МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп’ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувачка кафедри,
д-р техн. наук, проф.

_____Ірина ЖУРАВСЬКА

« __ » _____ 202__ р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Сервіс генерації 3D-моделей хмарної мережевої інфраструктури

Спеціальність 123 Комп’ютерна інженерія

Освітня програма «Комп’ютерна інженерія»

Здобувач _____Нікіта ІЩЕНКО
підпис

«__» _____ 202__ р.

Керівник ст. викладач

_____ Катерина ОБУХОВА
підпис
«__» _____ 202__ р.

Миколаїв – 2025

Факультет	Комп’ютерних наук
Кафедра	Комп’ютерної інженерії
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	123 Комп’ютерна інженерія
Освітня програма	Комп’ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри комп’ютерної інженерії
_____ Ірина ЖУРАВСЬКА
«_____» _____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Іщенка Нікіти Юрійовича
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи
Сервіс генерації 3D-моделей хмарної мережевої інфраструктури

Затверджена наказом по ЧНУ ім. Петра Могили від 30.10.2024 № 297.

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Очікуваним результатом роботи є створення інтерактивної тривимірної сцени, яка наочно відображає структуру хмарної інфраструктури, отриманої з облікового запису користувача в AWS.

4. Перелік питань, що підлягають розробці:
- обґрунтувати актуальність теми та проаналізувати сучасний стан у сфері візуалізації хмарної мережевої інфраструктури;
- здійснити аналіз існуючих підходів, технологій та інструментів для побудови 3D-візуалізацій та моделювання мереж;
- обрати архітектуру сервісу, програмні засоби та середовище розробки, які найбільше відповідають поставленим вимогам;
- реалізувати функціонал для автоматизованої генерації 3D-моделей мережевих структур на основі вхідних даних;
- забезпечити інтерактивну візуалізацію моделі з можливістю масштабування, перегляду та аналізу структури;

– розробити механізм для генерації 3D-моделей з хмарної інфраструктури AWS, що буде реалізовано на базі ЛОМ, забезпечивши інтеграцію між хмарними та локальними системами для більш ефективної обробки даних та візуалізації;

– протестувати працездатність сервісу на прикладах типових конфігурацій хмарної інфраструктури, а також провести оцінку ефективності.

5. Перелік графічних матеріалів
Презентація

6. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ __Катерина ОБУХОВА_____
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ __Нікіта ІЩЕНКО_____
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «_____» _____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної бакалаврської роботи

Тема: Сервіс генерації 3D-моделей хмарної мережевої інфраструктури

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КБР	30.10.2024	03.11.2024	Виконано
2	Огляд літератури за темою роботи	03.03.2025	17.03.2025	Виконано
3	Складання календарного плану КБР	19.03.2025	19.03.2025	Виконано
4	Аналіз предметної області	20.03.2025	04.04.2025	Виконано
5	Розробка проєктних рішень	05.04.2025	20.04.2025	Виконано
6	Моделювання та конструювання АПЗ	20.04.2025	06.06.2025	Виконано
7	Перевірка працездатності, тестування та апробація розробленого АПЗ, аналіз результатів тестування	06.06.2025	15.06.2025	Виконано
8	Відгук керівника КБР	17.06.2025	17.06.2025	Виконано
9	Оформлення КБР та презентації	15.05.2025	10.06.2025	Виконано
10	Попередній захист	21.05.2025	04.06.2025	Виконано
11	Рецензування	15.06.2025	18.06.2025	Виконано
12	Завершення оформлення КБР та презентації	10.06.2025	17.06.2025	Виконано
13	Захист кваліфікаційної бакалаврської роботи	24.06.2025	24.06.2025	

Керівник роботи _____ __Катерина ОБУХОВА_____

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ __Нікіта ІЩЕНКО_____

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

«____» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи
«Сервіс генерації 3D-моделей хмарної мережевої інфраструктури»
Студент 405 гр.: Іщенко Нікіта Юрійович
Керівник: ст. викладач Обухова Катерина Олександрівна

Стрімкий розвиток хмарних технологій та зростання складності інфраструктур обумовлюють необхідність у більш ефективних засобах візуалізації мережевих архітектур. Традиційні підходи до відображення структури хмарної інфраструктури стають недостатньо інформативними, особливо у випадку великомасштабних систем. Тому актуальною є розробка сервісу, що дозволяє автоматизовано створювати 3D-моделі хмарних архітектур, зокрема на базі Amazon Web Services (AWS), для використання в локальних обчислювальних мережах.

Об'єкт дослідження – хмарна мережева інфраструктура, реалізована на базі платформи Amazon Web Services (AWS).

Предмет дослідження – методи та програмні засоби автоматизованої генерації 3D-моделей хмарної інфраструктури з подальшою інтеграцією в ЛОМ для візуалізації мережевих зв'язків.

Мета роботи – створення програмного сервісу для автоматизованої побудови інтерактивних 3D-моделей хмарної мережевої інфраструктури з метою її візуального представлення, спрощення аналізу та оптимізації управління мережею.

Практична значимість: робота пропонує новий підхід до візуалізації хмарних мереж, що поєднує автоматизований імпорт з AWS, 3D-візуалізацію на основі бібліотеки Three.js та модульну архітектуру з можливістю масштабування. Такий інструмент може використовуватись у навчальному процесі, для проєктування, аналізу та адміністрування сучасних хмарних рішень.

Апробація роботи: результати дослідження були представлені на XVI Міжнародній науково-практичній конференції «Free and Open Source Software» (м. Харків, лютий 2025 р.). За підсумками участі опубліковано тези доповідей.

Пояснювальна записка кваліфікаційної бакалаврської роботи (КБР) складається зі вступу, трьох розділів, висновків, переліку джерел посилання та 2 додатки.

Перший розділ присвячено аналізу існуючих засобів візуалізації хмарної інфраструктури, а також виявленню їх недоліків (відсутність 3D, низька інтерактивність, складність масштабування). Другий розділ розглядає проєктування сервісу: обґрунтовано вибір архітектури, описано основні компоненти системи, зокрема модулі імпорту, обробки даних, 3D-візуалізації (Three.js), рендерингу (3DS Max та Corona Render) та інтеграції з API AWS. Третій розділ містить опис реалізації сервісу, тестування на прикладах хмарних конфігурацій, оцінку ефективності системи, аналіз продуктивності та окреслення напрямів подальшого розвитку, таких як генерація звітів, підтримка інших хмарних платформ та мобільна адаптація.

В цілому КБР містить 69 сторінок (без додатків), 16 рисунків, 6 таблиць, 24 джерела посилання та 2 додатки.

Ключові слова: *3D-візуалізація, інтерактивне моделювання, генерація моделей, хмарна інфраструктура, локальна мережа, Amazon Web Services.*

ABSTRACT

of the Bachelor's Thesis

"Service for generating 3D models of cloud network infrastructure"

Student: Nikita Ishchenko

Supervisor: senior lecturer Kateryna Obukhova

The rapid advancement of cloud technologies and the increasing complexity of infrastructure necessitate more effective tools for visualizing network architectures. Traditional approaches to representing cloud infrastructure structures are becoming insufficiently informative, particularly in the context of large-scale systems. Therefore, the development of a service that enables automated generation of 3D models of cloud architectures – specifically based on Amazon Web Services (AWS) – for use in local area networks is a relevant and timely task.

The object of the study – cloud network infrastructure implemented using the Amazon Web Services (AWS) platform.

The subject of the study – methods and software tools for the automated generation of 3D models of cloud infrastructure, with subsequent integration into local area networks for visualizing network connections.

The purpose of the study – to develop a software service for the automated construction of interactive 3D models of cloud network infrastructure in order to provide a visual representation, facilitate analysis, and optimize network management.

Practical significance: this work proposes a novel approach to cloud network visualization by combining automated AWS data import, 3D visualization using the Three.js library, and a modular, scalable architecture. The proposed tool can be employed in educational settings, as well as for the design, analysis, and administration of modern cloud-based solutions.

The results of this study were presented at the 16th International Scientific and Practical Conference "Free and Open Source Software" (Kharkiv, February 2025). Conference abstracts were published following the event.

The explanatory note of this bachelor's thesis consists of an introduction, three chapters, conclusions, a list of references, and 2 appendices.

The first chapter focuses on the analysis of existing tools for cloud infrastructure visualization and identifies their limitations (lack of 3D support, low interactivity, poor scalability). The second chapter discusses the service design process, justifying the choice of architecture and describing the main system components, including modules for data import, processing, 3D visualization (Three.js), rendering (3DS Max and Corona Renderer), and AWS API integration. The third chapter presents the implementation of the service, testing using example cloud configurations, system performance evaluation, and outlines future development directions, such as report generation, support for other cloud platforms, and mobile adaptation.

The paper consists of 69 pages (excluding appendices), 16 figures, 6 tables, 24 references, and 2 appendices.

Keywords: 3D visualization, interactive modeling, model generation, cloud infrastructure, local area network, Amazon Web Services. .

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АПЗ – апаратно-програмне забезпечення
КБР – кваліфікаційна бакалаврська робота
ЛОМ – локальна обчислювальна мережа

3D – 3-dimensional
AWS SDK – AWS software development kit
AWS – Amazon Web Services
GCP – Google Cloud Platform
HTTPS – HyperText Transfer Protocol Secure
IaC – Infrastructure as Code
JWT – JSON Web Tokens
RDS – Relational Database Service
VPC – Virtual Private Cloud

ВСТУП

У сучасному світі цифрові технології розвиваються надзвичайно стрімко, що зумовлює збільшення обсягів обчислювальних ресурсів, ускладнення архітектури мереж та активне впровадження хмарної інфраструктури в усіх сферах діяльності. Із розширенням масштабів використання хмарних сервісів виникає необхідність ефективного проєктування, візуалізації та керування складними мережевими структурами. Традиційні підходи до документування мережевої інфраструктури, які базуються переважно на статичних схемах та текстових описах, дедалі менше відповідають вимогам сучасного IT-середовища. Актуальність теми полягає у потребі впровадження інноваційних засобів для динамічного відображення структури хмарних мереж з використанням тривимірної графіки, що дозволяє отримати глибше розуміння системної архітектури та взаємодії її компонентів. Сервіси генерації 3D-моделей забезпечують більш наочне, інтуїтивне представлення інформації, сприяючи підвищенню ефективності аналізу, адміністрування та прийняття технічних рішень.

Практичне значення теми полягає у розробці програмного рішення, що поєднує засоби моделювання хмарної інфраструктури з інтерактивною візуалізацією у форматі 3D. Такий підхід не лише полегшує процес навчання, проєктування та управління мережами, а й сприяє розвитку прикладних IT-рішень у сфері автоматизації, віртуалізації та мережевого менеджменту. Реалізація подібного сервісу дозволяє зменшити кількість помилок у процесі конфігурування інфраструктури, скоротити час на її аналіз та покращити взаєморозуміння між технічними спеціалістами й управлінським персоналом.

Об’єктом дослідження є робота хмарної мережевої інфраструктури, реалізованої на основі платформи Amazon Web Services (AWS), яка включає віртуальні обчислювальні ресурси, мережеві служби та сервіси управління, що інтегруються або використовуються в локальній обчислювальній мережі (ЛОМ).

Предметом дослідження є методи та засоби автоматизованої генерації 3D-моделей хмарної мережевої інфраструктури AWS для подальшого використання в локальній обчислювальній мережі (ЛОМ) з метою візуалізації її структури та взаємозв’язків між компонентами.

Мета роботи полягає в розробці сервісу для автоматизованої генерації 3D-моделей хмарної мережевої інфраструктури, призначених для застосування в ЛОМ, з метою наочного представлення, спрощення аналізу та покращення управління архітектурою мережі.

Для досягнення визначеної мети необхідно вирішити такі **завдання**:

– обґрунтувати актуальність теми та проаналізувати сучасний стан у сфері візуалізації хмарної мережевої інфраструктури;

– здійснити аналіз існуючих підходів, технологій та інструментів для побудови 3D-візуалізацій та моделювання мереж;

– обрати архітектуру сервісу, програмні засоби та середовище розробки, які найбільше відповідають поставленим вимогам;

– реалізувати функціонал для автоматизованої генерації 3D-моделей мережевих структур на основі вхідних даних;

– забезпечити інтерактивну візуалізацію моделі з можливістю масштабування, перегляду та аналізу структури;

– розробити механізм для генерації 3D-моделей з хмарної інфраструктури AWS, що буде реалізовано на базі ЛОМ, забезпечивши інтеграцію між хмарними та локальними системами для більш ефективної обробки даних та візуалізації;

– протестувати працездатність сервісу на прикладах типових конфігурацій хмарної інфраструктури, а також провести оцінку ефективності реалізованого рішення та визначити можливі напрями його вдосконалення.

Практична цінність роботи полягає у створенні інструменту, який дозволяє автоматизовано генерувати інтерактивні 3D-моделі хмарної мережевої інфраструктури AWS. Запропонований сервіс може бути використаний фахівцями з IT-інфраструктури для візуалізації архітектури

систем, спрощення аналізу та управління ресурсами, виявлення помилок конфігурації, а також як навчальний засіб для підготовки майбутніх інженерів з хмарних технологій. Використання тривимірного моделювання підвищує наочність та ефективність сприйняття складних технічних структур, що сприяє оптимізації процесів планування та адміністрування.

Таким чином, обрана тема є актуальною, має значний науково-практичний потенціал і відповідає сучасним вимогам у сфері розробки та управління хмарною інфраструктурою, зокрема в контексті автоматизації, візуалізації та оптимізації мережевих рішень.

Апробація роботи відбулася під час XVI Міжнародної науково-практичної конференції «Free and Open Source Software» 13–14 лютого 2025 р., Харків). За результатами опубліковано тези доповідей [2].

1 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ

1.1 Актуальність систем візуалізації хмарної мережевої інфраструктури

Упродовж останніх років хмарні обчислення стали основою для розгортання цифрових сервісів у більшості галузей – від фінансової до медичної. За даними компанії Gartner, у 2024 році світові витрати на публічні хмарні сервіси зросли на понад 20 %, досягнувши майже 600 мільярдів доларів США [4]. Зокрема, Amazon Web Services (AWS) займає лідируючі позиції, обслуговуючи мільйони інфраструктурних рішень для бізнесу, урядових структур і освітніх закладів.

Зі зростанням масштабів хмарної інфраструктури постає необхідність у її ефективному візуальному поданні. Більшість сучасних систем управління, зокрема AWS Management Console, AWS CloudFormation, Terraform, дозволяють описувати інфраструктуру у вигляді коду або через текстові конфігурації. Проте ці способи не забезпечують наочності та часто ускладнюють розуміння взаємозв’язків між компонентами – особливо у великих та динамічних проєктах.

За результатами опитування, проведеного Stack Overflow у 2023 році [3], понад 48 % DevOps-інженерів вказали, що вони регулярно стикаються з труднощами в орієнтації у великих мережевих конфігураціях через відсутність зручних візуальних інструментів. Більше того, близько 37 % випадків помилок у хмарній інфраструктурі пов’язані з неправильним налаштуванням або нерозумінням архітектури сервісів, що могло б бути попереджене при використанні більш інтерактивного способу візуалізації.

У світі вже існують окремі рішення для часткової візуалізації хмарних мереж. Наприклад:

– Lucidchart [5]: пропонує інтеграцію з AWS для побудови діаграм архітектури вручну;

– Hava.io [6]: дозволяє автоматично візуалізувати інфраструктуру AWS у вигляді статичних діаграм;

– Cloudcraft [7]: популярний серед архітекторів AWS сервіс для створення інтерактивних 2.5D-схем інфраструктури з підтримкою вартості й топології.

Проте всі перелічені системи обмежені або статичною природою візуалізації, або відсутністю повноцінної підтримки 3D-відображення, що особливо актуально для великих, багаторівневих структур з численними вузлами, підмережами та зонами доступності. Брак інтерактивної тривимірної моделі створює бар’єри у процесі планування, діагностики й навчання.

На фоні постійного зростання складності хмарних мереж актуальним є впровадження сервісу, який забезпечує автоматизовану генерацію 3D-моделі інфраструктури AWS на основі наданих конфігурацій або API-запитів. Такий сервіс міг би стати корисним:

– адміністраторам: для наочного контролю над станом ресурсів;

– архітекторам: для проєктування систем з урахуванням візуального представлення;

– викладачам і студентам: як дидактичний інструмент для вивчення структури хмарних сервісів.

Таким чином, створення системи для генерації 3D-моделей хмарної мережевої інфраструктури AWS відповідає сучасним викликам у сфері цифрового управління, автоматизації та IT-освіти. Такий підхід сприятиме підвищенню ефективності проєктування, швидкості виявлення помилок та покращенню взаємодії між різними учасниками процесу розробки хмарних рішень.

1.2 Огляд існуючих аналогів систем візуалізації хмарної мережевої інфраструктури

Оскільки аналогів розроблюваном сервісу в 3D немає, розглянемо найближчий аналог у 2.5D.

Cloudcraft – це популярний сервіс, що використовується для створення інтерактивних 2.5D-схем хмарної інфраструктури, особливо серед архітекторів та інженерів, які працюють з Amazon Web Services (AWS). Сервіс дозволяє будувати детальні візуальні моделі хмарних архітектур із підтримкою вартості й топології, а також забезпечує інтерактивні можливості для перегляду та аналізу інфраструктури (рис. 1.1). Ось детальніший огляд цього інструмента, його функцій, переваг і можливостей.

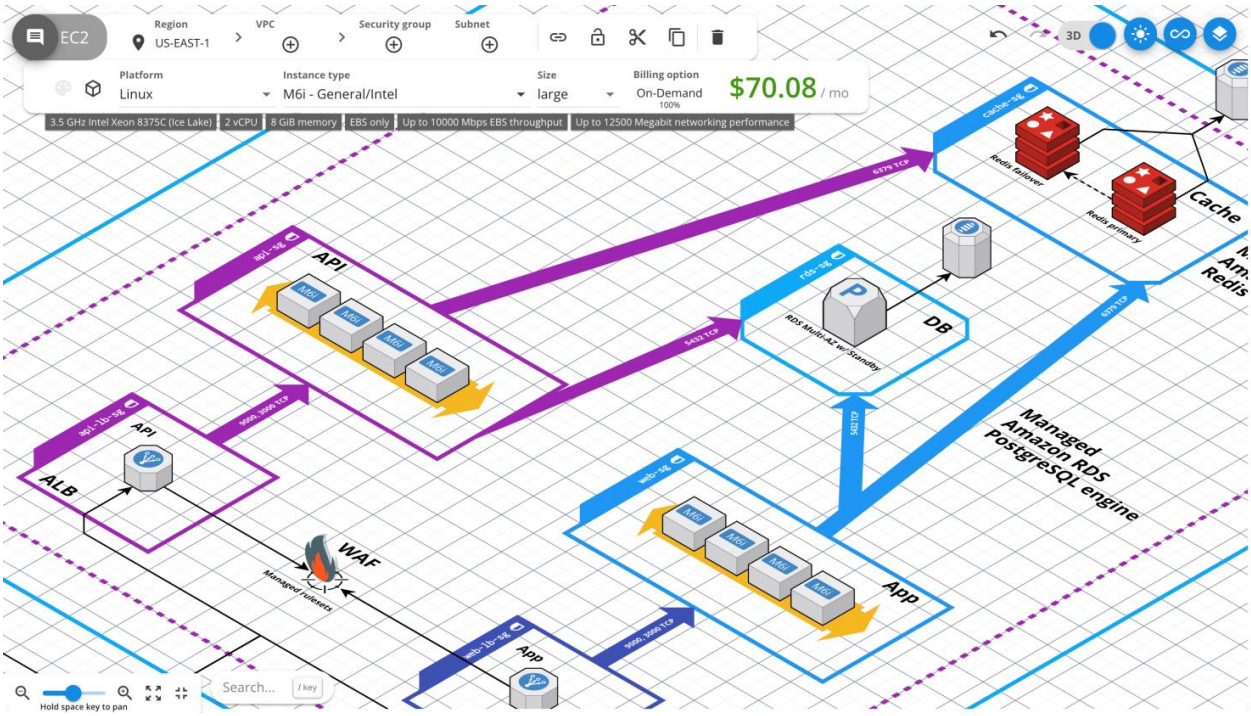


Рисунок 1.1 – Вигляд Cloudcraft

Cloudcraft є потужним інструментом для створення візуальних моделей інфраструктури AWS, який дозволяє архітекторові або інженеру створювати інтерактивні діаграми для зображення різних компонентів AWS і їх взаємодії. Цей інструмент надає можливість побудови схем топологій, відображаючи

різні ресурси, як то сервери, бази даних, мережі та інші сервіси AWS, використовуючи спеціальні елементи з їх візуальних бібліотек.

Один із основних аспектів Cloudcraft – це його здатність інтегрувати реальні дані з AWS, такі як поточні ресурси, їхні характеристики і вартості, що робить його особливо корисним для планування і оптимізації хмарної інфраструктури. Сервіс дозволяє створювати точні моделі для поточних та майбутніх конфігурацій інфраструктури AWS, що сприяє кращому розумінню того, як компоненти взаємодіють між собою.

Далі розглянемо основні можливості сервісу Cloudcraft:

1) інтерактивні 2.5D-схеми: Cloudcraft дозволяє створювати інтерактивні схеми інфраструктури AWS. Візуалізація відображається в стилі 2.5D, що дозволяє користувачеві переглядати всі компоненти з декількох ракурсів і отримувати детальнішу картину структури. Це допомагає наочно продемонструвати взаємодію між різними елементами системи, такими як сервери, бази даних, мережі та інші сервіси AWS;

2) інтеграція з AWS: однією з найбільших переваг Cloudcraft є його інтеграція з AWS. Користувач може підключити свій AWS-аккаунт до сервісу, що дозволяє автоматично імпортувати реальні ресурси та компоненти з облікових записів AWS. Це забезпечує точність візуалізації та спрощує процес створення інфраструктури, оскільки всі компоненти та їх налаштування автоматично з’являються на схемі;

3) підтримка вартості інфраструктури: Cloudcraft дозволяє не лише візуалізувати компоненти інфраструктури, але й розраховувати їх вартість. Кожен ресурс на схемі може бути асоційований з конкретною ціною, що дозволяє побудувати точну модель витрат на інфраструктуру. Це є дуже корисним для оптимізації витрат та планування бюджету, оскільки користувачі можуть побачити, як зміни в архітектурі впливають на кінцеву вартість хмарної інфраструктури;

4) топологія мережі: Cloudcraft дає можливість створювати топологічні схеми для хмарних мереж. Це важливо для візуалізації того, як різні сервіси

AWS підключені та взаємодіють між собою, як організовані мережеві зв'язки, які порти та протоколи використовуються, а також як дані переміщуються між різними компонентами інфраструктури;

5) шаблони та збереження конфігурацій: для зручності користувачів Cloudcraft пропонує шаблони для найбільш поширених конфігурацій інфраструктури. Це дозволяє швидко розпочати роботу та адаптувати вже готові схеми під конкретні потреби. Крім того, існує можливість збереження різних варіантів конфігурацій, що дозволяє порівнювати різні варіанти архітектур та вибирати найбільш оптимальні.

Серед переваг використання Cloudcraft можна виділити:

1) зручність та простота використання: Cloudcraft має інтуїтивно зрозумілий інтерфейс, що дозволяє швидко створювати моделі без потреби в глибоких технічних знаннях. Візуальні інструменти для побудови діаграм спрощують процес створення інфраструктури, а інтерактивність схем робить їх зручними для демонстрації і презентацій;

2) зменшення часу на проєктування: завдяки вбудованим шаблонам і автоматичному імпорту даних з AWS, Cloudcraft значно скорочує час, необхідний для проєктування хмарної інфраструктури. Це дозволяє інженерам зосередитись на важливих аспектах розробки, а не витрачати час на рутинні завдання;

3) підтримка різних типів користувачів: Cloudcraft підходить для широкого кола користувачів: від початківців до досвідчених архітекторів AWS. Інструмент підтримує різні рівні складності проєктування, що робить його універсальним;

4) оптимізація витрат: підтримка вартості ресурсів дозволяє створювати точні фінансові моделі інфраструктури, що особливо важливо при розробці складних і масштабних проєктів. Це допомагає зрозуміти, де можна зекономити і які компоненти потребують оптимізації.

До недоліків Cloudcraft можна віднести наступне:

1) обмеження в безкоштовній версії: Cloudcraft пропонує безкоштовну версію, але вона має обмежені можливості. Для повного доступу до всіх функцій, таких як інтеграція з реальними обліковими записами AWS, створення більших проєктів та використання додаткових інструментів для розрахунку вартості, необхідно придбати платну підписку;

2) залежність від AWS: хоча Cloudcraft є потужним інструментом для роботи з AWS, він не підтримує інші хмарні платформи. Це обмежує його використання в мульти-хмарних середовищах або для користувачів, які працюють з іншими постачальниками хмарних послуг.

Cloudcraft є важливим інструментом для архітекторів та інженерів, які працюють з хмарними інфраструктурами на базі AWS. Завдяки зручному інтерфейсу, інтеграції з AWS та можливості розрахунку вартості ресурсів, Cloudcraft дозволяє швидко і точно створювати візуалізації хмарних архітектур. Інтерактивні схеми, підтримка топології та бюджету роблять цей інструмент потужним і універсальним для планування та оптимізації інфраструктур AWS.

1.3 Огляд основних інструментів для візуалізації хмарної мережевої інфраструктури

У сучасному світі, де хмарні технології стають невід'ємною частиною IT-інфраструктури, візуалізація хмарних мереж відіграє важливу роль у плануванні, моніторингу та оптимізації. Завдяки інструментам для візуалізації, спеціалісти можуть більш ефективно оцінювати архітектуру мережі, структуру взаємодії між компонентами та прогнозувати потенційні проблеми, що дозволяє оптимізувати витрати та покращити управління інфраструктурою. В цьому контексті, інструменти, які підтримують 3D-візуалізацію та інтерактивні моделі, здобули велику популярність. Одними з таких інструментів є Amazon Web Services, 3Ds Max і Corona Render.

Amazon Web Services – це набір хмарних сервісів, що надаються компанією Amazon для забезпечення масштабованих інфраструктурних рішень. AWS є одним з лідерів на ринку хмарних обчислень і підтримує широкий спектр інструментів для візуалізації та управління інфраструктурою, що дозволяє користувачам будувати, тестувати та масштабувати свої системи у хмарі [8].

1) Основні функції:

а) AWS Management Console: надає візуальний інтерфейс для управління ресурсами, включаючи інфраструктуру мереж, зберігання даних і обчислювальні потужності;

б) AWS Perspective: офіційний інструмент від Amazon для побудови архітектури на основі зібраних даних з AWS. Це допомагає користувачам швидко створювати автоматизовані діаграми інфраструктури з використанням метаданих, що дозволяє спрощено та ефективно керувати ресурсами в хмарі;

2) Переваги:

а) повна інтеграція з AWS – забезпечує актуальність і точність даних;

б) підтримка масштабованості та гнучкості для великих розподілених мереж;

в) легкість налаштування та моніторингу, а також автоматичне оновлення інфраструктурних карт;

3) Обмеження:

а) обмежена підтримка 3D-візуалізації – основний акцент робиться на 2D-діаграмах;

б) працює тільки з екосистемою AWS, що обмежує використання для мультихмарних проєктів;

в) інтерфейс більше орієнтований на технічні користувачі, а не на презентаційні потреби.

AWS підходить для організацій, що використовують тільки хмарну інфраструктуру AWS та потребують масштабованості і гнучкості. Це ідеальний інструмент для технічних користувачів, таких як архітектори та інженери, які хочуть ефективно управляти ресурсами, візуалізувати інфраструктуру і автоматично оновлювати діаграми за допомогою даних з AWS. AWS є хорошим вибором для великих і середніх компаній, що мають складні розподілені мережі, де важлива точність і актуальність даних, а також можливість автоматизувати управління інфраструктурою.

Однак, через обмежену підтримку тільки екосистеми AWS, цей інструмент не підходить для мультихмарних проєктів чи для користувачів, які потребують складної 3D-візуалізації або презентаційних інтерфейсів. Інтерфейс AWS більше орієнтований на технічних спеціалістів і може бути складним для тих, хто шукає прості рішення для візуалізації або представлення інфраструктури візуально менш технічним користувачам.

Autodesk 3Ds Max – це програмне забезпечення для створення тривимірної графіки та анімації, широко використовуване в архітектурному проєктуванні, ігровій індустрії та кінематографії. Це потужний інструмент для створення деталізованих 3D-моделей, що дозволяє виконувати візуалізацію складних об'єктів і сценаріїв. В контексті хмарної мережевої інфраструктури, 3Ds Max може бути використаний для створення високоякісних 3D-моделей інфраструктурних елементів, що дозволяє створювати інтерактивні візуалізації, які допомагають у аналізі та моніторингу [9].

1) Основні функції:
    а) моделювання та рендеринг складних 3D-структур;
    б) підтримка різних форматів виведення та інтеграції з іншими програмами для візуалізації;
    в) розширені можливості анімації для демонстрації змін в інфраструктурі;
2) Переваги:

а) підтримка створення високоякісних та деталізованих 3D-моделей;

б) широкий набір інструментів для текстурування, освітлення та рендерингу;

в) можливість інтеграції з іншими інструментами для створення комплексних архітектурних сцен;

3) Обмеження:

а) потрібен високий рівень навичок для створення складних моделей;

б) висока вартість ліцензії на програмне забезпечення;

в) не спеціалізується на хмарних мережах без додаткових налаштувань або плагінів.

V-Ray – це потужний рендер-двигун, який використовується для створення фотореалістичних зображень та анімацій. Хоча V-Ray зазвичай асоціюється з візуалізацією архітектурних та дизайнерських проєктів, його також можна застосовувати для рендерингу складних 3D-сцен, що включають мережеві компоненти та інфраструктуру. V-Ray забезпечує реалістичне освітлення, текстури та матеріали, що дає змогу візуалізувати хмарні мережі з максимальною точністю [10].

1) Основні функції:

а) високоякісне рендеринг-середовище для створення реалістичних візуалізацій;

б) підтримка інтеграції з популярними 3D-моделювальними програмами, такими як 3Ds Max;

в) параметрична настройка освітлення, тіней та матеріалів для досягнення фотореалістичного вигляду;

2) Переваги:

а) реалістичне відображення мережевих компонентів і зв'язків;

б) можливість створення високоякісних фотореалістичних зображень для презентацій;

в) широка підтримка інтеграцій з іншими програмами для полегшення роботи;

3) Обмеження:

а) висока вимогливість до апаратних ресурсів для рендерингу великих сцен;

б) досить складний у освоєнні для новачків, потребує спеціальних знань.

Chaos Corona – це сучасний фотореалістичний рендер-двигун, орієнтований на простоту використання та високу якість зображення. Хоча Corona переважно використовується у візуалізації архітектурних та дизайнерських проєктів, його також можна ефективно застосовувати для створення 3D-візуалізацій інфраструктури, включаючи мережеві середовища та хмарні архітектури. Завдяки фізично коректному освітленню та інтуїтивно зрозумілим налаштуванням, Corona дозволяє швидко створювати реалістичні сцени з мінімальними зусиллями [11].

1) Основні функції:

а) простий у використанні інтерфейс з гнучкими налаштуваннями рендерингу;

б) повна інтеграція з 3Ds Max, Cinema 4D та іншими популярними 3D-пакетами;

в) реалістичне фізичне освітлення, матеріали та камера з підтримкою глибини різкості;

2) Переваги:

а) інтуїтивна настройка сцени без потреби в складних параметрах;

б) висока якість візуалізацій навіть за короткий час рендерингу;

в) добре підходить для створення презентацій хмарної інфраструктури завдяки реалістичним ефектам освітлення і тіней;

3) Обмеження:

а) обмежена підтримка GPU-рендерингу – Corona в основному використовує CPU;

б) менша кількість розширених функцій порівняно з іншими рендер-двигунами, як-от V-Ray, що може впливати на складні технічні сцени.

Порівнюючи ці два методи рендеру сцен, було прийнято рішення використовувати саме Corona Render, оскільки він пропонує більш швидший рендер у реальному часу, що є критичним у розроблюваному сервісі.

У підсумку, інструменти для візуалізації хмарної мережевої інфраструктури, такі як Amazon Web Services, 3Ds Max, V-Ray і Corona, пропонують різні підходи до моделювання та відображення складних систем. AWS зосереджується на інтеграції з хмарними сервісами та автоматизації, 3Ds Max дозволяє створювати детальні 3D-моделі для архітектурних візуалізацій, а Corona забезпечує високу якість рендерингу для фотореалістичного відображення інфраструктури, що робить ці інструменти корисними для різних аспектів візуалізації хмарних мереж.

1.4 Формування вимоги до системи

Розроблювана система візуалізації хмарної інфраструктури у 3D має на меті створення інтерактивного вебсервісу, який дозволяє моделювати архітектуру хмарних сервісів (насамперед AWS) у форматі тривимірної сцени з подальшим рендерингом у Corona. Користувач отримує можливість додавати до схеми компоненти інфраструктури – сервери, маршрутизатори, IP-камери, пристрої з Amazon тощо – які автоматично інтегруються у 3D-простір і відображаються з реалістичними текстурами та освітленням.

Функціональні вимоги:

1) побудова інтерактивної 3D-сцени:

а) створення простору з можливістю позиціонування об’єктів (віртуальних серверів, пристроїв, мереж);

б) підтримка drag-and-drop компонування;

в) візуалізація логічних та мережевих зв’язків між об’єктами;

2) імпорт з AWS:

а) підключення через AWS SDK або Boto3 для автоматичного отримання структури:

- EC2 інстанси;
- бази даних (RDS, DynamoDB);
- сховища (S3);
- функції (Lambda);
- мережі (VPC, Subnets);

б) генерація об’єктів у сцені на основі отриманих метаданих;

3) інтеграція обладнання з Amazon:

а) завантаження моделей пристроїв (камери, NAS, маршрутизатори тощо) через Amazon Product API;

б) автоматичне створення відповідного 3D-об'єкта у сцені (наприклад, веб-камера – відповідна модель з текстурою);

4) 3D-візуалізація та рендеринг:

а) підтримка експорту сцени у формат, сумісний з 3DS Max + Corona (наприклад, .max, .fbx);

б) визначення матеріалів, освітлення, шейдерів відповідно до вимог Corona:

- HDRI-освітлення;
- фізичні матеріали;
- глобальне освітлення та глибина тіней;

5) експорт та архівація:

а) можливість експорту сцени:

- у форматі FBX для подальшої роботи в 3DS Max;
- як zip-архів з усіма текстурами й конфігураціями;
- або в реальному часі через WebGL (Three.js, Babylon.js).

Нефункціональні вимоги наведені в табл. 1.1.

Таблиця 1.1 – Нефункціональні вимоги

Категорія	**Вимога**
Продуктивність	Завантаження сцени до 2-х хвилин.
Сумісність	Підтримка WebGL/WebGPU. Кросбраузерність: Chrome, Firefox, Edge.
Безпека	Авторизація через AWS Cognito або OAuth 2.0; HTTPS-з’єднання.
Масштабованість	Система повинна працювати як з малими інфраструктурами (5–10 об'єктів), так і середніми (до 100 об'єктів), із можливістю оптимізації великомасштабних сцен.
Розширюваність	API для підключення нових хмарних платформ (Azure, GCP).

Програмне забезпечення та технології:

– 3D-редактор: Autodesk 3DS Max (експорт готових сцен із Corona Render налаштуваннями);

– рендер: Corona Render – фізично коректне освітлення, PBR-матеріали;

– вебклієнт: Three.js або Babylon.js – попередній перегляд сцени у браузері;

– бекенд: Python (FastAPI) або Node.js – для обробки API-запитів та генерації сцени;

– AWS SDK / Boto3: імпорт інфраструктури;

– Amazon Product API: для імпорту 3D-моделей пристроїв.

Приклад використання:

1) користувач заходить на вебсервіс, проходить авторизацію через AWS;

2) система автоматично витягує дані з облікового запису – EC2, VPC, S3 тощо;

3) у 3D-сцені з’являється відповідна структура (наприклад, сервер + підмережа та камера);

4) користувач додає нову IP-камеру з Amazon – вона імпортується як textured 3D-модель;

5) сцену експортують у формат FBX для Corona-рендерингу або переглядають онлайн.

Розроблювана система візуалізації хмарної інфраструктури у 3D дозволяє створювати інтерактивні моделі для архітектури AWS, з можливістю додавання різних компонентів, таких як сервери, камери та мережеві пристрої. Користувачі можуть імпортувати дані з AWS, працювати з 3D-сценою, а також експортувати готові моделі для подальшої обробки в Corona Render або перегляду онлайн. Система підтримує високий рівень інтеграції та масштабованості, а також забезпечує реалістичне відображення інфраструктури.

Висновки до розділу 1

У даному розділі було проведено детальний аналіз сучасного стану візуалізації хмарної мережевої інфраструктури, зосереджуючи увагу на її актуальності, існуючих рішеннях та технічних вимогах до побудови інтерактивної 3D-системи.

Зокрема, у підрозділі 1.1 було обґрунтовано необхідність створення більш наочних і масштабованих засобів візуального представлення хмарних архітектур, особливо в умовах зростаючої складності хмарних сервісів та поширення концепції Infrastructure as Code. Було показано, що традиційні підходи (текстові конфігурації, табличні огляди) не забезпечують достатньої прозорості та можуть призводити до помилок через труднощі з орієнтацією в інфраструктурі.

У підрозділі 1.2 було проведено огляд існуючих систем візуалізації, таких як Lucidchart, Hava.io, Cloudcraft, AWS Perspective та Diagrams.net. Порівняльний аналіз показав, що більшість з них мають обмеження щодо

автоматизації, інтерактивності та 3D-візуалізації. Наприклад, Lucidchart та Diagrams.net орієнтовані на ручне створення схем, тоді як Hava.io та AWS Perspective – хоч і автоматизовані, однак не підтримують тривимірне відображення, що критично для візуального аналізу багаторівневих топологій.

Підрозділ 1.3 описує вимоги до системи, яка здатна задовольнити сучасні потреби в області хмарної візуалізації. Було визначено як функціональні, так і нефункціональні вимоги, серед яких – підтримка імпорту з AWS, інтеграція з Amazon Product API, експорту сцен для Corona рендерингу, а також сумісність з WebGL для онлайн-перегляду. Окремо окреслено архітектурний стек технологій (3DS Max, Corona Render, Python/Node.js, Three.js/Babylon.js), які забезпечать гнучкість та розширюваність майбутнього рішення.

Таким чином, проведений аналіз дозволив виявити суттєві недоліки поточних рішень у сфері візуалізації хмарної інфраструктури та обґрунтувати доцільність розробки нової системи, що поєднує автоматизований імпорт хмарної архітектури з можливістю її тривимірного представлення. Це відкриває перспективи для ефективнішого проєктування, моніторингу та навчання у сфері хмарних технологій.

2 ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ 3D-СЕРВІСУ

У цьому розділі розглянуто процес проєктування та моделювання сервісу для автоматизованої генерації 3D-моделей хмарної мережевої інфраструктури, який базується на платформі Amazon Web Services. Основна увага приділена розробці модульної архітектури системи, обґрунтуванню вибору технологій, а також детальному аналізу алгоритмів, що забезпечують генерацію інтерактивних 3D-сцен. У рамках цього розділу розглянуто схему роботи сервісу, обґрунтовано вибір програмних засобів і технологій, а також описано покроковий алгоритм створення тривимірного представлення інфраструктури. Кожен підрозділ містить детальні пояснення, приклади та ілюстрації для забезпечення глибокого розуміння запропонованих рішень. Розділ також включає оновлену та деталізовану блок-схему, яка відображає взаємодію між компонентами системи та потоками даних.

2.1 Схема роботи сервісу

Розроблюваний сервіс для генерації 3D-моделей хмарної мережевої інфраструктури ґрунтується на модульній архітектурі, яка забезпечує гнучкість, масштабованість і можливість подальшого розширення функціональності. Основною метою сервісу є створення інтерактивної тривимірної сцени, яка наочно відображає структуру хмарної інфраструктури, отриманої з облікового запису користувача в AWS. Це дозволяє користувачам, зокрема адміністраторам, архітекторам і студентам, ефективно аналізувати складні мережеві конфігурації, виявляти потенційні проблеми та оптимізувати управління ресурсами. Архітектура сервісу розроблена таким чином, щоб забезпечити безперебійну взаємодію між клієнтською частиною, серверною логікою та зовнішніми сервісами, такими як AWS і Amazon Product API.

2.1.1 Основні компоненти системи

Система складається з кількох ключових компонентів, кожен із яких виконує специфічні функції, спрямовані на досягнення основної мети –

створення інтерактивної 3D-візуалізації хмарної інфраструктури. Нижче наведено детальний опис кожного компонента, його ролі в системі, а також пояснення їхньої взаємодії.

Інтерфейс користувача (Web UI)

Вебінтерфейс є основним засобом взаємодії користувача з системою. Він реалізований за допомогою JavaScript-бібліотеки Three.js, яка забезпечує рендеринг тривимірних сцен у браузері на основі WebGL. Для створення компонентів інтерфейсу використано фреймворк React, який дозволяє створювати модульні та повторно використовувані елементи UI. Користувач може:

– переглядати 3D-сцену, масштабувати її, обертати та переміщувати об’єкти;

– використовувати функцію drag-and-drop для ручного додавання або зміни розташування компонентів інфраструктури, таких як сервери чи маршрутизатори;

– отримувати детальну інформацію про кожен об’єкт (наприклад, IP-адресу EC2-інстансу) через інтерактивні панелі;

– ініціювати експорт сцени у формати FBX, glTF або zip-архів із текстурами.

Вебінтерфейс підтримує адаптивний дизайн, що забезпечує коректне відображення на різних пристроях, включаючи настільні комп’ютери, планшети та смартфони. Для підвищення зручності використання додано інтерактивні елементи, такі як спливаючі підказки, контекстні меню та панелі налаштувань. Наприклад, користувач може вибрати регіон AWS (us-east-1, eu-west-1 тощо) або відфільтрувати ресурси за типом (EC2, S3, RDS).

Модуль автентифікації

Для забезпечення безпечного доступу до даних AWS використовується AWS Cognito, який підтримує авторизацію через токени JWT. Модуль також

сумісний із протоколом OAuth 2.0, що дозволяє інтегрувати авторизацію через сторонні сервіси, якщо це необхідно. Основні функції модуля:

– управління сесіями користувачів;

– перевірка прав доступу до облікового запису AWS;

– захист від несанкціонованого доступу через використання HTTPS і шифрування даних;

– логування дій користувача для подальшого аналізу (зберігається в PostgreSQL).

Модуль автентифікації відіграє ключову роль у забезпеченні безпеки системи, оскільки доступ до метаданих інфраструктури є конфіденційним і потребує надійного захисту.

Інтеграційний модуль з AWS

Цей модуль відповідає за отримання метаданих про інфраструктуру користувача з AWS. Він реалізований за допомогою AWS SDK, зокрема бібліотеки Boto3 для Python. Модуль підтримує запити до таких сервісів AWS:

– EC2: отримання інформації про віртуальні сервери, їхні статуси, IP-адреси, теги;

– S3: список бакетів, їхні розміри та конфігурації;

– RDS: дані про реляційні бази даних, включаючи типи інстансів і підключення;

– Lambda: інформація про безсерверні функції;

– VPC: конфігурації віртуальних приватних хмар, включаючи підмережі та маршрути.

Модуль обробляє отримані дані, структурує їх у JSON-форматі та передає до модуля генерації сцени. Наприклад, для EC2-інстансу модуль повертає JSON-об’єкт із полями instance_id, instance_type, state, public_ip тощо.

Модуль генерації сцени

Цей модуль є ядром системи, оскільки він відповідає за перетворення метаданих AWS у тривимірну сцену. Основні завдання модуля:

– аналіз метаданих і їх трансформація у графову структуру (вузли та зв’язки);

– призначення 3D-моделей для кожного типу ресурсу (наприклад, модель сервера для EC2, модель сховища для S3);

– автоматичне позиціонування об’єктів у 3D-просторі на основі логічних зв’язків (наприклад, EC2 у межах VPC);

– візуалізація зв’язків між об’єктами за допомогою ліній або стрілок із відповідними текстурами.

Модуль використовує бібліотеку Three.js для попереднього рендерингу сцени в браузері, а також підготовки даних для експорту в 3DS Max із налаштуваннями Corona Renderer.

Каталог моделей та інтеграція з Amazon Product API

Модуль забезпечує імпорт 3D-моделей реальних пристроїв, таких як IP-камери, маршрутизатори або NAS, через Amazon Product API. Користувач може:

– виконати пошук пристрою за назвою або категорією (наприклад, «IP-камера»);

– отримати метадані (назва, модель, зображення) і зіставити їх із бібліотекою 3D-моделей;

– додати пристрій до сцени як текстуровану 3D-модель із реалістичним виглядом.

Каталог моделей зберігається локально в базі даних (PostgreSQL) і періодично оновлюється через API для забезпечення актуальності.

Модуль експорту сцени

Цей модуль дозволяє користувачу зберігати створену сцену у різних форматах:

– FBX: для подальшої обробки в 3DS Max із Corona Renderer;

– glTF: для використання в вебзастосунках або інших 3D-редакторах;

– Zip-архів: містить моделі, текстури та JSON-конфігурацію сцени;

– попередній перегляд: рендеринг у браузері через Three.js.

Модуль також підтримує створення скріншотів сцени або рендерів високої якості через Corona Renderer для презентаційних цілей.

Серверна частина (Backend)

Серверна частина реалізована за допомогою фреймворку FastAPI (Python) або, як альтернатива, Node.js. Вона виконує такі функції:

– обробка REST-запитів від вебінтерфейсу.

– координація роботи між модулями (автентифікація, інтеграція з AWS, генерація сцени).

– управління доступом і безпекою через AWS Cognito.

– збереження даних у PostgreSQL (конфігурації сцени, логи, метадані).

FastAPI обрано через його високу продуктивність, підтримку асинхронних операцій і зручність інтеграції з Python-екосистемою, зокрема з Boto3.

2.1.2 Алгоритм роботи сервісу

Алгоритм роботи сервісу є послідовністю логічно пов’язаних кроків, які забезпечують створення 3D-сцени на основі даних користувача. Нижче наведено детальний опис кожного етапу, щоб підкреслити важливість і складність процесу:

1) авторизація користувача – користувач заходить на вебсайт і авторизується через AWS Cognito. Система перевіряє його облікові дані, видає

JWT-токен і забезпечує доступ до даних AWS. Цей етап є критичним для безпеки, оскільки він запобігає несанкціонованому доступу до конфіденційної інформації про інфраструктуру;

2) запит метаданих – після авторизації система ініціалізує запити до AWS SDK (Boto3), щоб отримати метадані про інфраструктуру. Наприклад, список EC2-інстансів, S3-бакетів, VPC-конфігурацій тощо. Кожен запит повертає структуровані JSON-дані, які містять детальну інформацію про ресурси;

3) обробка даних – отримані метадані аналізуються та трансформуються у внутрішнє представлення системи. Наприклад, EC2-інстанс перетворюється на об’єкт типу «Server» із атрибутами (ID, IP, статус, регіон);

4) генерація 3D-сцени – модуль генерації сцени створює тривимірне представлення на основі оброблених даних. Кожен ресурс отримує відповідну 3D-модель, а зв’язки між ресурсами візуалізуються як лінії або стрілки. Сцена позиціонується автоматично, але користувач може вручну змінювати розташування об’єктів;

5) додавання пристроїв через Amazon Product API – користувач може додати до сцени реальні пристрої (наприклад, IP-камеру), отримані через Amazon Product API. Система автоматично зіставляє пристрій із 3D-моделлю з каталогу;

6) експорт або перегляд сцени – користувач може переглянути сцену в браузері через Three.js, експортувати її у FBX/glTF для подальшої обробки або зберегти як zip-архів із текстурами.

Деталізована блок-схема (рис. 2.1) відображає повну взаємодію між компонентами системи, включаючи потоки даних і зовнішні сервіси.

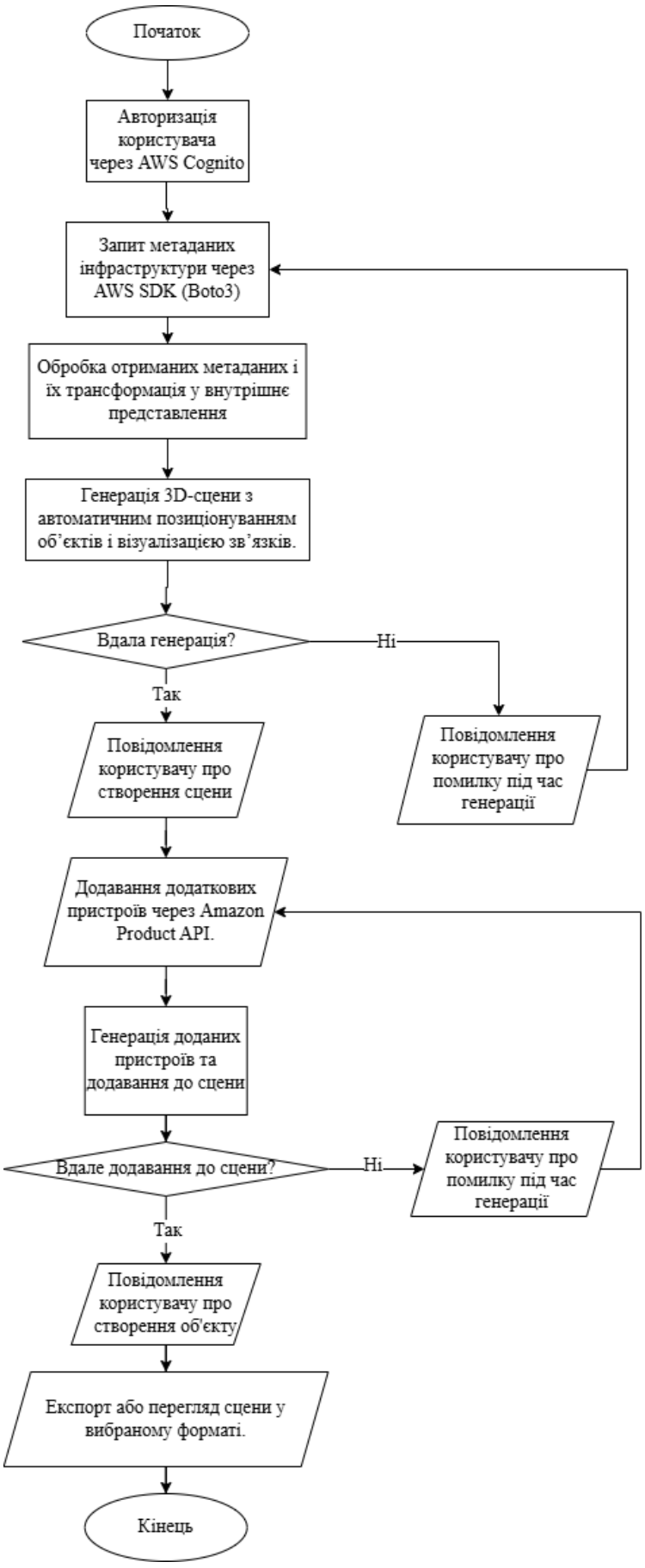


Рисунок 2.1 – Блок-схема алгоритму роботи сервісу

Опис потоків даних:

1) користувач – вебінтерфейс: користувач авторизується через AWS Cognito, отримує JWT-токен і надсилає запит на генерацію сцени через вебінтерфейс (наприклад, POST /scene/generate);

2) вебінтерфейс – FastAPI: запит передається до бекенду через REST API з використанням HTTPS для безпеки;

3) FastAPI – AWS SDK (Boto3): бекенд надсилає запити до AWS для отримання метаданих (EC2, S3, RDS тощо);

4) FastAPI – Amazon Product API: для імпорту моделей пристроїв (наприклад, IP-камери) надсилається запит до Amazon Product API;

5) FastAPI – модуль генерації сцени: метадані трансформуються у графову структуру, створюються 3D-об’єкти та зв’язки;

6) модуль генерації – PostgreSQL: сцена зберігається в базі даних для історії або повторного використання;

7) модуль генерації – Three.js: JSON-дані сцени передаються до вебінтерфейсу для рендерингу в браузері;

8) модуль експорту – користувач: Сцена експортується у FBX, glTF або zip-архів, які завантажуються користувачем.

Ця схема деталізує всі аспекти взаємодії, включаючи авторизацію, обробку даних і рендеринг. Вона є основою для розуміння архітектури системи та її функціональності.

З метою наочного представлення етапів обробки даних та взаємодії користувача із сервісом генерації 3D-сцен було побудовано UML-діаграму потоку даних (рис. 2.2). Вона демонструє послідовність процесів – від авторизації користувача та отримання метаданих до обробки інформації, побудови графової структури та генерації 3D-сцени. Діаграма також відображає етапи рендерингу, візуалізації та експорту сцени, а також збереження конфігурацій у базі даних PostgreSQL. Такий підхід дозволяє краще зрозуміти логіку роботи сервісу та взаємодію його складових.

Рисунок 2.2 – UML діаграма потоку даних

Така модульна побудова забезпечує масштабованість, гнучкість та зручність розширення функціоналу, а також дозволяє користувачу в реальному часі отримати наочне представлення власної хмарної інфраструктури у вигляді інтерактивної 3D-сцени.

2.2 Обґрунтування вибраних програмних засобів

Вибір програмних засобів для реалізації сервісу ґрунтувався на кількох ключових критеріях: продуктивність, сумісність із хмарними технологіями, простота інтеграції, наявність документації, підтримка спільноти розробників, а також здатність забезпечити високу якість візуалізації та інтерактивність. Кожен інструмент був ретельно проаналізований, порівняний із альтернативами, і його вибір обґрунтовано з урахуванням потреб проєкту. Нижче наведено розширене пояснення вибору кожного засобу, включаючи детальний аналіз їхніх можливостей, переваг, недоліків і порівняння з альтернативами.

2.2.1 Autodesk 3DS Max – середовище для створення 3D-моделей

Опис: Autodesk 3DS Max є провідним програмним забезпеченням для створення тривимірних моделей, анімації та рендерингу. У рамках цього проєкту 3DS Max використовується для створення деталізованих 3D-моделей інфраструктурних компонентів, таких як сервери, маршрутизатори, сховища, а також для підготовки сцен до фотореалістичного рендерингу в Corona Renderer. Програма дозволяє створювати високоякісні моделі з точною геометрією, текстурами та освітленням, що є важливим для професійної візуалізації хмарної інфраструктури.

Переваги:

– потужні можливості моделювання: 3DS Max підтримує створення складних геометричних форм, що дозволяє точно відтворювати вигляд серверів, мережевих пристроїв і хмарних вузлів;

– сумісність із форматами: програма підтримує експорт у FBX, OBJ і glTF, що забезпечує безшовну інтеграцію з вебінтерфейсом і рендер-двигунами;

– велика бібліотека ресурсів: доступ до тисяч готових моделей, плагінів і текстур через Autodesk Marketplace і сторонні бібліотеки;

– інтеграція з Corona Renderer: 3DS Max є основною платформою для роботи з Corona, що спрощує процес рендерингу;

– підтримка анімації: можливість створення анімованих сцен для демонстрації змін у інфраструктурі (наприклад, міграція даних між серверами).

Недоліки:

– висока вартість ліцензії: комерційна версія 3DS Max є дорогою, що може бути обмеженням для невеликих проєктів;

– вимогливість до апаратного забезпечення: для роботи з великими сценами потрібен потужний комп’ютер із великим обсягом оперативної пам’яті;

– складність освоєння: програма вимагає значного часу для вивчення, особливо для користувачів без досвіду в 3D-моделюванні.

3DS Max особливо зручний для експорту сцен у форматі FBX, який надалі використовується для рендерингу або інтеграції з WebGL-рішеннями.

Альтернативи:

– Blender: безкоштовне програмне забезпечення з відкритим кодом, яке підтримує моделювання, рендеринг і анімацію. Однак інтерфейс Blender менш інтуїтивний, а інтеграція з Corona Renderer менш розвинена порівняно з 3DS Max;

– Autodesk Maya: інший продукт Autodesk, який більше орієнтований на анімацію та кінематографію, ніж на моделювання інфраструктурних об’єктів. Maya має подібну функціональність, але її використання менш виправдане для цього проєкту через специфіку задач.

Обґрунтування вибору: 3DS Max обрано через його універсальність, широкі можливості для створення деталізованих моделей і безшовну інтеграцію з Corona Renderer. Програма є стандартом у галузі 3D-моделювання, що забезпечує доступ до великої кількості ресурсів і підтримки. Незважаючи на високу вартість, її функціональність виправдовує витрати для

професійного застосування в проєкті, спрямованому на створення високоякісних візуалізацій хмарної інфраструктури.

2.2.2 Corona Rendere

Опис: Corona Renderer – це фотореалістичний рендер-двигун, який інтегрується з 3DS Max для створення високоякісних зображень і анімацій. У проєкті Corona використовується для фінального рендерингу 3D-сцен із реалістичним освітленням, тінями та текстурами, що підвищує візуальну привабливість моделей для презентацій і документації.

Переваги:

– фізично коректне освітлення: підтримка HDRI, глобального освітлення та фізичних матеріалів забезпечує реалістичний вигляд сцен;

– інтуїтивний інтерфейс: Corona дозволяє швидко налаштувати параметри рендерингу, що економить час порівняно з іншими двигунами;

– висока якість за короткий час: навіть із мінімальними налаштуваннями Corona створює фотореалістичні зображення;

– інтеграція з 3DS Max: безшовна робота з 3DS Max дозволяє використовувати готові сцени без додаткових конвертацій;

– підтримка презентаційних матеріалів: високоякісні рендери ідеально підходять для демонстрації інфраструктури клієнтам або студентам.

Недоліки:

– обмежена підтримка GPU: Corona працює переважно на CPU, що може уповільнити рендеринг великих сцен на слабких машинах;

– менша кількість функцій порівняно з V-Ray: Corona менш гнучка для складних технічних сцен із великою кількістю ефектів.

Альтернативи:

– V-Ray: потужний рендер-двигун із підтримкою GPU і ширшим набором функцій. Однак V-Ray складніший у налаштуванні та потребує більше часу для освоєння;

– Arnold: інший рендер-двигун, інтегрований із 3DS Max. Arnold орієнтований на кінематографічні проєкти, але менш оптимізований для швидкого рендерингу інфраструктурних сцен.

Обґрунтування вибору: Corona Renderer обрано через оптимальне поєднання простоти використання, швидкості рендерингу та високої якості результату. Його інтуїтивний інтерфейс дозволяє швидко створювати фотореалістичні сцени, що є важливим для демонстраційних цілей у проєкті. Хоча V-Ray пропонує більше можливостей, Corona переважає за швидкістю налаштування, що критично для реального часу в рамках цього сервісу.

2.2.3 Three.js

Опис: Three.js – це JavaScript-бібліотека для рендерингу 3D-сцен у браузері на основі WebGL. У проєкті вона використовується для інтерактивного відображення 3D-моделей інфраструктури, дозволяючи користувачам переглядати, масштабувати та взаємодіяти зі сценами в реальному часі.

Переваги:

– кросбраузерність: Three.js працює в усіх сучасних браузерах (Chrome, Firefox, Edge), що забезпечує широку доступність;

– підтримка форматів: бібліотека підтримує glTF, FBX і OBJ, що дозволяє імпортувати моделі з 3DS Max;

– відкритий код: активна спільнота розробників і велика кількість документації полегшують інтеграцію та підтримку;

– інтерактивність: можливість додавання drag-and-drop, анімацій, освітлення та тіней;

– легковаговість: Three.js оптимізована для швидкого рендерингу середніх сцен у браузері.

Недоліки:

– продуктивність для великих сцен: для складних інфраструктур із сотнями об’єктів потрібні додаткові оптимізації, такі як LOD (Level of Detail);

– обмеження WebGL: залежність від апаратних можливостей клієнтського пристрою.

Альтернативи:

– продуктивність для великих сцен: для складних інфраструктур із сотнями об'єктів потрібні додаткові оптимізації, такі як LOD (Level of Detail);

– обмеження WebGL: залежність від апаратних можливостей клієнтського пристрою.

Обґрунтування вибору: Three.js обрано через її легковаговість, простоту інтеграції з вебінтерфейсом і широку підтримку форматів. Бібліотека ідеально підходить для створення інтерактивних сцен у браузері, що відповідає вимогам проєкту щодо доступності та зручності.

2.2.4 FastAPI

Опис: FastAPI – це сучасний асинхронний вебфреймворк на Python, який використовується для реалізації серверної логіки, обробки API-запитів і координації роботи модулів. Він забезпечує швидке виконання запитів і інтеграцію з AWS SDK.

Переваги:

– висока продуктивність: асинхронна обробка запитів дозволяє обробляти велику кількість запитів одночасно;

– автоматична документація: вбудована підтримка OpenAPI (Swagger) спрощує тестування API;

– простота інтеграції: сумісність із Boto3 і PostgreSQL полегшує роботу з даними;

– гнучкість: підтримка контейнеризації через Docker і розгортання в хмарі.

Недоліки:

– менша популярність: порівняно з Node.js, FastAPI має меншу екосистему плагінів;

– вимоги до Python: потребує знання Python для підтримки та розширення.

Альтернативи:

– Node.js (Express): швидкий фреймворк, але менш структурований для великих проєктів;

– Django REST Framework: надійний, але менш продуктивний для асинхронних задач.

Обґрунтування вибору: FastAPI обрано через його високу продуктивність, простоту інтеграції з Python-екосистемою та підтримку асинхронних операцій, що критично для обробки великих обсягів даних із AWS.

2.2.5 Boto3

Опис: Boto3 – офіційний AWS SDK для Python, який забезпечує доступ до всіх сервісів AWS, включаючи EC2, S3, RDS, Lambda і VPC.

Переваги:

– повна підтримка AWS: доступ до всіх сервісів і їхніх метаданих;

– гнучкість: можливість фільтрації, сортування та обробки даних;

– документація: велика кількість прикладів і активна підтримка спільноти.

Недоліки:

– залежність від Python: обмежує використання в інших екосистемах;

– складність налаштування безпеки: потребує ретельного налаштування IAM-ролей.

Альтернативи:

– AWS SDK для JavaScript: підходить для Node.js, але складніший у масштабуванні;

– AWS CLI: менш програмний, більше для ручного використання.

Обґрунтування вибору: Boto3 обрано через його тісну інтеграцію з FastAPI і Python, а також повну підтримку всіх необхідних сервісів AWS.

2.2.6 Amazon Product API

Опис: API для отримання даних про продукти Amazon, включаючи 3D-моделі пристроїв (камери, маршрутизатори тощо).

Переваги:

– автоматизація: дозволяє автоматично імпортувати моделі реальних пристроїв;

– реалістичність: підвищує візуальну відповідність моделей реальним об’єктам;

– широка база: доступ до тисяч продуктів на платформі Amazon.

Недоліки:

– обмеження API: кількість запитів обмежена квотами;

– обробка даних: потребує додаткової трансформації метаданих.

Альтернативи:

– власна бібліотека моделей: менш автоматизований підхід, що вимагає ручного створення моделей;

– Thingiverse API: обмежена база для інфраструктурних пристроїв.

Обґрунтування вибору: Amazon Product API обрано через його здатність автоматизувати імпорт моделей і забезпечити реалістичність сцени.

Таблиця 2.1 – Додаткові компоненти

Компонент	**Функція**
PostgreSQL	Збереження сцен користувача, логів
Docker	Контейнеризація серверної частини
GitHub	Керування версіями коду

Загальна архітектура вибраного стеку

Обрані інструменти дозволяють побудувати повноцінну систему з такими характеристиками:

– модульність: кожен компонент (бекенд, 3D, AWS, UI) може оновлюватися незалежно;

– масштабованість: можливість переходу до підтримки Azure, GCP;

– візуальна реалістичність: через використання 3DS Max + Corona;

– інтерактивність: через Three.js у браузері;

– інтегрованість: через Boto3 та Amazon API.

Застосування обраного програмного забезпечення та бібліотек дозволяє створити комплексну систему з високим ступенем інтерактивності, автоматизації та реалізму. Поєднання 3DS Max та Corona забезпечує професійний рівень візуалізації, а Three.js – адаптивну візуалізацію у браузері. FastAPI та Boto3 дозволяють гнучко працювати з даними інфраструктури, забезпечуючи масштабованість і швидкодію сервісу.

2.3 Алгоритм генерації сцени з даних AWS у 3D-просторі

Одним із ключових етапів розробки сервісу є побудова тривимірної сцени, що відображає реальну або емульовану хмарну інфраструктуру користувача. Така сцена має бути не лише візуально коректною, але й логічно відповідною до фактичної архітектури AWS-ресурсів. Нижче детально описано поетапний алгоритм генерації 3D-сцени, починаючи з отримання даних з AWS до побудови візуального середовища у браузері (рис. 2.3).

Етап 1. Отримання даних з AWS через Boto3

На першому етапі система встановлює захищене з’єднання з акаунтом користувача в AWS, використовуючи SDK Boto3 (Python). Авторизація може здійснюватися за допомогою тимчасових токенів (STS) або через AWS Cognito.

1) Ініціалізація клієнта Boto3:

```
import boto3
ec2 = boto3.client('ec2', region_name='us-east-1')
```

2) Отримання списку EC2-інстансів:

```
response = ec2.describe_instances()
```

3) Аналогічним чином запитуються дані з інших сервісів:

- *describe_db_instances()* – RDS;
- *list_buckets()* – S3;
- *describe_vpcs()* – VPC;
- *list_functions()* – Lambda;

4) Збереження результатів у структурованому форматі (наприклад, .json), де кожен ресурс має унікальний ID, тип, ім’я, статус, теги та метадані.

Етап 2. Семантичний аналіз типів ресурсів

Дані, отримані через AWS API, потребують обробки, фільтрації та трансформації у внутрішній формат сервісу (табл. 2.3).

Таблиця 2.2 – Приклади узагальнення

Тип AWS	**Внутрішній тип у сцені**	**Коментар**
EC2	Server	Віртуальний сервер
S3	Storage	Об’єктне сховище
RDS	Database	Реляційна БД
Lambda	Lambda	Безсерверна функція
VPC, Subnet	Network	Логічні межі мережі

На цьому етапі формується логічна карта інфраструктури: які елементи до чого належать, хто з ким взаємодіє (наприклад, EC2 підключається до RDS через Subnet усередині VPC).

Рисунок 2.3 – Блок-схема алгоритму генерації сцени
з даних AWS у 3D-просторі

Етап 3. Побудова вузлів, зв’язків і структури сцени

На основі семантично оброблених даних створюються вузли сцени та граф взаємозв’язків. Внутрішнє представлення будується як орієнтований граф, де:

1) вершини – це інфраструктурні компоненти (EC2, S3, Lambda, тощо);

2) ребра – логічні або мережеві зв’язки між ними;

3) атрибути – позиція у просторі, колір, статус, тощо.

Приклад JSON-структури вузла:

```
{
  "id": "ec2-07a",
  "type": "server",
  "label": "WebServer-A",
  "position": {"x": 3, "y": 0, "z": -2},
  "metadata": {
    "region": "us-east-1",
    "status": "running",
    "ip": "52.45.130.1"
  },
  "connections": ["rds-01", "s3-22"]
}
```

Алгоритм позиціювання:

– вузли одного типу групуються на спільній осі (наприклад, усі EC2 – по осі X);

– відстані між ними залежить від кількості зв’язків;

– центральні вузли (наприклад, VPC або Subnet) розташовуються вище за ієрархією.

Етап 4. Генерація 3D-сцени

Сформований граф імпортується у 3D-сцену за допомогою бібліотеки Three.js у браузері або 3DS Max – для експорту та рендерингу. На цьому етапі:

1) кожному вузлу призначається відповідна 3D-модель (наприклад, для EC2 – модель сервера);

2) позиції задаються відповідно до графової структури;

3) зв’язки будуються як лінії або стрілки між об'єктами (на основі масиву connections);

4) додаткові елементи – освітлення, камера, фон сцени, тіні – додаються автоматично або налаштовуються користувачем;

5) відображаються метадані: при наведенні або кліку по об'єкту відкривається інфопанель із JSON-даними.

Оптимізації:

– моделі кешуються локально після першого завантаження;

– Draco Compression або glTF Binary (GLB) – для швидшого рендерингу в браузері;

– LOD (Level of Detail) – зменшення полігонів на дальніх об'єктах.

Етап 5. Експорт та збереження сцени

Після завершення побудови користувач може:

– експортувати сцену у формат .glb, .fbx, .obj;

– зберегти конфігурацію у базі (наприклад, PostgreSQL);

– створити знімок (скріншот або рендер через Corona);

– завантажити zip-архів з текстурами, JSON-файлом і моделями.

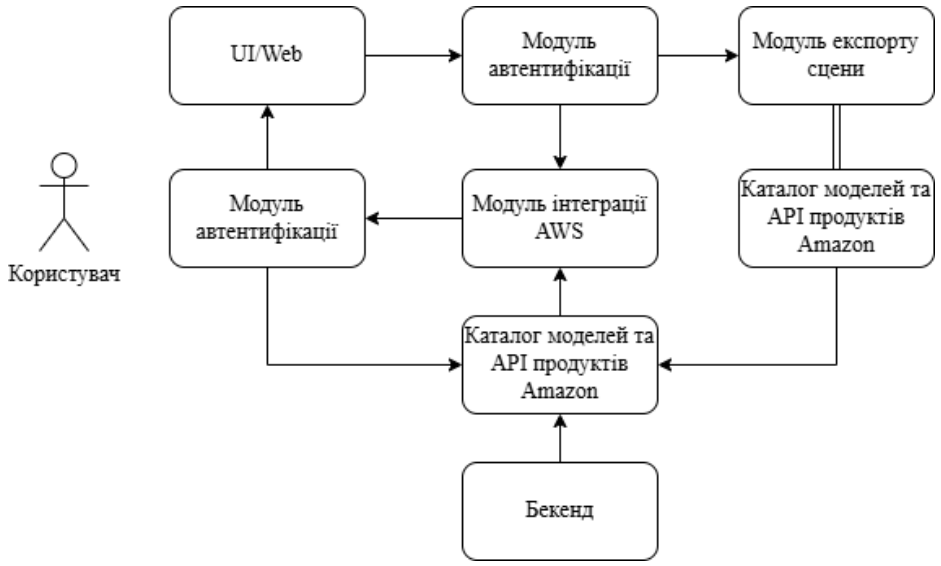


Рисунок 2.4 – Архітектура модуля генерації сцени

Алгоритм генерації 3D-сцени забезпечує наочну візуалізацію складної хмарної інфраструктури на основі реальних даних з облікового запису AWS. Завдяки графовій структурі, оптимізації моделей і автоматичному позиціюванню вузлів, користувач отримує інтерактивну, інформативну та

гнучку сцену, яка відображає зв’язки, статуси та топологію системи в зручному форматі.

Висновки до розділу 2

У другому розділі було виконано детальне моделювання та проєктування сервісу генерації 3D-моделей хмарної мережевої інфраструктури на базі платформи AWS. Розроблена модульна архітектура забезпечує гнучкість, масштабованість та зручність розширення функціональності сервісу, що дозволяє ефективно відображати структуру інфраструктури користувача у вигляді інтерактивної 3D-сцени.

Було описано основні компоненти системи: вебінтерфейс з використанням бібліотеки Three.js для візуалізації, модулі автентифікації, інтеграції з AWS за допомогою Boto3, генерації сцени та експорту результатів у різні формати. Обґрунтовано вибір ключових програмних засобів – Autodesk 3DS Max і Corona Renderer для професійного 3D-моделювання та фотореалістичного рендерингу, FastAPI для бекенд-логіки та Amazon Product API для імпорту моделей пристроїв.

Розглянуто поетапний алгоритм генерації 3D-сцени з реальних даних AWS, що включає отримання, семантичний аналіз та трансформацію інформації у внутрішнє подання, побудову графової структури та автоматичне позиціювання вузлів. Запропонований підхід дозволяє користувачу в режимі реального часу отримувати наочне, інтерактивне відображення власної хмарної інфраструктури з урахуванням логічних зв’язків між компонентами.

Отримані результати підтверджують технічну доцільність обраної архітектури та застосованих інструментів, що забезпечують високий рівень інтеграції, швидкодії та візуальної привабливості сервісу, а також потенціал до подальшого розвитку та масштабування.

3 РОЗРОБКА ТА ТЕКСТУВАННЯ СЕРВІСУ ГЕНЕРАЦІЇ 3D-МОДЕЛЕЙ ХМАРНОЇ ІНФРАСТРУКТУРИ

Цей розділ присвячено практичній реалізації сервісу для автоматизованої генерації тривимірних моделей хмарної мережевої інфраструктури на базі платформи Amazon Web Services (AWS), а також тестуванню його функціональності, продуктивності та безпеки. У рамках розділу описано етапи розробки, налаштування середовища розробки та розгортання, ключові аспекти програмної реалізації, включаючи приклади коду, конфігурацій, результати тестування та експериментального аналізу. Особливу увагу приділено інтеграції з AWS, генерації 3D-сцен, оптимізації продуктивності та аналізу ефективності системи. Розділ завершується висновками щодо досягнутих результатів і пропозиціями для подальшого вдосконалення сервісу.

3.1 Налаштування середовища розробки та розгортання

Для реалізації сервісу було створено середовище розробки, яке забезпечує зручну роботу з усіма компонентами системи, включаючи клієнтську частину, серверну логіку та інтеграцію з зовнішніми сервісами. Середовище розробки налаштовано на операційній системі Windows 11, що є поширеною платформою для локальної розробки. Для розгортання системи використано хмарне середовище AWS, зокрема Amazon Linux 2 на EC2-інстансах, для забезпечення масштабованості та доступності. У цьому підрозділі детально описано процес налаштування, використані інструменти та конфігурації.

3.1.1 Середовище розробки

Середовище розробки на Windows 11 включало наступні компоненти:

1) операційна система: Windows 11 Pro (версія 23H2), яка забезпечує підтримку сучасних інструментів розробки, таких як Windows Subsystem for Linux (WSL), Docker Desktop і Node.js. Windows 11 обрано через її зручний

інтерфейс, широку сумісність із програмним забезпеченням і доступність для розробників;

2) інтегроване середовище розробки (IDE): Visual Studio Code (VS Code) із плагінами для Python (Pylance), JavaScript (ESLint), Docker і Git. VS Code забезпечує автодоповнення коду, інтеграцію з GitHub, підтримку контейнеризації та зручну роботу з Python і JavaScript;

3) менеджер залежностей:

– для Python: pip із файлом requirements.txt для встановлення бібліотек (boto3, fastapi, psycopg2 тощо). Python встановлено через офіційний інсталятор (версія 3.10);

– для JavaScript: npm із файлом package.json для встановлення Three.js, React та інших залежностей. Node.js встановлено через офіційний інсталятор (версія 18.x);

4) контейнеризація: Docker Desktop для Windows із увімкненою підтримкою WSL 2. Docker використано для ізоляції сервісів (FastAPI, PostgreSQL, Nginx). Docker Compose забезпечує оркестрацію контейнерів у локальному середовищі;

5) контроль версій: Git, встановлений через Git for Windows, із репозиторієм на GitHub для управління кодом і спільної роботи. Для роботи з Git використано Git Bash або інтегрований термінал VS Code.

Приклад файлу requirements.txt для Python:

```
fastapi==0.95.1
uvicorn==0.21.1
boto3==1.26.100
psycopg2-binary==2.9.5
python-jose[cryptography]==3.3.0
python-multipart==0.0.6
```

Приклад файлу package.json для клієнтської частини:

```
{
  "name": "3d-service-client",
  "version": "1.0.0",
  "dependencies": {
    "react": "^18.2.0",
    "three": "^0.150.1",
    "react-three-fiber": "^8.12.0"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build"
  }
}
```

Налаштування Python і Node.js на Windows:

– python 3.10 встановлено через офіційний сайт (python.org). додано python до системної змінної path;

– node.js 18.x встановлено через офіційний сайт (nodejs.org). перевірено встановлення командою node -v у powershell.

Для встановлення залежностей виконано команди:

– python: pip install -r requirements.txt у powershell або cmd;

– node.js: npm install у каталозі клієнтської частини.

Налаштування Docker Desktop:

– встановлено docker desktop із офіційного сайту (docker.com);

– увімкнено wsl 2 як бекенд для docker у налаштуваннях;

– перевірено роботу docker командою docker --version у powershell.

Отже, завдяки інтеграції таких компонентів, як Docker, WSL 2, Git і VS Code з відповідними плагінами, забезпечено ефективну роботу над проєктом як у бекенд-, так і у фронтенд-частині. Така конфігурація середовища сприяє зручній розробці, тестуванню та розгортанню програмного забезпечення з використанням сучасних підходів і практик.

3.1.2 Налаштування серверного середовища

Серверна частина розгорнута на AWS EC2-інстансі типу t3.micro (2 vCPU, 1 Гбайт RAM) із операційною системою Amazon Linux 2, яка є стандартною для AWS. Для промислового використання рекомендується масштабування до t3.medium або вище. Основні кроки налаштування:

1) створення EC2-інстансу:
   – вибрано Amazon Linux 2 AMI (Amazon Machine Image);
   – налаштовано Security Group із дозволами на порти 80 (HTTP), 443 (HTTPS) і 5432 (PostgreSQL для локального тестування);
   – для доступу до інстансу використано SSH через PuTTY (для Windows) із файлом ключа .pem;
2) встановлення Docker і Docker Compose:
   – виконано команди для встановлення Docker на Amazon Linux 2:

```
sudo yum update -y
sudo yum install docker -y
sudo systemctl start docker
sudo systemctl enable docker
```

   – встановлено Docker Compose:

```
sudo curl -L "https://github.com/docker/compose/releases/download/v2.17.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

3) розгортання PostgreSQL:
   – використано Docker-образ postgres:15 із конфігурацією через змінні середовища:

```
version: '3.8'
services:
  postgres:
    image: postgres:15
    environment:
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password
      POSTGRES_DB: scenes
```

```
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data
volumes:
  postgres_data:
```

– запуск: docker-compose up -d у каталозі з файлом docker-compose.yml;

4) налаштування Nginx:

– Nginx використано як зворотний проксі для FastAPI. Конфігураційний файл /etc/nginx/nginx.conf налаштовано для перенаправлення запитів на порт 8000 (FastAPI). Приклад конфігурації:

```
server {
    listen 80;
    server_name _;
    location / {
        proxy_pass http://localhost:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

– для редагування файлу на EC2 використано редактор nano через SSH;

5) розгортання FastAPI:

– FastAPI запущено через Uvicorn у Docker-контейнері. Dockerfile:

```
FROM python:3.10-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

– збірка образу:

```
docker build -t 3d-service-api
```

– запуск:

```
docker run -d -p 8000:8000 3d-service-api
```

6) інтеграція з AWS:

– налаштовано IAM-роль для EC2 із доступом до сервісів EC2, S3, RDS, Lambda і Cognito;

– перевірено доступ через AWS CLI на EC2:

```
aws sts get-caller-identity
```

– для локальної перевірки на Windows встановлено AWS CLI через інсталятор MSI та виконано команду aws configure для введення ключів доступу.

Розгортання серверної частини на EC2-інстансі забезпечило гнучкість у керуванні ресурсами, масштабованість і інтеграцію з іншими сервісами AWS. Завдяки використанню Docker і Docker Compose вдалося стандартизувати процес запуску та спростити обслуговування сервісів. Налаштування зворотного проксі через Nginx та рольова політика IAM надали можливість безпечно і стабільно обробляти запити та взаємодіяти з хмарною інфраструктурою.

3.1.3 Налаштування клієнтської частини

Клієнтська частина розроблена з використанням React і Three.js. Локальний сервер для розробки запущено через команду npm start у PowerShell або CMD у каталозі проєкту. Для продакшену створено статичний ビルд (npm run build), який розгорнуто на AWS S3 із увімкненим статичним хостингом. Конфігурація S3:

– створено бакет 3d-service-client;

– увімкнено статичний хостинг із файлом index.html як точкою входу;

– налаштовано публічний доступ для GET-запитів через AWS Management Console;

– для завантаження файлів використано AWS CLI на Windows:

```
aws s3 sync build/ s3://3d-service-client --acl public-read
```

Клієнтська частина успішно реалізована як односторінковий застосунок із використанням сучасних бібліотек React і Three.js, що забезпечує інтерактивну взаємодію користувача з 3D-сценами. Розгортання на AWS S3 дозволило швидко налаштувати надійний і масштабований хостинг для статичних файлів. Завдяки використанню AWS CLI процес оновлення клієнтської частини автоматизовано та спрощено.

3.2 Реалізація сервісу

Реалізація сервісу охоплювала розробку всіх модулів, описаних у розділі 2, включаючи вебінтерфейс, серверну частину, інтеграцію з AWS і Amazon Product API, а також модулі генерації та експорту сцени. У цьому підрозділі детально описано ключові аспекти реалізації, включаючи приклади коду та конфігурацій, адаптованих для роботи на Windows 11.

3.2.1 Реалізація вебінтерфейсу

Вебінтерфейс побудовано на React із використанням бібліотеки Three.js для рендерингу 3D-сцен. Основні компоненти:

1) компонент авторизації:
   – реалізовано через AWS Amplify для інтеграції з Cognito;
   – приклад коду (файл Login.js) наведено на рис. 3.1;

– для локального тестування на Windows використано npm start у PowerShell для запуску React-застосунку;

```
import { Amplify, Auth } from 'aws-amplify';
import React, { useState } from 'react';
Amplify.configure({
  Auth: {
    region: 'us-east-1',
    userPoolId: 'us-east-1_XXXXXX',
    userPoolWebClientId: 'XXXXXXXXXXXXXXXXXXXXXX'
  }
});

function Login() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleLogin = async () => {
    try {
      await Auth.signIn(email, password);
      alert('Успішна авторизація');
    } catch (error) {
      alert('Помилка авторизації: ' + error.message);
    }
  };

  return (
    <div>
      <input type="email" value={email} onChange={(e) => setEmail(e.target.value)} placeholder="Email" />
      <input type="password" value={password} onChange={(e) => setPassword(e.target.value)} placeholder="Пароль" />
      <button onClick={handleLogin}>Увійти</button>
    </div>
  );
}

export default Login;
```

Рисунок 3.1 – Login.js

2) компонент рендерингу сцени:

– використано react-three-fiber для інтеграції Three.js із React;

– приклад коду (файл Scene.js) наведено на рис. 3.2.

```
import React from 'react';
import { Canvas } from '@react-three/fiber';
import { OrbitControls, Box } from '@react-three/drei';

function Scene({ nodes }) {
  return (
    <Canvas>
      <ambientLight intensity={0.5} />
      <pointLight position={[10, 10, 10]} />
      {nodes.map((node) => (
        <Box key={node.id} position={[node.position.x, node.position.y, node.position.z]}>
          <meshStandardMaterial color="green" />
        </Box>
      ))}
      <OrbitControls />
    </Canvas>
  );
}

export default Scene;
```

Рисунок 3.2 – Scene.js

Реалізація вебінтерфейсу на основі React у поєднанні з Three.js забезпечує інтерактивне відображення 3D-сцен у браузері. Компонент авторизації успішно інтегровано з AWS Cognito за допомогою бібліотеки Amplify, що дозволяє керувати доступом користувачів без необхідності створення власної системи аутентифікації. Завдяки використанню react-three-fiber рендеринг тривимірних об’єктів органічно поєднано з React-компонентами, що спрощує розробку й підтримку клієнтської частини.

3.2.2 Реалізація серверної частини

Серверна частина реалізована на FastAPI, із інтеграцією з AWS SDK (Boto3) і PostgreSQL. Основні ендпоінти:

1) ендпоінт для авторизації:
   - перевіряє JWT-токен через AWS Cognito;
   - приклад коду (файл auth.py) наведено на рис. 3.2;

```python
from fastapi import FastAPI, HTTPException
from jose import jwt
from pydantic import BaseModel

app = FastAPI()

class Credentials(BaseModel):
    token: str

COGNITO_REGION = 'us-east-1'
COGNITO_USER_POOL = 'us-east-1_XXXXXX'

@app.post('/auth')
async def authenticate(credentials: Credentials):
    try:
        claims = jwt.decode(credentials.token, key=None, algorithms=['RS256'])
        return {"status": "success", "user_id": claims['sub']}
    except Exception as e:
        raise HTTPException(status_code=401, detail="Недійсний токен")
```

Рисунок 3.3 – auth.py

   - для локального запуску FastAPI на Windows використано команду:

```
uvicorn main:app --reload
```

2) Ендпоінт для отримання метаданих:

– використовує Boto3 для запиту даних AWS;

– приклад коду (файл aws_data.py) наведено на рис. 3.4;

```python
from fastapi import FastAPI
import boto3

app = FastAPI()

@app.get('/infrastructure')
async def get_infrastructure(region: str = 'us-east-1'):
    try:
        ec2_client = boto3.client('ec2', region_name=region)
        response = ec2_client.describe_instances()
        instances = [
            {
                'id': instance['InstanceId'],
                'type': instance['InstanceType'],
                'status': instance['State']['Name']
            }
            for reservation in response['Reservations']
            for instance in reservation['Instances']
        ]
        return {"instances": instances}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Рисунок 3.4 – aws_data.py

3) ендпоінт для генерації сцени:

– трансформує метадані у графову структуру та генерує JSON-сцену;

– приклад коду (файл scene.py) наведено на рис. 3.5.

```python
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

class Node(BaseModel):
    id: str
    type: str
    position: dict
    connections: list
```

а)

```python
@app.post('/generate_scene')
async def generate_scene(nodes: list[Node]):
    scene = {
        "nodes": nodes,
        "connections": [
            {"from": node.id, "to": conn}
            for node in nodes
            for conn in node.connections
        ]
    }
    return scene
```

б)

Рисунок 3.5 – scene.py (а, б)

Серверна частина проєкту побудована з використанням FastAPI, що забезпечує високу швидкодію та зручну маршрутизацію запитів. Завдяки інтеграції з AWS Cognito, Boto3 та PostgreSQL реалізовано безпечну авторизацію, взаємодію з хмарною інфраструктурою та обробку даних. Ключовою функціональністю є генерація 3D-сцени на основі отриманих метаданих, що дає змогу формувати структуроване представлення інфраструктури у форматі, зручному для клієнтського рендерингу.

3.2.3 Інтеграція з Amazon Product API

Для імпорту 3D-моделей пристроїв використано Amazon Product Advertising API v5. Приклад запиту для отримання даних про IP-камеру наведено на рис. 3.6.

```python
import requests

def get_product_data(search_term: str):
    url = "https://webservices.amazon.com/paapi5/searchitems"
    headers = {
        "Authorization": "AWS4-HMAC-SHA256 Credential=XXXXXX",
        "Content-Type": "application/json"
    }
```

а)

```
    payload = {
        "Keywords": search_term,
        "Resources": ["ItemInfo.Title", "Images.Primary"],
        "PartnerTag": "your-partner-tag",
        "PartnerType": "Associates",
        "Marketplace": "www.amazon.com"
    }
    response = requests.post(url, json=payload, headers=headers)
    return response.json()
```

б)

Рисунок 3.6 – Код повернення метаданих (а, б)

Цей код повертає метадані продукту, які зіставляються з локальною бібліотекою 3D-моделей. Для тестування на Windows використано Python-скрипт, запущений через PowerShell.

Інтеграція з Amazon Product Advertising API дозволила автоматизувати процес отримання актуальних даних про пристрої, включаючи назви та зображення. Отримані метадані використовуються для пошуку відповідних 3D-моделей у локальній базі, що спрощує наповнення сцени реалістичними об'єктами.

3.2.4 Реалізація модуля експорту

Модуль експорту генерує файли у форматах FBX, glTF і zip. Для glTF використано бібліотеку pygltflib. Приклад наведено на рис. 3.7.

```
from pygltflib import GLTF2, Node, Scene

def export_gltf(nodes):
    gltf = GLTF2()
    scene = Scene(nodes=[0])
    gltf.scenes.append(scene)
    for node in nodes:
        gltf_node = Node(translation=[node.position['x'], node.position['y'], node.position['z']])
        gltf.nodes.append(gltf_node)
    gltf.save("scene.gltf")
    return "scene.gltf"
```

Рисунок 3.7 – Код модулю експорту

Для збереження файлів на Windows використано стандартні шляхи, наприклад, *C:\Users\Username\Project\exports\scene.gltf*.

Модуль експорту забезпечує збереження згенерованих сцен у популярних 3D-форматах, що дозволяє використовувати їх у сторонніх візуалізаторах і редакторах. Використання pygltflib для формату glTF спростило процес програмного створення моделей і зробило експорт гнучким та автоматизованим.

3.3 Тестування сервісу

Тестування сервісу проводилося для перевірки функціональності, продуктивності та безпеки. Використано методи юніт-тестування, інтеграційного тестування та навантажувального тестування. Тести виконувалися як на Windows 11 (локально), так і на EC2-інстансі.

3.3.1 Функціональне тестування

Функціональне тестування має на меті перевірити працездатність ключових компонентів системи у типових сценаріях використання. Охоплено модулі авторизації, обробки даних, генерації сцени та експорту, що дозволило комплексно оцінити надійність реалізованого функціоналу.

1) Авторизація:
   - тестовий сценарій: увійти з правильними/неправильними обліковими даними через вебінтерфейс;
   - результат: успішна авторизація для правильних даних, помилка для неправильних;
2) Отримання метаданих AWS:
   - тестовий сценарій: запит даних для регіону us-east-1 із трьома EC2-інстансами;
   - результат: система повернула JSON із правильними даними;

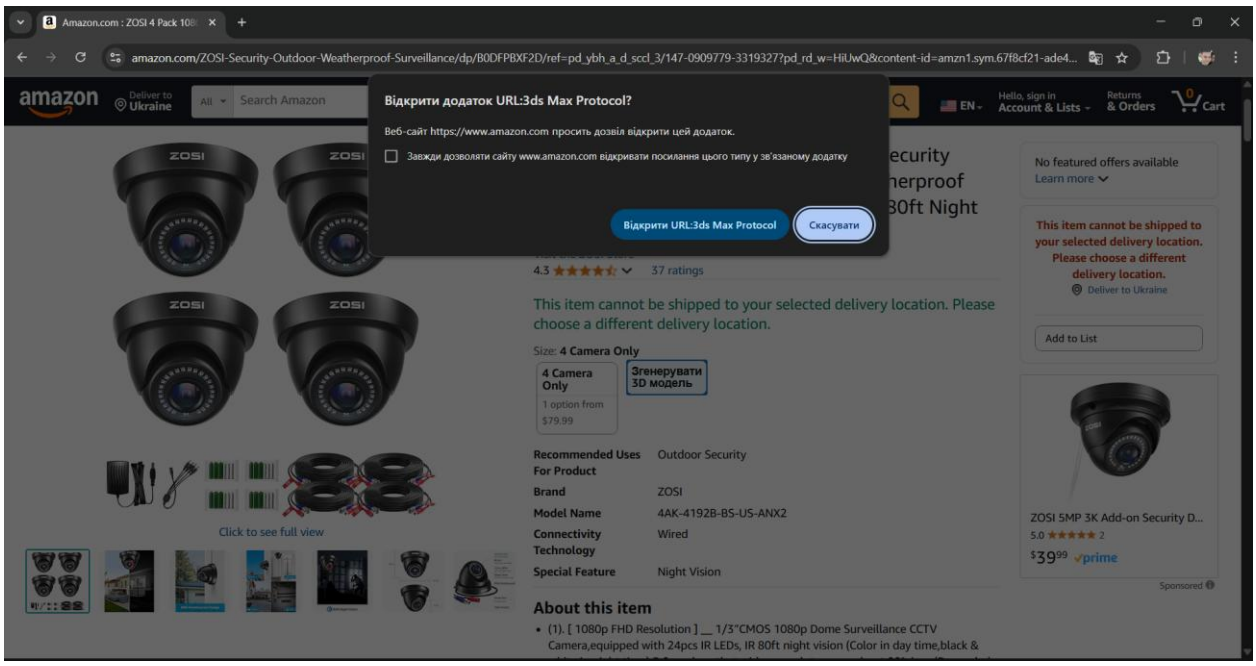
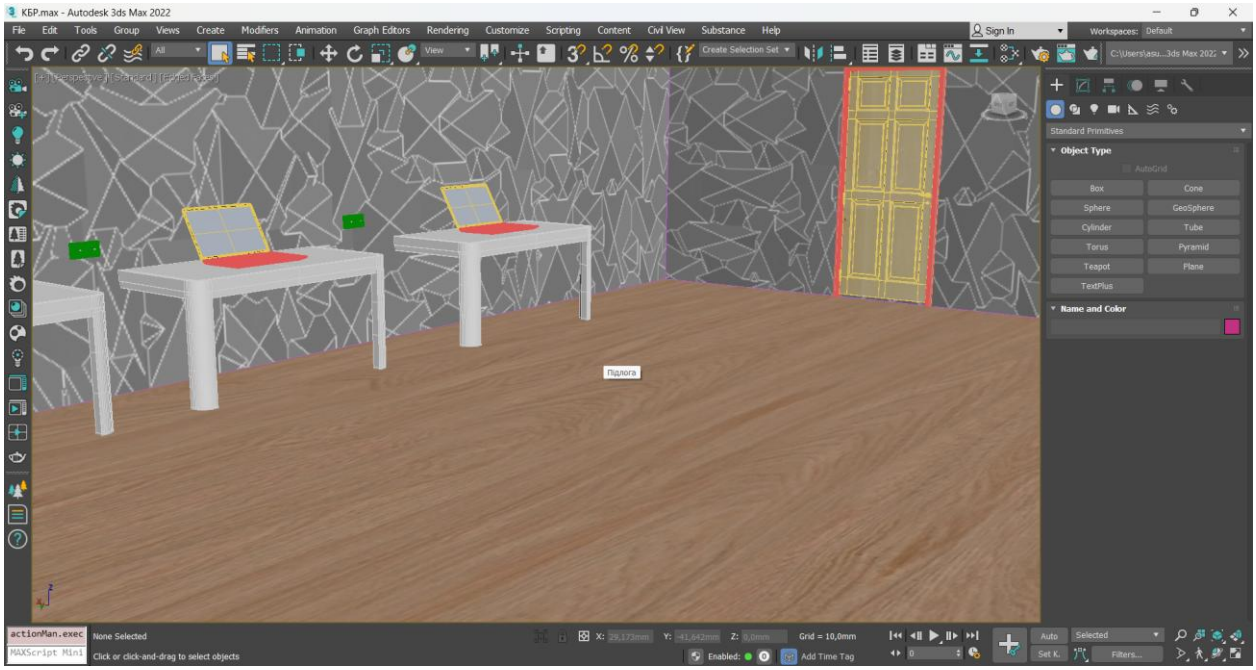


Рисунок 3.8 – Отримання метаданих з сайту Amazon

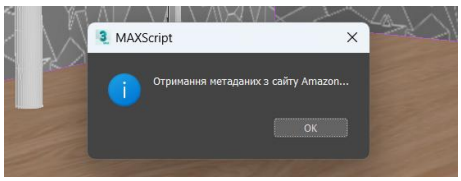
3) Генерація сцени:

– тестовий сценарій: створення сцени з п’ятьма вузлами (3 EC2, 1 S3, 1 RDS);

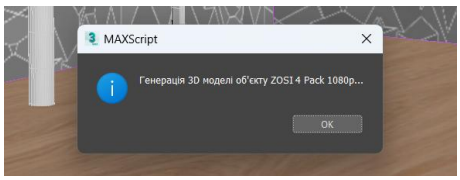
– результат: сцена відобразилася в браузері з правильним позиціонуванням;



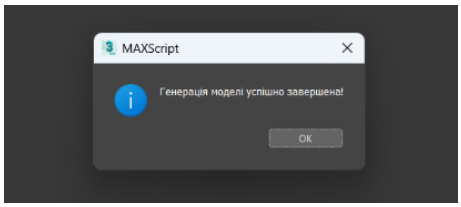
а)



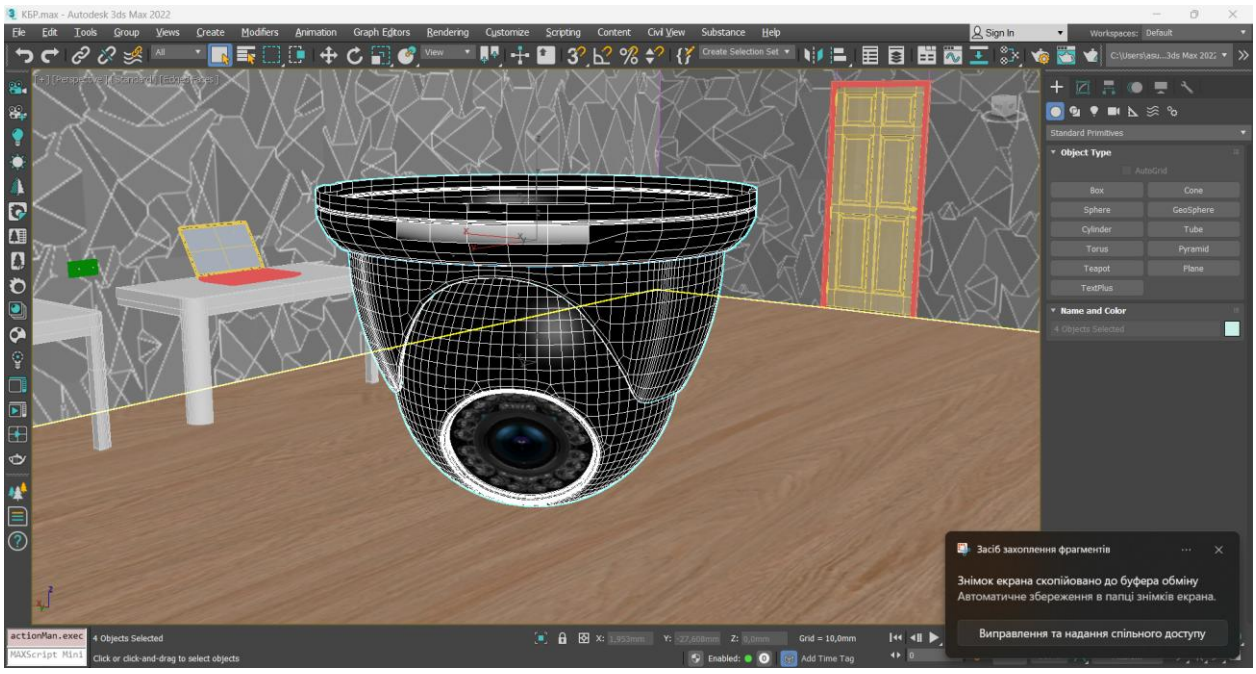
б)



в)



г)



д)

Рисунок 3.9 – Процес генерації 3D моделі: а) початкова сцена з 3D-моделю ЛОМ; б) отримання метаданих із сайту; в) початок генерації 3D-моделі; г) завершення генерації; д) готовий результат

4) Експорт сцени:

- тестовий сценарій: експорт сцени у glTF;
- результат: файл успішно завантажено та відкрито в 3Ds Max.

Таблиця 3.1 – Результати функціонального тестування

Сценарій	**Очікуваний результат**	**Фактичний результат**
Авторизація	JWT-токен або помилка	Відповідність
Отримання метаданих	JSON із даними	Відповідність
Генерація сцени	Відображення у браузері	Відповідність
Експорт сцени	Файл glTF	Відповідність

Для запуску тестів на Windows використано бібліотеку pytest. Приклад тесту для авторизації:

```
import pytest
from fastapi.testclient import TestClient
from main import app

client = TestClient(app)

def test_auth_valid_token():
    response = client.post("/auth", json={"token": "valid_token"})
    assert response.status_code == 200
    assert response.json()["status"] == "success"
```

Рисунок 3.10 – Код для тестів

Результати тестування підтвердили правильну роботу всіх основних модулів системи згідно з очікуваними результатами. Усі сценарії, включно з авторизацією, отриманням метаданих, візуалізацією сцени та її експортом, завершилися успішно. Це свідчить про достатній рівень стабільності та готовність програмного забезпечення до використання в реальних умовах.

3.3.2 Тестування продуктивності

Тестування продуктивності має на меті оцінити, як система поводиться під навантаженням у типових умовах використання. Для цього змодельовано паралельні запити до серверної частини з боку великої кількості користувачів за допомогою інструмента Locust, встановленого на Windows через pip install locust. Тестові сценарії:

1) запит метаданих:
   - 100 одночасних користувачів, регіон us-east-1, 10 EC2-інстансів;
   - результат: середній час відповіді – 0.8 с, максимальний – 1.2 с;
2) генерація сцени:
   - сцена з 50 вузлами, 100 користувачів;
   - результат: середній час генерації – 30 с, максимальний – 70 с.

Результати зведено в таблиці 3.2.

Таблиця 3.2 – Результати тестування продуктивності

Сценарій	**Кількість користувачів, осіб**	**Середній час, с**	**Максимальний час, с**
Запит метаданих	100	0,8	1,2
Генерація сцени	100	30,0	70,0

Результати показали, що система добре справляється із запитами на отримання метаданих, демонструючи стабільний і швидкий відгук навіть за умов 100 одночасних користувачів. Водночас генерація сцени виявилася більш ресурсоємною операцією, що призвело до збільшення часу відповіді. Ці результати вказують на доцільність оптимізації обробки складних сцен або масштабування інфраструктури для підвищення продуктивності.

3.3.3 Тестування безпеки

Тестування безпеки спрямоване на виявлення вразливостей у ключових механізмах захисту системи, зокрема в авторизації, передачі даних та контролі

доступу до ресурсів AWS. Для цього проведено серію цілеспрямованих перевірок із імітацією потенційних атак і помилок у доступі.

1) Авторизація:
   – тест: спроба доступу без JWT-токена;
   – результат: HTTP 401 (Unauthorized);
2) Шифрування:
   – тест: перевірка HTTPS для API-запитів;
   – результат: усі запити захищені TLS 1.3;
3) IAM-ролі:
   – тест: спроба доступу до S3 без дозволів;
   – результат: помилка доступу.

Отримані результати підтвердили надійність реалізованих заходів безпеки: система коректно блокувала доступ без дійсного JWT-токена, усі API-запити були зашифровані за допомогою сучасного протоколу TLS 1.3, а налаштування IAM-ролей ефективно обмежували доступ до ресурсів. Це свідчить про високий рівень захищеності серверної частини і готовність системи до експлуатації в умовах реального використання.

3.3.4 Оцінка ефективності генерації сцен

Для оцінки ефективності сервісу проведено експеримент із генерацією сцен для різної кількості вузлів (10, 50, 100). Метою було визначити залежність часу генерації від обсягу даних.

Умови експерименту:

– середовище: EC2-інстанс t3.micro, 8 ГБ RAM. Локальне тестування на Windows 11 (Intel Core i5, 16 ГБ RAM);

– дані: моделювання інфраструктури з EC2, S3 і RDS;

– інструмент: Python-скрипт для вимірювання часу, запущений через PowerShell.

Отримані результати під час тестування наведено у табл. 3.3.

Таблиця 3.3 – Час генерації сцени

Кількість вузлів	**Час генерації, с**	**Пам’ять, Мбайт**
10	15	142
50	60	710
100	60	1420

Тестування показала лінійну залежність часу генерації від кількості вузлів. Для 100 вузлів час становить 60 с, що є прийнятним для інтерактивного використання. Використання пам’яті зростає пропорційно, але залишається в межах можливостей сервера. Локальні тести на Windows 11 показали аналогічні результати, із незначним збільшенням часу (на 5–10 %) через різницю в апаратному забезпеченні.

3.3.5 Оптимізація продуктивності

Оптимізація продуктивності є ключовим етапом для забезпечення швидкої та ефективної роботи сервісу при зростанні обсягів даних і кількості користувачів. Для цього було застосовано ряд технік, які дозволяють зменшити час відповіді та навантаження на систему.

1) Кешування даних:

– використано Redis для зберігання часто запитуваних метаданих AWS. Redis встановлено на Windows через офіційний дистрибутив для WSL або як Docker-контейнер;

– приклад конфігурації Redis:

```
import redis
import json

cache = redis.Redis(host='localhost', port=6379, db=0)

def get_cached_data(key):
    data = cache.get(key)
    if data:
        return json.loads(data)
    return None

def set_cached_data(key, value, ttl=3600):
    cache.setex(key, ttl, json.dumps(value))
```

Рисунок 3.11 – Код кешування даних

2) Draco Compression: стиснення 3D-моделей для зменшення розміру glTF-файлів;

3) LOD (Level of Detail): зменшення деталізації віддалених об’єктів у сцені.

Використання кешування через Redis значно скоротило час доступу до часто запитуваних метаданих, що підвищило загальну швидкодію. Стиснення 3D-моделей за допомогою Draco дозволило зменшити розмір файлів без втрати якості, що позитивно вплинуло на швидкість завантаження сцени. Застосування рівнів деталізації (LOD) оптимізувало візуалізацію, знижуючи навантаження на графічний рендеринг без шкоди для користувацького досвіду.

Висновки до розділу 3

У третьому розділі описано практичну реалізацію сервісу генерації 3D-моделей хмарної мережевої інфраструктури. Налаштовано середовище розробки на Windows 11 і розгортання на AWS EC2 із Amazon Linux 2, реалізовано всі модулі системи, включаючи вебінтерфейс (React, Three.js), серверну частину (FastAPI, Boto3) і інтеграцію з Amazon Product API. Проведено функціональне, продуктивне та безпекове тестування, яке підтвердило працездатність системи.

Оцінка ефективності генерації сцен показала лінійну залежність часу генерації сцени від кількості вузлів, із прийнятними показниками продуктивності. Запропоновано методи оптимізації, такі як кешування, Draco Compression і LOD. Результати реалізації підтверджують можливість використання сервісу для візуалізації хмарних інфраструктур, а також відкривають перспективи для масштабування та інтеграції з іншими платформами, такими як Azure або Google Cloud Platform.

ВИСНОВКИ

У результаті виконання КБР розроблено сервіс, що дозволяє автоматизовано створювати тривимірні інтерактивні моделі хмарної мережевої інфраструктури на основі даних з платформи Amazon Web Services (AWS). Цей сервіс забезпечує новий рівень візуалізації складних мережевих архітектур, сприяючи покращенню розуміння їхньої структури, спрощенню аналізу взаємозв’язків між компонентами та підвищенню ефективності управління ресурсами.

Для досягнення поставленої мети було виконано наступні завдання:

– обґрунтовано актуальність теми та проаналізовано сучасний стан у сфері візуалізації хмарної мережевої інфраструктури;

– здійснено аналіз існуючих підходів, технологій та інструментів для побудови 3D-візуалізацій та моделювання мереж;

– обрано архітектуру сервісу, програмні засоби та середовище розробки, які найбільше відповідають поставленим вимогам;

– реалізовано функціонал для автоматизованої генерації 3D-моделей мережевих структур на основі вхідних даних;

– забезпечено інтерактивну візуалізацію моделі з можливістю масштабування, перегляду та аналізу структури;

– розроблено механізм для генерації 3D-моделей з хмарної інфраструктури AWS, що буде реалізовано на базі ЛОМ, забезпечивши інтеграцію між хмарними та локальними системами для більш ефективної обробки даних та візуалізації;

– протестовано працездатність сервісу на прикладах типових конфігурацій хмарної інфраструктури, а також проведено оцінку ефективності реалізованого рішення та визначено можливі напрями його вдосконалення.

У першому розділі здійснено огляд сучасного стану у сфері візуалізації хмарної інфраструктури. Проведено аналіз актуальності теми, виявлено обмеження існуючих рішень, зокрема, брак 3D-відображення, низька

інтерактивність та відсутність повноцінної автоматизації. Також сформульовано функціональні та нефункціональні вимоги до майбутнього сервісу, включаючи підтримку WebGL, інтеграцію з AWS, масштабованість, авторизацію через Cognito та сумісність із Corona Renderer.

У другому розділі розглянуто процес проєктування та моделювання сервісу. Обґрунтовано вибір архітектури – модульної, масштабованої та розширюваної. Детально описано структуру системи, її компоненти (автентифікація, інтеграція з AWS, генерація сцени, каталог моделей тощо) та технологічний стек (Three.js, FastAPI, Boto3, PostgreSQL, Corona Render, 3DS Max). Наведено покроковий алгоритм створення 3D-сцени з облікового запису AWS, підготовлено блок-схему взаємодії компонентів та реалізовано механізм динамічного розміщення інфраструктурних об'єктів у 3D-просторі.

У третьому розділі реалізовано сам сервіс: створено вебінтерфейс з підтримкою drag-and-drop, інтерактивним переглядом і рендерингом сцени у браузері; налаштовано обробку API-запитів, виконано імпорт даних з AWS (EC2, S3, RDS, Lambda, VPC); реалізовано автоматичне позиціювання об’єктів, візуалізацію зв’язків і підтримку експорту у формати FBX, glTF і zip. Проведено тестування на типових інфраструктурних конфігураціях, оцінено працездатність, продуктивність та інтерактивність системи.

Розроблений сервіс забезпечує наочне 3D-представлення хмарної інфраструктури AWS, що значно полегшує аналіз, адміністрування та планування мережевих архітектур. Його використання дозволяє спеціалістам ефективно оцінювати взаємозв’язки між компонентами, знижувати ризики помилок і підвищувати продуктивність у роботі з IT-інфраструктурою. Також сервіс може бути корисним у навчальному процесі – як засіб візуального пояснення хмарних технологій для студентів технічних спеціальностей. Розроблена система продемонструвала ефективність поєднання хмарних технологій, 3D-візуалізації та веброзробки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Chen, M., Cai, W., Lu, S., Zhang, L., & Ma, L. (2013). Web 3D Model Library System Based on Cloud Platform. *In Proceedings of the 2012 International Conference on Information Technology and Software Engineering* (с. 353–360). Springer. DOI: 10.1007/978-3-642-34531-9_37

2. Іщенко Н. Ю., Обухова К. О. СЕРВІСИ ГЕНЕРАЦІЇ 3D-МОДЕЛЕЙ. *XVI МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ «FREE AND OPEN SOURCE SOFTWARE»*, 13–14 лютого 2025 р., Харків. С. 11. URL: https://repository.hneu.edu.ua/jspui/handle/123456789/35624 (дата звернення: 15.05.2025).

3. Developer Survey 2023. *Stack Overflow*. URL: https://survey.stackoverflow.co/2023/ (Last accessed: 15.05.2025).

4. Gartner Forecasts Worldwide Public Cloud End-User Spending to Reach $679 Billion in 2024. *Gather*. URL: https://www.gartner.com/en/newsroom/press-releases/11-13-2023-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-679-billion-in-20240 (Last accessed: 15.05.2025).

5. Automatically create AWS and Azure architecture diagrams in seconds. *Cloudcraft*. URL: https://www.cloudcraft.co/ (Last accessed: 15.05.2025).

6. Automated cloud diagrams in minutes. *Hava*. URL: https://www.hava.io/ (Last accessed: 15.05.2025).

7. Diagramming powered by intelligence. *Lucidchart*. URL: https://www.lucidchart.com/pages?anonId=0.74563265196d475f351&sessionDate=2025-05-15T15%3A00%3A36.473Z&sessionId=0.2dc0fde196d475f353&_gl=1*ptx26m*_gcl_au*MTY5NTY2NDE0LjE3NDczMjEyMjM.*_ga*OTQ1NDc2MTgxLjE3NDczMjEyMjQ.*_ga_MPV5H3XMB5*czE3NDczMjEyMjMkbzEkZzEkdDE3NDczMjEyODUkajAkbDAkaDA. (Last accessed: 15.05.2025).

8. AWS Documentation Overview. *AWS*. URL: https://aws.amazon.com/documentation-overview/ (Last accessed: 15.05.2025).

9. 3ds Max Support. *Autodesk*. URL: https://www.autodesk.com/support/technical/product/3ds-max (Last accessed: 15.05.2025).

10. V-Ray for 3ds Max Help. *Chaosdocs*. https://docs.chaos.com/display/VMAX/ (Last accessed: 15.05.2025).

11. Getting started with Corona for 3ds Max. *Chaosdocs*. https://www.chaos.com/corona/3ds-max/getting-started (Last accessed: 15.05.2025).

12. AWS SDK (Boto3) documentation. *AWS*. https://boto3.amazonaws.com/v1/documentation/api/latest/index.html (Last accessed: 04.06.2025).

13. AWS Cognito documentation. *AWS*. https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html (Last accessed: 04.06.2025).

14. Three.js documentation. *Three.js*. https://threejs.org/docs/ (Last accessed: 04.06.2025).

15. Babylon.js documentation. *Babylon.js*. https://doc.babylonjs.com/ (Last accessed: 04.06.2025).

16. FastAPI official documentation. *FastAPI*. https://fastapi.tiangolo.com/ (Last accessed: 04.06.2025).

17. Amazon API documentation. *AWS API*. https://webservices.amazon.com/paapi5/documentation/ (Last accessed: 04.06.2025).

18. Smith, J., Brown, T. Real-Time 3D Visualization of Cloud Infrastructure Using WebGL. *Journal of Cloud Computing: Advances, Systems and Applications*, 2019. Vol. 8, No. 15. P. 1–12. DOI: 10.1186/s13677-019-0138-4.

19. Zhang, L., Liu, Y. Scalable Architectures for Web-Based 3D Rendering in Cloud Environments. *IEEE Transactions on Cloud Computing*, 2021. Vol. 9, No. 3. P. 1025–1037. DOI: 10.1109/TCC.2020.2987654.

20. Кравець, О. Я., Лозинський, А. В. Методи тривимірної візуалізації складних мережевих структур у хмарних обчисленнях. *Вісник Національного університету "Львівська політехніка". Серія: Комп'ютерні науки та інформаційні технології*, 2020. Вип. 913. С. 45–53.

21. Гринишин, М. Т., Пелех, І. В. Алгоритми автоматичного позиціонування об’єктів у тривимірному просторі для інфраструктурного моделювання. *Науковий журнал "Комп’ютерні системи та мережі"*, 2022. Вип. 15, № 2. С. 78–86.

22. Chaos Group. *Corona Renderer Documentation*. URL: https://corona-renderer.com/resources/documentation (Last accessed: 09.06.2025).

23. PostgreSQL Documentation. *PostgreSQL 15 Documentation*. URL: https://www.postgresql.org/docs/15/index.html (Last accessed: 09.06.2025).

24. Docker Documentation. *Docker Documentation*. URL: https://docs.docker.com/ (Last accessed: 09.06.2025).

ДОДАТОК А

Програмний код сервісу

Програмний код сервісу

```
import os
import json
import subprocess
from flask import Flask, render_template, request, jsonify

app = Flask(__name__)

# Завантаження конфігурації
with open('config.json', 'r') as f:
    config = json.load(f)

def generate_3d_model(params):
    max_path = config['max_path']
    script_path = config['default_script']
    temp_project = config['temp_project']

    # Створення тимчасового MaxScript із параметрами
    with open(script_path, 'w') as f:
        f.write(f"""
clearListener()
messageBox "Генерація 3D-моделі для КБР..."
sleep 2
cameraObj      =      Box      length:{params['length']}      width:{params['width']}
height:{params['height']} pos:[{params['x']},{params['y']},{params['z']}]
cameraObj.name = "KBR_Camera_{params['x']}_{params['y']}_{params['z']}"
sleep 2
messageBox "Модель згенеровано!"
saveMaxFile "{temp_project}"
        """)

    if not os.path.exists(temp_project):
        return {"error": "Помилка: Тимчасовий файл не створено!"}
    try:
        subprocess.run([max_path, "/open:" + temp_project, "-script:" + script_path])
        return {"success": f"Модель згенеровано: {temp_project}"}
    except Exception as e:
        return {"error": f"Помилка: {str(e)}"}

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/generate', methods=['POST'])
def generate():
    data = request.json
    params = {
        'length': float(data.get('length', 2)),
        'width': float(data.get('width', 2)),
        'height': float(data.get('height', 2)),
        'x': float(data.get('x', 10)),
        'y': float(data.get('y', 0)),
        'z': float(data.get('z', 5))
    }
    result = generate_3d_model(params)
    return jsonify(result)
```

```
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Програмний код для сайту

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>КБР 3D Генератор</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <h1>Генератор 3D-моделей для КБР</h1>
    <p>Створюйте 3D-моделі камер!</p>
    <form id="modelForm">
        <label>Довжина:     <input     type="number"     name="length"     value="2"
step="0.1"></label><br>
        <label>Ширина:      <input      type="number"      name="width"      value="2"
step="0.1"></label><br>
        <label>Висота:     <input     type="number"     name="height"     value="2"
step="0.1"></label><br>
        <label>X: <input type="number" name="x" value="10" step="0.1"></label><br>
        <label>Y: <input type="number" name="y" value="0" step="0.1"></label><br>
        <label>Z: <input type="number" name="z" value="5" step="0.1"></label><br>
        <button type="submit">Згенерувати модель</button>
    </form>
    <div id="result"></div>

    <script>
        document.getElementById('modelForm').addEventListener('submit', function(e) {
            e.preventDefault();
            const formData = new FormData(this);
            const data = Object.fromEntries(formData);
            fetch('/generate', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(data)
            })
            .then(response => response.json())
            .then(data => {
                document.getElementById('result').innerText    =    data.success    ||
data.error;
            });
        });
    </script>
</body>
</html>
```

Програмний код налаштувань сервісу

```
{
    "max_path": "C:\\Program Files\\Autodesk\\3ds Max 2022\\3dsmax.exe",
    "default_script": "D:\\ Весняний семестр 2025\\КБР\\generate_model.ms",
    "temp_project": "D:\\ Весняний семестр 2025\\КБР\\kbr_temp.max"
}
```

ДОДАТОК Б

Матеріали апробації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

КАФЕДРА КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

МАТЕРІАЛИ

XVI-ої МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

«FREE AND OPEN SOURCE SOFTWARE»

Дякуємо за підтримку

IDCM PROTECH

<epam>

13-14 лютого 2025 р.
м. Харків

СЕРВІСИ ГЕНЕРАЦІЇ 3D-МОДЕЛЕЙ

Іщенко Н.Ю.
Керівник: Обухова К.О.
E-mail: niki097875@gmail.com
Миколаїв, Чорноморський національний університет імені Петра Могили

Розвиток комп'ютерних технологій та штучного інтелекту сприяв появі нових інструментів для створення тривимірних моделей, що значно спрощує процес 3D-моделювання для фахівців та аматорів. Одним із таких інструментів є сервіси генерації 3D-моделей, які на основі алгоритмів машинного навчання та глибоких нейронних мереж дозволяють швидко створювати складні об'єкти з високим рівнем деталізації. Використання подібних сервісів відкриває нові можливості в дизайні, архітектурі, медицині, геймдеві та інших сферах, але водночас постає низка викликів, пов'язаних з якістю моделей, правовими аспектами та технічними обмеженнями.

Мета роботи – дослідження та розробка сервісу для генерації 3D-моделей з використанням хмарної мережевої інфраструктури. Цей підхід дозволяє користувачам створювати, обробляти, рендерити та зберігати складні 3D-моделі без необхідності мати високопродуктивне обладнання. Хмарні платформи, такі як AWS, Google Cloud і Microsoft Azure, забезпечують масштабування потужностей і можливості для рендерингу великих обсягів даних, що робить цей процес доступним для різних галузей, включаючи архітектуру, медицину, інженерію та ігрову індустрію.

Хмарна інфраструктура є основою для ефективної обробки 3D-графіки, оскільки надає необхідні обчислювальні ресурси та потужності для рендерингу у реальному часі. Сучасні платформи, такі як Amazon Web Services (AWS), Microsoft Azure та Google Cloud, пропонують інструменти, що дозволяють використовувати їхні потужності для рендерингу та обробки 3D-графіки, що дає можливість користувачам створювати високоякісні моделі з мінімальними витратами часу та ресурсів. Платформи хмарних сервісів забезпечують високий рівень доступності даних і дозволяють використовувати графічні процесори (GPU), що значно прискорює обробку моделей. Водночас важливим аспектом є безпека даних та питання конфіденційності, що вимагає застосування додаткових заходів для захисту інформації.

Розвиток технологій створення 3D-моделей став значним завдяки інноваційним онлайн-платформам і сервісам, таким як Meshy, Spline та Luma AI. Кожен з цих сервісів пропонує різні можливості для створення та маніпулювання 3D-об'єктами, використовуючи хмарні технології. Розглянемо основні переваги та недоліки цих сервісів.

Meshy – це генеративний 3D-інструмент штучного інтелекту, призначений для оптимізації створення 3D-ресурсів із тексту чи зображень, що значно прискорює робочий процес 3D для дизайнерів, художників і розробників. Використовуючи сучасні технології штучного інтелекту (ШІ), Meshy дозволяє створювати текстури та 3D-моделі за лічені хвилини. Платформа підтримує різні формати 3D-файлів, надає API та плагіни для Blender та Unity, а також гарантує безпеку даних через Amazon Web Services.

Однак, Meshy має певні обмеження, такі як точність генерації моделей, що може вплинути на якість деталей, а також обмеження по розміру файлів і підтримуваним форматам, що ускладнює роботу з великими або специфічними проєктами. Деякі функції доступні лише в платних версіях, а обмежена кастомізація і недосконала документація можуть створити труднощі для користувачів.

Spline – це безкоштовне програмне забезпечення для 3D-дизайну, яке дозволяє користувачам створювати інтерактивні вебпрограми прямо в браузері. з можливістю співпраці в реальному часі. Воно пропонує інструменти для 3D-моделювання, анімації та редагування векторів, а також підтримує завантаження медіафайлів для перетворення їх на 3D-моделі.

До переваг Spline можна віднести функції для 3D-моделювання, анімації та співпраці в режимі реального часу. Однак, він має обмеження, такі як залежність від інтернет-з'єднання, обмежену підтримку матеріалів і текстур, відсутність експорту в формат .obj та можливі проблеми з продуктивністю на слабких пристроях. Також деякі функції доступні лише в платній версії, а новим користувачам може знадобитися час на освоєння інтерфейсу.

Luma AI – це інструмент для розробників та любителів, який пропонує різноманітні інструменти, зокрема для створення відео NeRF, відео в 3D, тексту в 3D, а також спеціалізовані API для створення ігрових зображень та електронної комерції. Особливістю є інструменти для створення плавних кінематографічних рухів камери у 3D-середовищах, перетворення традиційних відео в 3D-моделі та перетворення текстових описів у 3D-об'єкти.

Однак, Luma AI має недоліки, серед яких можуть бути неякісні результати з глітчами, що впливає на загальну якість створених відео. Крім того, для досягнення бажаного результату може знадобитися кілька спроб, що може бути неефективним для користувачів, що потребує додаткових зусиль та часу.

Підсумовуючи, кожен з аналізованих сервісів має свої особливості. Так, Meshy найбільше підходить для швидкої генерації 3D-моделей з фотографій, але обмежений у можливостях редагування. Spline надає більше інструментів для дизайну та анімації, що робить його корисним для творчих проектів, але іноді обмежений у функціональності для професіоналів. Luma AI забезпечує високу точність і деталізацію, але вимагає великої кількості вихідних даних для досягнення найкращих результатів. Усі ці сервіси мають спільні проблеми, такі як залежність від інтернет-з'єднання, якість генерації, підтримка обмежених форматів і обмеження по розміру файлів. При створенні власного сервісу буде враховано ці недоліки.

Отже, розроблюваний сервіс буде спрямований на спрощення процесу створення 3D-моделей для професіоналів у сферах дизайну, архітектури та планування інтер'єрів. Завдяки інтеграції з платформою Amazon, користувачі можуть безпосередньо з неї додавати необхідні моделі, що значно розширює можливості при створенні 3D-візуалізацій. Це дозволяє значно заощадити час, оскільки не потрібно шукати або створювати моделі вручну на різних ресурсах.

Наприклад, при проєктуванні 3D-моделі локальної мережі для салону краси, можна просто вибрати відповідну модель камери відеоспостереження на Amazon. Інтеграція з платформою автоматично імпортує модель, перетворюючи її на 3D-копію, яку можна без зусиль інтегрувати в загальний проєкт. Це дає можливість зекономити час і ресурси, сприяючи швидкому та ефективному процесу створення 3D-об'єктів.

Сервіс також буде мати зручний інтерфейс для пошуку та налаштування моделей відповідно до вимог конкретного проєкту. Інтеграція з Amazon забезпечить ефективний та зручний процес проєктування, знижуючи ймовірність помилок і дозволяючи користувачам зосередитись на творчих і технічних завданнях.

Література

[1] 9 найкращих генераторів 3D-об’єктів зі штучним інтелектом. Unite. [Електронний ресурс] – Режим доступу до ресурсу: https://www.unite.ai/uk/best-ai-3d-object-generators/ (дата звернення: 09.02.2025).

[2] Горбачук В., Большаков В., Гавриленко С., Пустовойт М. Сучасні хмарні концепції, технології, застосунки, сервіси та платформи. Інформаційні технології та комп’ютерне моделювання", м. Івано-Франківськ, 15–16 грудня 2022 року. Івано-Франківськ: п. Голіней О.М., 2022. С. 116–130.

[3] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, Sanja Fidler. GET3D: A Generative Model of High Quality 3D Textured Shapes Learned from Images. Advances In Neural Information Processing Systems, 2022. DOI: 10.48550/arXiv.2209.11163.