014. 160 p.

22. Augarten B., Kuo M., Lin E. Express. Js Blueprints. Packt Publishing, Limited, 2015. 198 p.

23. Express – Node.js web application framework. *Express – Node.js web application framework*. URL: https://expressjs.com/ (дата звернення: 15.04.2025).

ДОДАТОК А

Посилання на Git

Програмна реалізація розміщена у відкритому доступі за посиланням на GitHub: https://github.com/grish0dante/Diplom-Project.