НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютеризованих систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Програмні засоби для рендерингу об'ємних моделей

Виконав студент IV курсу, групи ІП-81в
(шифр групи)
Суприган Артем Сергійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ст. викладач Головченко М. М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації ст. викладач Головченко М. М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент, к.т.н., Шимкович В. М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення комп'ютеризованих систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Супригану Артему Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Програмні засоби для рендерингу об'ємних моделей
керівник проєкту Головченко Максим Миколайович, ст. викладач
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «02» червня 2023 р. №2160-с

2. Термін подання студентом проєкту « 19 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.
3) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.
4) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема бази даних

3) Схема структурна компонентів програмного забезпечення

6. Консультанти розділів проєкту

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

7. Дата видачі завдання «10» березня 2023 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв’язання задачі	17.03.2023	
3	Постановка та формалізація задачі	01.04.2023	
4	Розробка інформаційного забезпечення	04.04.2023	
5	Алгоритмізація задачі	10.04.2023	
6	Обґрунтування вибору використаних технічних засобів	13.04.2023	
7	Розробка програмного забезпечення	01.05.2023	
8	Налагодження програми	07.05.2023	
9	Виконання графічних документів	14.05.2023	
10	Оформлення пояснювальної записки	18.05.2023	
11	Подання ДП на попередній захист	05.06.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	16.06.2023	

Студент _____ Артем СУПРИГАН
(підпис) (ініціали, прізвище)

Керівник _____ Максим ГОЛОВЧЕНКО
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 53 таблиці, 45 рисунків та 13 джерел – загалом 85 сторінок.

Дипломний проєкт присвячений розробці програмних засобів для рендерингу об’ємних моделей.

Метою розробки є надання можливості виконання рендерингу об’ємних моделей при обмежених технічних ресурсах за рахунок виконання процесу з будь-якого пристрою з доступом до всесвітньої мережі Інтернет.

У розділі першому описано предметну область, проаналізовано відомі технічні рішення, проведено порівняння з аналогом, створено діаграму варіантів використання, сформовано функціональні та нефункціональні вимоги до програмного забезпечення.

Розділ другий присвячений створенню BPMN діаграми та опису основного бізнес-процесу, обґрунтуванню архітектурних рішень, побудові схеми взаємодії між сервісами, опису використаних пакетів, детальному розбору архітектури та функціоналу кожного з мікросервісів, опису принципів, за якими було побудовано API кожного з сервісів, опису використаних утиліт, опису баз даних і таблиць кожного сервісу, побудові схеми баз даних.

У розділі третьому описано процеси і методи тестування, наведено список утиліт, що використовуються для аналізу якості коду, та наведено контрольний приклад використання програмного забезпечення.

Розділ четвертий присвячений опису процесів безперервного розгортання та безперервного супроводу програмного забезпечення.

КЛЮЧОВІ СЛОВА: ПРОГРАМНІ ЗАСОБИ, РЕНДЕРИНГ, ОБ’ЄМНА МОДЕЛЬ, OCTREE, АЛГОРИТМ МОЛЛЕРА – ТРУМБОРА, C#, .NET, ASP.NET CORE, БАЗА ДАНИХ, МІКРОСЕРВІСНА АРХІТЕКТУРА, ТРИЯРУСНА АРХІТЕКТУРА, MICROSOFT SQL SERVER, REDIS, RABBITMQ, DOCKER

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 53 tables, 45 figures and 13 sources – in total 85 pages.

The diploma project is devoted to the development of software tools for rendering volumetric models.

The purpose of the development is to provide the possibility of rendering volumetric models with limited technical resource by performing the process from any device with to the Internet.

In the first section, the subject area is described, known technical solutions are analyzed, a comparison with an analogue is made, a use case diagram is created, and functional and non-functional requirements for the software are formed.

The second section is devoted to the creation of a BPMN diagram and a description of the main business process, justification of architectural solutions, construction of an interaction scheme between services, a description of the packages used, a detailed analysis of the architecture and functionality of each of the microservices, a description of the principles by which the API of each of the services was developed, a description of the used utilities, description of databases and tables of each service, construction of a database scheme.

In the third section, the testing processes and methods are described, a list of utilities used for code quality analysis is given, and a control example of the use of the software is given.

The fourth section is devoted to the description of the processes of continuous deployment and continuous integration of the software.

KEYWORDS: SOFTWARE TOOLS, RENDERING, VOLUMETRIC MODEL, OCTREE, MÖLLER–TRUMBORE INTERSECTION ALGORITHM, C#, .NET, ASP.NET CORE, DATABASE, MICROSERVICE ARCHITECTURE, THREE-TIER ARCHITECTURE, MICROSOFT SQL SERVER, REDIS, RABBITMQ, DOCKER

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-8132.045440.01.91	Технічне завдання	15	
3	А4	КПІ.ІП-8132.045440.02.81	Пояснювальна записка	85	
4	А4	КПІ.ІП-8132.045440.03.12	Текст програми	34	
5	А4	КПІ.ІП-8132.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІП-8132.045440.05.34	Керівництво користувача	7	
7	А3	КПІ.ІП-8132.045440.06.99	Схема структурна варіантів	1	
			використань		
8	А3	КПІ.ІП-8132.045440.07.99	Схема бази даних	1	
9	А3	КПІ.ІП-8132.045440.08.99	Схема структурна компонентів	1	
			програмного забезпечення		

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІП-8132.045440.00.90		
Розроб.		Суприган А.С.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Головченко М.М					
Н.контр.		Головченко М.М				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-81в	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

Програмні засоби для рендерингу об’ємних моделей

Технічне завдання

КПІ.ІП-8132.045440.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Максим ГОЛОВЧЕНКО

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Артем СУПРИГАН

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки:

Програмні засоби для рендерингу об’ємних моделей

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмних засобів для рендерингу об’ємних моделей, котрі використовуються для користувачів, що працюють у галузі графічного моделювання.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки програмних засобів для рендерингу об’ємних моделей є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для:

– користувачів, що мають необхідність у рендерингу об’ємних моделей, проте не мають відповідних програмних засобів;

– користувачів, що не мають технічних засобів, які здатні швидко виконувати велику кількість обрахунків;

– користувачів, що хочуть мати можливість рендерити свої об’ємні моделі будь-де, будь-коли та з будь-якого девайсу;

– користувачів, що мають потребу виконувати одночасний рендеринг декількох об’ємних моделей.

Метою розробки є:

– надання можливості виконання рендерингу об’ємних моделей при обмежених технічних ресурсах за рахунок виконання процесу з будь-якого пристрою з доступом до всесвітньої мережі Інтернет.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Для незареєстрованого користувача:

4.1.1.1 реєстрація за електронною поштою та паролем через REST запит:

4.1.1.1.1 запит повинен мати метод POST;

4.1.1.1.2 електронна пошта має відповідати наступним вимогам:

- має містити символ «@»;
- символ «@» не має бути останнім символом у пошті;
- символ «@» не має бути першим символом у пошті;

4.1.1.1.3 пароль має відповідати наступним вимогам:

- має містити хоча б 10 символів;
- має містити не більше 100 символів;
- має містити хоча б 5 унікальних символів;
- має містити хоча б одну велику літеру;
- має містити хоча б одну маленьку літеру;
- має містити хоча б одну цифру;
- має містити хоча б один спеціальний символ;
- не має містити пробілів;

4.1.1.1.4 якщо пароль чи електронна пошта не пройшли валідацію, то видати у відповіді причину непроходження валідації;

4.1.1.1.5 якщо користувач із введеною електронною поштою уже існує, то видати у відповіді відповідне повідомлення про помилку;

4.1.1.1.6 якщо реєстрація успішна, то видати у відповіді пару Refresh Token та Access Token і профіль користувача.

4.1.1.2 авторизація за електронною поштою та паролем через REST запит:

4.1.1.2.1 запит повинен мати метод GET;

4.1.1.2.2 якщо користувач з введеною електронною поштою відсутній у базі даних, або його пароль не співпадає з введеним, то видати у відповіді повідомлення про те, що електронна пошта або пароль є невірними;

4.1.1.2.3 якщо авторизація успішна, то видати у відповіді пару Refresh Token та Access Token і профіль користувача.

4.1.2 Для зареєстрованого користувача:

4.1.2.1 перегляд своїх замовлень на рендеринг через REST запит:

4.1.2.1.1 запит повинен мати метод GET;

4.1.2.1.2 запит повинен містити заголовок Authorization, де повинен бути вказаний Access Token:

– якщо Access Token не є валідним, то видати відповідь із кодом стану HTTP 401 Unauthorized;

– якщо Access Token є валідним, то видати у відповіді список замовлень користувача;

4.1.2.2 створення нового замовлення на рендеринг:

4.1.2.2.1 запит повинен мати метод POST;

4.1.2.2.2 тіло запиту повинно мати наступний опис замовлення на рендеринг:

4.1.2.2.2.1 назва;

4.1.2.2.2.2 налаштування сцени:

– позиція камери;

– позиція джерела світла;

– кут огляду;

4.1.2.2.2.3 об'ємна модель;

4.1.2.2.3 якщо тіло запиту не пройшло валідацію, то у відповіді видати список помилок, які необхідно виправити;

4.1.2.2.4 запит повинен містити заголовок Autorization, де повинен бути вказаний Access Token:

– якщо Access Token не є валідним, то видати відповідь із кодом стану HTTP 401 Unauthorized;

– якщо Access Token є валідним, то видати у відповіді список замовлень користувача.

4.1.2.3 відстежування прогресу виконання замовлення.

4.1.3 Додаткові вимоги:

– розширюваність.

4.2 Вимоги до надійності

4.2.1 передбачити контроль введення інформації;

4.2.2 передбачити захист від некоректних дій користувача;

4.2.3 забезпечити цілісність інформації в базі даних;

4.2.4 забезпечити надійність зберігання інформації в базі даних;

4.2.5 забезпечити захист від неправомірного доступу до захищених ресурсів.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

Не висуваються.

4.3.2 Обслуговування

Не висуваються.

4.3.3 Обслуговування

Не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп'ютерах.

4.4.2 Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i7;
– об’єм ОЗП: 8 Гб;
– підключення до мережі Інтернет зі швидкістю від 10 мегабіт.

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows`XP, Windows NT і т.д.) або Unix.

4.5.2 Вимоги до вхідних даних

4.5.2.1 Об’ємна модель повинна бути представлена у вигляді файлу формату .obj, який повинен мати наступні дані:

– список вершин;
– список нормалей;
– визначення поверхонь.

4.5.2.2 Всі інші дані повинні бути представлені у файловому форматі JSON.

4.5.3 Вимоги до вихідних даних

4.5.3.1 Всі відповіді мають бути представлені у файловому форматі JSON;

4.5.3.2 Результат рендерингу об’ємної моделі повинен бути представлений посиланням на файл зображення у форматі .jpeg.

4.5.4 Вимоги до мови програмування

4.5.4.1 Розробку виконати на мові програмування C#.

4.5.5 Вимоги до середовища розробки

4.5.5.1 Розробку виконати на платформі .NET. Для розробки серверної частини використати фреймворк ASP.NET Core. Для розробки клієнтської частини використати фреймворк Avalonia.

4.5.6 Вимоги до представлення вихідних кодів

4.5.6.1 Вихідний код програми має бути представлений у вигляді документа «Текст програми».

4.5.7 Вимоги до маркування та пакування

4.5.7.1 Вимоги до маркування та пакування не висуваються.

4.5.8 Вимоги до транспортування та зберігання

4.5.8.1 Вимоги до транспортування та зберігання не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;

– технічне завдання;

– текст програми;

– програма та методика тестування;

– керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;

– схема бази даних;

– схема структурна компонентів програмного забезпечення.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення рекомендованої літератури	10.03	
2.	Аналіз існуючих методів розв’язання задачі	17.03	Вимоги до програмного забезпечення
3.	Постановка та формалізація задачі	01.04	Специфікації програмного забезпечення
4.	Розробка інформаційного забезпечення	04.04	Схема структурна програмного забезпечення та специфікація компонентів
5.	Алгоритмізація задачі	10.04	Схема бізнес процесу створення замовлення на рендеринг
6.	Обґрунтування вибору використаних технічних засобів	13.04	Тести, результати тестування
7.	Розробка програмного забезпечення	01.05	Тексти програмного забезпечення
8.	Налагодження програми	07.05	Тести, результати тестування
9.	Виконання графічних документів	14.05	Графічний матеріал проекту
10.	Оформлення пояснювальної записки	18.05	Пояснювальна записка

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: Програмні засоби для рендерингу об’ємних моделей

КПІ.ІП-8132.045440.02.81

Київ – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
API	–	Application programming interface, прикладний програмний Інтерфейс.
SDK	–	Software development kit.
IT	–	Information technology, інформаційні технології.
DB	–	Database, база даних.
CTS	–	Common Type System, система спільних типів.
CLR	–	Common Language Runtime, загальномовне середовище виконання.
IL	–	Intermediate Language, проміжна мова.
GC	–	Garbage Collector, збирач сміття.
DI	–	Dependency Injection, ін’єкція залежностей.
СУБД	–	Система управління базами даних.
BPMN	–	Business Process Model and Notation, нотація і модель бізнес-процесів.
SSE	–	Server-sent Events, події, що надіслані сервером.
ORM	–	Object-Relation Mapping, об’єктно-реляційне відображення.
JWT	–	JSON Web Token, токен у форматі файлу JSON, який використовується для аутентифікації користувача у системі.
Бітмапа	–	Структура даних, що являє собою двовимірний масив пікселів.
DAL	–	Data Access Layer, шар доступу до даних.
BLL	–	Business Logic Layer, шар бізнес-логіки.
PL	–	Presentation Layer, шар представлення.
VCS	–	Version Control System, система контролю версій.

ВСТУП

У сучасному світі все частіше можна зустріти використання технологій тривимірного моделювання для вирішення багатьох у найрізноманітніших сферах людської діяльності: від звичних комп'ютерних ігор та кінематографу, до будівництва та медицини.

Проте загальнодоступними для використання ці технології були не одразу. Дуже тісно це пов'язано з наявними на той час технологіями моделювання та розповсюдженістю персональних комп'ютерів, на яких, власне, створюються та обробляються об'ємних моделей.

На початку своє історії у середині XX сторіччя, засоби моделювання були доступні лише дуже вузькому колу спеціалістів з інженерії та автоматизації у галузях аерокосмічної та автомобільної промисловості. Працювали такі засоби саме з математичними моделями.

Для використання об'ємних моделей недостатньо лише їх створити. Необхідно також мати можливість представити створену модель для кінцевого користувача. Одним із способів представлення моделей є рендеринг.

Даний процес дозволяє представити модель у вигляді зображення. Алгоритми рендерингу відрізняються один від одного швидкістю та якістю отриманого результату.

Моделі мають тенденцію ставати деталізованішими, алгоритми – ресурсовимогловішими, а набір вимог до кінцевого результату – все більшим. Для отримання результату, що відповідає сформульованому набору вимог, необхідні досить потужні обчислювальні засоби, оскільки алгоритми рендерингу є вкрай ресурсовитратними. Навіть найдорожчі персональні комп'ютери можуть досить швидко перестати відповідати потребам, а заміна комплектуючих є особливо дороговартісною в наш час у зв'язку зі світовим дефіцитом напівпровідників, які необхідні для виготовлення комп'ютерних комплектуючих.

Для подолання цієї проблеми було вирішено створити програмні засоби, що надаватимуть можливість виконати рендеринг з будь-якого пристрою з доступом до всесвітньої мережі Інтернет.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Об’ємною моделлю називають математичне представлення якого-небудь об’єкта. Об’єктом може бути як абстрактна фігура, так і що-небудь з реального світу.

Математичним представленням моделі є множина точок, що поєднуються між собою гранями, тим самим утворюючи пласкі багатокутники, які й визначають зовнішній вигляд моделі. Такі багатокутники також називають полігонами. Кількість полігонів визначає деталізованість моделі. Чим їх більше – тим деталізованішою є модель.

Рендерингом у комп’ютерній графіці називають процес перетворення моделі у зображення. Результуюче зображення різниться, та може бути як фотореалістичним, так і нефотореалістичним.

Нефотореалістичні зображення характеризуються тим, що вони виконані у художньому стилі та по суті являють собою вхідну модель, яка була модифікована нанесенням матеріалу. Вихідна геометрія також ідентична вхідній. [1]

На рисунку 1.1 зображено приклад нефотореалістичного рендерингу.

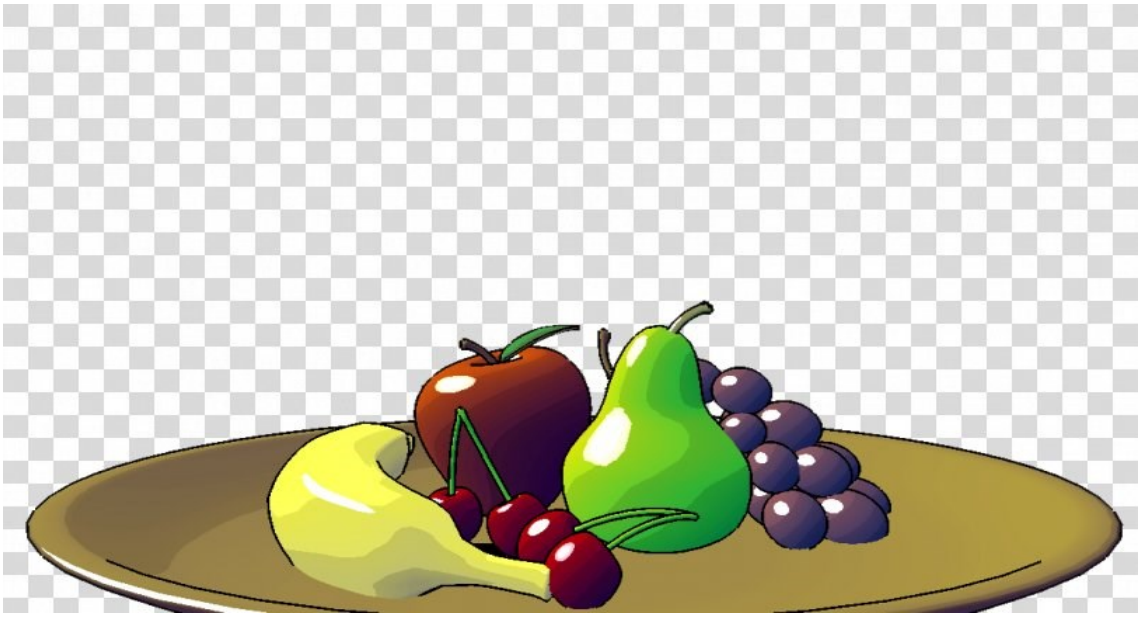


Рисунок 1.1 – Приклад нефотореалістичного рендерингу

На відміну від нефотореалістичного зображення, фотореалістичне дуже тяжко відрізнити від реального, особливо з використанням сучасних технологій рендерингу. Зазвичай моделі для такого типу рендерингу є вкрай деталізованими та потребують грамотного виставлення світла і реалістичних текстур і шейдерів. Для максимального ефекту фотореалістичності, текстури не повинні бути суцільними та ідеальними, тобто мати деякі недоліки. Розширення вихідного зображення має бути достатньо великим, щоб можна було розрізнити деталі. Також важливе місце у фотореалістичному рендерингу займає пост-продакшн. Встановлені на цьому етапі декорації та задній фон роблять зображення ще більш фотореалістичним. [2]

На рисунку 1.2 зображено приклад фотореалістичного рендерингу.

Рисунок 1.2 – Приклад фотореалістичного рендерингу

Рендеринг може відбуватися з використанням певного ряду особливостей. Деякі з них притаманні лише деякій множині алгоритмів рендерингу, або взагалі одному. До найбільш розповсюджених особливостей відносяться:

– шейдинг. Відповідає за зміну кольору поверхні в залежності від куту падіння, яскравості та відстані від джерела світла;

– тіні. Відповідає за створення ефекту затінення від точкового джерела світла;

– м’які тіні. Відповідає за створення ефекту затінення різного ступеня від частково перекритих джерел світла;

– накладення текстур. Відповідає за накладення матеріалів на поверхні моделі за допомогою мапи текстур. Мапою текстур називають зображення, яке лягає на полігон моделі. Також існує варіант накладення декількох текстур на одну поверхню. На рисунку 1.3 зображено приклад накладання текстур;

Рисунок 1.3 – Приклад накладання текстур

– фогінг. Відповідає за зменшення видимості на контрастності об’єктів в залежності від їх віддаленості. Досить часто цю особливість можна зустріти у відеоіграх, де використовується рендеринг у реальному часі. Зазвичай віддалені від гравця об’єкти роблять низькодеталізованими, тобто такими, що досить легко рендеряться, та покривають туманом, щоб вони не створювали візуального дискомфорту. На рисунку 1.4 зображено приклад використання фогінгу у відеогрі;

Рисунок 1.4 – Приклад використання фогінгу у відеогрі

– рельєфне текстурування. Відповідає за створення ефекту нерівності поверхні. На рисунках 1.5 та 1.6 зображено приклад без використання та з використанням рельєфного текстурування відповідно;

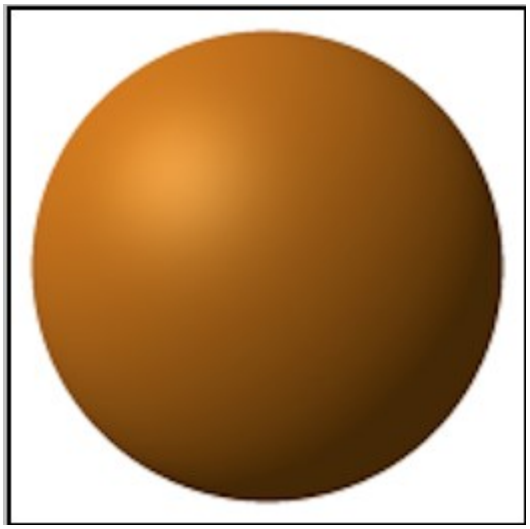


Рисунок 1.5 – Приклад рендерингу без використання рельєфного текстурування

Рисунок 1.6 – Приклад моделі з використанням рельєфного текстурування

– відображення. Відповідає за відбиття світла від дзеркальних та світлих поверхонь. Тип відображення може різнитися в залежності від поверхні, від якої відбивається світло. На рисунку 1.7 зображено приклад дзеркального відображення;

Рисунок 1.7 – Приклад дзеркального відображення

– прозорість поверхонь. Відповідає за ефект проникнення світла через поверхню;

– глибина різко зображуваного простору. Відповідає за розмиття об’єктів, які знаходяться на певній відстані від об’єкта, що у фокусі. На рисунку 1.8 зображено приклад використання глибини різко зображуваного простору.

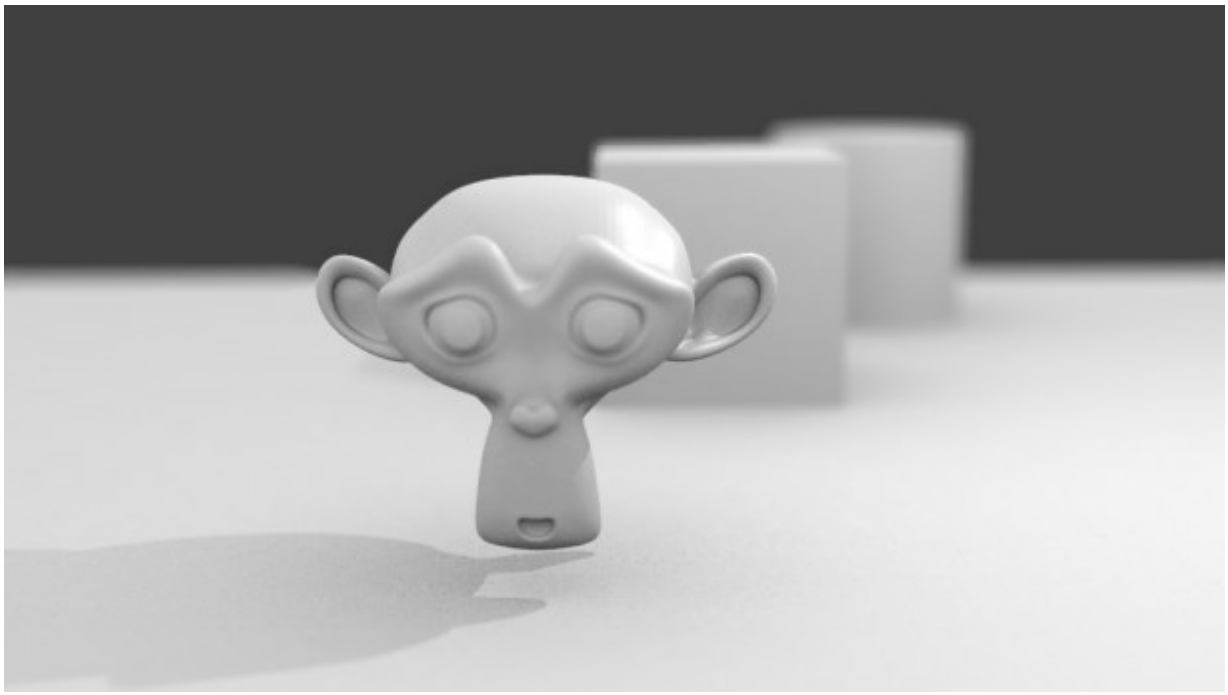


Рисунок 1.8 – Приклад використання глибини різко зображуваного простору

Існує багато технік шейдингу, які застосовуються при рендерингу, проте в більшості випадків вони поділяються на «плаский» шейдинг та «гладкий» шейдинг. [3]

В таблиці 1.1 наведено порівняльну характеристику «плаского» шейдингу та «гладкого» шейдингу.

Таблиця 1.1 – Порівняння різних технік шейдингу

Характеристика	«плаский» шейдинг	«гладкий» шейдинг
Складність обчислення	Невелика	Дуже велика
Можливість використання на гладких поверхнях	Відсутня	Наявна
Спосіб зміни кольору полігону	Один колір для всього полігону	Кожна точка полігону має власний колір
Вираженість граней об’єкта	Сильно виражені	Відсутні

На рисунку 1.9 зображено приклад використання «плаского» шейдингу.

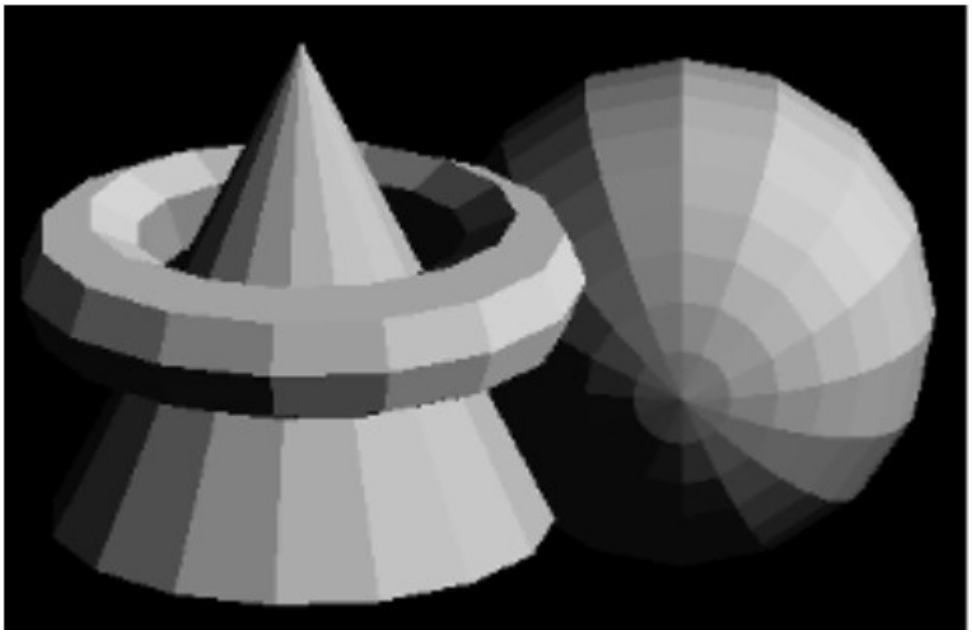


Рисунок 1.9 – Приклад використання «плаского» шейдингу

На рисунку 1.10 зображено приклад використання «гладкого» шейдингу.

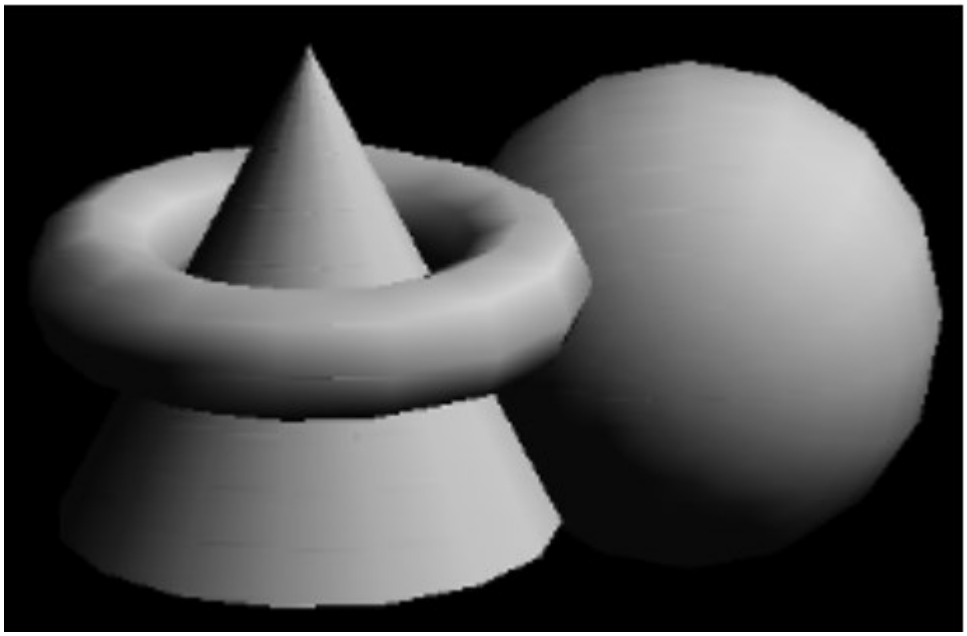


Рисунок 1.10 – Приклад використання «гладкого» шейдингу

1.2 Змістовний опис і аналіз предметної області

Кожне програмне забезпечення, яке здатне рендерити об’ємні моделі, оперує одним або декількома алгоритмами рендерингу. Їх існує багато, але серед них найбільш популярним є трасування променів. [4]

Його суть полягає у тому, що у просторі задається камера, яка по суті є точкою у просторі з вектором напряму зйомки. В даному випадку вектор напрямку зйомки залежить від параметрів висоти та ширини екрану, а також куту огляду камери.

Модель, яку необхідно зобразити, задається у вигляді набору полігонів(трикутників) в об'ємному просторі. Кожен полігон є набором із трьох точок у просторі.

Також перед камерою на невеликій відстані задається екран, що являє собор набір пікселів. Кожен піксель екрану відповідає пікселю у вихідному зображенні. За замовчуванням, кожен піксель екрану має певний початковий колір.

Через кожний піксель у екрані проходить промінь з камери у бік моделі. Якщо промінь, випущений камерою, перетинає модель, то піксель має бути пофарбований у певний колір. У протилежному випадку колір пікселя має залишитись сталим.

Колір, в який буде пофарбовано піксель, може залежати від багатьох факторів. Наприклад, кут падіння променю на модель, глобальне освітлення, яскравість світла тощо.

Для того, щоб перевірити, чи перетинає промінь полігон, було використано алгоритм Моллера-Трумбора, який є найшвидшим для такої перевірки.[5]

Проблема даного алгоритму в тому, що необхідно перевірити перетин усіх полігонів моделі з усіма променями. Зазвичай трикутників у моделі більше мільйона, а тому алгоритм буде виконуватись занадто довго, особливо в однопотоковому режимі.

Для вирішення даної проблеми зазвичай використовують дерева, які поділятимуть модель на вузли за певним алгоритмом. Суть застосування дерев у тому, щоб одразу опускати ті полігони, які точно ніколи не будуть перетинатись з певним променем.

Для даної роботи була обрана структура даних octree, кожен вузол якої має вісім дочірніх вузлів. В якості вузлів було використано просторові куби. Найбільший куб, тобто весь простір, визначається за найбільшими за модулем координатами простору серед усіх полігонів. Таким чином став можливий рекурсивний обхід дерева в пошуках найменшого кубу, який перетинається з променем, що значно зменшує тривалість роботи алгоритму. На рисунку 1.11 зображено схему побудови octree.

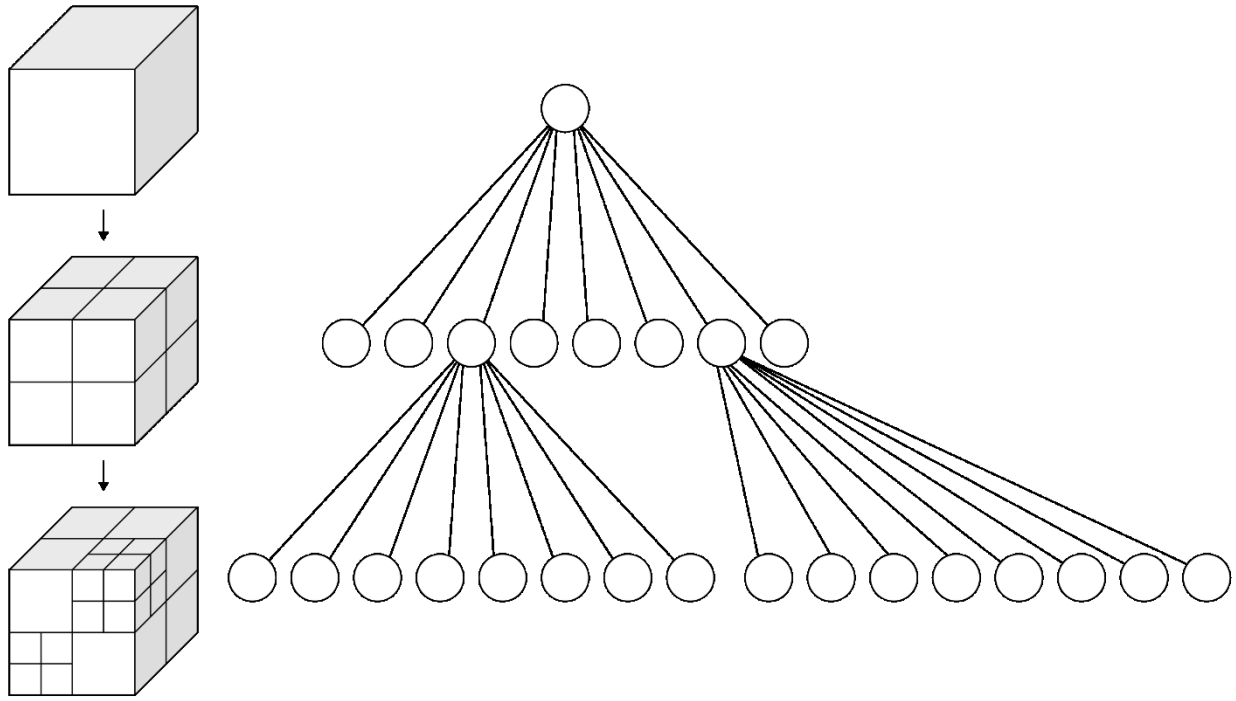


Рисунок. 1.11 – Схема побудови octree

1.3 Аналіз успішних IT-проектів

1.3.1 Аналіз відомих технічних рішень

Серед відомих технічний рішень зазвичай розрізняють такі, що здійснюють розрахунки на стороні кінцевого користувача та на стороні сервера. У даному випадку доречним є використання саме клієнт-серверної моделі, оскільки основною особливістю проекту є перенесення усіх розрахунків з обладнання користувача на серверну частину додатку.

Найбільш популярними на момент написання роботи мовами програмування та платформами для створення подібного роду додатків є C#,

Java та Node.js Для написання серверної частини платформа Node.js не підходить, оскільки не надає можливостей для роботи з багатопотоковістю. Принципових різниць для написання даної роботи між Java та C# немає.

Серверна частина дипломного проекту були виконана на платформі .NET 6 на мові C#.

Згідно з [6] .NET – це платформа з відкритим кодом від компанії Microsoft для створення різноманітних типів кросплатформових додатків. До типів створюваних додатків відносяться:

– web-застосунки;
– desktop застосунки;
– mobile застосунки;
– ігри;
– застосунки для штучного інтелекту та машинного навчання;
– класичні консольні застосунки;
– служби Windows.

Платформою підтримуються наступні мови програмування:

– C#;
– F#;
– Visual Basic.

Для розробки був обраний саме C#, оскільки він підтримує найбільшу кількість сторонніх бібліотек та пакетів, які були використані при реалізації.

Платформа .NET підтримує CTS, що надає усім підтримуваним мовам програмування спільний набір примітивних типів даних та створює можливості для інтеграції між ними.

У [7] C# - це сучасна високорівнева об’єктно-орієнтована мова програмування зі строгою типізацією. Вона належить до родини C-подібних мов та має схожий синтаксис з такими мовами, як:

– C;
– C++;
– Java;

– JavaScript.

Хоч C# і строго типізована мова програмування, у ній є можливість використовувати динамічну типізацію, про зазвичай цього не роблять.

Увесь код, написаний на мові C#, виконується у CLR та компілюється у IL. Код та вбудовані у проект ресурси зберігаються у файлах збірок, які мають розширення .dll.

У CLR також міститься механізм GC, який автоматично керує виділенням пам’яті. Також у розробника є можливість власноруч керувати пам’яттю, як, наприклад, мові C++, завдяки механізму unsafe-коду. Такий блок коду дає можливість працювати напряму з пам’яттю, проте через це є небезпечним і такий підхід не варто використовувати за наявності вбудованого GC.

Для розробки веб застосунків різних видів на .NET, існує фреймворк ASP .NET Core. У [8] є наступне:

– можливість розгортання на багатьох серверах;

– відкритий код;

– можливість розробки під Windows та Unix-подібні операційні системи;

– вбудована підтримка патерну розробки DI.

В якості СУБД була обрана Microsoft SQL Server. Згідно з [9] перевагами даної СУБД є:

– висока продуктивність завдяки компресії та шифруванню;

– безпечність зберігання даних завдяки складним алгоритмам шифрування та закритому коду;

– велика кількість механізмів відновлення даних;

– легкість інтеграції з платформою .NET.

Для зберігання «гарячих» даних була обрана No-SQL резидентна СУБД Redis. У [10] наведені такі переваги:

– висока продуктивність завдяки зберіганню даних прямо в оперативній пам’яті;

– підтримка багатьох типів даних;

– розширюваність та масштабованість;

– наявність зручної SDK для платформи .NET.

Крім переваг слід також виділити той недолік, що дана СУБД не є надійною, а тому не підходить для зберігання критичних даних, втрата яких є недопустимою.

Для того, щоб забезпечити оперативну комунікацію між сервісами був обраний такий брокер повідомлень, як RabbitMQ. Згідно з [11] має наступні переваги та можливості:

– надійність зберігання та передачі повідомлень;

– гнучкість налаштування маршрутизації повідомлень;

– можливість формування кластерів із декількох брокерів;

– можливість рівномірного розподілення повідомлення між підписниками;

– наявність зручної SDK для платформи .NET.

У якості API-шлюзу було обрано Amazon API Gateway. У [12] наведено наступні переваги:

– гнучкі налаштування безпеки;

– продуктивність при будь-якому масштабі;

– наявність моніторингу викликів API.

1.3.2 Аналіз відомих програмних продуктів

Одним із сервісів зі схожим функціоналом можна вважати сервіс TurboRender.

TurboRender – це сервіс, що надає можливість виконати рендеринг на віддалених серверах. Сервіс доступний через браузерний клієнт та десктопний додаток. Для аналізу був обраний браузерний клієнт. На рисунку 1.12 зображено головну сторінку сайту.

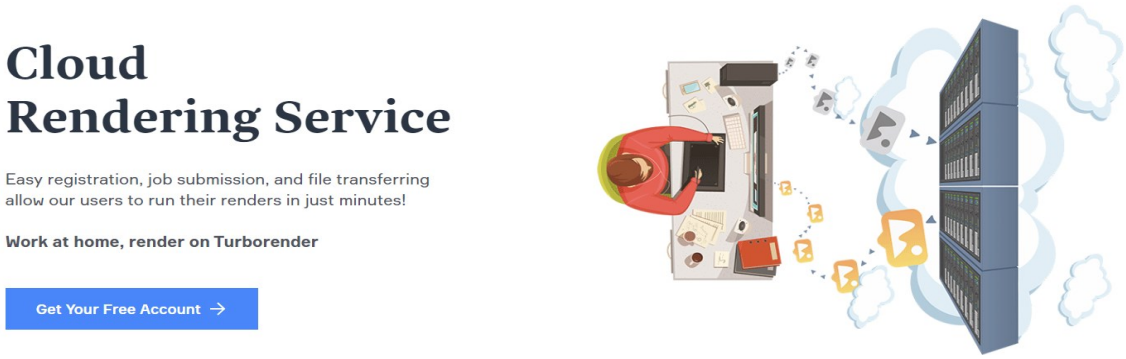


Рисунок 1.12 – Головна сторінка сервісу TurboRender

Даний сервіс підтримує інтеграцію з багатьма десктопними рендерерами, серед яких найбільш відомими є:

– 3Ds Max;
– Cinema 4D;
– After Effects;
– Maya;
– Houdini;
– Blender.

При створенні замовлення на рендеринг, сервіс пропонує завантажити файл проекту, обрати кількість серверів, сцену, кількість кадрів в секунду та формат результуючого зображення. На рисунку 1.13 зображено процес створення замовлення на рендеринг.

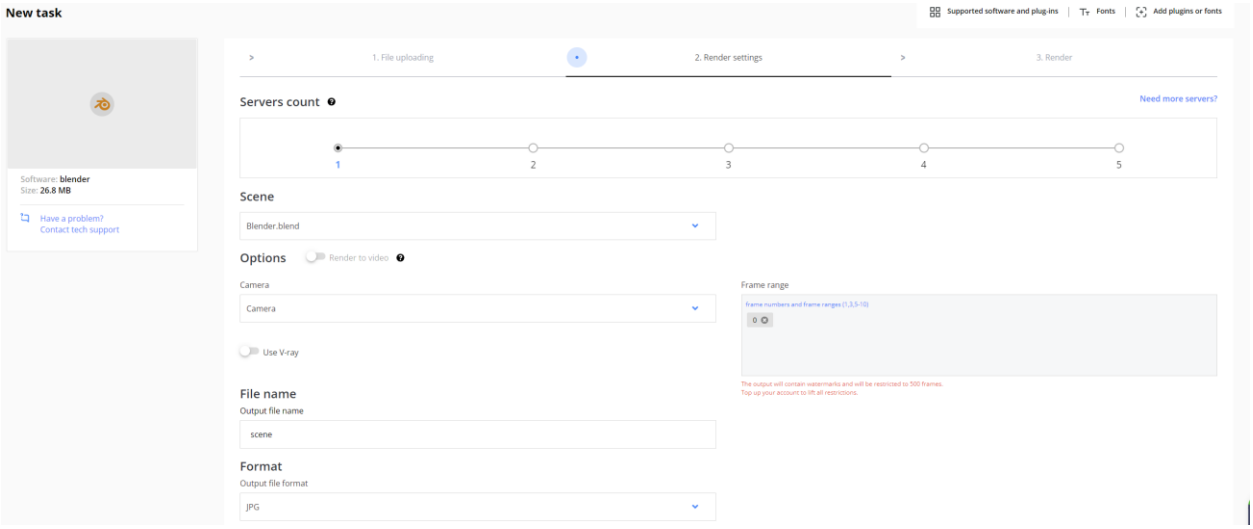


Рисунок 1.13 – Процес створення замовлення на рендеринг

Після створення замовлення, воно потрапляє в чергу, а користувач бачить основну інформацію про нього та поточний статус виконання. На рисунку 1.14 зображено створене замовлення.

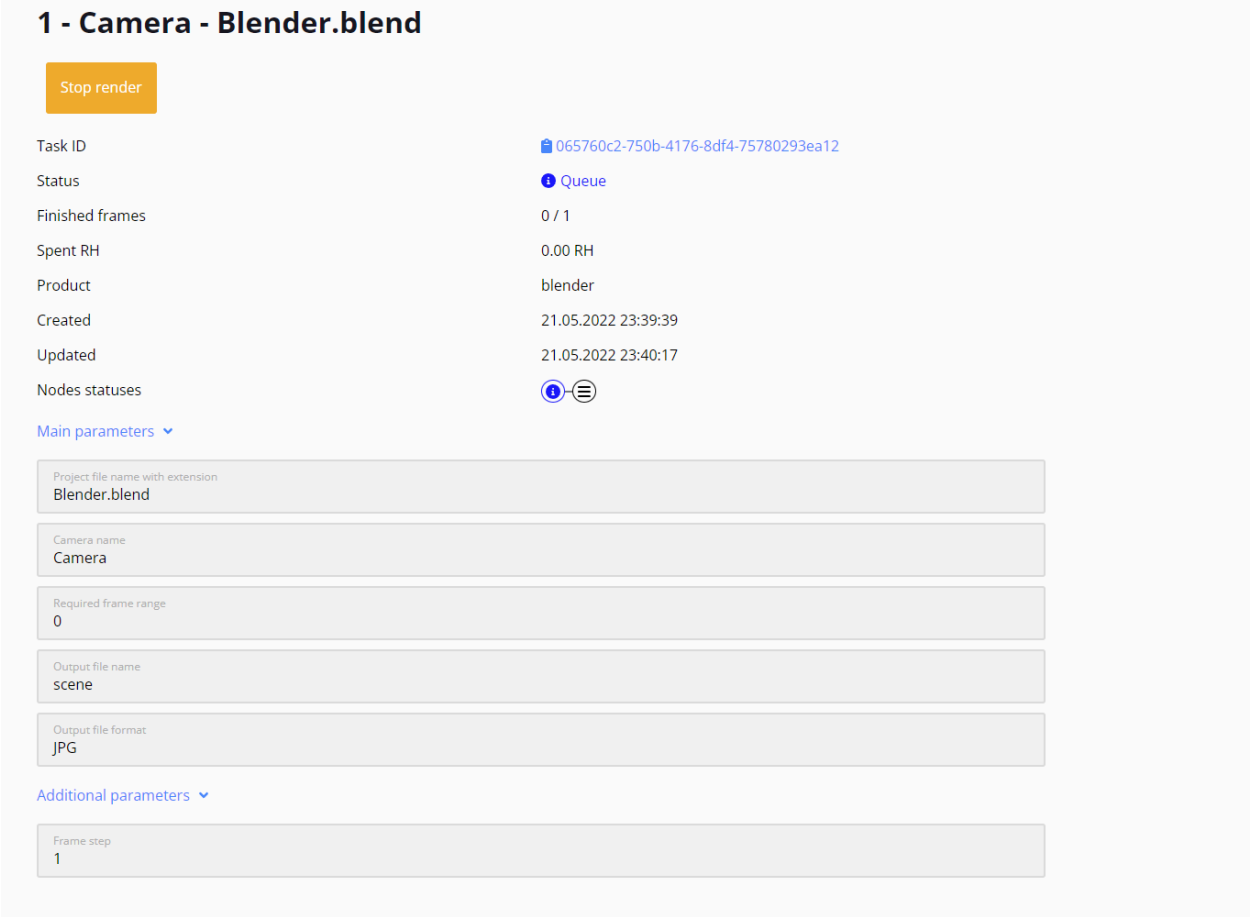


Рисунок 1.14 – Створене замовлення

Порівняння даного сервіса з дипломним проектом представлене у таблиці 1.2.

Таблиця 1.2 – Порівняння з аналогом

Критерій	Дипломна робота	TurboRender	Пояснення
Можливість вибору формату результуючого зображення	Відсутній	Наявний	У роботі для збереження зображення використовується лише формат .jpeg

Продовження таблиці 1.2

Критерій	Дипломна робота	TurboRender	Пояснення
Можливість збереження результуючого зображення на хмарне сховище	Наявний	Наявний	Обидва проекти зберігають результуюче зображення на хмарне сховище
Можливість одночасного рендерингу декількох замовлень	Наявний	Наявний	Обидна проекти здатні одночасно рендерити декілька замовлень
Можливість інтеграції зі сторонніми рендерерами та плагінами	Відсутній	Наявний	Робота приймає лише саму модель та надає власний список налаштувань при створенні замовлення на рендеринг
Можливість налаштування сцени без використання сторонніх програмних засобів	Наявний	Відсутній	TurboRender здатен виконувати рендеринг лише тих сцен, які були створені у сторонніх рендерерах

Продовження таблиці 1.2

Критерій	Дипломна робота	TurboRender	Пояснення
Можливість перегляду стану замовлень у реальному часі	Наявний	Наявний	Обидна проекти оновлюють інформацію про замовлення у реальному часі
Необхідність оформлення платної підписки	Відсутня	Наявна	Для користування TurboRender необхідне оформлення платної підписки

1.4 Аналіз вимог до програмного забезпечення

Головним функціоналом розроблюваного проекту є саме рендеринг об’ємних моделей. Перш ніж почати рендерити свої моделі, користувачу необхідно зареєструватися у системі за допомогою логіну та паролю, або авторизуватися, якщо він уже має акаунт. При створені замовлення на рендеринг, користувачу необхідно завантажити свою модель та обрати певні налаштування для сцени. Діаграма варіантів використання знаходиться у графічному матеріалі «Схема структурна варіантів використання».

1.4.1 Розроблення функціональних вимог

У таблицях 1.3 – 1.8 наведено варіанти використання проекту.

Таблиця 1.3 – Варіант використання UC-1

Назва	Реєстрація
Опис	Реєстрація нового користувача
Учасники	Неавторизований користувач

Продовження таблиці 1.3

Постумови	Створення пари Access Token та Refresh Token, та збереження облікового запису користувача у базі даних
Основний сценарій	Учасник надсилає HTTP запит з методом POST з тілом запиту у вигляді електронної пошти та пароля у форматі JSON. У відповіді отримує пару Access Token та Refresh Token, та профіль користувача.
Розширення сценаріїв	У випадку непроходження валідації введених даних, користувачу повертається у відповіді список помилок.

Таблиця 1.4 – Варіант використання UC-2

Назва	Авторизація
Опис	Авторизація учасника в системі
Учасники	Неавтризований користувач
Постумови	Створення пари Access та Refresh Token
Основний сценарій	Учасник надсилає HTTP запит з методом GET з тілом запиту у вигляді електронної пошти та пароля у форматі JSON. У відповіді отримує пару Access Token та Refresh Toke, та профіль користувача.

Продовження таблиці 1.4

Розширення сценаріїв	У випадку непроходження валідації введених даних, користувачу повертається у відповіді список помилок.

Таблиця 1.5 – Варіант використання UC-3

Назва	Перегляд своїх замовлень на рендеринг
Опис	Отримання списку замовлень учасника на рендеринг
Учасники	Авторизований користувач
Постумови	У відповіді міститься список отриманих замовлень
Основний сценарій	Учасник надсилає HTTP запит з методом GET, де міститься заголовок «Authorization» зі значенням у вигляді Access Token. У відповіді отримує список замовлень користувача, якому належить введений Access Token.
Розширення сценаріїв	Якщо вказаний Access Token є невалідним, то відповідь матиме код стану 401 Unauthorized. Якщо у користувача немає замовлень на рендеринг, то відповідь матиме код 404 Not Found.

Таблиця 1.6 – Варіант використання UC-4

Назва	Створення нового замовлення на рендеринг
Опис	Створення нового замовлення на рендеринг зі вказаними налаштуваннями
Учасники	Авторизований користувач
Постумови	Створення замовлення на рендеринг
Основник сценарій	Учасник надсилає HTTP запит з методом POST, де містяться заголовки «Authorization» зі значенням Access Token, та «Content-Type» зі значенням «multipart/form-data». Запит повинен містити форму, де у полі Model повинен знаходитись файл моделі, а у полі Order – замовлення у форматі JSON. Замовлення на рендеринг повинно містити назву замовлення, налаштування екрану, кольору об’єкта та заднього фону, камери, та список джерел світла.
Розширення сценаріїв	Якщо вказаний Access Token є невалідним, то відповідь матиме код стану 401 Unauthorized. У випадку непроходження валідації введених даних, користувачу повертається у відповіді список помилок.

Таблиця 1.7 – Варіант використання UC-5

Назва	Видалення замовлення на рендеринг
Опис	Видалення вказаного замовлення на рендеринг користувача
Учасники	Авторизований користувач
Постумови	Видалення вказаного замовлення із бази даних
Основний сценарій	Учасник надсилає HTTP запит з методом DELETE, де міститься заголовок «Authorization» зі значенням у вигляді Access Token. Кінцевий сегмент URL повинен містити ідентифікатор замовлення, яке необхідно видалити. Відповідь має статус код 204 No Content.
Розширення сценаріїв	Якщо вказаний Access Token є невалідним, то відповідь матиме код стану 401 Unauthorized. Якщо вказане замовлення не належить користувачу, то відповідь матиме код 403 Forbidden.

Таблиця 1.8 – Варіант використання UC-6

Назва	Перегляд стану замовлення в реальному часі
Опис	Оновлення інформації про певне замовлення на рендеринг у реальному часі
Учасники	Авторизований користувач

Продовження таблиці 1.8

Постумови	Учасник отримує повідомлення при зміні стану певного замовлення
Основний сценарій	Учасник надсилає HTTP запит з методом GET, де міститься заголовок «Authorization» зі значенням у вигляді Access Token. Запит повинен містити параметр id, який має значення ідентифікатора замовлення, стан якого треба відслідковувати. Відповіддю на запит є потік даних, який містить повідомлення про оновлення статусу виконання замовлення.
Розширення сценаріїв	Якщо вказаний Access Token є невалідним, то відповідь матиме код стану 401 Unauthorized. Якщо вказане замовлення не належить користувачу, то відповідь матиме код 403 Forbidden.

В таблицях 1.9 – 1.19 наведено опис функціональних вимог проекту.

Таблиця 1.9 – Функціональна вимога FR-1

Назва	Реєстрація
Опис	Система повинна надавати можливість реєстрації за електронною поштою і паролем

Таблиця 1.10 – Функціональна вимога FR-2

Назва	Авторизація
Опис	Система повинна надавати можливість авторизації за електронною пошту і паролем

Таблиця 1.11 – Функціональна вимога FR-3

Назва	Перевірка валідності введених даних
Опис	Система повинна валідувати введені користувачем дані, та надавати користувачу список помилок, якщо дані виявились невалідними. Електронна пошта повинна відповідати наступним вимогам: – має містити символ «@»; – символ «@» не має бути останнім у пошті; – символ «@» не має бути першим у пошті. Пароль має відповідати наступним вимогам: – має містити хоча б 10 символів; – має містити не більше 100 символі; – має містити хоча б 5 унікальних символів; – має містити хоча б одну велику літеру; – має містити хоча б одну маленьку літеру; – має містити хоча б одну цифру; – має містити хоча б один спеціальний символ; – не має містити пробілів. Файл моделі має містити хоча б одну поверхню. Назва замовлення на рендеринг повинна бути непустою. Висота екрану, ширини екрану, кут огляду та інтенсивність освітлення не повинні бути менше нуля. Рівні червоного, зеленого та синього повинні знаходитись в діапазоні 0-255.

Таблиця 1.12 – Функціональна вимога FR-4

Назва	Оновлення доступу до ресурсів
Опис	Система повинна надавати можливість оновити доступ до ресурсів, якщо строк його дії закінчився

Таблиця 1.13 – Функціональна вимога FR-5

Назва	Перегляд замовлень на рендеринг
Опис	Система повинна надавати можливість переглянути користувачу список своїх замовлень

Таблиця 1.14 – Функціональна вимога FR-6

Назва	Створення нового замовлення
Опис	Система повинна надавати можливість користувачу створити нове замовлення на рендеринг

Таблиця 1.15 – Функціональна вимога FR-7

Назва	Рендеринг об’ємної моделі
Опис	Система повинна надавати можливість рендерити об’ємні моделі

Таблиця 1.16 – Функціональна вимога FR-8

Назва	Зберігання файлів на хмарному сховищі
Опис	Система повинна завантажувати надані та результуючі файли до хмарного сховища, та зберігати посилання на них у базі даних

Таблиця 1.17 – Функціональна вимога FR-9

Назва	Оновлення статусу замовлення у реальному часі
Опис	Система повинна надавати можливість користувачу стежити за статусом виконання у режимі реального часу

Таблиця 1.18 – Функціональна вимога FR-10

Назва	Видалення замовлення
Опис	Система повинна надавати можливість видаляти користувачі свої замовлення

Таблиця 1.19 – Функціональна вимога FR-11

Назва	Перевірка доступу до ресурсу
Опис	Система повинна забороняти доступ до захищених ресурсів від неавторизованих користувачів, або користувачів, які не є власниками даних ресурсів

На рисунку 1.15 зображено матрицю трасування вимог.

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11
UC-1	+		+								
UC-2		+	+								
UC-3			+	+	+						+
UC-4			+	+		+		+			+
UC-5			+	+						+	+
UC-6			+	+			+	+	+		+

Рисунок 1.15 – Матриця трасування вимог

1.4.2 Розроблення нефункціональних вимог

Програмне забезпечення повинно бути:

– розширюваним;

– масштабованим;

– безпечним;

– задокументованим.

1.5 Постановка задачі

Розробка призначена для користувачів, що:

– мають необхідність у рендерингу об’ємних моделей, проте не мають відповідних програмних засобів;

– не мають технічних засобів, які здатні швидко виконувати велику кількість обрахунків;

– хочуть мати можливість рендерити свої об’ємні моделі будь-де, будь-коли та з будь-якого девайсу;

– мають потребу виконувати одночасний рендеринг декількох об’ємних моделей.

Метою розробки є:

– надання можливості виконання рендерингу об’ємних моделей при обмежених технічних ресурсах за рахунок виконання процесу з будь-якого пристрою з доступом до всесвітньої мережі Інтернет.

Мета досягається за рахунок розв’язання наступних задач:

– реалізації функціоналу управління обліковими записами користувачів програмних засобів для рендерингу об’ємних моделей;

– реалізації функціоналу менеджменту замовлень на рендеринг об’ємних моделей;

– реалізації алгоритму рендерингу шляхом трасування променів;

– реалізації функціоналу нотифікації клієнта про зміну стану замовлення на рендеринг у режимі реального часу.

Програмне забезпечення повинно виконувати усі функціональні та нефункціональні вимоги.

1.6 Висновки до розділу

В рамках першого розділу пояснювальної записки було досліджено предметну область програмного забезпечення та можливу галузь його застосування.

Під час аналізу успішних технічних рішень були обрані та обґрунтовані програмні засоби, які в подальшому будуть використовуватись для розробки проекту.

Порівняння зі схожими продуктами показало, що розроблюваний проект має низку переваг, серед яких особливою є те, що даний проект не потребує ніяких сторонніх програмних засобів для користування ним, та є повністю самостійним.

Для аналізу вимог до програмного забезпечення, було створено діаграму варіантів використання і розроблені функціональні вимоги. На основі детального опису варіантів використання та вимог, було створено матрицю трасування вимог.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

На рисунку 2.1 зображено BPMN діаграму, що описує бізнес процес рендерингу зображення від отримання нового замовлення до отримання готового результату.

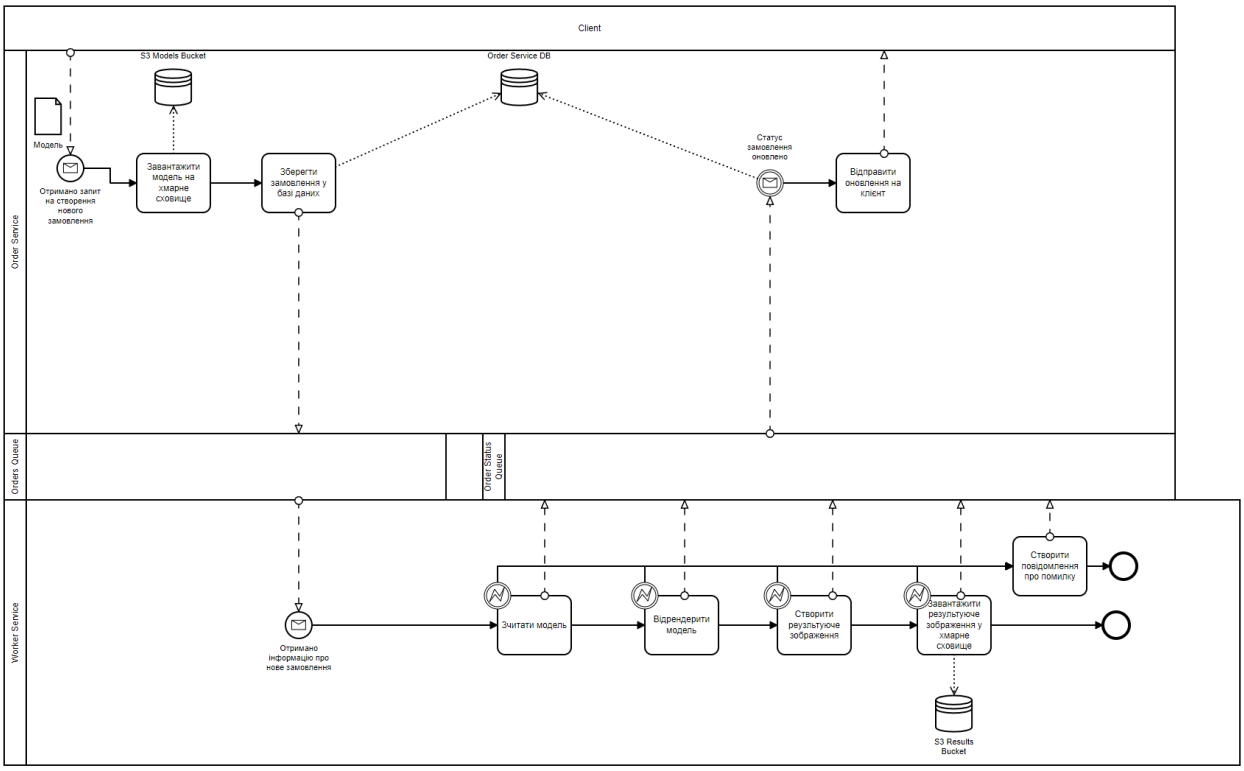


Рисунок 2.1 – Опис бізнес процесу рендерингу

Опис послідовності створення замовлення та рендерингу об’ємної моделі:

– клієнт надсилає на сервіс замовлень файл моделі та налаштування замовлення;

– якщо хоча б одне поле у налаштуваннях не пройшло валідацію, то сервіс замовлень повертає клієнту список помилок;

– якщо поля пройшли валідацію, то сервіс замовлень завантажує файл моделі у хмарне сховище S3 до бакету з моделями. У результаті сервіс замовлень отримує посилання на файл у сховищі;

– посилання на модель разом із налаштуваннями зберігаються у базу даних;

– сервіс замовлень створює повідомлення про нове замовлення та кладе його у чергу у брокер повідомлень;

– один із вільних воркерів, що підписаний на оновлення даної черги, отримує повідомлення;

– воркер отримує із повідомлення налаштування замовлення та завантажує із хмарного сховища файл моделі;

– воркер створює відповідне повідомлення та надсилає його у брокер повідомлень. Варто зазначити, що будь-яке повідомлення, що надсилається воркером, являє собою модель замовлення та обробляється сервісом замовлень, який оновлює запис відповідного замовлення у базі даних. Після оновлення, сервіс замовлень пересилає повідомлення на сторону клієнта за допомогою SSE. Таким чином забезпечується оновлення стану замовлення в реальному часі;

– воркер намагається зчитати модель та отримати список полігонів;

– якщо воркер не може зчитати файл, або він не містить полігонів, то створює повідомлення про відповідну помилку та кладе його у чергу у брокер повідомлень;

– якщо зчитування успішне, то воркер присвоює замовленню відповідний статус та кладе повідомлення у брокер повідомлень;

– воркер починає рендеринг отриманих полігонів з файлу моделі з вказаними у замовленні налаштуваннями;

– результатом рендерингу є бітмапа, де кожному пікселю екрану присвоєний певний колір. Воркер створює відповідне повідомлення для брокера повідомлень;

– наступним кроком є перетворення отриманої бітмапи у результуюче зображення формату .jpeg;

– воркер завантажує отримане зображення до хмарного сховища S3 у бакет з результатами рендерингу і створює останнє повідомлення для брокера повідомлень.

2.2 Архітектура програмного забезпечення

При виборі архітектурного стилю необхідно було враховувати те, що проект має бути масштабованим та легко розширюватись. Серед найбільш популярних та успішних стилів зазвичай вирізняють монолітну та мікросервісну архітектури.

Монолітна архітектура характеризується тим, що усі компоненти проекту є пов’язаними та залежать один від одного. Така архітектура вважається найбільш розповсюдженою та ціниться за свою зрозумілість та легкість розробки. Проблемою даного стилю є те, що йому бракує можливостей для розширення та масштабування. Додавання нової логіки призводить до необхідності по перерозгортання усього проекту.

Мікросервісна архітектура дозволяє вирішувати проблеми масштабування та розширювання завдяки повній незалежності одних компонентів від інших. Така система здатна повторно використовувати уже створені компоненти та нарощувати продуктивність за рахунок збільшення кількості інстансів сервісів. Проблемою такої архітектури є те, що вона більш складна у розумінні, потребує загалом більше часу для розробки та вимагає володіння додатковими інструментами для налаштування взаємодії між сервісами. Незважаючи на усі недоліки, було вирішено використати мікросервісний підхід для вирішення поставленої задачі.

Схему взаємодії між сервісами зображено у графічному матеріалі «Схема структурна компонентів програмного забезпечення».

Кожен з мікросервісів був побудований з використанням трьохшарової архітектури. Шар являє собою абстракцію, яка слугує для розмежування методів за їх призначенням. Кожен шар має доступ лише до методів сусіднього шару нижнього рівня. Опис кожного з шарів описано в таблиці 2.1.

Таблиця 2.1 – Опис шарів трьохшарової архітектури

Назва шару	Опис
DAL	Даний шар слугує для взаємодії з джерелами даних. Зазвичай такими джерелами є бази даних. Тут зберігаються класи-сутності, які представлені у базах даних у вигляді таблиць. Також шар може містити інтерфейси, які описують методи взаємодії з даними для шару бізнес логіки.
BLL	Даний шар слугує для обробки даних, отриманих від клієнта, в рамках певних бізнес процесів. Має доступ до DAL і за потреби може використовувати його методи для доступу до джерел даних.
PL	Даний шар слугує для прямої взаємодії з клієнтом. Передає та отримує результуючі дані у BLL. Може описувати правила та формати, за якими дані повинні передаватись від клієнта до сервіса, та навпаки.

Типова схема трьохшарової архітектури зображена на рисунку 2.2

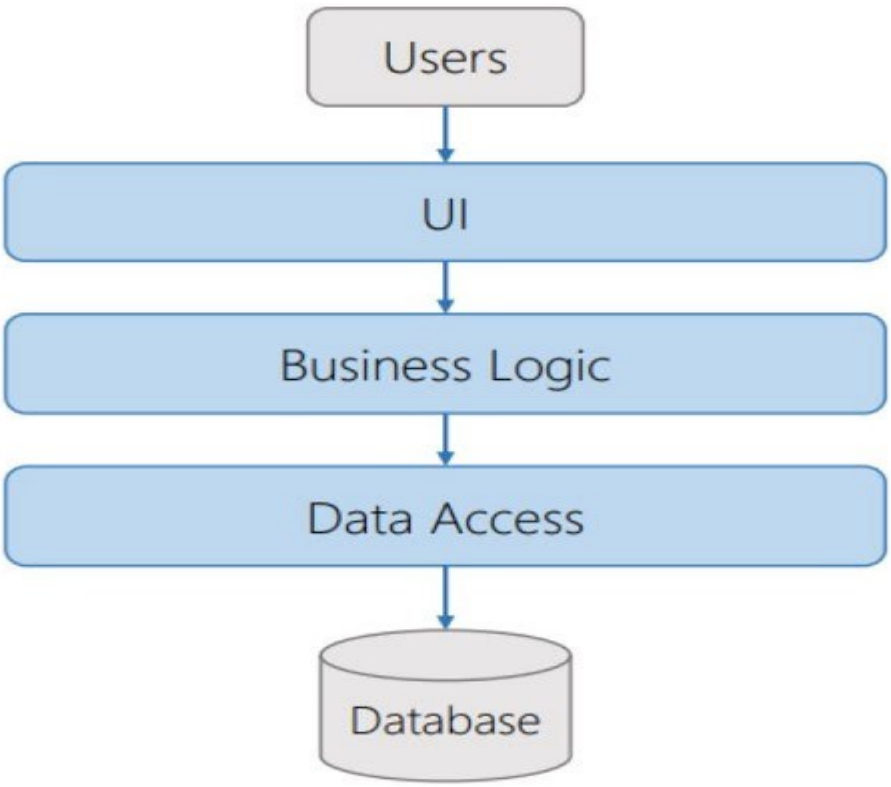


Рисунок 2.2 – Типова схема трьохшарової архітектури

Для додавання пакетів до проектів на платформі .NET використовується менеджер пакетів Nuget. Він містить у собі великий спектр різних пакетів та надає можливості для версіонування встановлених пакетів. Список використаних пакетів, бібліотек та наборів інструментів зазначений у таблиці 2.2.

Таблиця 2.2 – Список використаних пакетів, бібліотек та фреймворків

Назва залежності	Призначення
Microsoft.Entity.FrameworkCore	Являє собою ORM для спрощення роботи із базами даних
Microsoft.Entity.FrameworkCore.Relational	Додає можливості гнучкого налаштування відношень між сутностями у базі даних
StackExchange.Redis	Клієнт для роботи із СУБД Redis
AutoMapper	Інструмент для мапінгу об’єктів між різними шарами архітектури
Isopoh.Cryptography.Argon2	Бібліотека для хешування паролів за допомогою алгоритму Argon2

Продовження таблиці 2.2

Назва залежності	Призначення
Microsoft.IdentityModel.Tokens	Бібліотека, що надає криптографічні методи для шифрування токенів доступу
Microsoft.IdentityModel.Tokens.Jwt	Бібліотека для створення та валідації JWT
Newtonsoft.Json	Фреймворк для роботи з файлами формату JSON
AutoMapper.Extensions.Microsoft.DependencyInjection	Додає DI механізм для AutoMapper
Microsoft.EntityFrameworkCore.SqlServer	Додає DI механізм для створення підключення до СУБД Microsoft SQL Server
AWSSDK.Core	SDK для роботи із клієнтом AWS
AWSSDK.S3	SDK для роботи із сервісом S3
Microsoft.Extensions.Hosting	Бібліотека для створення бекграунд-процесів
Microsoft.Extensions.Logging	Додає можливість логування інформації

Продовження таблиці 2.2

Назва залежності	Призначення
Microsoft.Extensions.Options	Додає DI механізм для налаштувань проекту
RabbitMQ.Client	Бібліотека, що містить клієнт брокера повідомлень RabbitMQ
SkiaSharp	Бібліотека для створення зображень на основі бітмапи
Swashbuckle.AspNetCore	Бібліотека для генерування Swagger документації для API
Lib.AspNetCore.ServerSentEvents	Бібліотека для підтримки SSE

Дипломний проект складається із трьох сервісів, які представлені у платформі .NET трьома рішеннями. Далі наводиться опис класів, що відповідають за бізнес логіку кожного із сервісів.

2.2.1 Опис архітектури сервісу авторизації

Argon2PasswordService – клас-сервіс, який за допомогою бібліотеки Isopoh.Cryptography.Argon2 виконує хешування паролів та валідацію їх хешів. Методи даного класу наведені в таблиці 2.3.

Таблиця 2.3 – Методи класу Argos2PasswordService

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
1	GenerateHash	Пароль у вигляді рядку	Хешує пароль алгоритмом Argon2	Об’єкт класу PasswordResult, який містить захешований пароль, сіль та версію алгоритму шифрування
2	Verify	Хеш, сіль та пароль, який необхідно перевірити	Перевіряє, чи введений пароль відповідає хешу	True, якщо пароль відповідає хешу, у протилежному випадку false

AuthorizationService – клас-сервіс, який авторизує користувача в системі за введеними електронною поштою та паролем. Методи даного класу наведено в таблиці 2.4.

Таблиця 2.4 – Методи класу AuthorizationService

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
1	Authorize	Електронна пошта та пароль у вигляді рядків	Авторизує користувача в системі	Пара Access Token та Refresh Token, а також профіль користувача

На рисунку 2.3 зображено алгоритм авторизації.

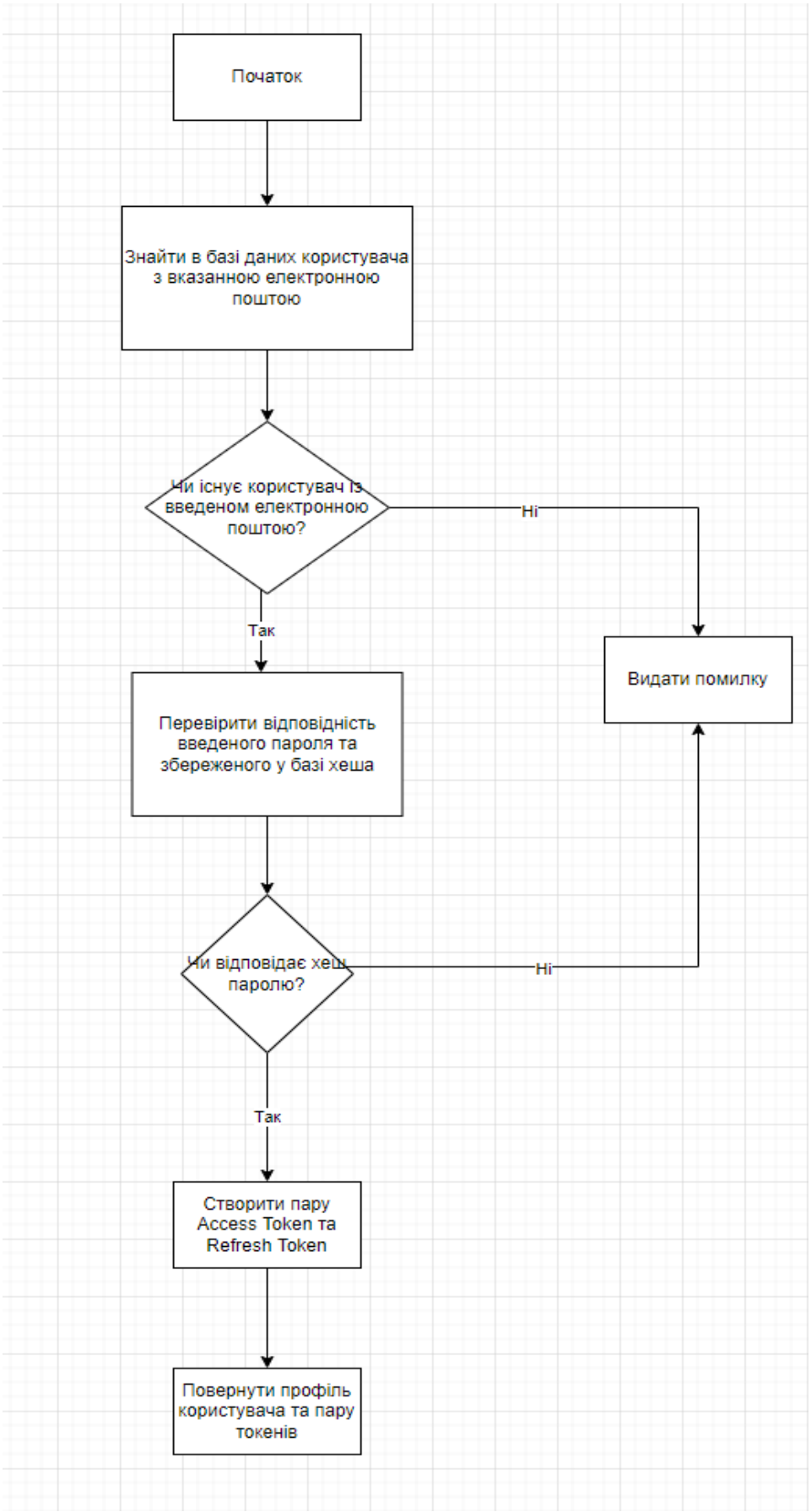


Рисунок 2.3 – Алгоритм авторизації

RegistrationService – клас-сервіс, що відповідає за реєстрацію користувачів в системі за введеними електронною поштою і паролем. Методи даного класу наведені в таблиці 2.5.

Таблиця 2.5 – Методи класу RegistrationService

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
1	RegisterUser	Електронна пошта та пароль у вигляді рядків	Реєструє нового користувача в системі	Пара Access Token і Refresh Token, а також профіль користувача
2	ValidateUserEmail	Електронна пошта у вигляді рядка	Перевіряє, чи існує користувач з введеною поштою у базі даних	True, якщо такий користувач існує, у протилежному випадку false

На рисунку 2.4 зображено алгоритм реєстрації.

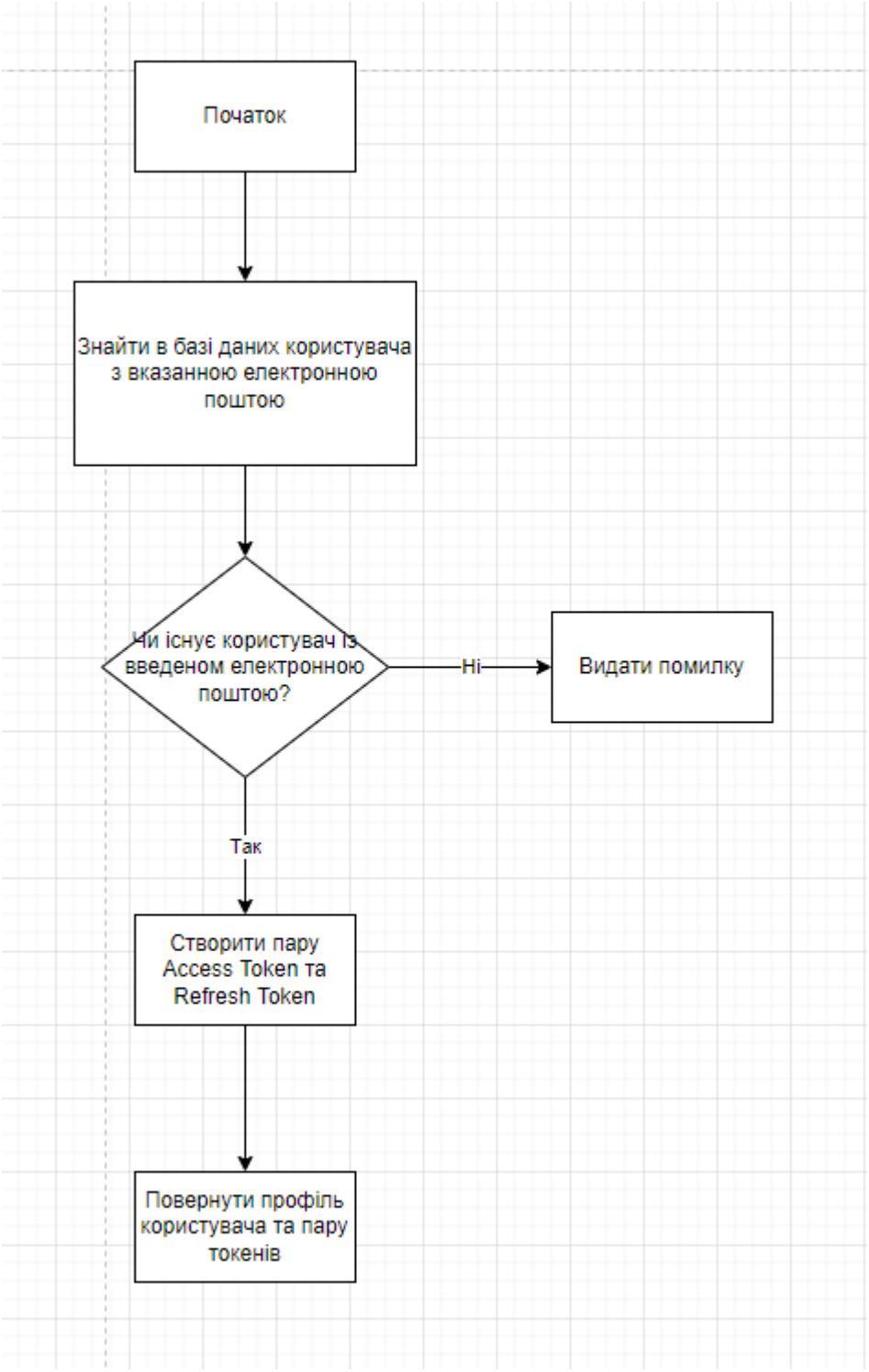


Рисунок 2.4 – Алгоритм реєстрації

TokenManager – клас-утиліта, що відповідає за створення та валідацію токенів. В таблиці 2.6 описано методи класу TokenManager.

Таблиця 2.6 – Методи класу TokenManager

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
1	GenerateTokens	Дані про власника токену, зокрема його ідентифікатор та електронна пошта	Створює пару Access Token і Refresh Token	Повертає пару Access Token і Refresh Token
2	GenerateJwt	Дані про власника токену, зокрема його ідентифікатор та електронна пошта	Створює JWT	Повертає JWT у вигляді рядка
3	GenerateRefreshToken	Відсутні	Створює Refresh Token	Повертає Refresh Token у вигляді рядка
4	GetClaim	JWT та ключ у вигляді рядків	Зчитує значення даних про власника JWT за вказаним ключем	Повертає рядок

Продовження таблиці 2.6

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
5	TryValidateJwt	JWT у вигляді рядка та посилання на змінну, в яку необхідно записати розшифрований JWT	Перевіряє валідність вказаного JWT	Повертає true і записує у передане посилання значення розшифрованого JWT, якщо токен валідний. У протилежному випадку повертає false.

UserModel – це клас-сутність, який повертається користувачу при успішній авторизації чи реєстрації. На рисунку 2.5 показано клас UserModel.

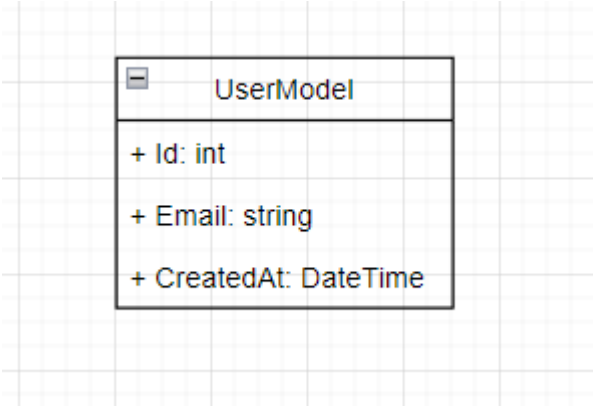


Рисунок 2.5 – Клас UserModel

2.2.2 Опис архітектури сервісу замовлень

CloudStorageService – клас-сервіс, який відповідає за роботу з хмарним сховищем S3 у таблиці 2.7 наведено опис методів класу CloudStorageService.

Таблиця 2.7 – Методи класу CloudStorageService

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
1	UploadFileAsync	Ім’я файлу та послідовність байтів файлу	Завантажує файл на хмарне сховище S3	Посилання на завантажений файл
2	DeleteFileAsync	Посилання на файл	Видаляє файл по вказаному посилання із хмарного сховища	Файл видалено
3	GetObjectStreamAsync	Посилання на файл	Зчитує файл за вказаним посиланням	Дані файлу у вигляді потоку байтів

OrdersProducer – клас-сервіс, який відповідає створення та відправку повідомлень про створення нового замовлення у брокер повідомлень RabbitMQ. У таблиці 2.8 наведено методи класу OrdersProducer.

Таблиця 2.8 – Методи класу OrdersProducer

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
1	PushMessage	Об’єкт, який необхідно подати у вигляді повідомлення	Створює та надсилає повідомлення про створення нового замовлення до брокеру повідомлень	Повідомлення надіслано

OrderMessage – це клас повідомлення, яким обмінюються між собою сервіси. Являє собою модель замовлення. На рисунку 2.6 показано клас OrderMessage.

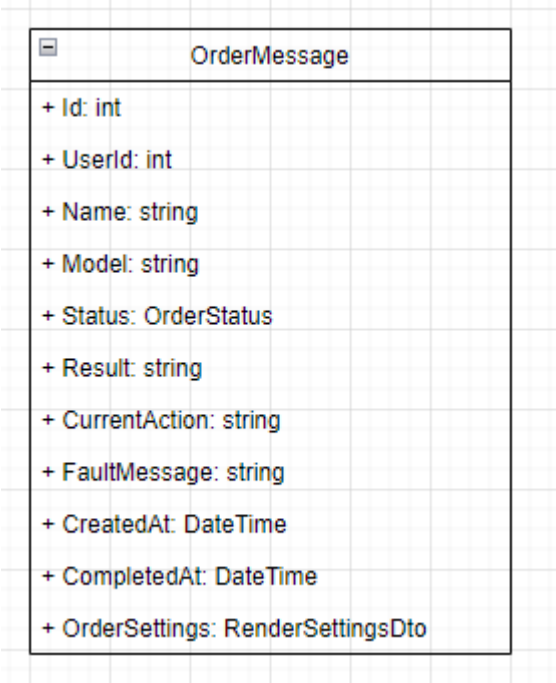


Рисунок 2.6 – Клас OrderMessage

У таблиці 2.8 зазначено опис класу OrderMessage.

Таблиця 2.9 – Опис класу OrderMessage

Назва поля	Опис
Id	Ідентифікатор замовлення
UserId	Ідентифікатор користувача, що створив замовлення
Name	Ім’я замовлення
Model	Посилання на файл моделі у хмарному сховищі
Status	Поточний статус замовлення
Result	Посилання на файл результуючого зображення у хмарному сховищі
CurrentAction	Поточна дія, що виконує воркер над замовленням
FaultMessage	Характеризує помилку, що сталася під час рендерингу
CreatedAt	Дата створення замовлення

Продовження таблиці 2.9

Назва поля	Опис
CompletedAt	Дата завершення замовлення
OrderSettings	Налаштування замовлення

OrderSettingsDto – клас, що агрегує у собі всі налаштування замовлення. На рисунку 2.7 зображено описано класи налаштувань замовлення.

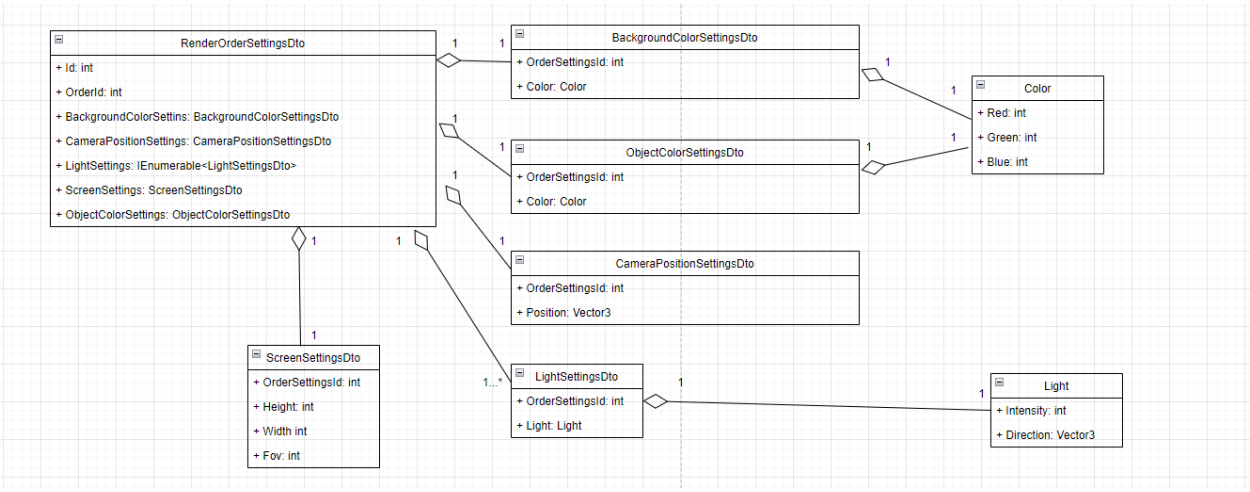


Рисунок 2.7 – Опис класів налаштування замовлень

2.2.3 Опис архітектури воркера

JpegImageWriter – клас-сервіс, що відповідає за створення зображень формату .JPEG за допомогою бібліотеки SkiaSharp. У таблиці 2.10 наведено методи класу JpegImageWorker.

Таблиця 2.10 – Методи класу JpegImageWorker

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
1	WriteImage	Бітмапа у вигляді двовимірного масиву векторів	Перетворює векторну бітмапу на зображення формату .JPEG	Зображення формату .JPEG у вигляді потоку байтів

Tree – це клас, що являє собою структуру даних octree. Кожний вузол даного дерева представлений класом TreeNode. Вузли складається із восьми

дочірніх вузлів та просторового кубу. Просторові куби задаються класом Cube, який зберігає у собі вісім точок у просторі, а також містить логіку перевірки, чи перетинається від заданим променем. У таблицях 2.11-2.13 наведено методи, які описують структуру octree.

Таблиця 2.11 – Методи класу Tree

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
1	Initialize	Список полігонів та максимальна за модулем координата у просторі	Створює та ініціалізує вузли дерева	Створено вузли дерева
2	FindIntersections	Точка початку променю, вектор напрямку променю, список полігонів, в який необхідно додати полігони, які перетинаються з променем	Проводить рекурсивний обхід дерева в пошуку полігонів, які перетинаються з променем починаючи з кореня	Список полігонів, що перетинаються з променем

Таблиця 2.12 – Методи класу TreeNode

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
1	Rebuild	Число, яке характеризує мінімальну можливу довжину ребра куба, який вставляється у дерево	Ділить куб у вузлі на вісім однакових менших кубів, створює дочірні вузли та розподіляє полігони між ними	Утворено вісім дочірніх вузлів та розподілено полігони між ними
2	FindIntersections	Точка початку променю, вектор напрямку променю, список полігонів, в який необхідно додати полігони, які перетинаються з променем	Рекурсивно перевіряє перетин променю для кожного з дочірніх вузлів	Список трикутників, що перетинаються з променем

Продовження таблиці 2.12

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
3	PositionFinder	Точка у просторі та центр куба	Визначає індекс дочірнього вузла, якому належить введена точка	Індекс дочірнього вузла у списку дочірніх вузлів

Таблиця 2.13 – Методи класу Cube

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
1	ThereIsIntersectionBetweenRayAndTriangle	Точка початку променю, вектор напрямку променю та одна з площин куба	За допомогою алгоритму Моллера-Трумбора, перевіряє чи перетинає промінь задану площину	True, якщо промінь перетинає вказану площину, у протилежному у випадку - false

Продовження таблиці 2.13

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
2	IntersectionBetweenRayAndCube	Точка початку променю, вектор напрямку променю	Перевіряє, чи перетинає промінь хоча одну з площин куба	True, якщо промінь перетинає хоча б одну з площин куба, у протилежному у випадку - false

ObjModelReader – клас-сервіс для зчитування об’ємних моделей з файлів формату .OBJ. В таблиці 2.14 наведені методи класу ObjModelReader.

Таблиця 2.14 – Методи класу ObjModelReader

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
1	ReadModel	Файл моделі у вигляді потоку байтів	Зчитує вміст об’ємної моделі з файлу	Список полігонів моделі
2	ProcessVertex	Вміст моделі у вигляді масиву рядків	Зчитує з файлу список вершин	Список вершин
3	ProcessNormal	Вміст моделі у вигляді масиву рядків	Зчитує з файлу список нормалей	Список нормалей
4	ProcessFace	Вміст моделі у вигляді масиву рядків	Зчитує з файлу список полігонів	Список полігонів

Renderer – клас-сервіс, що рендерить об’ємну модель. В таблиці 2.15 наведено методи класу Renderer.

Таблиця 2.15 – Методи класу Renderer

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
1	GetCameraDirection	Індекси пікселя на екрані та налаштування екрану	Обчислює вектор напрямку камери відносно конкретного пікселя та налаштувань екрану	Вектор напрямку камери
2	Initialize	Двовимірний масив пікселів та налаштування кольору заднього фону	Фарбує усі пікселі у заданий в налаштуваннях колів	Двовимірний масив пікселів

Продовження таблиці 2.15

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
3	CastRay	Точка початку променю, вектор напрямку променю, налаштування світла та полігон	Пускає промінь у вибраному напрямку та обчислює значення кольору для заданих налаштувань світла	Якщо промінь перетинає полігон, то повертає точку перетину та колів. У протилежному випадку - нічого
4	ProcessPixel	Індекси пікселя на екрані, список трикутників та налаштування	Обчислює значення кольору для конкретного пікселя у результуючому зображенні	Значення кольору

RenderService – клас-сервіс, який запускає ланцюжок процесу рендерингу від зчитування моделі, до завантаження результуючого зображення до хмарного сховища. У таблиці 2.16 наведено методи класу RenderService.

Таблиця 2.16 – Методи класу RenderService

№ п/п	Назва методу	Опис параметрів	Опис методу	Результат
1	RenderModel	Об’єкт класу замовлення	Запускає ланцюжок рендерингу об’ємної моделі	Результуюче зображення завантажується до хмарного сховища

OrderStatusProducer – клас-сервіс, що відповідає за створення та доставку повідомлень про оновлення статусу замовлень. Його структура аналогічна структурі класу, наведеному у таблиці 2.7.

2.3 Конструювання програмного забезпечення

В рамках даного проекту PL кожного мікросервіса являє собою API, побудоване за принципами REST. Кожен ендпоїнт API являє собою окремий метод певного контролера. Перевагами використання REST архітектури є:

– масштабованість;
– простота, зрозумілість та одноманітність інтерфейсу для клієнта;
– кешування запитів;
– надійність.

Щоб відповідати архітектурі REST, API повинно відповідати наступним вимогам:

– повинно використовувати клієнт-серверну модель;
– стан клієнта не повинен зберігатися на стороні сервера;
– запити повинні кешуватися;
– інтерфейс повинен бути одноманітним та бути побудованим навколо «ресурсів», тобто абстракцій, які формують можливості інтерфейсу.

Типова схема REST API зображена на рисунку 2.8.

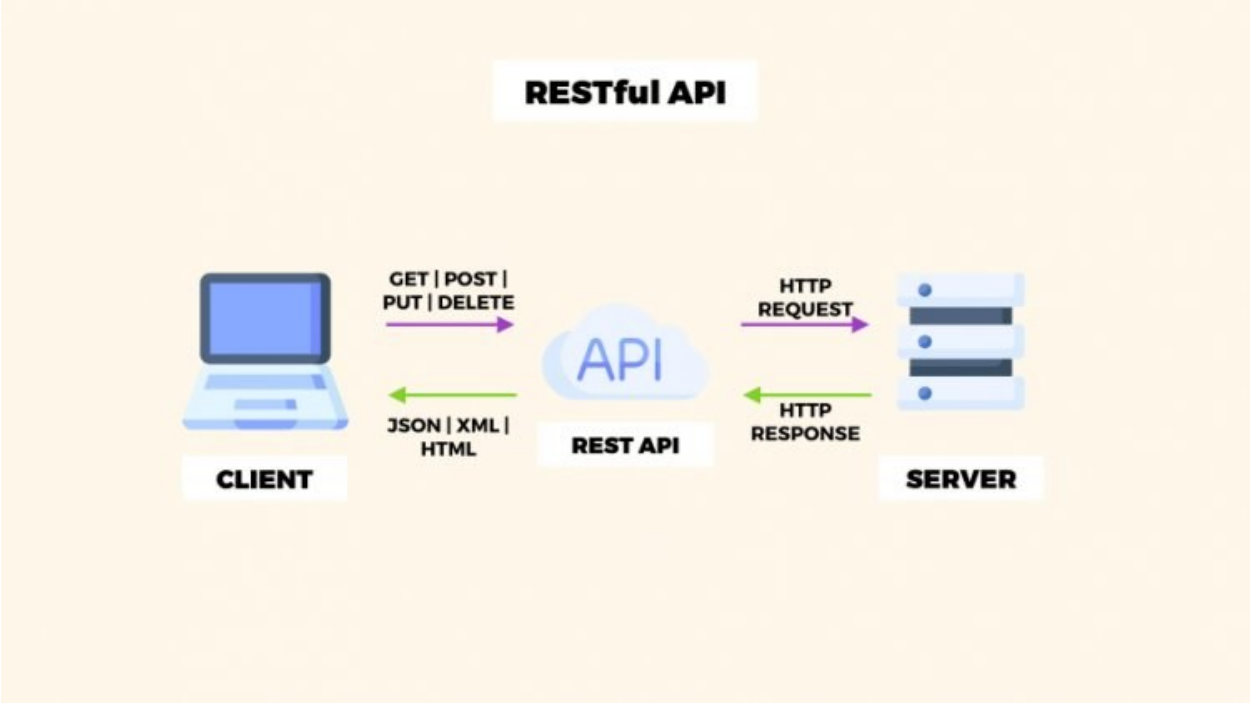


Рисунок 2.8 – Типова схема REST API

Для розробки дипломного проекту був використаний ряд утиліт. Опис кожної утиліти наведено в таблиці 2.17.

Таблиця 2.17 – Опис використаних утиліт

№ п/п	Назва утиліти	Опис застосування
1	JetBrains Rider	IDE для платформи .NET
2	Postman	Інструмент для тестування API
3	Telerik Fiddler	Інструмент для відстежування HTTP трафіку
4	Microsoft SQL Management Studio	Інструмент для управління всіх компонентів СУБД SQL Server
5	Git	VCS
6	RedisInsight	Інструмент для керування СУБД Redis
7	Docker	Інструмент для управління контейнерами

Сервіси авторизації та замовлень мають власні бази даних з використанням реляційної СУБД Microsoft SQL Server і резидентної СУБД

Redis. Для того, щоб уникнути аномалій при видаленні, додаванні та зміні записів у таблицях, бази даних були приведені до третьої нормальної форми. Схема усіх баз даних знаходиться у графічному матеріалі «Схема бази даних».

Далі наводиться опис кожної бази даних кожного сервісу.

2.3.1 Опис баз даних сервісу авторизації

Сервіс авторизації використовує дві бази даних. Для зберігання користувачів використовується база даних Users, що знаходиться у СУБД SQL Server. У таблиці 2.18 наведено опис таблиці Users.

Таблиця 2.18 – Опис таблиці Users

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
Users	Id	Int	Ідентифікатор користувача
	Email	Nvarchar(max)	Електронна пошта користувача
	PasswordHash	Nvarchar(max)	Хеш паролю користувача
	PasswordSalt	Nvarchar(max)	Сіль до хешу паролю
	PasswordVer	Nvarchar(max)	Версія алгоритму хешування
	CreatedAt	Datetime2(7)	Дата створення облікового запису

Для зберігання Refresh Token користувачів використовується база даних у СУБД Redis. Таблиці у Redis представлені у вигляді ключ-значення. В якості ключа було взято Refresh Token у вигляді рядку. У якості значення використано модель RefreshTokenData, яку було серіалізовано у файл формату .JSON. На рисунку 2.9 показано клас RefreshTokenData.

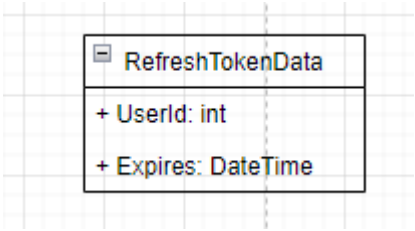


Рисунок 2.9 – Клас RefreshTokenData

2.3.2 Опис баз даних сервісу замовлень

Для зберігання даних про замовлення, сервіс використовує базу даних на СУБД SQL Server. У таблицях 2.19 – 2.24 наведено опис таблиць, що входять до бази даних.

Таблиця 2.19 – Опис таблиці Orders

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
Orders	Id	Int	Ідентифікатор замовлення
	UserId	Int	Ідентифікатор користувача, якому належить замовлення
	Name	Nvarchar(max)	Ім’я замовлення
	Status	Nvarchar(max)	Статус замовлення
	Result	Nvarchar(max)	Посилання на файл результуючого зображення
	Model	Nvarchar(max)	Посилання на файл моделі

Продовження таблиці 2.19

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
	FaultMessage	Nvarchar(max)	Опис помилки при виконанні замовлення
	CurrentAction	Nvarchar(max)	Опис поточної дії над замовленням
	CreatedAt	Datetime2(7)	Дата створення замовлення
	Completed	Datetime2(7)	Дата завершення замовлення

Таблиця 2.20 – Опис таблиці BackgroundColorSettings

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
BackgroundColorSettings	Id	Int	Ідентифікатор налаштування
	OrderId	int	Ідентифікатор замовлення
	Red	real	Рівень червоного
	Green	Real	Рівень зеленого
	Blue	real	Рівень синього

Таблиця 2.21 – Опис таблиці ObjectColorSettings

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
ObjectColorSettings	Id	Int	Ідентифікатор налаштування

Продовження таблиці 2.21

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
	OrderId	int	Ідентифікатор замовлення
	Red	real	Рівень червоного
	Green	Real	Рівень зеленого
	Blue	real	Рівень синього

Таблиця 2.22 – Опис таблиці CameraPositionSettings

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
CameraColorSettings	Id	Int	Ідентифікатор налаштування
	OrderId	int	Ідентифікатор замовлення
	PositionX	Real	Положення камери по осі абсцис
	PositionY	real	Положення камери по осі ординат
	PositionZ	Real	Положення камери по осі аплікат

Таблиця 2.23 – Опис таблиці LightSettings

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
LightSettings	Id	Int	Ідентифікатор налаштування

Продовження таблиці 2.23

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
	OrderId	int	Ідентифікатор замовлення
	PositionX	Real	Положення джерела світла по осі абсцис
	PositionY	real	Положення джерела світла по осі ординат
	PositionZ	Real	Положення джерела світла по осі аплікат
	Intensity	Int	Інтенсивність світла

Таблиця 2.24 – Опис таблиці ScreenSettings

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
ScreenSettings	Id	Int	Ідентифікатор налаштування
	OrderId	int	Ідентифікатор замовлення
	Width	Int	Ширина екрану
	Height	Int	Висота екрану
	Fov	Int	Кут огляду

Висновки до розділу

В рамках даного розділу було проведено наступне:

– описано основний бізнес процес рендерингу об’ємної моделі у вигляді BMPN-діаграми;

– обґрунтовано вибір архітектурно стилю;

– побудовано схему взаємодії між мікросервісами проекту;

– описано бібліотеки та пакети, що були використані під час розробки дипломного проекту;

– побудовані описи найбільш значущих класів та алгоритмів для кожного з сервісів;

– побудовано схеми баз даних та описані таблиці для кожного з сервісів;

– обґрунтовано використання трьохшарової архітекури в рамках кожного з сервісів;

– обґрунтовано побудову API з використанням REST архітектури.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Аналіз якості програмного забезпечення було проведено за допомогою методу ручного тестування. Даний метод тестування був обраний здебільшого через те, що якість результатів рендерингу неможливо оцінити з точки зору автоматизованої системи. Лише людське око здатне оцінити якість та виявити певні недоліки у результуючих зображеннях.

У рамках тестування було перевірено наступне:

– виконання запитів при введені валідних даних;

– виконання запитів при введені невалідних даних;

– час виконання рендерингу;

– наявність доступу до захищених ресурсів неавторизованим користувачем.

Для аналізу якості програмного забезпечення було використано розширення для IDE JetBrains ReSharper[13]. Дане розширення слугує для безперервного аналізу коду та здатне висвітлювати та виправляти проблемні місця.

3.2 Опис процесів тестування

В таблицях 3.1 – 3.10 наведено опис ручного тестування дипломного проекту.

Таблиця 3.1 – Тестування реєстрації

Тест	Реєстрація
Номер тесту	1.1
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Електронна пошта та пароль

Опис проведення тесту	У відповідні поля вводяться коректні дані та натискається кнопка «sign-up»
Очікуваний результат	Реєстрація успішна, відповідь містить пару Access Token і Refresh Token, а також профіль користувача
Стан системи після проведення тесту	Створено новий запис у таблиці користувачів і збережено Refresh Token в базі даних Redis

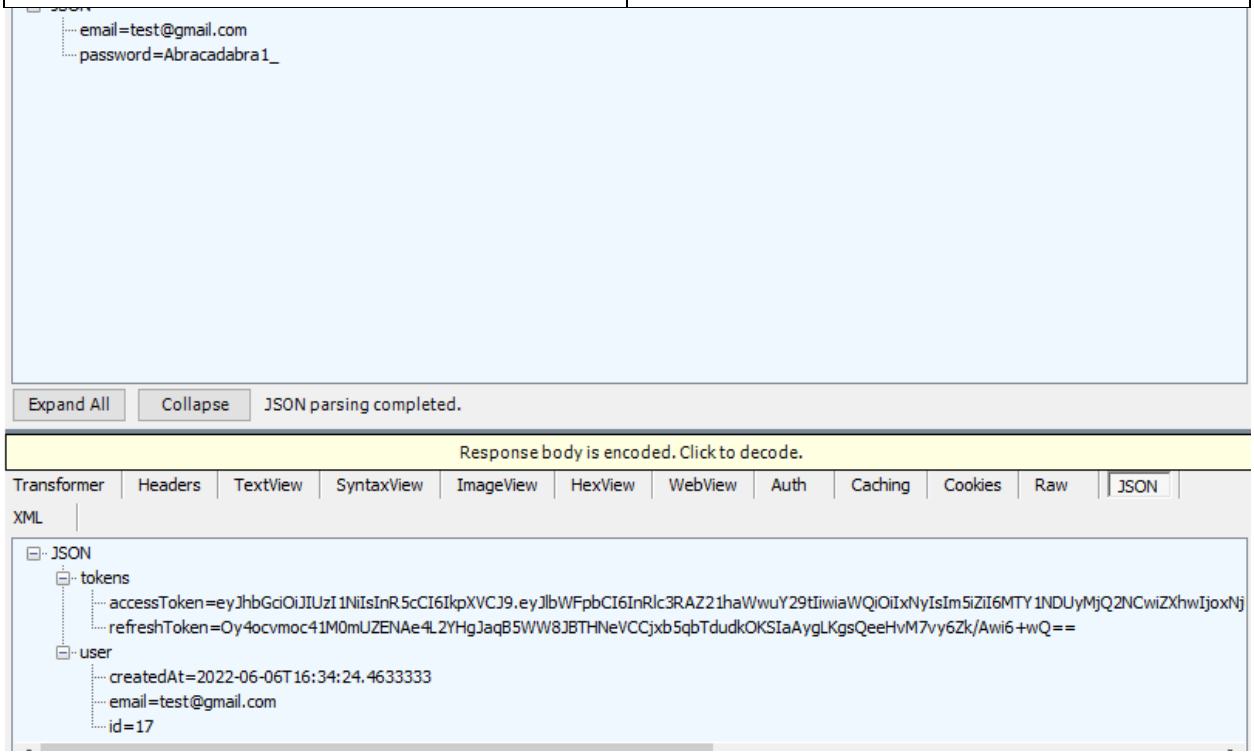


Рисунок 3.1 – Запит на реєстрацію та відповідь

Таблиця 3.2 – Тестування реєстрації із введенням невалідних даних

Тест	Реєстрація із введенням невалідних даних
Номер тесту	1.2
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Електронна пошта та пароль

Продовження таблиці 3.2

Опис проведення тесту	У відповідні поля вводяться невалідні електронна пошта та пароль
Очікуваний результат	Відповідь містить список помилок у введених даних
Стан системи після проведення тесту	Реєстрація неуспішна, відповідь містить список помилок

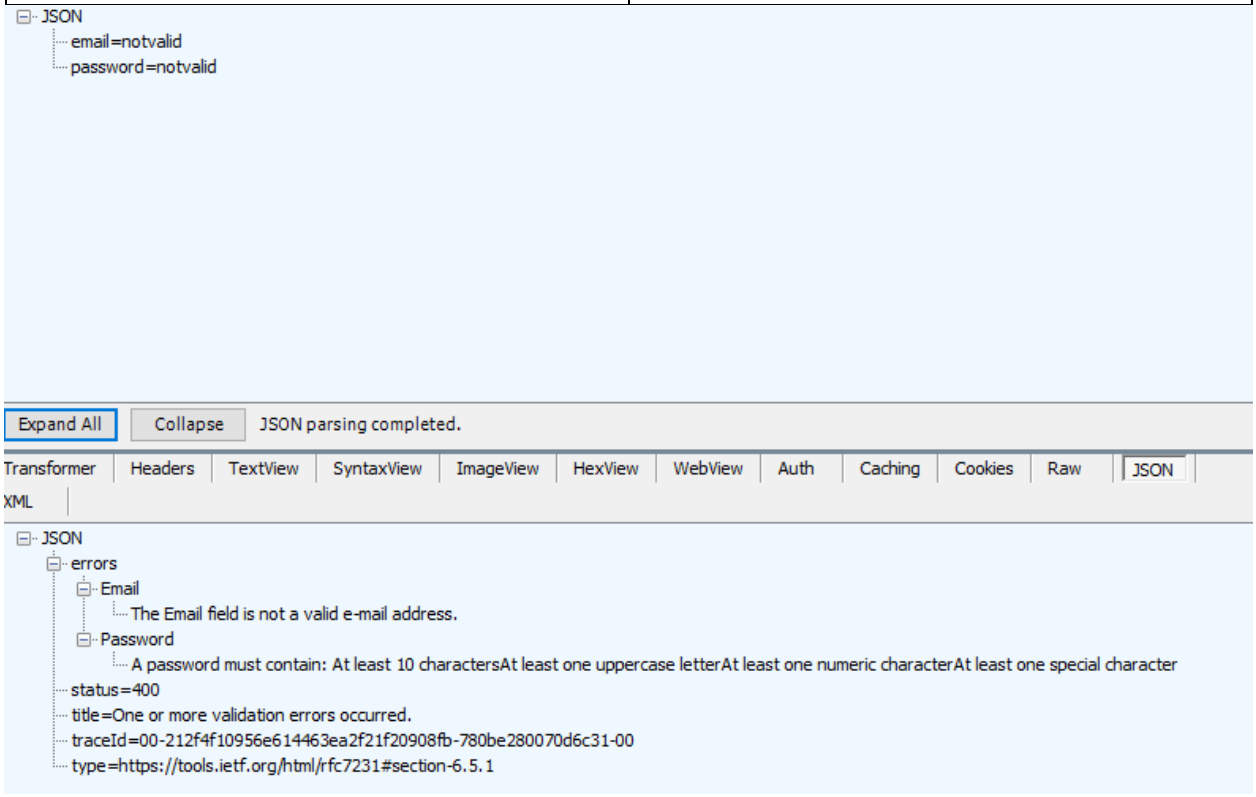


Рисунок 3.2 – Запит на реєстрацію із невалідними даними і відповідь

Таблиця 3.3 – Тестування реєстрації із введенням існуючої електронної пошти

Тест	Реєстрація із введенням існуючої електронної пошти
Номер тесту	1.3
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Електронна пошта і пароль

Продовження таблиці 3.3

Опис проведення тесту	У відповідні поля вводяться електронна пошта, яка існує в базі даних, та коректний пароль
Очікуваний результат	Відповідь містить повідомлення, що користувач із такою електронною поштою вже існує
Стан системи після проведення тесту	Реєстрація неуспішна, відповідь містить повідомлення, що користувач із такою електронною поштою вже існує

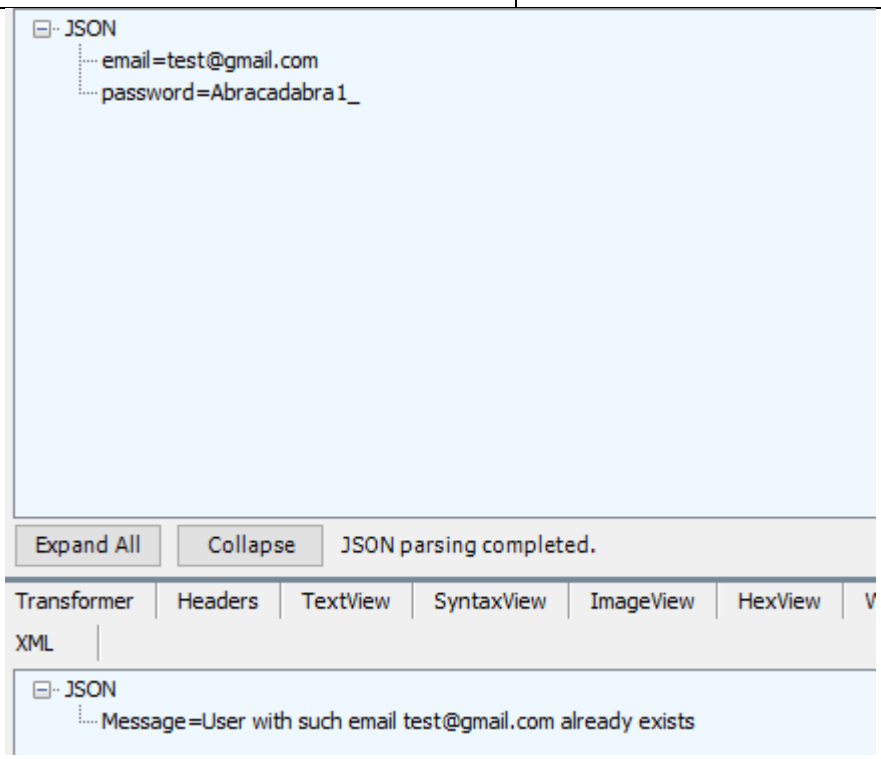


Рисунок 3.3 - Запит на реєстрацію із існуючою поштою і відповідь

Таблиця 3.4 – Тестування авторизації

Тест	Авторизація
Номер тесту	2.1
Початковий стан системи	Користувач зареєстрований та знаходиться на сторінці авторизації

Продовження таблиці 3.4

Вхідні дані	Електронна пошта та пароль
Опис проведення тесту	У відповідні поля вводяться коректні дані та натискається кнопка «sign-in»
Очікуваний результат	Авторизація успішна, відповідь містить пару Access Token і Refresh Token, а також профіль користувача
Стан системи після проведення тесту	Створено пару Access Token і Refresh Token, та збереження останнього в базі даних

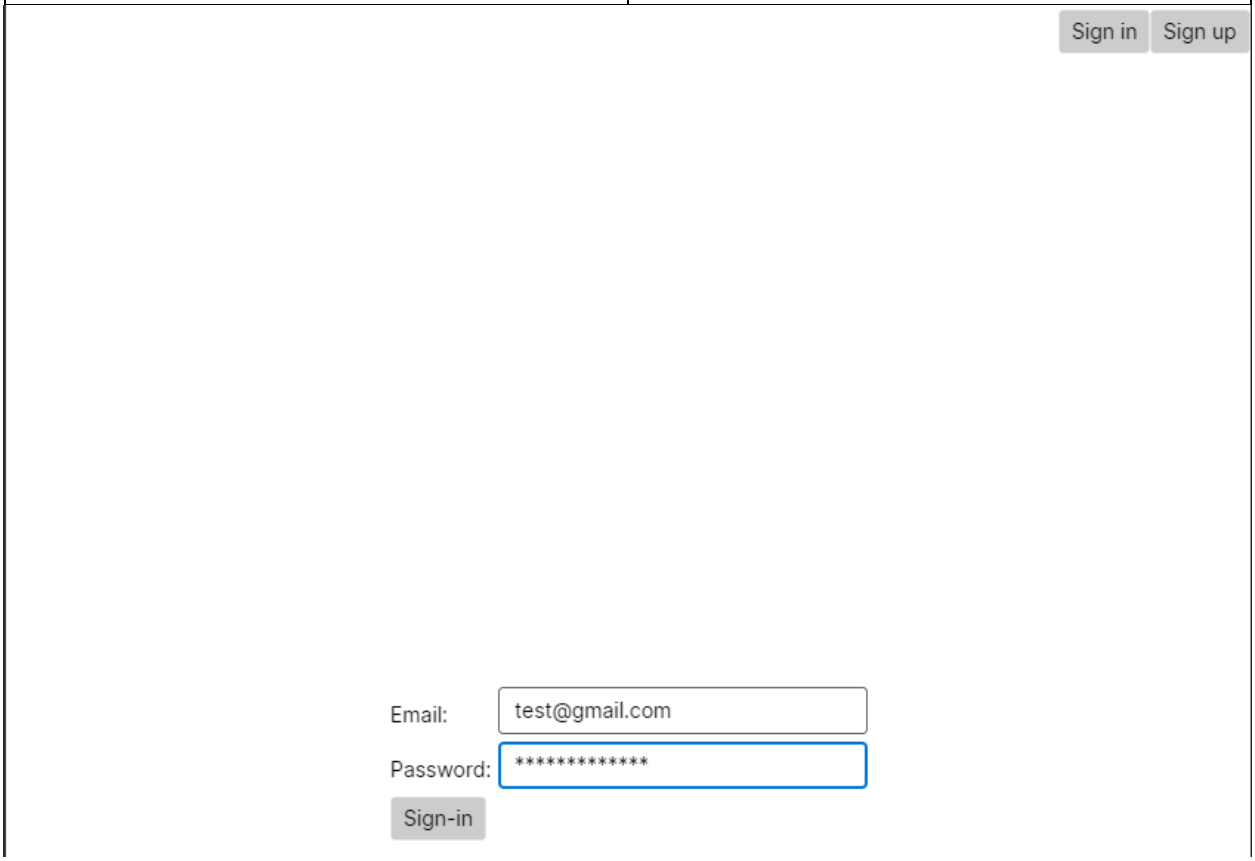


Рисунок 3.4 – Сторінка авторизації із введеними валідними даними

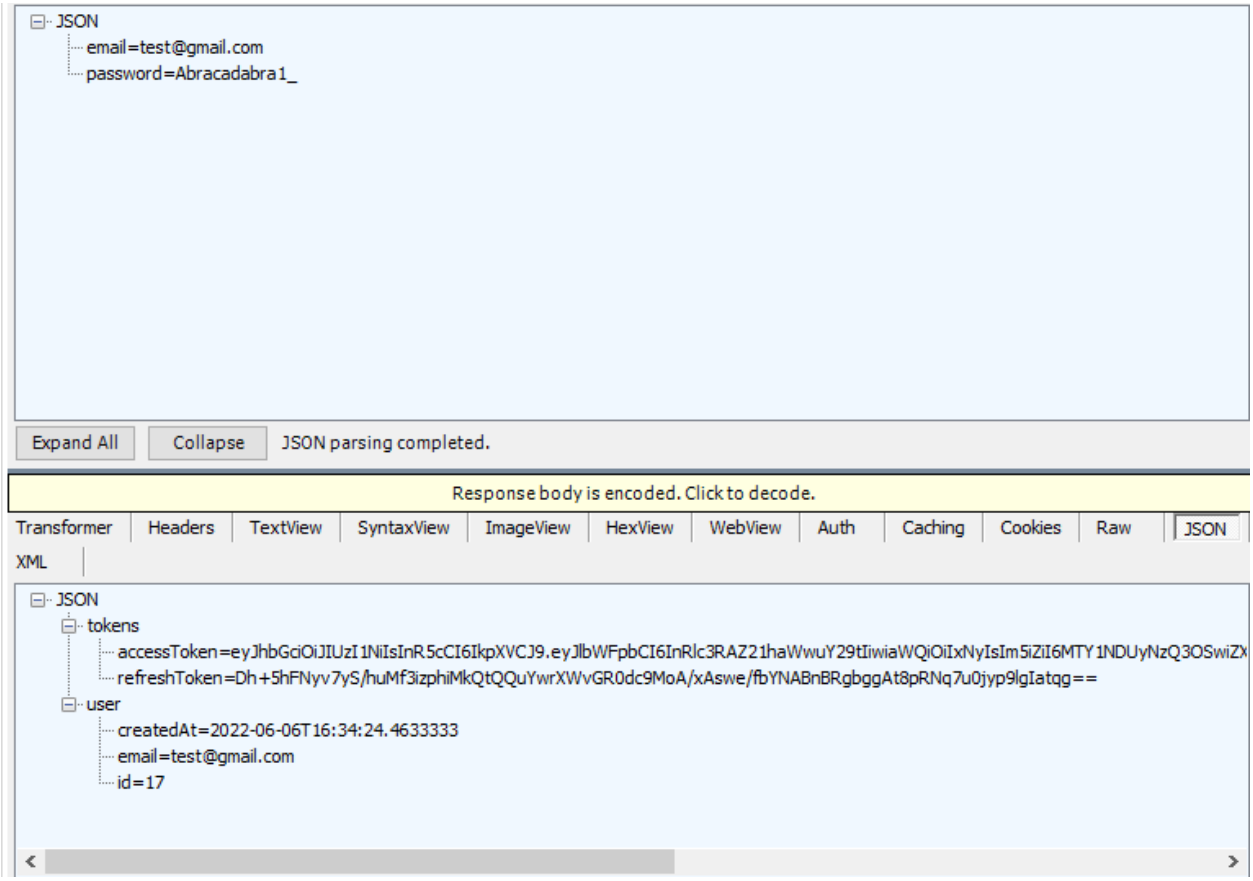


Рисунок 3.5 – Запит на авторизацію і відповідь

Таблиця 3.5 – Тестування авторизації із невалідними даними

Тест	Авторизація із невалідними даними
Номер тесту	2.2
Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні дані	Електронна пошта та пароль
Опис проведення тесту	У відповідні поля вводяться некоректні електронна пошта та пароль, та натискається кнопка «sign-in»
Очікуваний результат	Відповідь містить повідомлення, що електронна пошта або пароль є невірними

Продовження таблиці 3.5

Стан системи після проведення тесту	Авторизація неуспішна, відповідь містить повідомлення, що електронна пошта або пароль є невірними

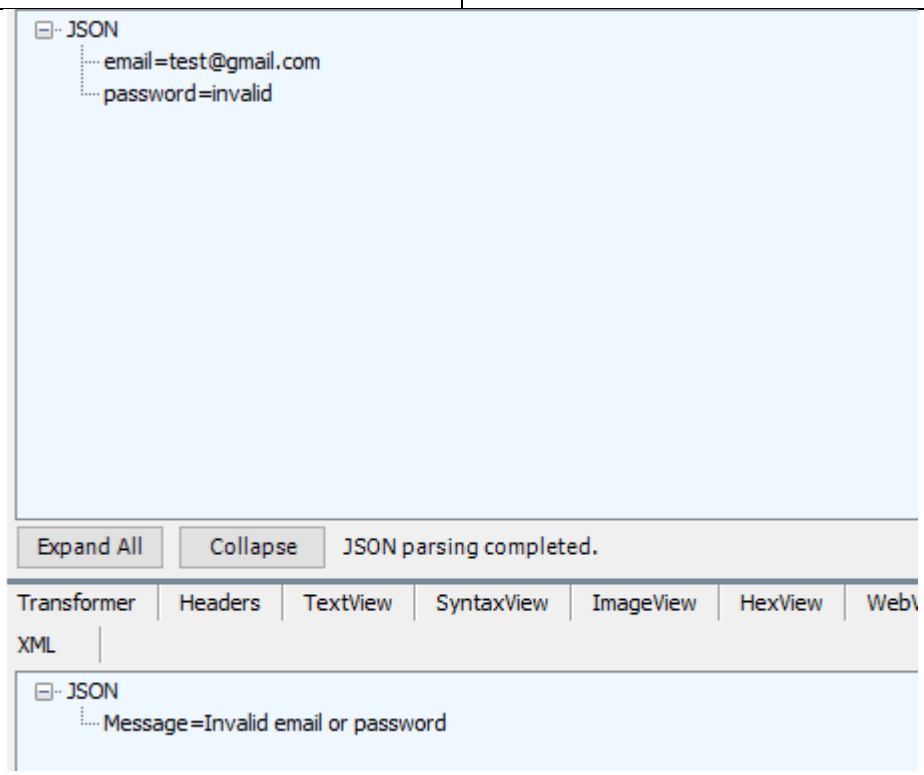


Рисунок 3.6 – Запит на авторизацію із некоректними даними і відповідь

Таблиця 3.6 – Тестування створення замовлення на рендеринг

Тест	Створення замовлення на рендеринг
Номер тесту	3.1
Початковий стан системи	Користувач авторизований та знаходиться на сторінці створення замовлення
Вхідні дані	Ім’я замовлення, модель, налаштування екрану, позиції камери, кольорів об’єкта та заднього фону та список джерел світла

Продовження таблиці 3.6

Опис проведення тесту	У відповідні поля вводяться коректні дані, завантажується коректна модель, та натискається кнопка «Create»
Очікуваний результат	Відповідь містить модель створеного замовлення
Стан системи після проведення тесту	Замовлення збережено у базі даних, воркери сповіщенні про створення нового замовлення

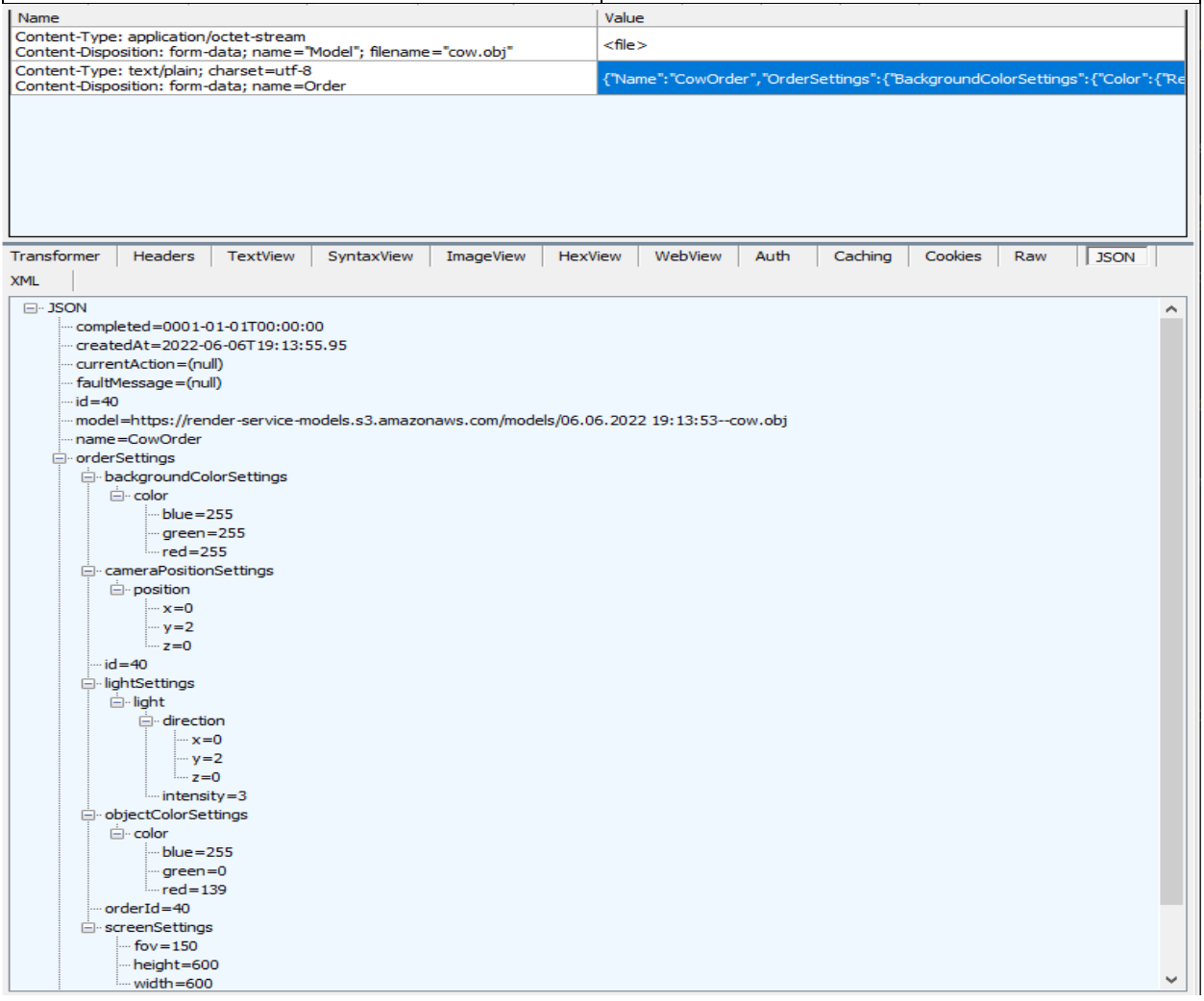


Рисунок 3.7 – Запит на створення замовлення і відповідь

Таблиця 3.7 – Тестування створення замовлення на рендеринг із некоректною моделлю

Тест	Створення замовлення на рендеринг із некоректною моделлю
Номер тесту	3.2
Початковий стан системи	Користувач авторизований та знаходиться на сторінці створення замовлення на рендеринг
Вхідні дані	Ім’я замовлення, модель у вигляді пустого файлу, налаштування екрану, позиції камери, кольорів об’єкта та заднього фону та список джерел світла
Опис проведення тесту	У відповідні поля вводяться коректні дані, завантажується некоректна модель, та натискається кнопка «Create»
Очікуваний результат	Під час рендерингу замовлення повинно отримати статус Faulted з відповідним повідомленням про помилку зчитування моделі
Стан системи після проведення тесту	Воркер сповістив про оновлення статусу замовлення сервіс замовлень, сервіс замовлень оновив відповідне замовлення у базі даних та сповістив клієнта

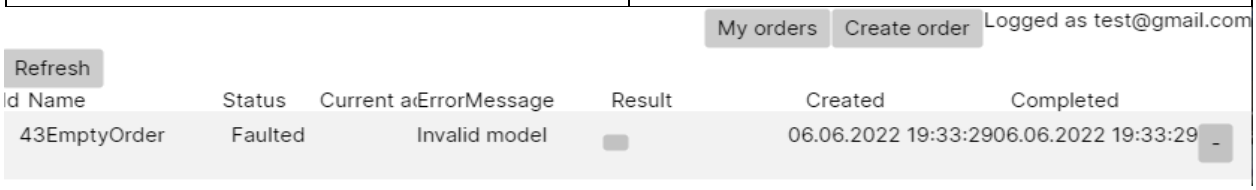


Рисунок 3.8 – Статус замовлення із некоректною моделлю

Таблиця 3.8 – Тестування перегляду власних замовлень на рендеринг

Тест	Перегляд власних замовлень на рендеринг
Номер тесту	4.1
Початковий стан системи	Користувач авторизований
Вхідні дані	-
Опис проведення тесту	Користувач натискає на кнопку «My orders»
Очікуваний результат	Відповідь містить список замовлень на рендеринг користувача
Стан системи після проведення тесту	Отримано список замовлень користувача та оформлено у вигляді таблиці

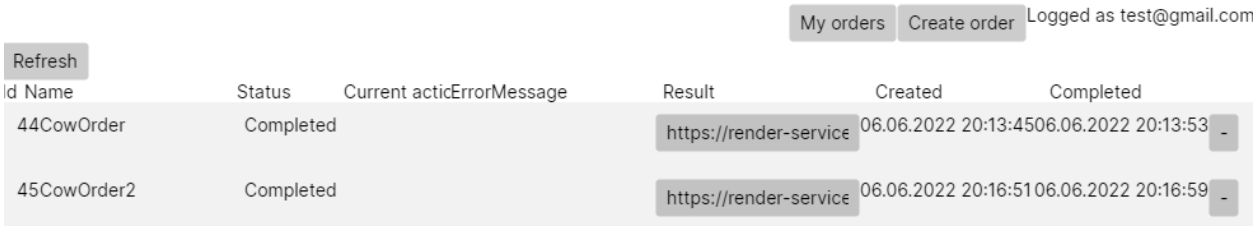


Рисунок 3.9 – Список замовлень користувача

Таблиця 3.9 – Тестування оновлення стану замовлення у реальному часі

Тест	Оновлення стану замовлення у реальному часі
Номер тесту	4.2
Початковий стан системи	Користувач авторизований та створив замовлення
Вхідні дані	-
Опис проведення тесту	Користувач натискає на кнопку «My orders» та за допомогою sse отримує дані про стан створеного замовлення

Продовження таблиці 3.9

Очікуваний результат	Замовлення оновлюються кожен раз, коли надходить сповіщення
Стан системи після проведення тесту	Дані оновлюються в таблиці замовлень

Таблиця 3.10 – Тестування видалення замовлення

Тест	Видалення замовлення
Номер тесту	5.1
Початковий стан системи	Користувач авторизований, знаходиться на сторінці замовлень та має створені замовлення
Вхідні дані	-
Опис проведення тесту	Користувач натискає кнопку «-» навпроти замовлення
Очікуваний результат	Замовлення видаляється зі списку
Стан системи після проведення тесту	Запис про замовлення видаляється із бази даних

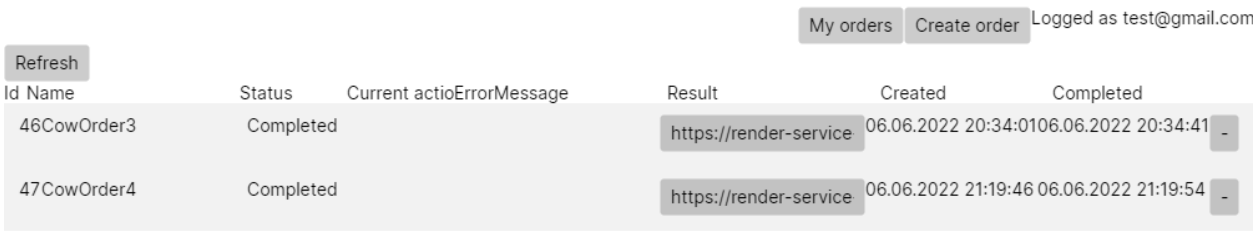


Рисунок 3.10 – Стан системи до тесту 5.1

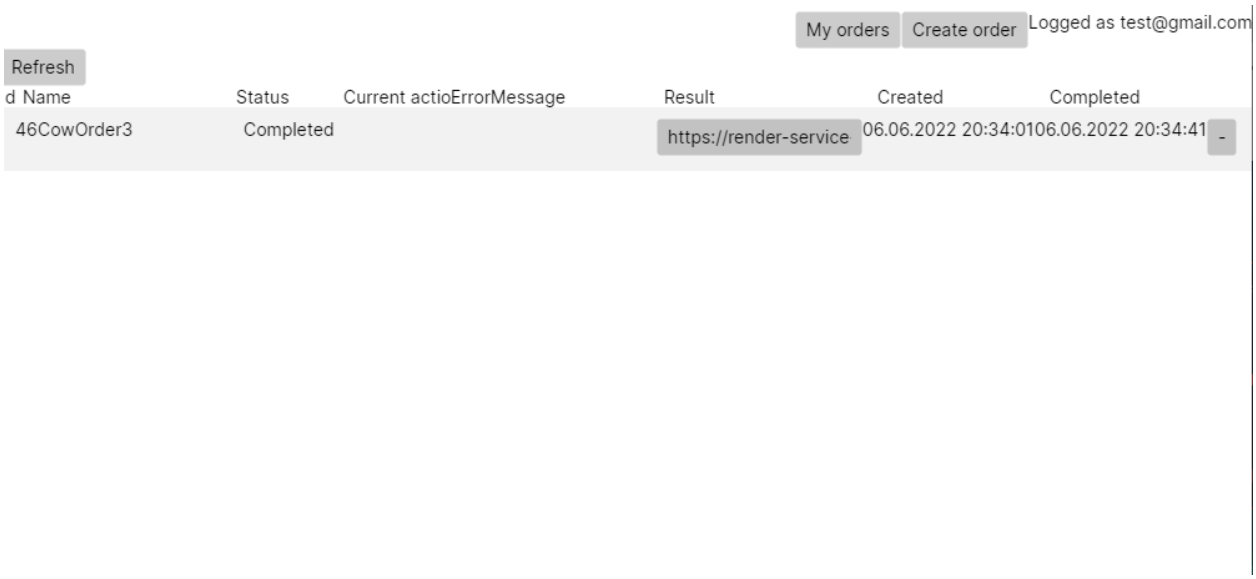


Рисунок 3.11 – Стан системи після тесту 5.1

3.3 Опис контрольного прикладу

Для контрольного прикладу було зареєстровано користувача з поштою test@gmail.com та паролем Abracadabra1_. На рисунку 3.12 зображено реєстрацію нового користувача.

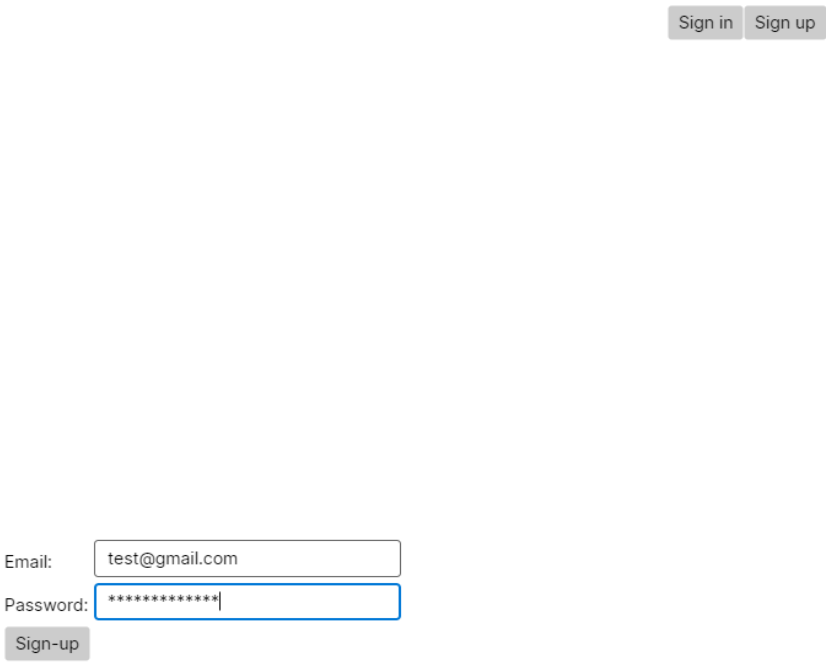


Рисунок 3.12 – реєстрація нового корстувача

Наступним кроком є створення нового замовлення на рендеринг. Для цього необхідно перейти до вкладки «Create order» та ввести параметри для замовлення. На рисунку 3.13 зображено вікно створення нового замовлення на рендеринг.

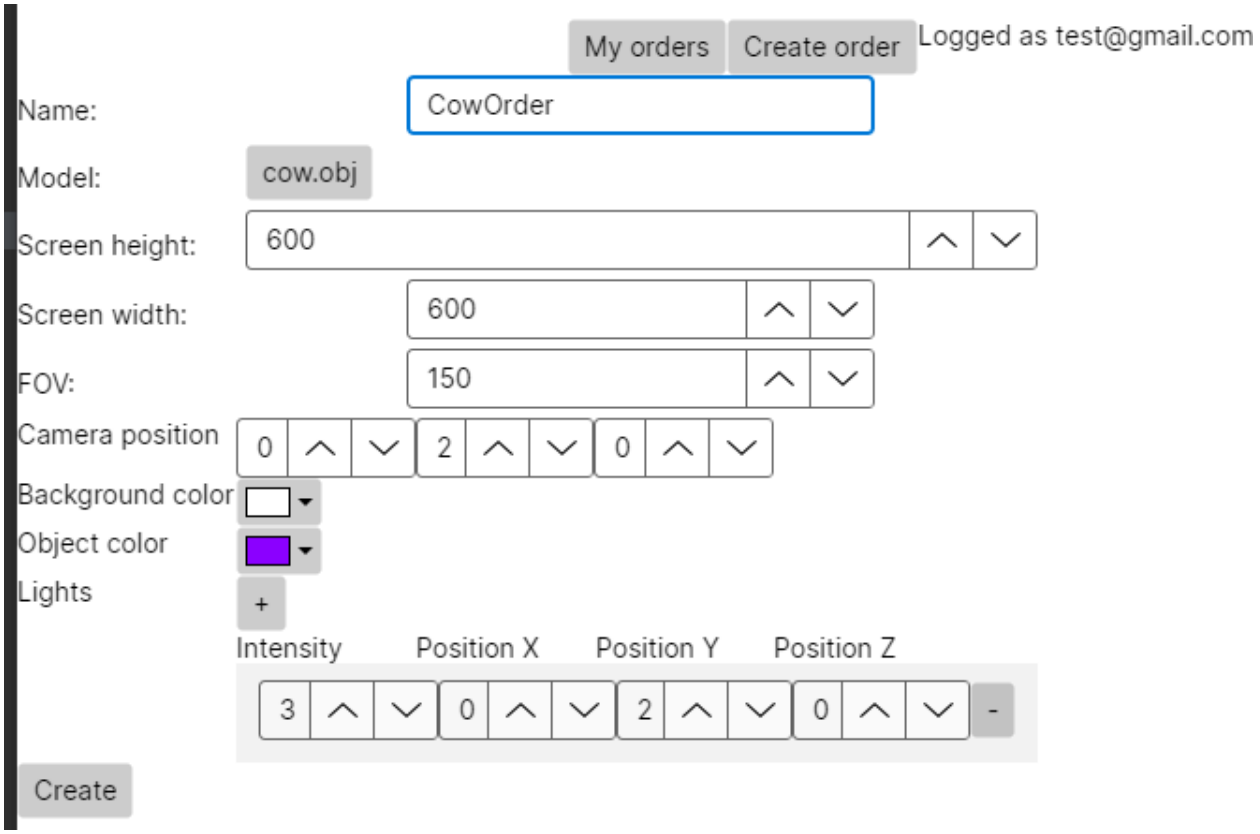


Рисунок 3.13 – створення нового замовлення на рендеринг

У якості об’ємної моделі було обрано файл cow.obj, який являє собою модель корови. На рисунку 3.14 зображено модель cow.obj у додатку Paint 3D.

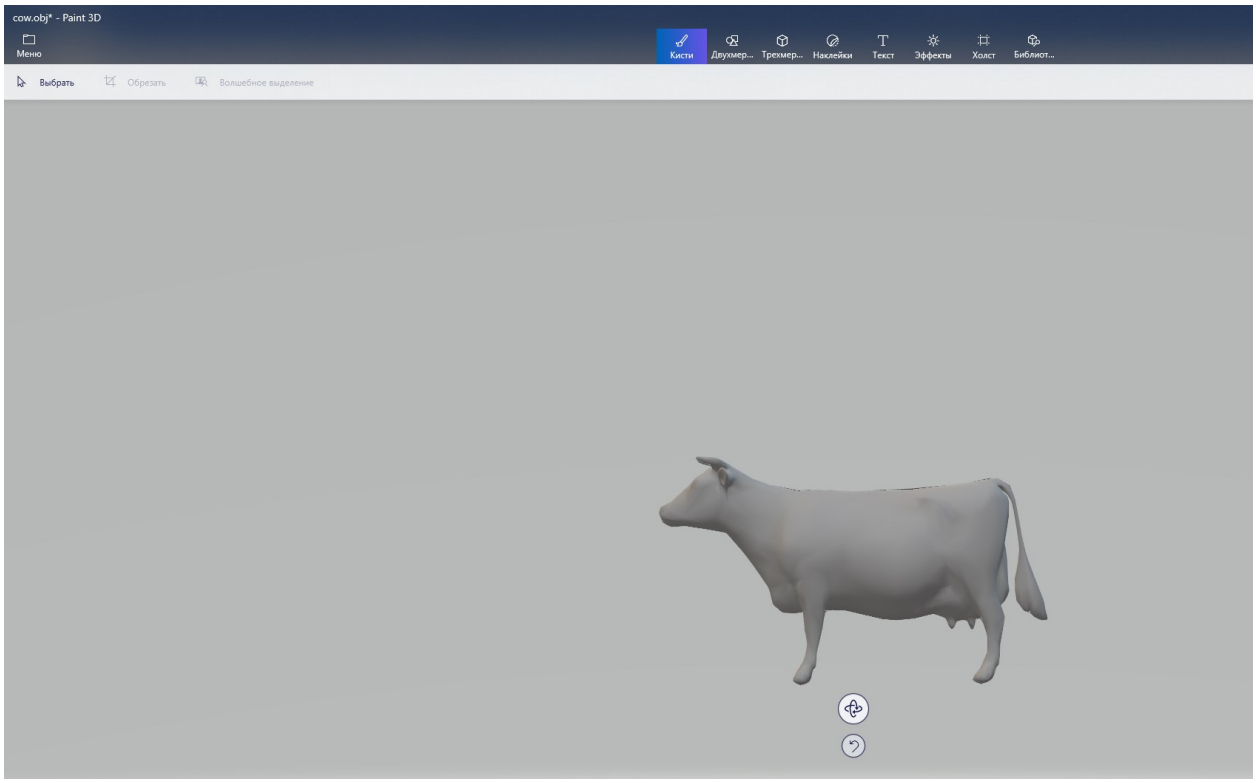


Рисунок 3.14 – вигляд cow.obj у додатку Paint 3D

Нове замовлення можна побачити у вкладці «My orders». У колонці «Status» можна побачити зміну стану замовлення у режимі реального часу. Початковим станом є стан «Created». Коли замовлення починає оброблятись, його стан змінюється на «InProgress», а у колонці «Current action» можна побачити поточний етап виконання. Коли замовлення виконано, його стан змінюється на «Completed» та у колонці «Result» з'являється посилання на результуюче зображення. На рисунках 3.15 – 3.17 зображено зміни стану замовлення.

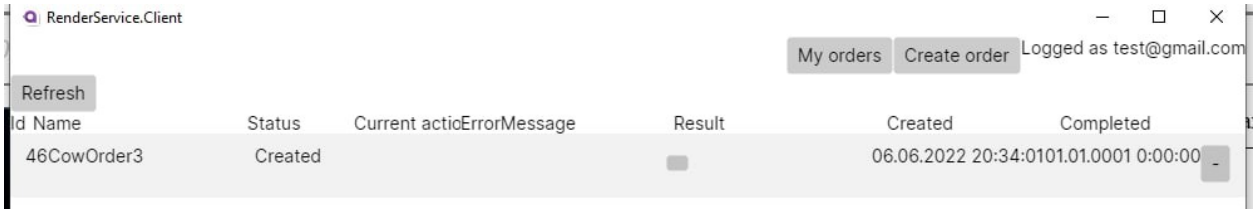


Рисунок 3.15 – початковий стан замовлення

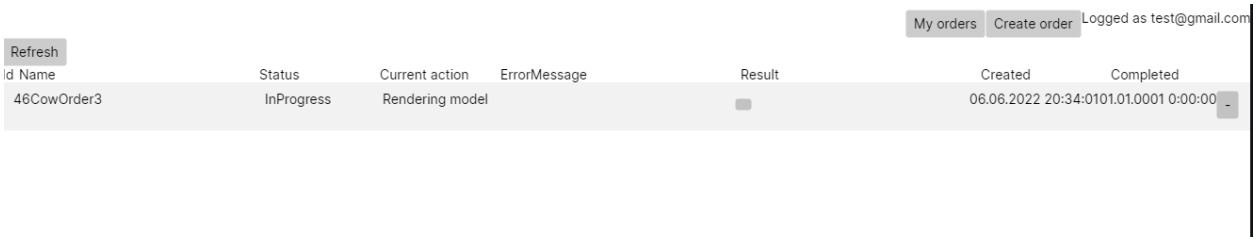


Рисунок 3.16 – стан замовлення змінено на «In progress»

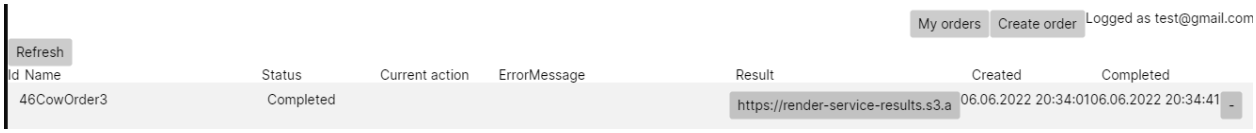


Рисунок 3.17 – стан замовлення змінено на «Completed»

При натисканні на посилання у колонці «Result», відкривається браузер з відповідним посиланням. На рисунку 3.18 зображено результат рендерингу.

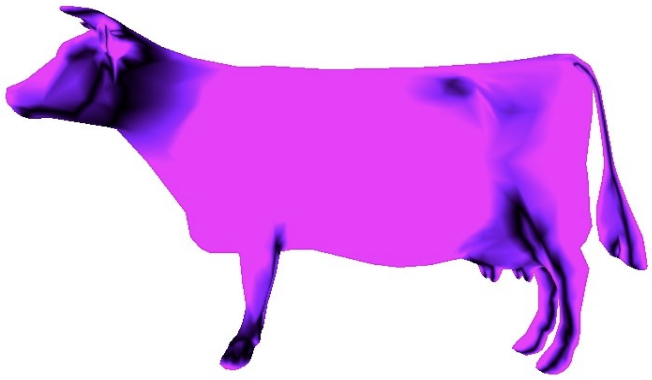


Рисунок 3.18 – результат рендерингу

Висновки до розділу

В рамках даного розділу було проведено тестування усіх сервісів, які складають програмне забезпечення, та описано наступне:

– методологію оцінки якості програмного забезпечення;

– утиліти, що застосовувались для оцінки якості програмного забезпечення;

– методику тестування;

– утиліти для тестування;

– показники тестування;

– основні тест кейси.

До основних тест кейсів належать:

– реєстрація;

– авторизація;

– створення замовлень на рендеринг;

– перегляд замовлень на рендеринг;

– видалення замовлень на рендеринг.

Усі описані основні тест кейси пройшли перевірку з урахуванням сценаріїв, коли користувач вводить некоректні дані.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Розгортання усіх сервісів програмних засобів для тестового середовища було виконано на платформі AWS. В рамках даного проекту було створено наступні ресурси:

– дві бази даних на основі СУБД SQL Server в RDS;
– база даних на основі СУБД Redis в Amazon Memory DB for Redis ;
– брокер повідомлень RabbitMQ у AmazonMQ;
– API шлюз у Amazon API Gateway;
– три віртуальні машини під кожен сервіс у EC2.

Для розгортання програмного забезпечення була використана система CI/CD GitHub Actions. Дана система дозволяє створювати робочі процеси, які виконуються кожен раз, коли у репозиторії виникають певні зміни. В даному випадку такими змінами є коміти, які додаються у головну гілку репозиторію. Робочий процес являє собою послідовність команд, або робіт, які записуються у файл формату .yaml.

На рисунках 4.1 – 4.3 зображено статус виконання останніх збірок для кожного із сервісів.

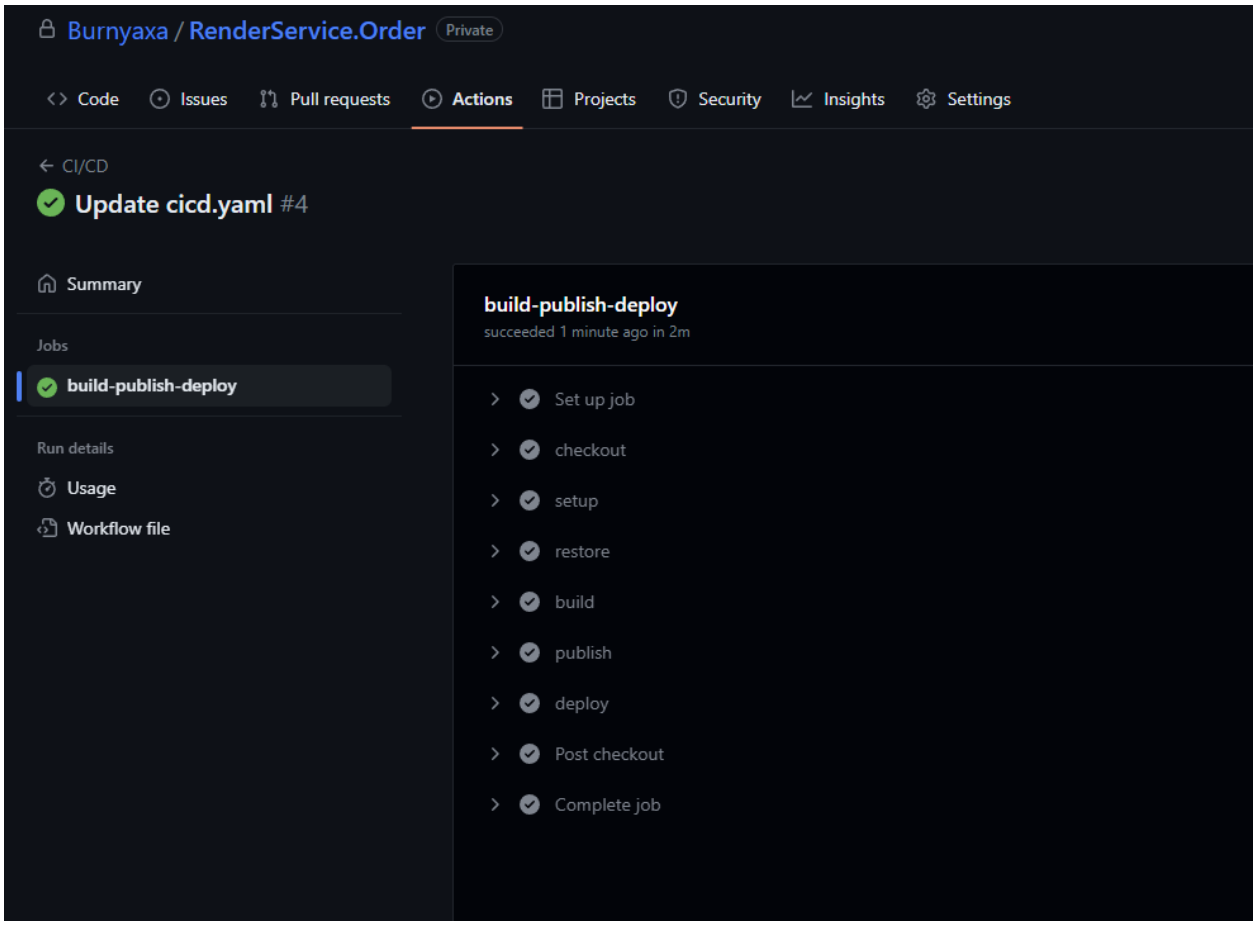


Рисунок 4.1 – Статус виконання останньої зборки для сервісу замовлень

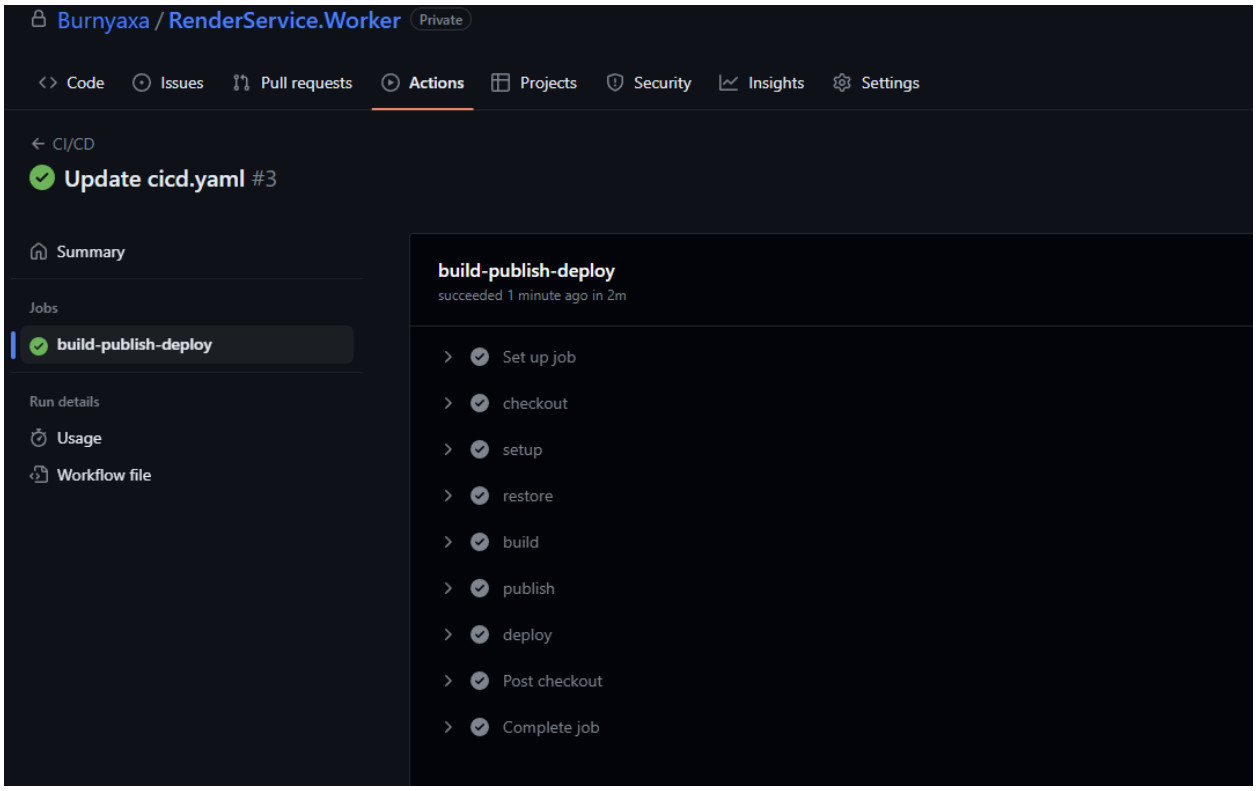


Рисунок 4.2 – Статус викоання останньої зборки для воркеру

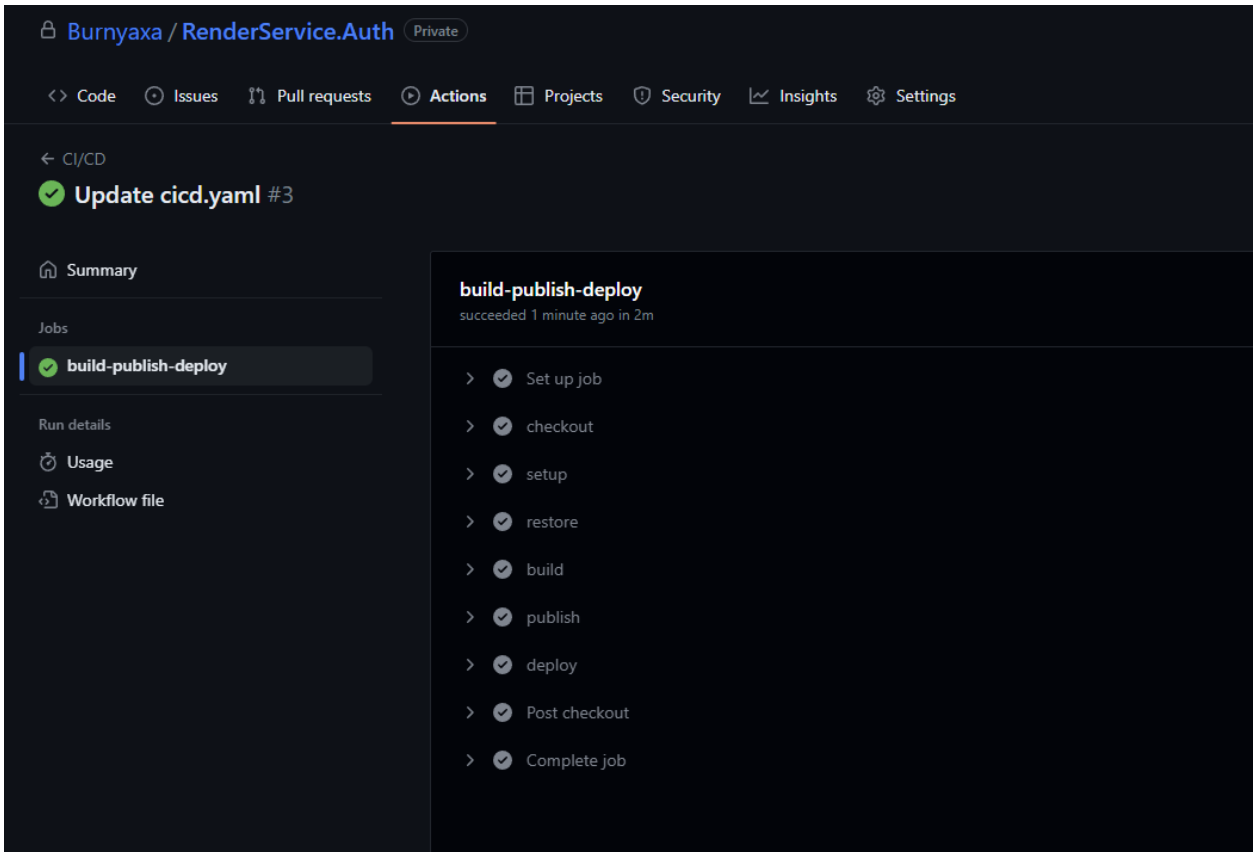


Рисунок 4.3 – Статус виконання останньої зборки для сервісу авторизації

Для кожного репозиторію виконується наступний список команд:

– отримання вихідного коду з гілки master;
– завантаження усіх необхідних пакетів з Nuget;
– збірка рішення;
– збірка docker-образу;
– публікація docker-образу у Docker Hub;
– розгортання на платформі AWS.

Висновки до розділу

В рамках даного розділу було наведено список ресурсів, що було створено під час розгортання проекту на платформі AWS.

Також покроково описано процес неперервної інтеграції та неперервного розгортання на базі GitHub Actions для кожного з мікросервісів.

ВИСНОВКИ

В рамках аналізу вимог до програмного забезпечення було проаналізовано та описано предметну область застосування розроблюваних програмних засобів. Була проведена порівняльна характеристика функціоналу одного з існуючих аналогів та розроблюваного проекту. Для формування списку вимог було описано функціональні та нефункціональні вимоги, а також побудовано діаграму варіантів використання і матрицю трасування. На основі отриманого переліку вимог були сформовані задачі, які необхідно виконати для задоволення усіх висунутих потреб, та перелік технічних рішень, які будуть використовуватись під час розробки.

В рамках моделювання та конструювання програмного забезпечення було побудовано BPMN-діаграму та покроково описано основний процес створення замовлення на рендеринг. Під час вибору архітектури були описані основні архітектурні підходи та вибрано найбільш підходящу для задоволення потреб, а саме мікросервісну архітектуру. На основі вибраної архітектури була побудована схема взаємодії між мікросервісами. Були описані пакети та основні класи, що входять в кожний з мікросервісів. У результаті були отримані три мікросервіси, що зконструйовані з використанням трьохшарової архітектури та мають API, побудоване за принципами REST. Двоє з мікросервісів використовують окремі бази даних, які відповідають 3НФ. Для кожної з баз даних була побудована схема.

Для ручного тестування був розроблений клієнтський додаток. В рамках тестування було покрито увесь функціонал програмних засобів. Усі сценарії тестування було описано у відповідній таблиці.

В рамках локального розгортання серверної частини проекту було описано список створених ресурсів на платформі AWS. Для супроводу програмного забезпечення для кожного мікросервіса було створено окремий git-репозиторій та налаштовано процедуру CI/CD. У результаті кожен

мікросервіс стає доступним у вигляді docker-образу та розгортається на платформі AWS.

Результуючі програмі засоби задовольняють поставленим вимогам та можуть бути використані у зазначених сферах використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Non-Photorealistic Rendering [Електронний ресурс]. – 2001. – Режим доступу: https://www.researchgate.net/publication/236973460_Non-Photorealistic_Rendering

2) Real-time Rendering of Complex Photorealistic Landscapes Using Hybrid Level-of-Detail Approaches [Електронний ресурс]. – 2005. – Режим доступу: https://www.researchgate.net/publication/30012777_Real-time_Rendering_of_Complex_Photorealistic_Landscapes_Using_Hybrid_Level-of-Detail_Approaches

3) What Is The Difference Between Smooth Shading And Flat Shading? [Електронний ресурс]. – 2022. – Режим доступу: https://www.blenderbasecamp.com/home/what-is-the-difference-between-smooth-shading-and-flat-shading/

4) An overview of rendering techniques [Електронний ресурс]. – 1990. – Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/009784939090014O

5) Fast Ray-Triangle Intersections by Coordinate Transformation [Електронний ресурс]. – 2016. – Режим доступу: https://jcgt.org/published/0005/03/03/

6) What is .NET? Introduction and overview [Електронний ресурс]. – 2022. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/core/introduction

7) A tour of the C# language [Електронний ресурс]. – 2022. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

8) Overview of ASP.NET Core [Електронний ресурс]. – 2022. – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0

9) MS SQL SERVER HISTORY AND ADVANTAGES [Електронний ресурс]. – 2014. – Режим доступу: https://bytescout.com/blog/2014/09/ms-sql-server-history-and-advantages.html

10) What is Redis? [Електронний ресурс]. – 2020. – Режим доступу: https://aws.amazon.com/ru/redis/

11) What can RabbitMQ do for you? [Електронний ресурс]. – 2022. – Режим доступу: https://www.rabbitmq.com/features.html

12) Amazon API Gateway [Електронний ресурс]. – 2022. – Режим доступу: https://aws.amazon.com/api-gateway/

13) ReSharper Documentation [Електронний ресурс]. – 2022. – Режим доступу: https://www.jetbrains.com/resharper/documentation/documentation.html

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

UNICHECK

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1015407425

Дата перевірки:
03.06.2023 17:30:03 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
03.06.2023 17:31:38 EEST

ID користувача:
76913

Назва документа: **ІП-81в_Суприган_ПЗ**

Кількість сторінок: **74** Кількість слів: **9270** Кількість символів: **72558** Розмір файлу: **2.79 MB** ID файлу: **1015070952**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

8.02%
Схожість

Найбільша схожість: **2.25%** з джерелом з Бібліотеки (ID файлу: **1008363913**)

3.24% Джерела з Інтернету	139	Сторінка 76
7.99% Джерела з Бібліотеки	233	Сторінка 78

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Підозріле форматування 13 сторінок

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНІ ЗАСОБИ ДЛЯ РЕНДЕРИНГУ ОБ’ЄМНИХ МОДЕЛЕЙ

Текст програми

КПІ.ІП-8132.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Максим ГОЛОВЧЕНКО

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Артем СУПРИГАН

Київ – 2023

Файл EntityBase.cs

```
public abstract class EntityBase : IEntity<int>
{
    public int Id { get; set; }
}
```

Файл UserDbContext.cs

```
public class UserDbContext : DbContext
{
    public DbSet<User> Users { get; set; }
    public DbSet<Role> Roles { get; set; }

    public UserDbContext() : base()
    {
        Database.EnsureCreated();
    }

    public UserDbContext(DbContextOptions<UserDbContext> options) :
base(options)
    {
        Database.EnsureCreated();
    }

    protected override void OnModelCreating(ModelBuilder builder)
    {
        base.OnModelCreating(builder);

        builder.Entity<User>()
            .HasMany(x => x.Roles)
            .WithOne(x => x.User)
            .OnDelete(DeleteBehavior.Cascade)
            .HasForeignKey(x => x.UserId)
            .IsRequired();

        builder.Entity<User>()
            .Property(x => x.CreatedAt)
            .HasDefaultValueSql("getDate()");

        builder.Entity<User>()
            .Property(x => x.LastPasswordReset)
            .HasDefaultValueSql("getDate()");

        builder.Entity<User>()
            .Property(x => x.LastVisit)
            .HasDefaultValueSql("getDate()");

        builder.Entity<Role>()
            .Property(x => x.UserRole)
            .HasDefaultValue(Enum.Roles.User);
    }
}
```

Файл RedisEntity.cs

```
public class RedisEntity : IEntity<RedisKey>
{
    public RedisKey Id { get; set; }
    public RedisValue Value { get; set; }
}
```

Файл User.cs

```
public class User : EntityBase
{
    public string Email { get; set; }
    public string PasswordHash { get; set; }
```

```
    public string PasswordSalt { get; set; }
    public string PasswordVer { get; set; }
    public DateTime CreatedAt { get; set; }
    public DateTime LastVisit { get; set; }
    public DateTime LastEdit { get; set; }
    public DateTime LastPasswordReset { get; set; }
    public IEnumerable<Role> Roles { get; set; }
}
```

Файл IEntity.cs

```
public interface IEntity<TKey> where TKey : IEquatable<TKey>
{
    TKey Id { get; set; }
}
```

Файл IRedisRepository.cs

```
public interface IRedisRepository<TEntity, in TKey>
    where TEntity : IEntity<TKey>
    where TKey : IEquatable<TKey>
{
    Task Delete(TEntity entity);
    Task<TEntity> Update(TEntity entity);
    Task<TEntity> Insert(TEntity entity);
    Task<TEntity> GetByIdAsync(TKey id);
}
```

Файл IRedisUnitOfWork.cs

```
public interface IRedisUnitOfWork : IDisposable
{
    IRedisRepository<RedisEntity, RedisKey> Repository { get; }
}
```

Файл IRepository.cs

```
public interface IRepository<TEntity, in TKey> : IDisposable
    where TEntity : IEntity<TKey>
    where TKey : IEquatable<TKey>
{
    Task Delete(TEntity entity);
    Task<TEntity> Update(TEntity entity);
    Task<TEntity> Insert(TEntity entity);
    Task<TEntity> GetByIdAsync(TKey id);
    IQueryable<TEntity> GetAll();
    Task<IEnumerable<TEntity>> GetAllAsync();
}
```

Файл IUnitOfWork.cs

```
public interface IUnitOfWork : IDisposable
{
    IRepository<User, int> UsersRepository { get; }
    IRepository<Role, int> RolesRepository { get; }

    Task SaveAsync();
}
```

Файл RedisRepository.cs

```
public class RedisRepository : IRedisRepository<RedisEntity, RedisKey>
{
    private readonly IConnectionMultiplexer _redis;

    public RedisRepository(IConnectionMultiplexer redis)
    {
        _redis = redis;
    }
```

```
    public Task Delete(RedisEntity entity)
    {
        var db = _redis.GetDatabase();
        return db.StringGetDeleteAsync(entity.Id);
    }

    public async Task<RedisEntity> Update(RedisEntity entity)
    {
        var db = _redis.GetDatabase();
        await db.StringSetAsync(entity.Id, entity.Value);

        return entity;
    }

    public async Task<RedisEntity> Insert(RedisEntity entity)
    {
        var db = _redis.GetDatabase();
        var res = await db.StringSetAsync(entity.Id, entity.Value, when:
When.NotExists);
        return res ?
            entity :
            throw new EntityAlreadyExistsException($"Pair with key
{entity.Id} already exists");
    }

    public async Task<RedisEntity> GetByIdAsync(RedisKey id)
    {
        var db = _redis.GetDatabase();
        var value = await db.StringGetAsync(new RedisKey(id));
        if (value == default)
        {
            throw new EntityNotFoundException($"Value with key {id} not
found");
        }

        return new RedisEntity
        {
            Id = id,
            Value = value
        };
    }
}
```

Файл Repository.cs

```
public class Repository<TEntity, TKey> : IRepository<TEntity, TKey>
    where TEntity : class, IEntity<TKey>
    where TKey : IEquatable<TKey>
{
    private bool _disposed = false;
    private readonly DbContext _context;
    private readonly DbSet<TEntity> _dbSet;

    public Repository(DbContext context)
    {
        _context = context ?? throw new
ArgumentNullException(nameof(context));
        _dbSet = context.Set<TEntity>();
    }

    public void Dispose()
    {
        Dispose(true);
        GC.SuppressFinalize(this);
    }
```

```
    public Task Delete(TEntity entity)
    {
        if (_context.Entry(entity).State == EntityState.Detached)
        {
            _dbSet.Attach(entity);
        }
        _dbSet.Remove(entity);
        return Task.CompletedTask;
    }

    public Task<TEntity> Update(TEntity entity)
    {
        _dbSet.Attach(entity);
        _context.Entry(entity).State = EntityState.Modified;
        return Task.FromResult(entity);
    }

    public Task<TEntity> Insert(TEntity entity)
    {
        var result = _dbSet.Add(entity).Entity;
        return Task.FromResult<TEntity>(result);
    }

    public async Task<TEntity> GetByIdAsync(TKey id)
    {
        var entities = await GetAll().ToListAsync();
        var result = entities.FirstOrDefault(x => id.Equals(x.Id));
        return result ?? throw new EntityNotFoundException($"Entity with id
{id} not found.");
    }

    public IQueryable<TEntity> GetAll()
    {
        var query = _dbSet.AsQueryable();
        var entityType = query.ElementType;
        var navigations = _context.Model.FindEntityType(entityType)
            ?.GetDerivedTypesInclusive()
            .SelectMany(type => type.GetNavigations())
            .Distinct();
        if (navigations != null)
        {
            query = navigations.Aggregate(query,
                (current, property) => current.Include(property.Name));
        }

        return query;
    }

    public async Task<IEnumerable<TEntity>> GetAllAsync()
    {
        var itemsQueryable = GetAll();
        var itemsList = await itemsQueryable.ToListAsync();
        return itemsList;
    }

    protected virtual void Dispose(bool disposing)
    {
        if (!_disposed)
        {
            if (disposing)
            {
                _context.Dispose();
            }
```

```
        }
        _disposed = true;
    }
}
```

Файл RedisUnitOfWork.cs

```
public class RedisUnitOfWork : IRedisUnitOfWork
{
    private readonly IConnectionMultiplexer _redis;
    private bool _disposed;
    public IRedisRepository<RedisEntity, RedisKey> Repository { get; private
set; }

    public RedisUnitOfWork(IConnectionMultiplexer redis)
    {
        _redis = redis;
        Repository = new RedisRepository(_redis);
    }

    public void Dispose()
    {
        Dispose(true);
        GC.SuppressFinalize(this);
    }

    protected virtual void Dispose(bool disposing)
    {
        if (!_disposed)
        {
            if (disposing)
            {
                _redis.Dispose();
            }
        }
        _disposed = true;
    }
}
```

Файл UnitOfWork.cs

```
public class UnitOfWork : IUnitOfWork
{
    private readonly DbContext _context;
    private bool disposed = false;

    private IRepository<User, int> _usersRepository;
    private IRepository<Role, int> _rolesRepository;

    public IRepository<User, int> UsersRepository => _usersRepository ??= new
Repository<User, int>(_context);
    public IRepository<Role, int> RolesRepository => _rolesRepository ??= new
Repository<Role, int>(_context);

    public UnitOfWork(UserDbContext context)
    {
        _context = context;
    }

    public void Dispose()
    {
        Dispose(true);
        GC.SuppressFinalize(this);
    }

    public Task SaveAsync()
    {
```

```
            return _context.SaveChangesAsync();
        }

        protected virtual void Dispose(bool disposing)
        {
            if (!this.disposed)
            {
                if (disposing)
                {
                    _context.Dispose();
                }
            }
            this.disposed = true;
        }
    }
```

Файл ICrudService.cs

```
public interface ICrudService<TDto, in TKey>
{
    Task<IEnumerable<TDto>> GetAll();
    Task<TDto> GetById(TKey id);
    Task<TDto> Create(TDto item);
    Task<TDto> Update(TDto item);
    Task Delete(TKey id);
}
```

Файл Argon2PasswordService.cs

```
public class Argon2PasswordService : IPasswordService
{
    public string PasswordVer => "1.0";

    public PasswordResult GenerateHash(string password)
    {
        var hash = Argon2.Hash(password);
        var result = new PasswordResult(PasswordVer, hash, string.Empty);
        return result;
    }

    public bool Verify(HashedPassword hashedPassword, string password)
    {
        return Argon2.Verify(hashedPassword.PasswordHash, password);
    }
}
```

Файл AuthorizationService.cs

```
public class AuthorizationService : IAuthorizationService
{
    private readonly ITokenManager _manager;
    private readonly IRefreshTokenService _refreshTokenService;
    private readonly IUserService _userService;
    private readonly IPasswordService _passwordService;

    public AuthorizationService(ITokenManager manager,
        IRefreshTokenService tokenService,
        IUserService userService,
        IPasswordService passwordService)
    {
        _manager = manager;
        _refreshTokenService = tokenService;
        _userService = userService;
        _passwordService = passwordService;
    }

    public async Task<ProfileDto> Authorize(CredentialsDto credentials)
    {
        UserDto user;
```

```
        try
        {
            user = await _userService.GetUserByEmail(credentials.Email);
        }
        catch (EntityNotFoundException)
        {
            throw new WrongCredentialsException($"No such email
{credentials.Email}");
        }

        var isPasswordValid = _passwordService.Verify(new HashedPassword()
        {
            PasswordHash = user.PasswordHash,
            Salt = user.PasswordSalt
        }, credentials.Password);

        if (!isPasswordValid)
        {
            throw new WrongCredentialsException($"Wrong password for email
{credentials.Email}");
        }

        var tokens = _manager.GenerateTokens(new JwtPayload()
        {
            UserId = user.Id,
            Claims = new List<Claim>()
            {
                new(JwtRegisteredClaimNames.Email, user.Email),
                new("id", user.Id.ToString())
            }
        });

        await _refreshTokenService.Update(tokens.RefreshToken);

        var profile = new ProfileDto()
        {
            Token = tokens,
            User = user
        };

        return profile;
    }
}
```

Файл RefreshTokenService.cs

```
public class RefreshTokenService : IRefreshTokenService
{
    private readonly IRedisUnitOfWork _unitOfWork;
    private readonly IMapper _mapper;

    public RefreshTokenService(IRedisUnitOfWork unitOfWork, IMapper mapper)
    {
        _unitOfWork = unitOfWork;
        _mapper = mapper;
    }

    public Task<IEnumerable<RefreshTokenDto>> GetAll()
    {
        throw new NotImplementedException();
    }

    public async Task<RefreshTokenDto> GetById(string id)
    {
        var result = await _unitOfWork.Repository.GetByIdAsync(id);
```

```
        return _mapper.Map<RefreshTokenDto>(result);
    }

    public async Task<RefreshTokenDto> Create(RefreshTokenDto item)
    {
        var entity = _mapper.Map<RedisEntity>(item);
        var result = await _unitOfWork.Repository.Insert(entity);
        return _mapper.Map<RefreshTokenDto>(result);
    }

    public async Task<RefreshTokenDto> Update(RefreshTokenDto item)
    {
        var entity = _mapper.Map<RedisEntity>(item);
        var result = await _unitOfWork.Repository.Update(entity);
        return _mapper.Map<RefreshTokenDto>(result);
    }

    public async Task Delete(string id)
    {
        var item = await GetById(id);
        var entityToDelete = _mapper.Map<RedisEntity>(item);
        await _unitOfWork.Repository.Delete(entityToDelete);
    }
}
```

Файл RegistrationService.cs

```
public class RegistrationService : IRegistrationService
{
    private readonly IUserService _userService;
    private readonly IPasswordService _passwordService;
    private readonly IRefreshTokenService _tokenService;
    private readonly ITokenManager _tokenManager;

    public RegistrationService(IUserService userService,
        IPasswordService passwordService,
        IRefreshTokenService tokenService,
        ITokenManager tokenManager)
    {
        _userService = userService;
        _passwordService = passwordService;
        _tokenService = tokenService;
        _tokenManager = tokenManager;
    }

    public async Task<ProfileDto> RegisterUser(UserDto user)
    {
        var isEmailNotExists = await ValidateUserEmail(user.Email);
        if (!isEmailNotExists)
        {
            throw new UserAlreadyExistsException(user.Email);
        }

        var hashedPassword = _passwordService.GenerateHash(user.Password);
        user.PasswordHash = hashedPassword.PasswordHash;
        user.PasswordVer = hashedPassword.PasswordVer;
        user.PasswordSalt = hashedPassword.PasswordSalt;

        var savedUser = await _userService.Create(user);
        var tokenPayload = new JwtPayload()
        {
            UserId = savedUser.Id,
            Claims = new List<Claim>()
            {
                new (JwtRegisteredClaimNames.Email, user.Email),
```

```
                new ("id", savedUser.Id.ToString())
            }
        };
        var tokens = _tokenManager.GenerateTokens(tokenPayload);

        await _tokenService.Create(tokens.RefreshToken);

        var result = new ProfileDto()
        {
            Token = tokens,
            User = savedUser
        };

        return result;
    }

    private async Task<bool> ValidateUserEmail(string email)
    {
        try
        {
            await _userService.GetUserByEmail(email);
        }
        catch (Exception e) when(e is EmailNotFoundException or
EntitiesNotFoundException)
        {
            return true;
        }

        return false;
    }
}
```

Файл TokenManager.cs

```
public class TokenManager : ITokenManager
{
    private readonly TokenSettings _settings;
    public TokenManager(IOptions<TokenSettings> options)
    {
        _settings = options.Value;
    }

    public TokenPairDto GenerateTokens(JwtPayload payload)
    {
        var jwt = GenerateJwt(payload);
        var refreshToken = GenerateRefreshToken();


        var refreshTokenData = new RefreshTokenData()
        {
            UserId = payload.UserId,
            Expires =
DateTime.Now.AddDays(_settings.RefreshTokenValidityInDays)
        };

        return new TokenPairDto()
        {
            Jwt = jwt,
            RefreshToken = new RefreshTokenDto()
            {
                Token = refreshToken,
                Data = refreshTokenData
            }
        };
    }
```

```
    public string GetClaim(string token, string claimType)
    {
        var tokenHandler = new JwtSecurityTokenHandler();
        var securityToken = tokenHandler.ReadJwtToken(token);
        var stringClaimValue = securityToken.Claims.First(claim => claim.Type
== claimType).Value;
        return stringClaimValue;
    }

    public bool TryValidateJwt(string jwt, out SecurityToken? validatedToken)
    {
        validatedToken = default;

        var symmetricKey =  new
SymmetricSecurityKey(Encoding.ASCII.GetBytes(_settings.Secret));

        var tokenHandler = new JwtSecurityTokenHandler();

        try
        {
            tokenHandler.ValidateToken(jwt, new TokenValidationParameters()
            {
                ValidateIssuerSigningKey = true,
                ValidateIssuer = true,
                ValidateAudience = true,
                ValidIssuer = _settings.Issuer,
                ValidAudience = _settings.Audience,
                IssuerSigningKey = symmetricKey
            }, out validatedToken);

            return true;
        }
        catch
        {
            return false;
        }
    }

    private string GenerateJwt(JwtPayload payload)
    {
        var symmetricKey =  new
SymmetricSecurityKey(Encoding.ASCII.GetBytes(_settings.Secret));

        var tokenHandler = new JwtSecurityTokenHandler();

        var tokenDescriptor = new SecurityTokenDescriptor()
        {
            Subject = new ClaimsIdentity(payload.Claims),
            Expires =
DateTime.Now.AddSeconds(_settings.JwtValidityInSeconds),
            Issuer = _settings.Issuer,
            Audience = _settings.Audience,
            SigningCredentials = new SigningCredentials(symmetricKey,
SecurityAlgorithms.HmacSha256Signature)
        };

        var token = tokenHandler.CreateToken(tokenDescriptor);
        return tokenHandler.WriteToken(token);
    }

    private string GenerateRefreshToken()
    {
        var randomNumber = new byte[64];
```

```
        using var rng = RandomNumberGenerator.Create();
        rng.GetBytes(randomNumber);
        return Convert.ToBase64String(randomNumber);
    }
}
```

Файл TokenService.cs

```
public class TokenService : ITokenService
{
    private readonly ITokenManager _tokenManager;
    private readonly IRefreshTokenService _refreshTokenService;
    private readonly IUserService _userService;

    public TokenService(ITokenManager tokenManager, IRefreshTokenService
service, IUserService userService)
    {
        _tokenManager = tokenManager;
        _refreshTokenService = service;
        _userService = userService;
    }

    public async Task<ProfileDto> RefreshAccessToken(RefreshTokenDto token)
    {
        RefreshTokenDto tokenDto;
        UserDto userDto;

        try
        {
            tokenDto = await _refreshTokenService.GetById(token.Token);
        }
        catch (EntityNotFoundException)
        {
            throw new RefreshTokenNotFoundException(token.Token);
        }

        if (tokenDto.IsExpired)
        {
            throw new RefreshTokenExpiredException(tokenDto.Token);
        }
        try
        {
            userDto = await _userService.GetById(tokenDto.Data.UserId);
        }
        catch (EntityNotFoundException)
        {
            await _refreshTokenService.Delete(token.Token);
            throw new
UserWasDeletedException(tokenDto.Data.UserId.ToString());
        }

        var newTokens = _tokenManager.GenerateTokens(new JwtPayload()
        {
            UserId = userDto.Id,
            Claims = new List<Claim>()
            {
                new(JwtRegisteredClaimNames.Email, userDto.Email)
            }
        });


        await _refreshTokenService.Update(newTokens.RefreshToken);

        return new ProfileDto()
        {
```

```
            Token = newTokens,
            User = userDto
        };
    }
}
```

Файл UserService.cs

```
public class UserService : IUserService
{
    private readonly IUnitOfWork _unitOfWork;
    private readonly IMapper _mapper;

    public UserService(IUnitOfWork unitOfWork, IMapper mapper)
    {
        _unitOfWork = unitOfWork;
        _mapper = mapper;
    }

    public Task<IEnumerable<UserDto>> GetAll()
    {
        var users = _unitOfWork.UsersRepository.GetAll().ToList();
        if (users.Count == 0)
        {
            throw new EntitiesNotFoundException(nameof(users));
        }
        return Task.FromResult(_mapper.Map<IEnumerable<UserDto>>(users));
    }

    public async Task<UserDto> GetById(int id)
    {
        var user = await _unitOfWork.UsersRepository.GetByIdAsync(id);
        if (user == null)
        {
            throw new EntityNotFoundException(nameof(user), id);
        }

        return _mapper.Map<UserDto>(user);
    }

    public async Task<UserDto> Create(UserDto item)
    {
        var user = _mapper.Map<User>(item);
        var result = await _unitOfWork.UsersRepository.Insert(user);
        await _unitOfWork.SaveAsync();
        return _mapper.Map<UserDto>(result);
    }

    public async Task<UserDto> Update(UserDto item)
    {
        var user = await _unitOfWork.UsersRepository.GetByIdAsync(item.Id);
        if (user == null)
        {
            throw new EntityNotFoundException(nameof(user), item.Id);
        }

        user.Email = item.Email;
        user.PasswordHash = item.PasswordHash;
        user.PasswordSalt = item.PasswordSalt;
        user.PasswordVer = item.PasswordVer;
        user.LastPasswordReset = item.LastPasswordReset;
        user.LastVisit = item.LastVisit;
        user.LastEdit = DateTime.Now;

        var result = await _unitOfWork.UsersRepository.Update(user);
```

```
        await _unitOfWork.SaveAsync();

        return _mapper.Map<UserDto>(result);
    }

    public async Task Delete(int id)
    {
        var user = await _unitOfWork.UsersRepository.GetByIdAsync(id);
        if (user == null)
        {
            throw new EntityNotFoundException(nameof(user), id);
        }

        await _unitOfWork.UsersRepository.Delete(user);
        await _unitOfWork.SaveAsync();
    }

    public async Task<UserDto> GetUserByEmail(string email)
    {
        var allUsers = await GetAll();
        var user = allUsers.FirstOrDefault(x => x.Email == email);

        return user ?? throw new EmailNotFoundException(nameof(user), email);
    }
}
```

Файл ExceptionHandlerMiddleware.cs

```
public class ExceptionHandlerMiddleware : IMiddleware
{
    private readonly ILogger _logger;

    public ExceptionHandlerMiddleware(ILogger<ExceptionHandlerMiddleware>
logger)
    {
        _logger = logger;
    }

    public async Task InvokeAsync(HttpContext context, RequestDelegate next)
    {
        try
        {
            await next(context);
        }
        catch (Exception ex)
        {
            var message = CreateMessage(context, ex);
            _logger.LogError(message, ex);

            await HandleExceptionAsync(context, ex);
        }
    }

    private async Task HandleExceptionAsync(HttpContext context, Exception e)
    {
        var result = new ErrorModel { Message = e.Message };
        int statusCode;

        switch (e)
        {
            case NotFoundException:
                statusCode = StatusCodes.Status404NotFound;
                break;
            case BadRequestException:
                statusCode = StatusCodes.Status400BadRequest;
```

```
                break;
            case ForbiddenException:
                statusCode = StatusCodes.Status403Forbidden;
                break;
            case UnauthorizedException:
                statusCode = StatusCodes.Status401Unauthorized;
                if (e is WrongCredentialsException)
                {
                    result.Message = "Invalid email or password";
                }
                break;
            default:
                statusCode = StatusCodes.Status500InternalServerError;
                result.Message = "Unknown error, please contact the system
administrator";
                break;
        }

        _logger.LogError(e, e.Message);
        var response = JsonConvert.SerializeObject(result,
Formatting.Indented,
            new JsonSerializerSettings
            {
                NullValueHandling = NullValueHandling.Ignore
            });

        context.Response.StatusCode = statusCode;
        await context.Response.WriteAsync(response);
    }

    private string CreateMessage(HttpContext context, Exception e)
    {
        var message = $"Exception caught in error handler middleware,
exception message: {e.Message}, stack: {e.StackTrace}";

        if (e.InnerException != null)
        {
            message = $"{message}, inner message {e.InnerException.Message},
inner stack {e.InnerException.StackTrace}";
        }

        return $"{message} RequestId: {context.TraceIdentifier}";
    }
```

Файл S3ClientExtensions.cs

```
public static class S3ClientExtensions
{
    public static string GetResourceUrl(this IAmazonS3 client, string bucket,
string key)
    {
        return @$"https://{bucket}.s3.amazonaws.com/{key}";
    }

    public static string GetKey(this IAmazonS3 client, string bucket, string
url)
    {
        return new
string(url.Skip(@$"https://{bucket}.s3.amazonaws.com/".Length).ToArray());
    }
}
```

Файл OrderStatusListener.cs

```
public class OrdersStatusListener : RabbitListener
{
    private readonly IServiceProvider _services;
```

```
    private readonly ILogger<RabbitListener> _logger;
    private readonly IServerSentEventsService _sse;
    public OrdersStatusListener(IOptions<RabbitMqSettings> settings,
        IOptions<RabbitMqQueuesSettings> queueSettings,
        IServiceProvider services,
        ILogger<RabbitListener> logger, IServerSentEventsService sse) :
base(settings)
    {
        _services = services;
        _logger = logger;
        _sse = sse;
        _queue = queueSettings.Value.OrdersStatusQueue;
    }

    public override async Task<bool> ProcessMessage(string message)
    {
        try
        {
            var order = JsonConvert.DeserializeObject<OrderMessage>(message);
            using var scope = _services.CreateScope();
            var ordersService =
scope.ServiceProvider.GetRequiredService<IRenderOrdersService>();
            var mapper = scope.ServiceProvider.GetRequiredService<IMapper>();
            _logger.LogInformation(1, $"Sse message {message}");
            await ordersService.Update(mapper.Map<RenderOrderDto>(order));

            await _sse.SendEventAsync(order.Id.ToString(), message);
        }
        catch (Exception e)
        {
            _logger.LogError(-1, e.Message);
            return false;
        }
        return true;
    }
}
```

Файл RabbitProducer.cs

```
public abstract class RabbitProducer : IRabbitProducer
{
    protected string Queue;
    private readonly IModel _channel;
    private readonly ILogger<RabbitProducer> _logger;
    public RabbitProducer(IOptions<RabbitMqSettings> settings,
ILogger<RabbitProducer> logger)
    {
        _logger = logger;
        var factory = new ConnectionFactory()
        {
            HostName = settings.Value.HostName,
            UserName = settings.Value.UserName,
            Password = settings.Value.Password,
            Port = settings.Value.Port
        };

        var connection = factory.CreateConnection();
        _channel = connection.CreateModel();
    }

    public virtual void PushMessage(object message)
    {
        _channel.QueueDeclare(Queue, true, false, false);
        var jsonObject = JsonConvert.SerializeObject(message, new
JsonSerializerSettings()
```

```
        {
            ReferenceLoopHandling = ReferenceLoopHandling.Ignore
        });
        var body = Encoding.UTF8.GetBytes(jsonObject);
        _channel.BasicPublish(string.Empty, Queue, null, body);
        _logger.LogInformation(1, $"Pushed {jsonObject} to queue {Queue}");
    }
}
```

Файл RabbitListener.cs

```
public abstract class RabbitListener : IHostedService
{
    private readonly IConnection _connection;
    private readonly IModel _channel;
    protected string _queue;

    public RabbitListener(IOptions<RabbitMqSettings> settings)
    {
        var factory = new ConnectionFactory()
        {
            HostName = settings.Value.HostName,
            UserName = settings.Value.UserName,
            Password = settings.Value.Password,
            Port = settings.Value.Port,
            DispatchConsumersAsync = true
        };

        _connection = factory.CreateConnection();
        _channel = _connection.CreateModel();
    }
    public Task StartAsync(CancellationToken cancellationToken)
    {
        _channel.QueueDeclare(_queue, true, false, false);
        var consumer = new AsyncEventingBasicConsumer(_channel);
        consumer.Received += async (_, eventArgs) =>
        {
            var message = Encoding.UTF8.GetString(eventArgs.Body.Span);
            var result = await ProcessMessage(message);
            if (result)
            {
                _channel.BasicAck(eventArgs.DeliveryTag, false);
            }
        };

        _channel.BasicConsume(_queue, false, consumer);
        return Task.CompletedTask;
    }

    public abstract Task<bool> ProcessMessage(string message);

    public Task StopAsync(CancellationToken cancellationToken)
    {
        _connection.Close();
        return Task.CompletedTask;
    }
}
```

Файл CloudStorageService.cs

```
public class CloudStorageService : ICloudStorageService
{
    private readonly RegionEndpoint _bucketRegion =
RegionEndpoint.EUCentral1;
    private readonly string _bucket;
    private readonly IAmazonS3 _s3Client;
```

```
    public CloudStorageService(IOptions<AmazonSettings> options)
    {
        _bucket = options.Value.ModelsBucket;
        _s3Client = new AmazonS3Client(options.Value.AwsAccessKeyId,
options.Value.AwsSecretKey, _bucketRegion);
    }

    public async Task<string> UploadFileAsync(FileDto file)
    {
        var key = "models/" + DateTime.Now + "--" + file.Name;
        var fileTransferUtility = new TransferUtility(_s3Client);
        var uploadRequest = new TransferUtilityUploadRequest()
        {
            InputStream = file.File,
            Key = key,
            BucketName = _bucket,
            AutoCloseStream = true
        };

        await fileTransferUtility.UploadAsync(uploadRequest);
        return _s3Client.GetResourceUrl(_bucket, key);
    }

    public Task DeleteFileAsync(string link)
    {
        var key = _s3Client.GetKey(_bucket, link);
        return _s3Client.DeleteObjectAsync(_bucket, key);
    }

    public string DownloadFile(string link)
    {
        var key = _s3Client.GetKey(_bucket, link);

        var request = new GetPreSignedUrlRequest()
        {
            BucketName = _bucket,
            Key = key,
            Expires = DateTime.Now.AddDays(1)
        };

        return _s3Client.GetPreSignedURL(request);
    }
}
```

Файл RenderOrdersService.cs

```
public class RenderOrdersService : IRenderOrdersService
{
    private readonly IMapper _mapper;
    private readonly IUnitOfWork _unitOfWork;
    private readonly ICloudStorageService _cloudStorageService;
    private readonly OrdersProducer _producer;
    public RenderOrdersService(IMapper mapper, IUnitOfWork unitOfWork,
ICloudStorageService cloudStorageService, OrdersProducer producer)
    {
        _mapper = mapper;
        _unitOfWork = unitOfWork;
        _cloudStorageService = cloudStorageService;
        _producer = producer;
    }

    public async Task<IEnumerable<RenderOrderDto>> GetAll()
    {
        var orders = await _unitOfWork.OrdersRepository.GetAllAsync();
        var ordersList = orders.ToList();
```

```
        if (ordersList.Count == 0)
        {
            throw new EntityNotFoundException("Orders not found");
        }

        return _mapper.Map<IEnumerable<RenderOrderDto>>(ordersList);
    }

    public async Task<RenderOrderDto> GetById(int id)
    {
        var order = await _unitOfWork.OrdersRepository.GetByIdAsync(id);
        if (order == null)
        {
            throw new EntityNotFoundException($"Order with id {id} not
found");
        }

        return _mapper.Map<RenderOrderDto>(order);
    }

    public async Task<RenderOrderDto> Create(RenderOrderDto item)
    {
        var linkToModel = await
_cloudStorageService.UploadFileAsync(item.ModelFile);
        item.Model = linkToModel;
        var order = _mapper.Map<RenderOrder>(item);
        var result = await _unitOfWork.OrdersRepository.Insert(order);
        await _unitOfWork.SaveAsync();
        var dto = _mapper.Map<RenderOrderDto>(result);
        _producer.PushMessage(_mapper.Map<OrderMessage>(dto));
        return dto;
    }

    public async Task<RenderOrderDto> Update(RenderOrderDto item)
    {
        var order = await _unitOfWork.OrdersRepository.GetByIdAsync(item.Id);
        if (order == null)
        {
            throw new EntityNotFoundException($"Order with id {item.Id} not
found");
        }

        order.Name = item.Name;
        order.Completed = item.Completed;
        order.FaultMessage = item.FaultMessage;
        order.CurrentAction = item.CurrentAction;
        order.Result = item.Result;
        order.Status = item.Status;

        var result = await _unitOfWork.OrdersRepository.Update(order);
        await _unitOfWork.SaveAsync();

        return _mapper.Map<RenderOrderDto>(result);
    }

    public async Task Delete(int id)
    {
        var order = await _unitOfWork.OrdersRepository.GetByIdAsync(id);
        if (order == null)
        {
            throw new EntityNotFoundException($"Order with id {id} not
found");
        }
```

```
        await _cloudStorageService.DeleteFileAsync(order.Model);
        await _unitOfWork.OrdersRepository.Delete(order);
        await _unitOfWork.SaveAsync();
    }

    public async Task<IEnumerable<RenderOrderDto>>
GetRenderOrdersByUserId(int userId)
    {
        var allOrders = await _unitOfWork.OrdersRepository.GetAllAsync();
        var allOrdersList = allOrders.ToList();

        var userOrders = allOrdersList
            .Where(x => x.UserId == userId)
            .ToList();

        if (userOrders.Count == 0)
        {
            throw new EntityNotFoundException($"Orders with user id {userId}
not found");
        }

        return _mapper.Map<IEnumerable<RenderOrderDto>>(userOrders);
    }
}
```

Файл JwtReader.cs

```
public static class JwtReader
{
    public static string ReadHeader(string header)
    {
        var token = header.Replace("Bearer ", "").Trim();
        return token;
    }

    public static string GetClaim(string token, string claimType)
    {
        var tokenHandler = new JwtSecurityTokenHandler();
        var securityToken = tokenHandler.ReadJwtToken(token);
        var stringClaimValue = securityToken.Claims.First(claim => claim.Type
== claimType).Value;
        return stringClaimValue;
    }
}
```

Файл SecureRequestMiddleware.cs

```
public class SecureRequestMiddleware : IMiddleware
{
    private readonly ILogger<SecureRequestMiddleware> _logger;
    private readonly HttpClient _client;


    public SecureRequestMiddleware(ILogger<SecureRequestMiddleware> logger,
IHttpClientFactory factory)
    {
        _logger = logger;
        _client = factory.CreateClient("AuthClient");
    }

    public async Task InvokeAsync(HttpContext context, RequestDelegate next)
    {
        var hasHeader = context.Request.Headers.TryGetValue("Authorization",
out var token);
        if (!hasHeader)
```

```
        {
            _logger.LogError(-1, "Unauthorized request");
            context.Response.StatusCode = 401; ;
            await context.Response.WriteAsync("Missing Authorization
header");
            return;
        }

        var request = new HttpRequestMessage(HttpMethod.Head,
"api/v1/token/verify");
        request.Headers.Add("Authorization", token.ToString());
        var response = await _client.SendAsync(request);
        if (!response.IsSuccessStatusCode)
        {
            _logger.LogError(-1, $"Invalid token {token}");
            context.Response.StatusCode = 401;
            await context.Response.WriteAsync("Invalid token");
            return;
        }

        await next.Invoke(context);
    }
}
```

Файл Triangle.cs

```
public class Triangle
{
    public Vector3 A { get; set; }
    public Vector3 B { get; set; }
    public Vector3 C { get; set; }

    public Vector3 N0 { get; set; }
    public Vector3 N1 { get; set; }
    public Vector3 N2 { get; set; }



    public Vector3 Center => new Vector3((A.X + B.X + C.X) / 3, (A.Y + B.Y +
C.Y) / 3, (A.Z + B.Z + C.Z) / 3);

    public Triangle(Vector3 a, Vector3 b, Vector3 c, Vector3 n0, Vector3 n1,
Vector3 n2)
    {
        A = a;
        B = b;
        C = c;
        N0 = n0;
        N1 = n1;
        N2 = n2;
    }

    public Vector3 NormVector()
    {
        var first = B - A;
        var second = C - A;
        var cross = Vector3.Cross(first, second);
        return Vector3.Normalize(cross);
    }

    public float Dot(Vector3 dir)
    {
        var first = B - A;
        var second = C - A;
        var cross = Vector3.Cross(dir, second);
```

```
            return Vector3.Dot(first, cross);
        }

        public static bool operator ==(Triangle first, Triangle second)
        {
            return first.A == second.A && first.B == second.B && first.C ==
second.C;
        }

        public static bool operator !=(Triangle first, Triangle second)
        {
            return !(first == second);
        }

        public override bool Equals(object? obj)
        {
            return base.Equals(obj);
        }

        public override int GetHashCode()
        {
            return Center.GetHashCode();
        }
}
```

Файл ObjModelReader.cs

```
public class ObjModelReader : IModelReader
{
    private readonly ILogger<ObjModelReader> _logger;

    public ObjModelReader(ILogger<ObjModelReader> logger)
    {
        _logger = logger;
    }

    public IEnumerable<Triangle> ReadModel(Stream modelStream)
    {
        var fileLines = new List<string>();
        var vertexes = new List<Vector3>();
        var normals = new List<Vector3>();
        var faces = new List<List<Vertex>>();

        try
        {
            using var reader = new StreamReader(modelStream);
            fileLines.AddRange(reader.ReadAllLines());

            foreach (var str in fileLines)
            {
                var line = str.Split(" ",
StringSplitOptions.RemoveEmptyEntries);

                if (line.Length == 0)
                {
                    continue;
                }

                switch (line[0])
                {
                    case "v":
                        vertexes.Add(ProcessVertex(line));
                        break;
                    case "vn":
                        normals.Add(ProcessNormal(line));
```

```
                        break;
                    case "f":
                        faces.Add(ProcessFace(line));
                        break;
                }
            }

            return faces
                .Select(face => new Triangle(
                    vertexes[face[0].V],
                    vertexes[face[1].V],
                    vertexes[face[2].V],
                    normals[face[0].Vn],
                    normals[face[1].Vn],
                    normals[face[2].Vn]))
                .ToList();
        }
        catch(Exception e)
        {
            _logger.LogError(-1, e.Message);
            throw new ArgumentException("Unable to read the model");
        }
    }

    private Vector3 ProcessVertex(IReadOnlyList<string> lines)
    {
        return new Vector3(
            float.Parse(lines[1], CultureInfo.InvariantCulture.NumberFormat),
            float.Parse(lines[2], CultureInfo.InvariantCulture.NumberFormat),
            float.Parse(lines[3],
CultureInfo.InvariantCulture.NumberFormat));
    }

    private Vector3 ProcessNormal(IReadOnlyList<string> lines)
    {
        return new Vector3(
            float.Parse(lines[1], CultureInfo.InvariantCulture.NumberFormat),
            float.Parse(lines[2], CultureInfo.InvariantCulture.NumberFormat),
            float.Parse(lines[3],
CultureInfo.InvariantCulture.NumberFormat));
    }

    private List<Vertex> ProcessFace(IEnumerable<string> lines)
    {
        var temp = new List<Vertex>();
        foreach (var line in lines.Skip(1).ToArray())
        {
            var vertices = line.Split('/');
            var vertex = new Vertex()
            {
                V = int.Parse(vertices[0],
CultureInfo.InvariantCulture.NumberFormat) - 1,
                Vn = int.Parse(vertices[2],
CultureInfo.InvariantCulture.NumberFormat) - 1
            };
            temp.Add(vertex);
        }
        return temp;
    }
}
```

Файл Renderer.cs

```
public class Renderer : IRenderer
{
```

```
    private const float Epsilon = 0.0000001f;
    private readonly ITree _tree;

    public Renderer(ITree tree)
    {
        _tree = tree;
    }

    public Vector3[,] Render(List<Triangle> triangles, RenderOrderDto order)
    {
        if (triangles.Count == 0)
        {
            throw new ArgumentException("Invalid model");
        }
        var result = Initialize(order);

        var listA = triangles.Select(x => x.A);
        var listB = triangles.Select(x => x.B);
        var listC = triangles.Select(x => x.C);
        var max = listA.Concat(listB)
            .Concat(listC)
            .Max(x => new[]{Math.Abs(x.X), Math.Abs(x.Y),
Math.Abs(x.Z)}.Max());

        _tree.Initialize(max, triangles);

        Parallel.For(0, order.OrderSettings.ScreenSettings.Height, x =>
        {
            Parallel.For(0, order.OrderSettings.ScreenSettings.Width, y =>
            {
                ProcessPixel(x, y);
                ProcessColor(ref result[x, y].X);
                ProcessColor(ref result[x, y].Y);
                ProcessColor(ref result[x, y].Z);
            });
        });

        var resultReversed = new
Vector3[order.OrderSettings.ScreenSettings.Height,
order.OrderSettings.ScreenSettings.Width];

        for (var i = 0; i < order.OrderSettings.ScreenSettings.Height; i++)
        {
            for (var j = 0; j < order.OrderSettings.ScreenSettings.Width;
j++)
            {
                resultReversed[i, j] =
result[order.OrderSettings.ScreenSettings.Height - i - 1,
order.OrderSettings.ScreenSettings.Width - j - 1];
            }
        }

        return resultReversed;

        void ProcessPixel(int height, int width)
        {
            var minDistance = float.MaxValue;
            var currentColor =
order.OrderSettings.BackgroundColorSettings.Color.ToVector();
            var resultTriangles = new List<Triangle>();
            var direction = GetCameraDirection(height, width,
order.OrderSettings.ScreenSettings);

_tree.FindIntersections(order.OrderSettings.CameraPositionSettings.Position,
```

```
direction, resultTriangles);

            foreach (var triangle in resultTriangles)
            {
                CastRay(order.OrderSettings.CameraPositionSettings.Position,
                    direction,
                    triangle,
                    order.OrderSettings.LightSettings.Select(x =>
x.Light).ToList(),
                    out var hit,
                    out var color);

                var distance = Vector3.Distance(hit,
order.OrderSettings.CameraPositionSettings.Position);

                if (minDistance > distance && color !=
order.OrderSettings.BackgroundColorSettings.Color.ToVector())
                {
                    minDistance = distance;
                    currentColor = color;
                }
            }

            result[height, width] = currentColor;
        }

        void CastRay(Vector3 vector, Vector3 direction, Triangle triangle,
List<Light> lights, out Vector3 hit,
            out Vector3 color)
        {
            var intersecting = ProcessScene(vector, direction, triangle, out
hit, out var intersectNormal);

            if (!intersecting)
            {
                color =
order.OrderSettings.BackgroundColorSettings.Color.ToVector();
                return;
            }

            var normal = Vector3.Dot(direction, intersectNormal) < 0 ? -
intersectNormal : intersectNormal;

            var diffuseLightIntensity = 0f;
            var reverseDiffuseLightIntensity = 0f;

            foreach (var light in lights)
            {
                var lightDirection = light.Position - hit;

                var square = Vector3.Dot(lightDirection, lightDirection);

                var distance = (float)Math.Sqrt(square);

                lightDirection.X /= distance;
                lightDirection.Y /= distance;
                lightDirection.Z /= distance;

                diffuseLightIntensity += light.Intensity * Math.Max(0,
Vector3.Dot(normal, lightDirection));
                reverseDiffuseLightIntensity +=
                    light.Intensity * Math.Max(0, Vector3.Dot(normal, -
lightDirection));
            }
```

```
            color =  order.OrderSettings.ObjectColorSettings.Color.ToVector()
* Math.Max(diffuseLightIntensity, reverseDiffuseLightIntensity);
        }

        bool IsIntersecting(Vector3 vector, Vector3 direction, Triangle
triangle, out float t, out float u, out float v)
        {
            t = 0;
            u = 0;
            v = 0;

            var first = triangle.B - triangle.A;
            var second = triangle.C - triangle.A;

            var pvector = Vector3.Cross(direction, second);

            var det = Vector3.Dot(first, pvector);

            if (det < Epsilon && det > -Epsilon)
            {
                return false;
            }

            var invertedDet = 1f / det;

            var tvector = vector - triangle.A;

            u = Vector3.Dot(tvector, pvector) * invertedDet;

            if (u < 0 || u > 1)
            {
                return false;
            }

            var qvector = Vector3.Cross(tvector, first);

            v = Vector3.Dot(direction, qvector) * invertedDet;

            if (v < 0 || u + v > 1)
            {
                return false;
            }

            t = Vector3.Dot(second, qvector) * invertedDet;
            return true;
        }

        bool ProcessScene(Vector3 vector, Vector3 direction, Triangle
triangle, out Vector3 hit, out Vector3 normal)
        {
            hit = new Vector3();
            normal = new Vector3();

            var intersecting = IsIntersecting(vector, direction, triangle,
out var t, out var u, out var v);

            if (intersecting)
            {
                hit = vector + direction * t;
                normal = triangle.N0 * (1f - u - v) + triangle.N1 * u +
triangle.N2 * v;
                return true;
            }
```

```
            return false;
        }

        void ProcessColor(ref float colorElement)
        {
            if (colorElement > 255f)
            {
                colorElement = 255f;
            }

            if (colorElement < 0f)
            {
                colorElement = 0f;
            }
        }
    }

    private Vector3[,] Initialize(RenderOrderDto order)
    {
        var result = new Vector3[order.OrderSettings.ScreenSettings.Height,
order.OrderSettings.ScreenSettings.Width];

        for (var i = 0; i < result.GetLength(0); i++)
        {
            for (var j = 0; j < result.GetLength(1); j++)
            {
                result[i, j] =
order.OrderSettings.BackgroundColorSettings.Color.ToVector();
            }
        }
        return result;
    }

    private Vector3 GetCameraDirection(int pixelHeight, int pixelWidth,
ScreenSettingsDto screenSettings)
    {
        var x = (float)(((2 * (pixelWidth + 0.5)) / screenSettings.Width -
1f) *
                        Math.Tan(screenSettings.Fov / 2f) *
                        screenSettings.Width) /
                screenSettings.Height;

        var z = -((2f * (pixelHeight + 0.5f)) / screenSettings.Height - 1f) *
                (float)Math.Tan(screenSettings.Fov / 2f);
        return new Vector3(x, -1, z);
    }
}
```

Файл JpegImageWriter.cs

```
public class JpegImageWriter : IImageWriter
{
    public Task<Stream> WriteImage(Vector3[,] colorMatrix)
    {
        var height = colorMatrix.GetLength(0);
        var width = colorMatrix.GetLength(1);
        var bitmap = new SKBitmap(colorMatrix.GetLength(0),
colorMatrix.GetLength(1));
        for (var i = 0; i < height; i++)
        {
            for (var j = 0; j < width; j++)
            {
                var red = (byte)colorMatrix[i, j].X;
                var green = (byte)colorMatrix[i, j].Y;
```

```
                var blue = (byte)colorMatrix[i, j].Z;
                bitmap.SetPixel(i, j, new SKColor(red, green, blue));
            }
        }

        var image = SKImage.FromBitmap(bitmap);
        var encodedImage = image.Encode(SKEncodedImageFormat.Jpeg, 100);
        return Task.FromResult(encodedImage.AsStream());
    }
}
```

Файл RenderService.cs

```
public class RenderService : IRenderService
{
    private readonly ICloudStorageService _cloudStorageService;
    private readonly IModelReader _reader;
    private readonly IRenderer _renderer;
    private readonly IImageWriter _writer;
    private readonly string _modelsBucket;
    private readonly string _resultsBucket;
    private readonly ILogger<IRenderService> _logger;
    private readonly OrderStatusProducer _producer;

    public RenderService(ICloudStorageService cloudStorageService,
        IModelReader reader,
        IOptions<S3Buckets> buckets,
        IRenderer renderer,
        IImageWriter writer,
        ILogger<IRenderService> logger,
        OrderStatusProducer producer)
    {
        _cloudStorageService = cloudStorageService;
        _reader = reader;
        _renderer = renderer;
        _writer = writer;
        _logger = logger;
        _producer = producer;
        _modelsBucket = buckets.Value.ModelsBucket;
        _resultsBucket = buckets.Value.ResultsBucket;
    }

    public async Task RenderModel(RenderOrderDto order)
    {
        try
        {
            order.Status = OrderStatus.InProgress;
            ChangeCurrentState(order, "Reading model");
            await using var modelStream = await
_cloudStorageService.GetObjectSteamAsync(order.Model, _modelsBucket);
            var trianglesToRender = _reader.ReadModel(modelStream).ToList();

            ChangeCurrentState(order, "Rendering model");
            var results = _renderer.Render(trianglesToRender, order);

            ChangeCurrentState(order, "Creating result image");
            var resultImageStream = await _writer.WriteImage(results);

            ChangeCurrentState(order, "Uploading the result to a cloud
storage");
            order.Result = await _cloudStorageService.UploadFileAsync(new
FileDto()
            {
                File = resultImageStream,
                Name = order.Name
```

```
            }, _resultsBucket);

            order.Status = OrderStatus.Completed;
        }
        catch (Exception e)
        {
            _logger.LogError(-1, e.Message);
            order.Status = OrderStatus.Faulted;
            order.FaultMessage = e.Message;
        }
        order.Completed = DateTime.Now;
        ChangeCurrentState(order, "");
    }

    private void ChangeCurrentState(RenderOrderDto order, string message)
    {
        order.CurrentAction = message;
        _producer.PushMessage(order);
    }
}
```

Файл Cube.cs

```
public class Cube
{
    public readonly Vector3 X1;
    public readonly Vector3 X2;
    public readonly Vector3 X3;
    public readonly Vector3 X4;
    public readonly Vector3 X5;
    public readonly Vector3 X6;
    public readonly Vector3 X7;
    public readonly Vector3 X8;

    private readonly List<Triangle> _sides;

    public Cube(Vector3 x, Vector3 y, Vector3 z, Vector3 k, Vector3 n1,
Vector3 y1, Vector3 z1, Vector3 k1)
    {
        _sides = new List<Triangle>();
        X1 = x;
        X2 = y;
        X3 = z;
        X4 = k;

        X5 = n1;
        X6 = y1;
        X7 = z1;
        X8 = k1;

        var temp = new Vector3(0, 0, 0);
        _sides.Add(new Triangle(X7, X6, X8, temp, temp, temp));
        _sides.Add(new Triangle(X5, X6, X8, temp, temp, temp));

        _sides.Add(new Triangle(X1, X5, X8, temp, temp, temp));
        _sides.Add(new Triangle(X1, X4, X8, temp, temp, temp));

        _sides.Add(new Triangle(X5, X2, X6, temp, temp, temp));
        _sides.Add(new Triangle(X5, X2, X1, temp, temp, temp));

        _sides.Add(new Triangle(X7, X2, X3, temp, temp, temp));
        _sides.Add(new Triangle(X7, X2, X6, temp, temp, temp));

        _sides.Add(new Triangle(X3, X7, X8, temp, temp, temp));
        _sides.Add(new Triangle(X3, X4, X8, temp, temp, temp));
```

```
            _sides.Add(new Triangle(X3, X2, X4, temp, temp, temp));
            _sides.Add(new Triangle(X1, X2, X4, temp, temp, temp));
        }

        bool ThereIsIntersectionBetweenRayAndTriangle(Vector3 rayOrigin, Vector3
rayVector, Triangle inTriangle)
        {
            var vertex0 = inTriangle.A;
            var vertex1 = inTriangle.B;
            var vertex2 = inTriangle.C;
            var edge1 = vertex1 - vertex0;
            var edge2 = vertex2 - vertex0;
            var h = Vector3.Cross(rayVector, edge2);

            var a = Vector3.Dot(edge1, h);
            var epsilon = 1e-5f;

            if (a > -epsilon && a < epsilon)
            {
                return false;
            }

            var f = 1 / a;
            var s = rayOrigin - vertex0;
            var u = f * Vector3.Dot(s, h);
            if (u < 0.0 || u > 1.0)
            {
                return false;
            }

            var q = Vector3.Cross(s, edge1);
            var v = f * Vector3.Dot(rayVector, q);
            if (v < 0.0 || u + v > 1.0)
            {
                return false;
            }


            var t = f * Vector3.Dot(edge2, q);
            return true;
        }

        public bool IntersectionBetweenRayAndCube(Vector3 rayOrigin, Vector3
rayVector)
        {
            return _sides.Any(i =>
ThereIsIntersectionBetweenRayAndTriangle(rayOrigin, rayVector, i));
        }
    }
```

Файл TreeNode.cs

```
public class TreeNode
{
    public List<Triangle> Triangles { get; set; }
      public List<TreeNode> DaughterNodes { get; set; }

      private bool _list = true;
      private Cube SpaceCube { get; set; }


      public TreeNode(float minX, float maxX, float minY, float maxY, float
minZ, float maxZ)
      {
```

```
            var x1 = new Vector3(maxX, maxY, maxZ);
            var x2 = new Vector3(minX, maxY, maxZ);
            var x3 = new Vector3(minX, minY, maxZ);
            var x4 = new Vector3(maxX, minY, maxZ);

            var x5 = new Vector3(maxX, maxY, minZ);
            var x6 = new Vector3(minX, maxY, minZ);
            var x7 = new Vector3(minX, minY, minZ);
            var x8 = new Vector3(maxX, minY, minZ);

            SpaceCube = new Cube(x1, x2, x3, x4, x5, x6, x7, x8);
            Triangles = new List<Triangle>();
            DaughterNodes = new List<TreeNode>();
        }

        public void Rebuild(float min)
        {
            float minXnew, maxXnew, minYnew, maxYnew, minZnew, maxZnew;


            var center = new Vector3((SpaceCube.X1.X + SpaceCube.X2.X) * 0.5f,
(SpaceCube.X1.Y + SpaceCube.X4.Y) * 0.5f, (SpaceCube.X1.Z + SpaceCube.X5.Z) *
0.5f);

            minXnew = (SpaceCube.X1.X + SpaceCube.X2.X) * 0.5f;
            maxXnew = SpaceCube.X1.X;
            minYnew = (SpaceCube.X1.Y + SpaceCube.X4.Y) * 0.5f;
            maxYnew = SpaceCube.X1.Y;
            minZnew = (SpaceCube.X1.Z + SpaceCube.X5.Z) * 0.5f;
            maxZnew = SpaceCube.X1.Z;

            if (Triangles.Count() < 100 || (maxXnew - minXnew) < min)
                return;

            DaughterNodes.Add(new TreeNode(minXnew, maxXnew, minYnew, maxYnew,
minZnew, maxZnew));

            maxXnew = (SpaceCube.X1.X + SpaceCube.X2.X) * 0.5f;
            minXnew = SpaceCube.X2.X;
            minYnew = (SpaceCube.X1.Y + SpaceCube.X4.Y) * 0.5f;
            maxYnew = SpaceCube.X1.Y;
            minZnew = (SpaceCube.X1.Z + SpaceCube.X5.Z) * 0.5f;
            maxZnew = SpaceCube.X1.Z;

            DaughterNodes.Add(new TreeNode(minXnew, maxXnew, minYnew, maxYnew,
minZnew, maxZnew));

            maxXnew = (SpaceCube.X1.X + SpaceCube.X2.X) * 0.5f;
            minXnew = SpaceCube.X2.X;
            maxYnew = (SpaceCube.X2.Y + SpaceCube.X3.Y) * 0.5f;
            minYnew = SpaceCube.X3.Y;
            minZnew = (SpaceCube.X1.Z + SpaceCube.X5.Z) * 0.5f;
            maxZnew = SpaceCube.X1.Z;

            DaughterNodes.Add(new TreeNode(minXnew, maxXnew, minYnew, maxYnew,
minZnew, maxZnew));

            minXnew = (SpaceCube.X3.X + SpaceCube.X4.X) * 0.5f;
            maxXnew = SpaceCube.X4.X;
            maxYnew = (SpaceCube.X1.Y + SpaceCube.X4.Y) * 0.5f;
            minYnew = SpaceCube.X4.Y;
            minZnew = (SpaceCube.X1.Z + SpaceCube.X5.Z) * 0.5f;
            maxZnew = SpaceCube.X1.Z;
```

```
            DaughterNodes.Add(new TreeNode(minXnew, maxXnew, minYnew, maxYnew,
minZnew, maxZnew));


            minXnew = (SpaceCube.X1.X + SpaceCube.X2.X) * 0.5f;
            maxXnew = SpaceCube.X1.X;
            minYnew = (SpaceCube.X1.Y + SpaceCube.X4.Y) * 0.5f;
            maxYnew = SpaceCube.X1.Y;
            maxZnew = (SpaceCube.X1.Z + SpaceCube.X5.Z) * 0.5f;
            minZnew = SpaceCube.X5.Z;

            DaughterNodes.Add(new TreeNode(minXnew, maxXnew, minYnew, maxYnew,
minZnew, maxZnew));

            maxXnew = (SpaceCube.X1.X + SpaceCube.X2.X) * 0.5f;
            minXnew = SpaceCube.X2.X;
            minYnew = (SpaceCube.X1.Y + SpaceCube.X4.Y) * 0.5f;
            maxYnew = SpaceCube.X1.Y;
            maxZnew = (SpaceCube.X1.Z + SpaceCube.X5.Z) * 0.5f;
            minZnew = SpaceCube.X5.Z;

            DaughterNodes.Add(new TreeNode(minXnew, maxXnew, minYnew, maxYnew,
minZnew, maxZnew));

            maxXnew = (SpaceCube.X1.X + SpaceCube.X2.X) * 0.5f;
            minXnew = SpaceCube.X2.X;
            maxYnew = (SpaceCube.X2.Y + SpaceCube.X3.Y) * 0.5f;
            minYnew = SpaceCube.X3.Y;
            maxZnew = (SpaceCube.X1.Z + SpaceCube.X5.Z) * 0.5f;
            minZnew = SpaceCube.X5.Z;

            DaughterNodes.Add(new TreeNode(minXnew, maxXnew, minYnew, maxYnew,
minZnew, maxZnew));

            minXnew = (SpaceCube.X3.X + SpaceCube.X4.X) * 0.5f;
            maxXnew = SpaceCube.X4.X;
            maxYnew = (SpaceCube.X1.Y + SpaceCube.X4.Y) * 0.5f;
            minYnew = SpaceCube.X4.Y;
            maxZnew = (SpaceCube.X1.Z + SpaceCube.X5.Z) * 0.5f;
            minZnew = SpaceCube.X5.Z;

            DaughterNodes.Add(new TreeNode(minXnew, maxXnew, minYnew, maxYnew,
minZnew, maxZnew));

            var newVectors = new List<Triangle>();

            while (Triangles.Count() != 0)
            {
               var counter1 = -1;
               var counter2 = -1;
               var counter3 = -1;

               var i = Triangles.Last();
               Triangles.RemoveAt(Triangles.Count() - 1);

               counter1 = PositionFinder(i.A, center);
               counter2 = PositionFinder(i.B, center);
               counter3 = PositionFinder(i.C, center);

               if (counter1 == counter2 && counter2 == counter3)
                  DaughterNodes[counter1].Triangles.Add(i);
               else
                  newVectors.Add(i);
            }
```

```
            Triangles = newVectors;

            _list = false;

            DaughterNodes[0]._list = true;
            DaughterNodes[0].Rebuild(min);
            DaughterNodes[1]._list = true;
            DaughterNodes[1].Rebuild(min);
            DaughterNodes[2]._list = true;
            DaughterNodes[2].Rebuild(min);
            DaughterNodes[3]._list = true;
            DaughterNodes[3].Rebuild(min);
            DaughterNodes[4]._list = true;
            DaughterNodes[4].Rebuild(min);
            DaughterNodes[5]._list = true;
            DaughterNodes[5].Rebuild(min);
            DaughterNodes[6]._list = true;
            DaughterNodes[6].Rebuild(min);
            DaughterNodes[7]._list = true;
            DaughterNodes[7].Rebuild(min);
        }

        public void FindIntersections(Vector3 rayOrigin, Vector3 rayVector,
List<Triangle> result)
        {

            foreach (var i in Triangles)
                result.Add(i);
            if (_list)
                return;

            for (var i = 0; i < 8; i++)
            {
                if (DaughterNodes[i] != null &&
DaughterNodes[i].SpaceCube.IntersectionBetweenRayAndCube(rayOrigin,
rayVector))
                {
                    DaughterNodes[i].FindIntersections(rayOrigin, rayVector,
result);
                }
            }

        }

        public int PositionFinder(Vector3 x, Vector3 center)
        {
            if (x.Z > center.Z)
            {
                if (x.Y > center.Y)
                {
                    return x.X > center.X ? 0 : 1;
                }

                return x.X > center.X ? 3 : 2;
            }

            if (x.Y > center.Y)
            {
                return x.X > center.X ? 4 : 5;
            }

            return x.X > center.X ? 7 : 6;
```

```
        }
}
```

Файл Tree.cs

```
public class Tree : ITree
{
    private float _maxSize;
    private float _minSize;
    private TreeNode _head;

    public void Initialize(float max, List<Triangle> triangles)
    {
        _minSize = max / 10;
        _maxSize = max;
        _head = new TreeNode(-max, max, -max, max, -max, max) {Triangles =
triangles};
        _head.Rebuild(_minSize);
    }

    public void FindIntersections(Vector3 rayOrigin, Vector3 rayVector,
List<Triangle> result)
    {
        _head.FindIntersections(rayOrigin, rayVector, result);
    }
}
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНІ ЗАСОБИ ДЛЯ РЕНДЕРИНГУ ОБ’ЄМНИХ МОДЕЛЕЙ

Програма та методика тестування

КПІ.ІП-8132.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Максим ГОЛОВЧЕНКО

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Артем СУПРИГАН

Київ – 2023

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є програмні засоби для рендерингу об’ємних моделей. Серверну та клієнтську частини виконано на платформі .NET.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка збереження даних;
- знаходження проблем, помилок і недоліків з метою їх усунення.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;

– динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;

– функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;

– системне тестування – перевіряється усе програмне забезпечення в цілому;

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з використанням наскрізного тестування, з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

– динамічне тестування на відповідність функціональним вимогам;

– тестування на наявність повідомлень про помилку у відповідях з серверу, коли це необхідно;

– тестування на швидкість виконання рендерингу;

– тестування на валідацію введених даних.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНІ ЗАСОБИ ДЛЯ РЕНДЕРИНГУ ОБ’ЄМНИХ МОДЕЛЕЙ

Керівництво користувача

КПІ.ІП-8132.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Максим ГОЛОВЧЕНКО

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Артем СУПРИГАН

Київ – 2023

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

Програмні засоби для рендерингу об’ємних моделей – це програмні засоби, що дозволяють їх власнику надавати свої обчислювальні потужності клієнтам для рендерингу об’ємних моделей.

Кінцева збірка складається з трьох мікросервісів та демонстраційної клієнтської частини.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

– наявність доступу до Інтернету.

2.2 Завантаження застосунку

На даний момент застосунок можна встановити власноруч, встановивши Docker та завантаживши і зібравши docker-образи наступними консольними командами:

– docker pull burnyaxa/render-service-auth:latest;

– docker pull burnyaxa/render-service-order:latest;

– docker pull burnyaxa/render-service-worker:latest.

Для коректної роботи також необхідно завантажити образи Microsoft SQL Server, RabbitMQ та Redis використовуючи наступні коносольні команди:

– docker pull mcr.microsoft.com/mssql/server;

– docker pull rabbitmq;

– docker pull redis.

Демонстраційну клієнтську частину можна завантажити з репозиторію за посиланням https://github.com/Burnyaxa/RenderService.Client.

2.3 Перевірка коректної роботи

По завершенню розгортання усіх docker-контейнерів, вони мають мати статус «Running». У разі, якщо контейнер має статус Exited, то його розгортання було неуспішним.

3 ВИКОНАННЯ ПРОГРАМИ

Для повноцінного користування програмними засобами, необхідно мати зареєстрований акаунт. Для реєстрації необхідно надіслати POST запит на ендпоїнт /api/v1/registration. В тілі запита необхідно вказати поля Email та Password.

Електронна пошта має відповідати наступним вимогам:

– має містити символ «@»;
– символ «@» не має бути останнім символом у пошті;
– символ «@» не має бути першим символом у пошті.

Пароль має відповідати наступним вимогам:

– має містити хоча б 10 символів;
– має містити не більше 100 символів;
– має містити хоча б 5 унікальних символів;
– має містити хоча б одну велику літеру;
– має містити хоча б одну маленьку літеру;
– має містити хоча б одну цифру;
– має містити хоча б один спеціальний символ;
– не має містити пробілів.

Якщо акаунт вже створено, то для отримання доступу до захищених ресурсів, необхідно провести процедуру авторизації. Для цього необхідно надіслати POST запит на ендпоїнт /api/v1/authorization. В тілі запита необхідно вказати поля Email та Password.

Після реєстрації або авторизації, у відповідь клієнт отримає профіль користувача, що складається з полів Id, Email та CreatedAt, а також пару токенів AccessToken та RefreshToken.

Для доступу до захищених ресурсів, у запити необхідно додавати заголовок Authorization зі значенням у форматі «Bearer {AccessToken}».

У випадку, якщо термін дії AccessToken закінчився, його можна оновити. Для цього необхідно надіслати POST запит на ендпоїнт

/api/v1/token/refresh та у тілі запиту вказати RefreshToken. У відповіді клієнт отримає нову пару AccessToken і RefreshToken.

Доступ до ендпоїнтів, пов’язаних з замовленнями на реднеринг є захищеним, тому для них обов’язково треба мати AccessToken.

Для отримання списку замовлень поточного користувача, треба надіслати GET запит на ендпоїнт /api/v1/orders/user. У відповідь клієнт отримає список замовлень, що складаються з наступних полів:

– Id;
– UserId;
– Name;
– Status;
– FaultMessage;
– CurrentAction;
– Result;
– CreatedAt;
– CompletedAt.

Клієнт має змогу редагувати замовлення, зокрема змінювати його ім’я. Для цього необхідно надіслати PUT запит на ендпоїнт /api/v1/orders та у тіло передати модель замовлення зі зміненими даними.

Для видалення замовлення, необхідно надіслати DELETE запит на ендпоїнт /api/v1/orders/{id}, де id – це Id замовлення, що необхідно видалити.

Для створення замовлення, необхідно надіслати POST запит на ендпоїнт /api/v1/orders. Заголовок Content-Type запиту повинен мати значення «multipart/form-data». У формі повинні міститися файл моделі та наступні поля:

– UserId;
– Name;
– OrderSettings.

Налаштування замовлення складаються з наступних полів:

– BackgroundColorSettings;

– CameraPositionSettings;
– LightsSettings;
– ScreenSettings;
– ObjectColorSettings.

Налаштування кольору задаються рівнями червоного, зеленого та синього від 0 до 255 включно. Рівні кольорів представлені у вигляді полів Red, Green та Blue відповідно.

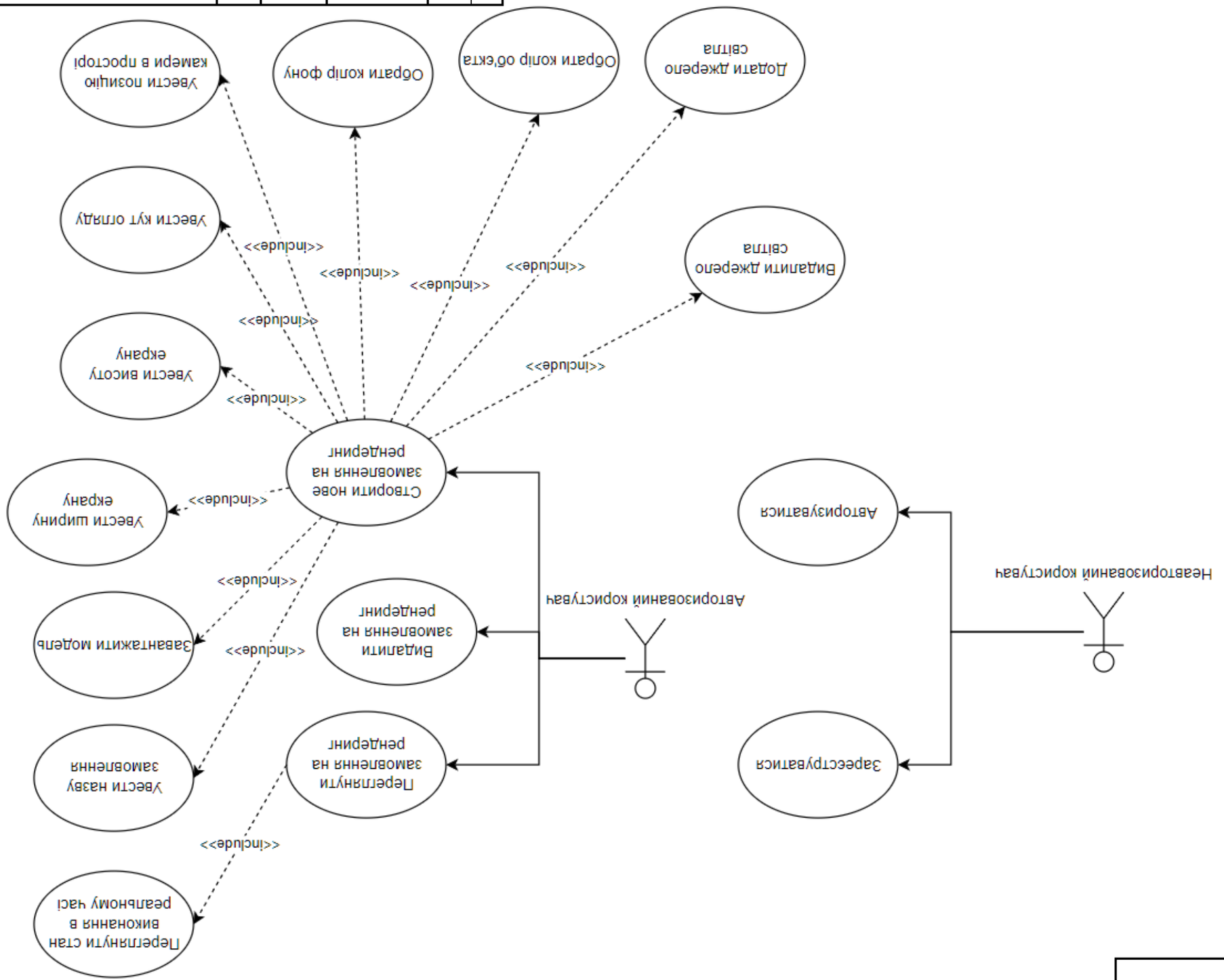
Налаштування позиції камери у просторі задаються у вигляді точки з координатами X, Y та Z.

Налаштування джерела світла задаються полями інтенсивність світла Intensity та вектор Direction.

Налаштування екрану задаються наступними полями:

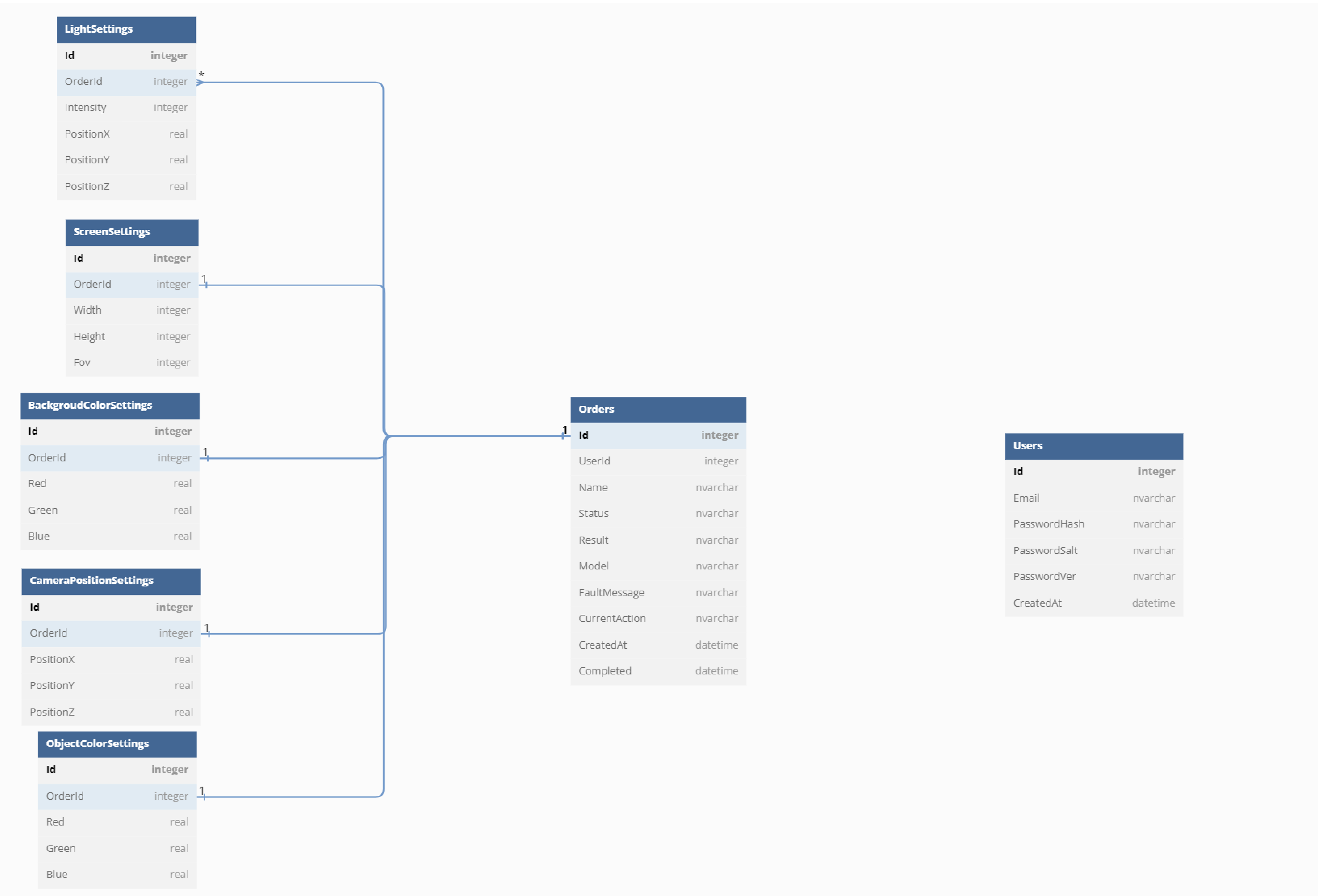
– Height;
– Width;
– Fov.

Для того, щоб отримувати сповіщення про зміну стану замовлення в режимі реального часу, клієнту необхідно надіслати GET запит на ендпоїнт /sse. Для доступу необхідно вказати AccessToken. У результаті клієнт, почне отримувати пакети даних, які являють собою пару ключ-значення, де ключем є Id замовлення, а значенням – модель замовлення. Пакети даних надсилаються кожен раз, коли змінюється стан будь-якого замовлення поточного користувача.

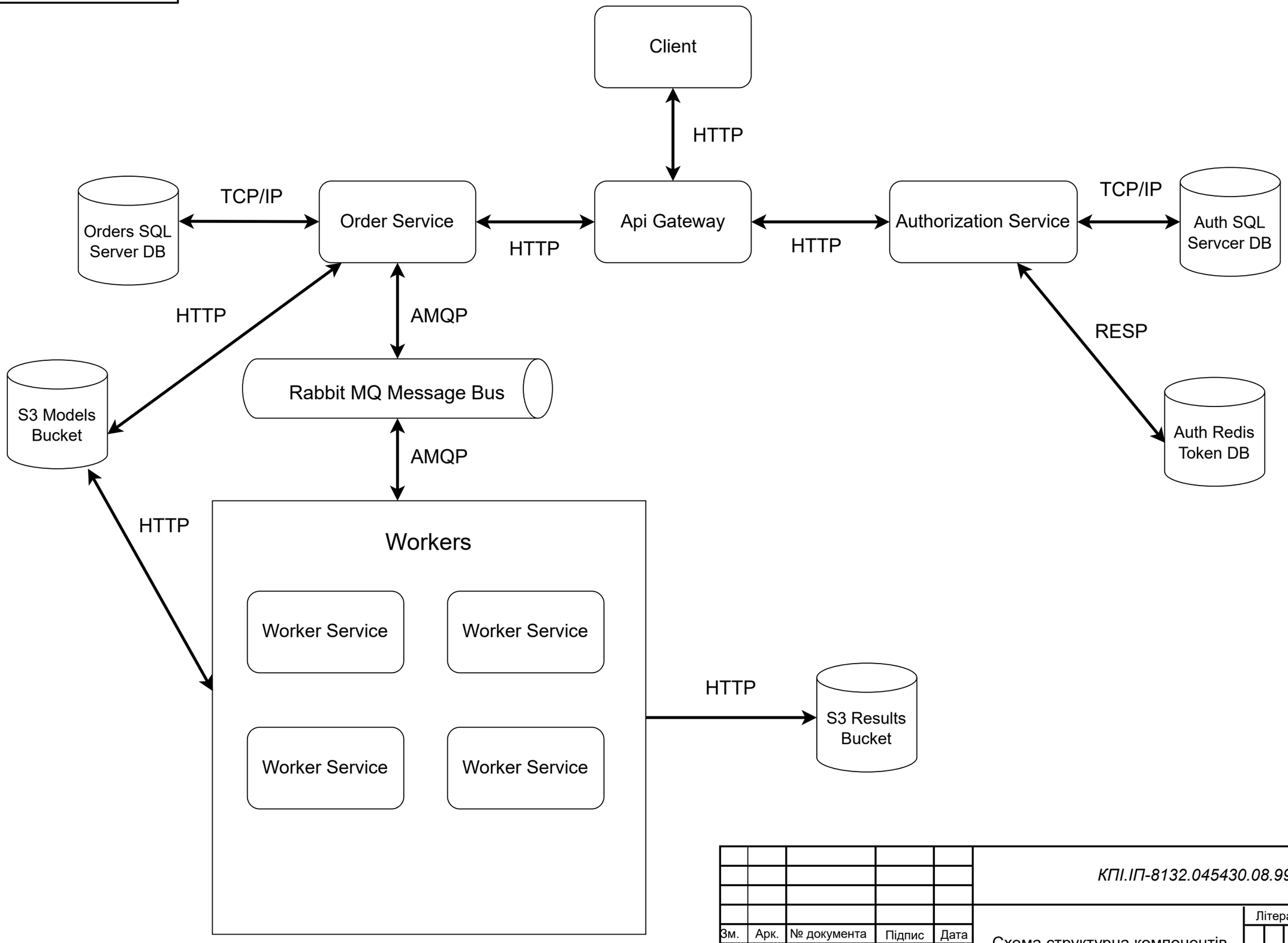


										КПІ.ІП-8132.045430.06.99.СС
Змн.	Арк.	№ документа	Підпис	Дата						
Розробив		Сухрипан А.С.								Схема структура варіантів використання
Перевірив		Голованенко М.М.								
Т. контр.										
Н. контр.		Голованенко М.М.								Програмні засоби для ремдвигту об'ємних моделей
Затвердив		Жарікова Е.В.								
Літера		Маса	Масштаб							КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-81в
Аркуш		Аркушів								

КПІ.ІП-8132.045430.07.99.СБД



Зм.	Арк.	№ документа	Підпис	Дата	КПІ.ІП-8132.045430.07.99.СБД		
Розробив		Суприган А.С.			Схема бази даних	Літера	Маса	Масштаб
Перевірив		Головченко М.М.				Аркуш	Аркушів
Т. кон.					Програмні засоби для рендерингу об'ємних моделей	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-81в
Н. кон.		Головченко М.М.				
Затвердив		Жаріков Е.В.				



					КПІ.ІП-8132.045430.08.99.СС	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна компонентів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Суприган А.С.				Аркуш / Аркушів
Перевірив		Головченко М.М.				
Т. кон.					Програмні засоби для рендерингу об'ємних моделей	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-81в
Н. кон.		Головченко М.М.				
Затвердив		Жаріков Е.В.				