МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»

Факультет ракетно-космічної техніки

Кафедра геоінформаційних технологій та космічного моніторингу Землі

Пояснювальна записка до дипломного проєкту (роботи)

(тип кваліфікаційної роботи)

магістр
(освітній ступінь)

на тему: «Методика створення 3D моделі будівлі на інтерактивній картографічній основі в структурі WEB-додатку за допомогою бібліотеки Three.js»

ХАІ.407.462м.24О193.1804042 ПЗ

Виконав: студент(ка) 2 курсу групи № 462-м

Спеціальність 193 Геодезія та землеустрій
(код та найменування)

Освітня програма Геоінформаційні системи та технології
(найменування)

Крапива А.А.
(прізвище та ініціали студента(ки))

Керівник: Нечаусов А.С.
(прізвище та ініціали)

Рецензент: Полупан А. В.
(прізвище та ініціали)

Харків – 2024

Міністерство освіти і науки України
Національний аерокосмічний університет ім. М.Є. ЖУКОВСЬКОГО
«Харківський авіаційний інститут»

Факультет Ракетно-космічної техніки
Кафедра Геоінформаційних технологій та космічного моніторингу Землі
Рівень вищої освіти магістр
Спеціальність 193 «Геодезія та землеустрій»
Освітня програма Геоінформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедри, голова циклової комісії
Станіслав ГОРЕЛИК
(підпис) (прізвище та ініціали)
« 23 »жовтня 20 23 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Крапива Анна Андріївна
(прізвище, ім'я, по-батькові)

1. Тема випускної роботи Методика створення 3D моделі будівлі на інтерактивній картографічній основі в структурі WEB-додатку за допомогою бібліотеки Three.js
2. Керівник кваліфікаційної роботи к.т.н., доцент Нечаусов Артем Сергійович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом Університету №2001-уч від «15» листопада 2023 року
3. Термін подання студентом кваліфікаційної роботи 11.01.2024
4. Вихідні дані до проєкту (роботи):
1) інтерактивна картографічна основа OSM;
2) програмний комплекс 3D-моделювання Blender та ГІС розширення до нього;
3) інтегроване середовище розробки Visual Studio Code з фреймворками та бібліотеками (Bootstrap та Angular);
4) Java Script бібліотека 3D-моделювання Three.js.
5. Зміст пояснювальної записки (перелік завдань, які потрібно розв'язати)
Вступ. Загальна характеристика WEB-програмування. Вибір технологій для розробки WEB-додатку. Застосування WEB-програмування у ГІС. WEB-геоінформаційні системи. Тривимірні WEB-геоінформаційні системи. Аналіз програмного забезпечення для WEB-програмування. Розробка методики створення 3D моделі будівлі на інтерактивній картографічній основі в структурі WEB-додатку за допомогою бібліотеки Three.js. Побудова структурної схеми методики. Висновки.
6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

Структурна схема основних етапів роботи. створення 3D моделі будівлі на інтерактивній картографічній основі в структурі WEB-додатку за допомогою бібліотеки Three.js.

7. Консультанти розділів кваліфікаційного проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	Нечаусов А.С. *доцент*		

Нормоконтроль Красовська І.Г. « 15 » 01 2024 р.

8. Дата видачі завдання 23.10.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітки
1	Загальна характеристика WEB-програмування. Вибір технологій для розробки WEB-додатку.	23.10.23 – 27.10.22	
2	Застосування WEB-програмування у ГІС. WEB-геоінформаційні системи. Тривимірні WEB-геоінформаційні системи.	30.10.23 – 02.11.23	
3	Аналіз програмного забезпечення для WEB-програмування.	03.11.23 – 08.11.23	
4	Розробка методики створення 3D моделі будівлі на інтерактивній картографічній основі в структурі WEB-додатку за допомогою бібліотеки Three.js.	09.11.23 – 08.12.23	
5	Побудова структурної схеми методики. Оформлення графічних матеріалів.	11.12.23 – 15.12.23	
6	Написання пояснювальної записки.	18.12.23 – 11.01.24	

Студенка _____ Анна КРАПИВА
(підпис) (прізвище та ініціали)

Керівник дипломної (кваліфікаційної) роботи _____ Артем НЕЧАУСОВ
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної бакалаврської роботи містить: 112 сторінок, 42 малюнки, 4 таблиці, 23 посилань на використану літературу.

Об’єкт дослідження: процес створення та відтворення 3D моделей та сцен картографічної спрямованості на базі інтерактивних веб-додатків.

Предмет дослідження – методи створення 3D моделей будівель на інтерактивній картографічній основі в структурі WEB-додатку за допомогою бібліотеки Three.js.

Мета роботи: підвищення ефективності процесу створення інтерактивних веб-додатків для демонстрації тривимірних сцен із картографічним змістом, зокрема місцевості та забудови за рахунок поєднання сучасних засобів тривимірного моделювання, засобів картографування, обробки геоінформаційних даних та сучасних засобів веб-розробки у єдину універсальну методику.

Методи дослідження: аналіз програмно-аналітичних засобів відображення 3D-моделей у веб-додатках, розробка веб-додатку, моделювання тривимірного зображення на картографічній основі, моделювання сцени відображення тривимірного зображення в структурі веб-додатку.

У результаті роботи було отримано методику творення 3D моделі будівлі на інтерактивній картографічній основі в структурі WEB-додатку за допомогою бібліотеки Three.js. В результаті застосування методики на реальних даних, було розгорнуто веб-додаток, створено тривимірні моделі забудов студмістечка «ХАІ» у форматі glTF та KMZ архів моделі, розроблено тривимірну сцену візуалізації отриманих моделей у структурі веб-додатку.

Ключові слова: WEB-ДОДАТОК, 3D-МОДЕЛЬ, КАРТОГРАФІЧНА ОСНОВА, ТРИВИМІРНЕ ЗОБРАЖЕННЯ, VISUAL STUDIO CODE, THREE.JS, ANGULAR, ТЕКСТУРА, BLENDER, COLLADA, KMZ, GLTF.

REVIEW

Calculation and explanatory note for the course work: 112 pages, 42 drawings, 4 table, 23 references to the literature used.

Object of research: processes of creating and rendering 3D models and cartographic scenes based on interactive web applications.

Subject of research: methods of creating 3D models of buildings on an interactive cartographic basis in the structure of a WEB application using the Three.js library.

Objective: increasing the effectiveness of the process of creating interactive web applications for the demonstration of three-dimensional scenes with cartographic content, in particular terrain and buildings, by combining modern three-dimensional modeling tools, mapping tools, geoinformation data processing, and modern web development tools into a single universal methodology.

Research methods: analysis of software and analytical tools for displaying 3D models in web applications, development of a web application, modeling of a three-dimensional image on a cartographic basis, modeling of a three-dimensional image display scene in the structure of a web application.

As a result of the work, a method of creating a 3D model of a building on an interactive cartographic basis in the structure of a WEB application using the Three.js library was obtained. As a result of applying the methodology to real data, a web application was deployed, three-dimensional models of buildings of the "KHAI" campus were created in glTF and KMZ format, a model archive was developed, and a three-dimensional visualization scene of the obtained models was developed in the structure of the web application.

Keywords: WEB APPLICATION, 3D MODEL, MAPBASE, THREE-DIMENSIONAL IMAGE, VISUAL STUDIO CODE, THREE.JS, ANGULAR, TEXTURE, BLENDER, COLLADA, KMZ, GLTF.

ЗМІСТ

ВСТУП

Сучасні геоінформаційні технології визначають новий рівень взаємодії з географічною інформацією, перетворюючи спосіб, яким ми сприймаємо та використовуємо просторові дані. Однією з ключових тенденцій у цьому напрямку є розширення можливостей візуалізації за допомогою тривимірних моделей, що дозволяють краще розуміти та аналізувати географічні об'єкти та явища.

Завдяки можливостям 3D і просторового аналізу, а також зростаючій інтеграції технологій, таких як доповнена реальність (AR) і віртуальна реальність (VR), ГІС пропонує величезні переваги для моделювання впливу, планування та розвитку розумних міст за допомогою використання технологій таких, як цифрові двійники.

Актуальність дослідження обумовлена сучасними тенденціями розвитку геоінформаційних технологій, зокрема технології цифрових двійників, які дозволяють створювати віртуальні моделі реальних об’єктів і процесів. За даними Євразійської економічної комісії використання цифрових двійників зросло з 3% у 2017 році до 13% у 2020 році, а прогнозується, що до 2025 року цей показник сягне 35%. Одним із ключових елементів цифрових двійників є 3D моделі будівель, які дозволяють візуалізувати архітектурні особливості, структурні характеристики, функціональні параметри та інші аспекти об'єктів нерухомості.

Створення 3D моделей будівель на інтерактивній картографічній основі в структурі WEB-додатку є актуальною та перспективною задачею, яка вимагає застосування сучасних методик та інструментів. Однак, не існує універсальних рішень для моделювання 3D графіки для веб-додатків, оскільки кожен проєкт має свої специфічні вимоги та обмеження. Наприклад, різні веб-додатки можуть мати різні цілі, аудиторії, бюджети, терміни, технології, архітектури тощо. Тому необхідно розробляти нові та перспективні методики моделювання, які б враховували всі ці фактори та забезпечували оптимальну якість та

продуктивність 3D графіки для веб-додатків. Такі методики мають вагу на ринку інформаційних продуктів, оскільки вони можуть дати конкурентну перевагу та збільшити зацікавленість користувачів.

Одним із таких інструментів є бібліотека Three.js, яка дозволяє створювати та маніпулювати 3D графікою в браузері за допомогою JavaScript. Використання бібліотеки Three.js для створення 3D моделей будівель на інтерактивній картографічній основі в структурі WEB-додатку є новітнім та інноваційним підходом, який має значний науковий та практичний потенціал. Тому обрана тема магістерської роботи є актуальною та відповідає сучасним вимогам геоінформаційної галузі. Структурна схема розкриття теми роботи зображено на рис. 1.

Для досягнення поставленої мети було виділено наступні цілі:

- аналіз методів візуалізації тривимірних сцен у веб-додатках;
- вибір технологій розробки веб-додатку;
- вибір програмного забезпечення;
- розгортка веб-додатку;
- розробка тривимірної моделі забудови;
- отримання glTF-файлу моделі;
- інтеграція тривимірної моделі у структуру веб-додатку;
- опис методики розробки веб-додатку з інтегрованою тривимірною моделлю забудови.

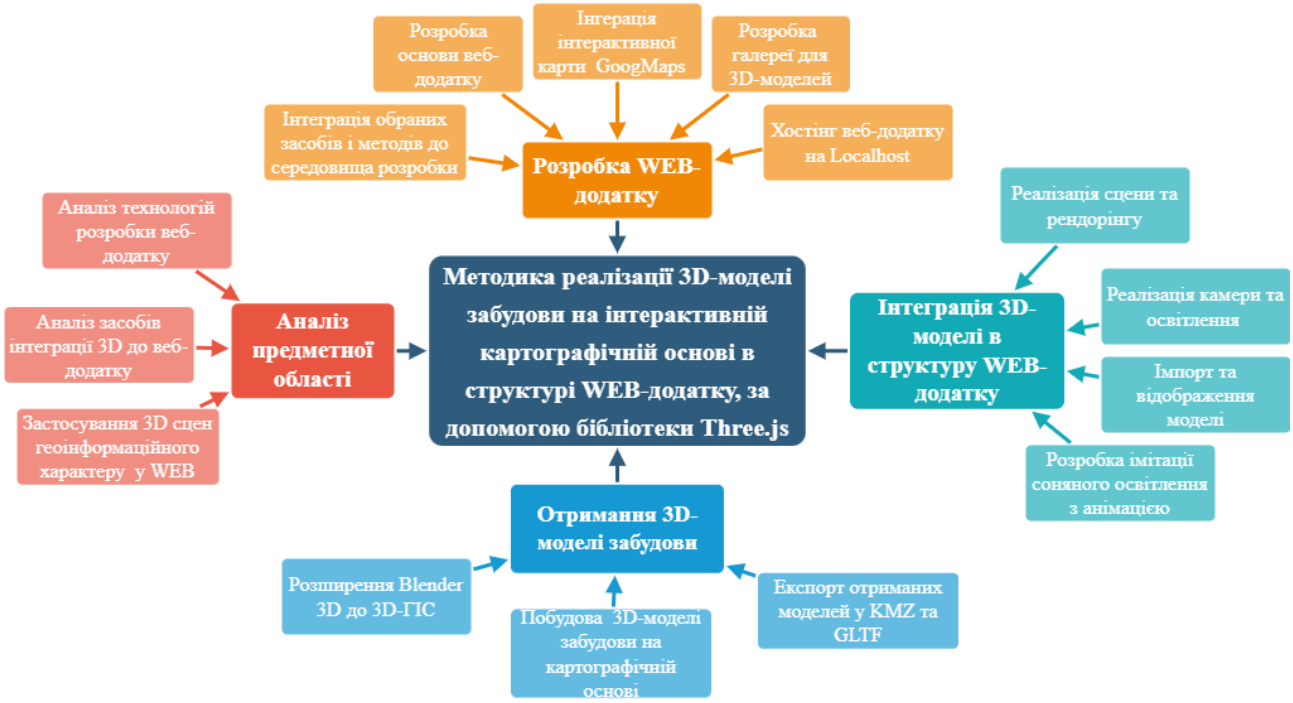


Рисунок 1 – Структурна схема розробки методики інтеграції тривимірної моделі у структуру WEB-додатку

Об’єктом дослідження є процеси створення та інтеграції тривимірних зображень будівель, створених з використанням ГІС технологій, у картографічні WEB-додатки.

Предметом дослідження є методи створення 3D моделей будівель на інтерактивній картографічній основі в структурі WEB-додатку за допомогою бібліотеки Three.js.

Робота виконана з використанням бібліотеки візуалізації графіки з відкритим кодом – Three.js, корсплатформного редактору коду Visual Studio Code, безкоштовних плагінів для розширення можливостей програмного продукту Blender – «Blender-OSM» та «Blender glTF 2.0 Importer and Exporter». Також в якості інтерактивної картографічної основи було використано безкоштовну базу геоданих веб-картографічного сервісу OpenStreetMap.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальна характеристика WEB-програмування

Веб-застосунки стали невід’ємною частиною нашого цифрового досвіду, починаючи від простих сайтів із вмістом і закінчуючи складними, багатофункціональними програмами, такими як платформи соціальних мереж, веб-сайти електронної комерції та інструменти для співпраці. Їх доступність і простота використання значно сприяли широкому впровадженню веб-технологій у різних областях.

Веб-програмування – це створення сайтів і програм, які працюють у мережі. Створюються спеціальні комп’ютерні програми – скрипти, які діляться на два типи: серверні і клієнтські. Сервер являє собою комп'ютер, на якому знаходяться файли сайту. Клієнт – це сам користувач, а якщо бути точніше, то браузер, встановлений на ПК. Подаючи запит до сервера, він виступає в ролі клієнта.

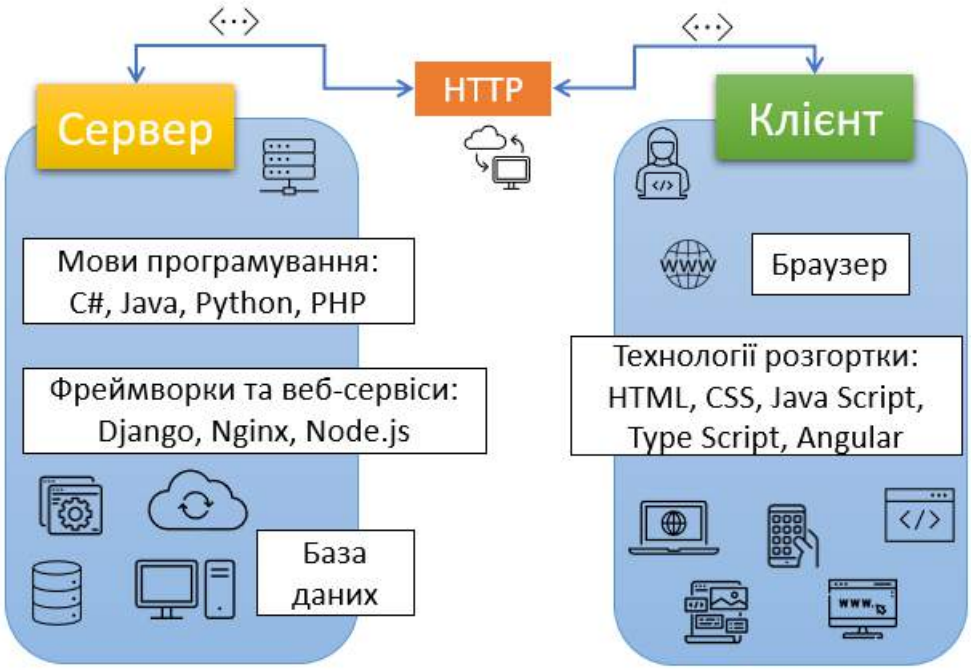


Рисунок 1.1 – Структурна схема веб-програмування

Сполучною ланкою між веб-розробником і сайтами є мова веб-програмування. Існує багато мов на яких пишуться і серверні, і клієнтські скрипти. Але деякі мови, навпаки, використовуються виключно для серверних чи клієнтських скриптів [1].

Розглянемо найпоширеніші технології, що застосовуються для веб-розробки:

1) Технології веб-дизайну:

- мова розмітки HTML – це код, який використовується для структурування і відображення веб-сторінки і її контенту;

- CSS – це фактично мова стилів, яка відповідає за відображення HTML-документів. CSS працює зі шрифтами, з кольорами символів і фону, з полями, з рядками, з висотою і з шириною елементів відображення, з фоновими зображеннями, з позиціонуванням елементів і багато з чим іншим;

- SASS – це препроцесор CSS, за допомогою якого можна знизити кількість повторюваного CSS коду і заощадити час. Це більш стабільне розширення CSS, чітко і структурно описує стилі документа;

- LESS – це надбудова над CSS. Має всі базові можливості препроцесорів і навіть більше, але не має умовних конструкцій і циклів в звичному для нас розумінні. Основним плюсом є його простота, практично стандартний для CSS синтаксис і можливість розширення функціоналу за рахунок системи плагінів.

2) Технології програмування на стороні клієнта:

- JavaScript – найпопулярніша і підтримувана всіма сучасними інтернет браузерами мова програмування. Вона створена для того, щоб зробити веб-сторінки «живими». Скрипти у браузері підключаються безпосередньо до HTML і, як тільки завантажується сторінка — тут же виконуються;

- React – це бібліотека JavaScript, яка використовується для створення призначеного для користувача інтерфейсу;

- VueJS – це JavaScript бібліотека для створення веб-інтерфейсів з використанням шаблону архітектури MVVM (Model-View-ViewModel);

- AngularJS – це JavaScript фреймворк призначений для створення односторінкових веб-додатків;

- jQuery – це бібліотека JavaScript, призначена для спрощення скриптинга при роботі з вузлами HTML-елементів в браузері або для роботи в браузері без графічного інтерфейсу;

- AJAX – це бібліотека, яка значно спрощує і прискорює написання JavaScript коду, також дозволяє працювати з усіма браузерами.

3) Технології програмування на стороні сервера:

- PHP – найбільш популярна мова веб-програмування на стороні сервера. PHP підтримується більшістю хостинг-провайдерів

- Node.js – це середовище для виконання вашого JavaScript коду, це просто ще один спосіб виконувати код на вашому комп'ютері

- ASP.NET. – модель для розробки веб-додатків із застосуванням мінімуму коду, яка містить служби, необхідні для побудови веб-додатків для підприємств

- Python – це мова програмування загального призначення, націлений в першу чергу на підвищення продуктивності самого програміста

- Ruby on Rails – веб-орієнтоване середовище розробки з відкритим кодом, оптимізована для зручності програмування та стійкої продуктивності

Таким чином, комбінуючи різні технології, будь вони клієнтські або серверні, розробник може отримати велику кількість різних інтерактивних можливостей. Ці технології продовжують розвиватися, щоб відповідати вимогам сучасної веб-розробки, надаючи розробникам інструменти, необхідні для створення ефективних, безпечних і масштабованих веб-рішень [2].

Можна виділити три види веб-розробки: frontend, backend та fullstack.

Фронтенд-фахівці займаються розробкою клієнтської частини, тобто відображенням даних. Програмісти, задіяні у цьому напрямі, взаємодіють із дизайнерами і відповідають за плавність анімації, правильність макета і всю фронтенд-частину, яку бачать користувачі. Саме frontend-сторона продукту взаємодіє з браузером.

Бекенд-фахівець працює з серверною частиною, тобто логікою, прихованою від користувача. Мова може йти й як про автентифікації користувачів, так і про балансування навантаження на сервер. Бекенд-фахівці можуть взаємодіяти з сисадмінами, оскільки велике значення має працездатність та швидкодія сервера. Програміст працює з backend-частиною проєкту, до якої звертається frontend або інший клієнт.

Fullstack-фахівець – універсальний «солдат», який відповідає за всі етапи реалізації проєкту. Фулстек-програмісти поєднують обов'язки бекенд- та фронтенд-розробників. У деяких випадках fullstack-фахівці можуть виконувати функції системних адміністраторів та дизайнерів.

Створення веб-додатку умовно можна розділити на такі етапи:

а) Попередній етап розробки сайту. На цьому етапі розв'язуються питання загального характеру. Обговорюється загальна концепція сайту, формулюються та фіксуються цілі створення сайту.

б) Етап проєктування сайту. Визначення структури сайту: меню, посилання, розміщення модулів, побудова списку компонентів, що підключаються, тощо.

в) Етап розробки й тестування сайту

г) Розміщення та розвиток ресурсу.

Веб-додаток — це програмне забезпечення або програма, до якої можна отримати доступ і взаємодіяти з якою через веб-браузер через мережу, як правило, Інтернет. На відміну від традиційних настільних програм, які запускаються вашою операційною системою, веб-програми мають бути доступні через веб-браузер. Ця взаємодія на основі браузера дозволяє користувачам отримувати доступ до програми незалежно від пристрою чи операційної системи, яку вони використовують, за умови, що вона має сумісний веб-браузер.

Web-додаток і web -сайт, здається два абсолютно ідентичних поняття, проте варто зауважити, що у них різні функції, а також компоненти розробки. Як ви вже зрозуміли веб-додаток це те, що запускається в браузері, проте він включає більшу кількість елементів з якими ми можемо взаємодіяти.

Таблиця 1.1 – Порівняльна характеристика web-додатоку і web -сайту

Параметри	Web-додаток	Web-сайт
Взаємодія з користувачем	Користувач може проводити маніпуляції з даними, але з обмеженим доступом	Користувач може читати інформаційний контент, проте не може його ніяк міняти
Завдання та складність	Має багато функцій, компонент та деталей	Відображає лише статичну сторінку, на якій зображена текстова інформація
Складність розробки	Зазвичай передбачає більш складну розробку, включаючи компоненти зовнішнього та внутрішнього планів	Загалом менш складна розробка, часто зосереджена на зовнішній презентації
Інтерактивність	Динамічний та інтерактивний, що дозволяє вводити дані користувача та оновлювати в режимі реального часу	В основному статичний вміст з обмеженою інтерактивністю. Зазвичай служить для інформаційних цілей
Зміна проєкту	Для внесення змін, потрібно робити ревью всього коду і змінювати багато частин коду	Досить просто вносити зміни, лише зробивши пару змін у html коді сторінки
Стек технологій	Використовує повний стек технологій, включно з зовнішніми та внутрішніми фреймворками	Зазвичай використовує простіший стек технологій, часто зосереджений на зовнішніх технологіях

Архітектура програмного забезпечення – структура, на базі якої створюється додаток, взаємодіють його модулі та компоненти.

Архітектура веб-додатку є макетом з усіма програмними компонентами (такими як бази даних, додатки та проміжне ПЗ) та їх взаємодією один з одним.

Архітектура визначає, як дані доставляються через HTTP, і гарантує, що сервер на стороні клієнта і внутрішній сервер можуть їх зрозуміти. Забезпечує наявність достовірних даних у всіх запитах користувачів, створення записів та керування ними, доступ та аутентифікацію на основі дозволів.

Розуміння та застосування принципів архітектури веб-додатків допомагає створювати масштабовані, гнучкі та надійні системи, які забезпечують задоволення потреб користувачів.

Загалом, існують різні архітектурні шаблони та варіанти веб-додатків, кожен із яких має свій набір переваг і недоліків. Будь-який проєкт можливо виконати за будь-якою архітектурою, але це може бути неефективним або у плані розробки, або у плані експлуатації проєкту. Тому при виборі архітектури потрібно враховувати поставлені задачі, проблематику, особливості потрібної взаємодії з БД, потреби клієнта та ін.

Розглянемо основні типи верхньорівневої архітектури веб-додатків: моноліт, мікросервіси, безсерверна та компонента архітектури.

Моноліт — це архітектурне рішення, у якому усі компоненти та модулі тісно повіязані між собою, залежать один від одного та спільно використовують один простір пам'яті.

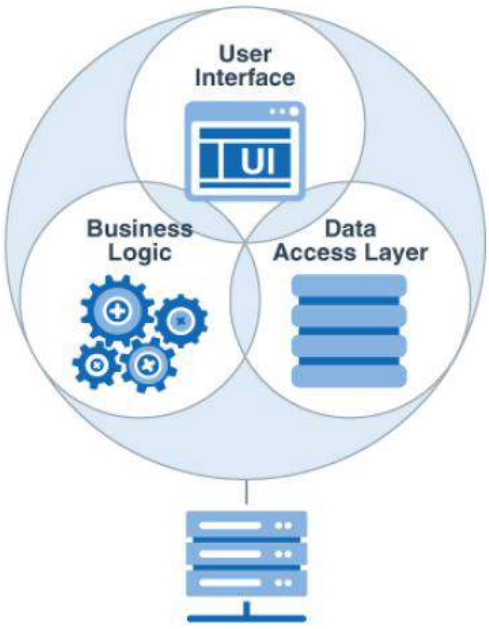


Рисунок 1.2 – Структура монолітної архітектури веб-додатку

Переваги монолітної архітектури:

1) простота розробки - процес розробки часто простіший, оскільки розробники працюють з єдиною кодовою базою, що полегшує його розуміння та підтримку;

2) спрощене розгортання - розгортання монолітної програми передбачає розгортання одного блоку, що зменшує складність розгортання порівняно з розподіленими архітектурами;

3) легше налагодження та тестування - налагодження та тестування спрощено, оскільки всі компоненти є частиною одного екземпляра програми, що полегшує виявлення та усунення проблем;

4) обмін ресурсами - компоненти в межах моноліту спільно використовують ті самі ресурси, що полегшує керування даними та контроль доступу в програмі.

Проблеми та недоліки:

1) обмеження масштабованості - масштабування монолітної програми може бути складним завданням, оскільки всі компоненти тісно інтегровані, вертикальне масштабування (додавання більше ресурсів до одного сервера) часто є основним варіантом;

2) технологічна гнучкість - запровадження нових технологій може бути складним, оскільки всю програму потрібно оновити або перенести, цей брак гнучкості може перешкоджати інноваціям;

3) Незалежність від розробника - у великих командах різні групи розробників можуть працювати над різними частинами програми, монолітна архітектура може призвести до залежностей між командами, потенційно сповільнюючи розвиток.

Мікросервіси – це архітектурний стиль, який структурує програму як набір невеликих незалежних служб, кожна з яких представляє певну бізнес-можливість. Ці служби спілкуються через API, і кожна служба розгортається та масштабується незалежно.

Архітектура мікросервісів вирішує кілька проблем, що виникають у монолітній архітектурі. Це архітектурне рішення, яке базується на розподілі

модулів на окремі системи, які спілкуються між собою за допомогою повідомлень. Вся система - це набір маленьких систем, пов'язаних між собою.

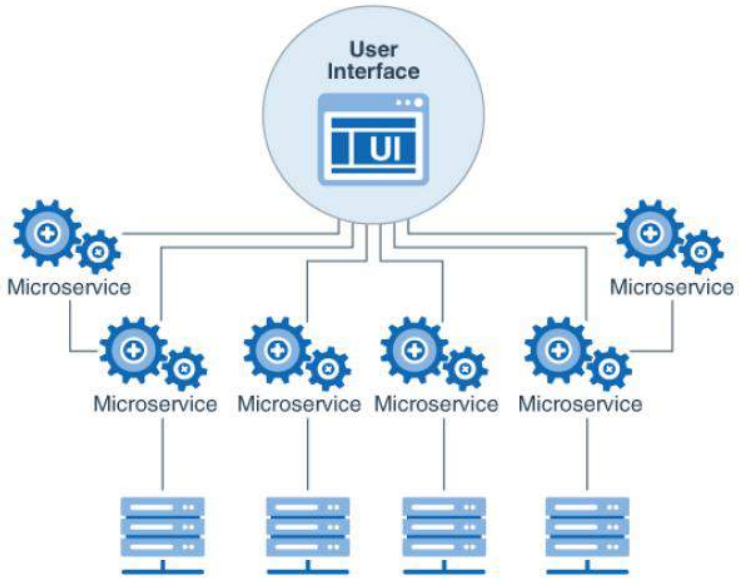


Рисунок 1.3 – Структура мікросерверної архітектури веб-додатку

Переваги:

1) масштабованість - сервіси можна масштабувати незалежно залежно від одного, оскільки вони є самостійними;

2) гнучкість - різні служби можуть використовувати різні технології, що забезпечує гнучкість вибору технологій;

3) усунення помилок - проблеми в одній службі не обов’язково впливають на всю програму.

Недоліки:

1) складність - мікросервісну архітектуру непросто розробляти, оскільки потрібно виробляти кілька підсистем і налагодити взаємодію між ними;

2) розгортання - початкове розгортання є непростим, і додавання нових сервісів вимагає налаштування ключових частин проєкту.

3) налагодження – так, як всі сервіси залежні, то якщо один зламається вся система перестає працювати, складно налагоджувати систему, оскільки потрібно знайти, який сервіс зламався і чому, враховуючи та аналізуючи всі залежності.

Безсерверна архітектура – це альтернатива мікросервісам, яка автоматизує все розгортання завдяки хмарним технологіям, дозволяючи створювати та запускати програми без керування серверами. Функції виконуються у відповідь на події, а розробники виставляють рахунок на основі фактичного використання.

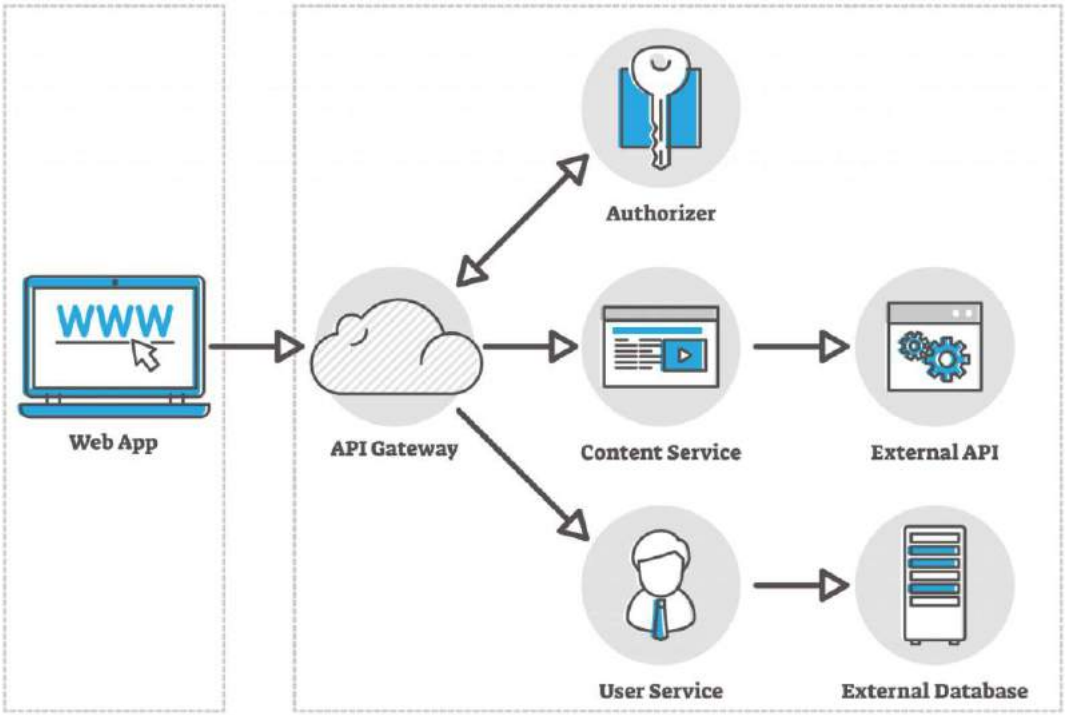


Рисунок 1.4 – Структура безсерверної архітектури веб-додатку

Переваги:

1) зменшені операційні витрати - відсутність потреб керування інфраструктурою сервера;

2) гнучкість - гнучка за рахунок відокремленості одного модуля від іншого;

3) масштабованість - автоматичне масштабування на основі попиту.

Недоліки:

1) затримка холодного запуску - функції можуть мати затримку, коли вони запускаються після періоду бездіяльності;

2) налагодження - як й у мікросервісній архітектурі, налагодження ускладнене взаємодією між компонентами;

3) прив’язка до постачальника послуг (Vendor Lock) - кожна хмара працює за власними правилами, тому переїзд з однієї хмари на іншу майже неможливий;

4) час виконання - функції зазвичай мають максимальний час виконання.

Компонентна архітектура – модель складної системи на основі компонентів передбачає створення програми шляхом розбиття її на модульні компоненти, які можна багаторазово використовувати. Кожен компонент інкапсулює певну функціональність або функцію.

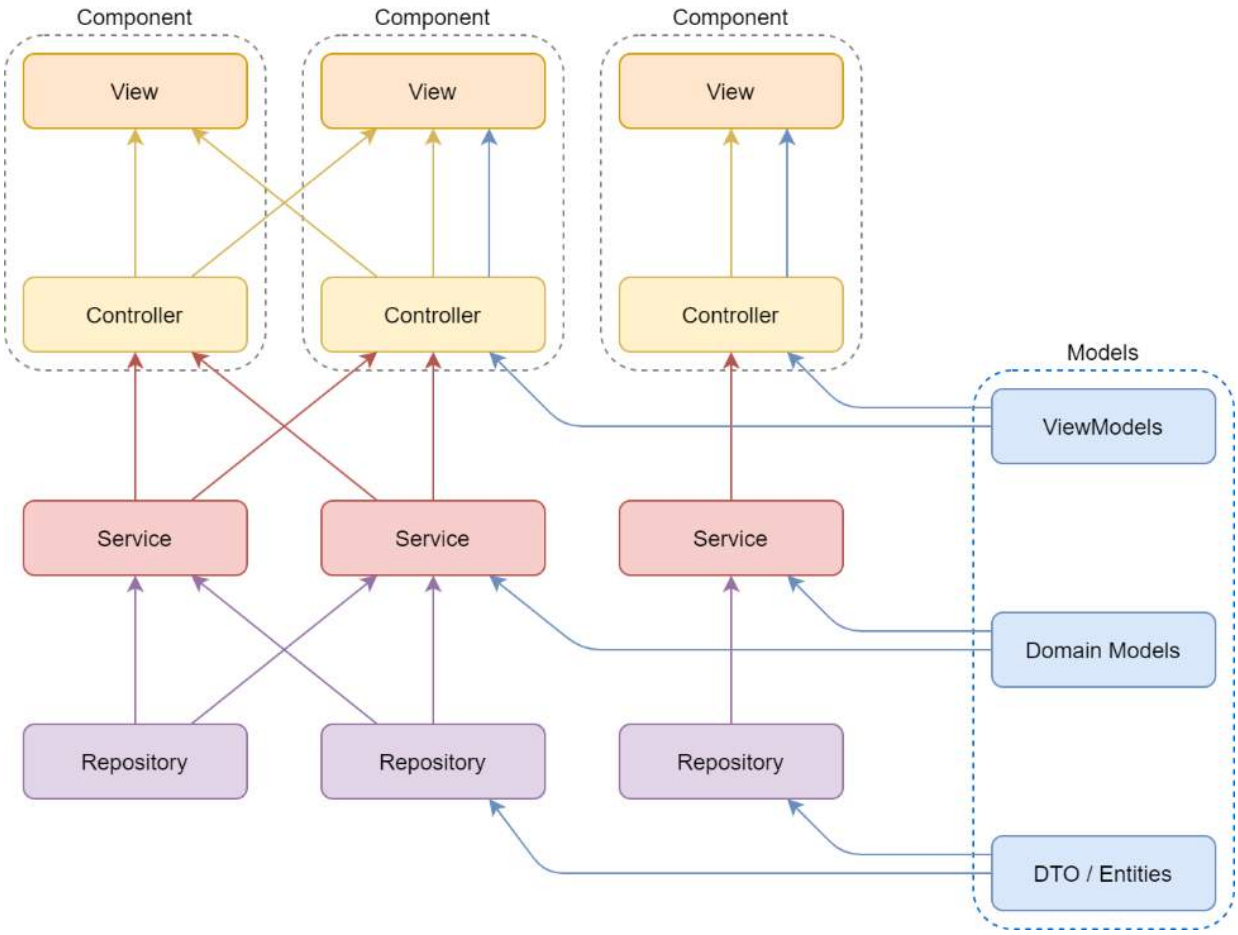


Рисунок 1.5 – Структура компонентної архітектури веб-додатку

Переваги:

1) повторне використання - компоненти можна повторно використовувати в різних частинах програми;

2) ремонтопридатність - простіше в обслуговуванні, оскільки зміни можна локалізувати для конкретних компонентів;

3) паралельна розробка - різні команди можуть працювати над різними компонентами одночасно.

Недоліки:

1) проблеми інтеграції - інтеграція компонентів може бути складною.

2) узгодженість - забезпечення узгодженості між компонентами вимагає ретельного проєктування та управління [3].

Ці архітектурні стилі не є взаємовиключними, і на практиці програми можуть використовувати комбінацію цих шаблонів на основі конкретних вимог і випадків використання. Вибір архітектури залежить від таких факторів, як природа програми, вимоги до масштабованості, досвід команди розробників і цілі проєкту.

1.2. Вибір технологій для розробки веб-додатку

На вибір технологій для розробки веб-додатків, зокрема односторінкових додатків, впливають різні фактори, включаючи вимоги до проєкту, досвід розробників, потреби в масштабованості та характер додатка. Виходячи з поставлених цілей, для програмування веб-додатку, було обрано: HTML, CSS, TypeScript – як основні мови програмування, Bootstrap й Angular – як фреймворки, та Three.js – як бібліотеку для відображення 3D у веб-застосунку.

Мова програмування – це система позначень для опису алгоритмів і структур даних, певна штучна формальна система, засобами якої можна виражати алгоритми. Мову програмування визначає набір лексичних, синтаксичних і семантичних правил, що задають зовнішній вигляд програми та дії, які виконує виконавець (комп'ютер) під її управлінням.

Фреймворк – це комплекс компонентів, бібліотек та інструментів, які пропонують структуру та готові рішення для роботи над певними завданнями.

Або дають вирішення певних проблем у розробці програмного забезпечення. Вони прискорюють розробку, покращують якість коду та стійкість додатків.

Фреймворки загалом підвищують ефективність веб-розробки та продуктивність, забезпечуючи узгоджену структуру, щоб розробникам не доводилося перебудовувати код з нуля. Фреймворки економлять час і пропонують розробникам безліч додаткових функцій, які можна додати до програмного забезпечення без додаткових зусиль.

Простіше кажучи, фреймворки є готовими інструментами для розробки програмного забезпечення. Вони містять набір шаблонів, коду та інших елементів, що дозволяють розробникам скоротити час на написання коду та покращити його якість.

Різні фреймворки створені, щоб задовольняти різні потреби розробників та проєктів. У кожного фреймворку своя унікальна функціональність, архітектура та методи розробки. Вони можуть бути більш сприятливими для певних типів проєктів або стилів розробки. Наприклад, деякі фреймворки задовольняють потреби у високій продуктивності, інші у зручності використання тощо. Це дозволяє розробникам вибрати фреймворк, який найкраще задовольняє їхні потреби [4].

Для вибору оптимальних технологічних рішень нижче проаналізовано технології веб-програмування, які можуть вирішити поставлені завдання.

1.2.1. HTML та CSS

HTML (Hypertext Markup Language) – стандартизована мова розмітки гіпертекстових документів. Використовується для інтерпретації та відображення браузером вмісту сторінки.

Документ, описаний за допомогою мови розмітки HTML – це текстовий файл, який можна створити й редагувати у найпростішому редакторі (Блокнот, Notrpad++, Kate тощо). Він містить містить різні елементи: заголовки, абзаци,

малюнки, таблиці та інші. Кожний елемент задають вказівкою мови HTML, яку називають тегом.

Як і більшість речей, HTML має кілька сильних сторін та недоліків.

Переваги HTML:

- широко вживана мова з великою кількістю ресурсів та величезною спільнотою;
- запускається в кожному веб-браузері;
- з відкритим кодом і абсолютно безкоштовна;
- чиста і послідовна розмітка;
- офіційні веб-стандарти підтримуються Консорціумом World Wide Web (W3C);
- легко інтегрується з серверними мовами, такими як PHP та Node.js.

Мінуси HTML:

- в основному використовується для статичних веб-сторінок. Для динамічної функціональності вам може знадобитися використовувати JavaScript або серверну мову, таку як PHP;
- не дозволяє користувачеві реалізовувати логіку, як результат, усі веб-сторінки потрібно створювати окремо, навіть якщо вони використовують однакові елементи, наприклад, верхній та нижній колонтитули;
- деякі браузери повільно застосовують нові функції.
- поведінку браузера іноді важко передбачити (наприклад, старі браузери не завжди відображають новіші теги).

CSS (абревіатура від Cascading Style Sheets, що в перекладі означає каскадні таблиці стилів) – це спеціальна мова (мова стилів), за допомогою якої описують вигляду документів (як і де відображати елементи веб-сторінки), написаних мовами розмітки даних. Найчастіше CSS використовується для документів, котрі розмічені мовою HTML, XHTML та XML.

Концепція стилів працює так - текст спочатку виводиться, а потім форматується, за допомогою CSS стилів.

Одна з головних переваг використання CSS - це можливість розділити зміст сторінки від її оформлення. Таке розділення дозволило покращити сприйняття та доступність змісту, забезпечити більшу гнучкість та контроль за відображенням змісту в різних умовах, зробити зміст більш структурованим та простим, прибрати повторення та ін. Власне це ж і була основна мета створення цієї технології. Крім цього, використання CSS дає вам набагато більше можливостей тонкого настроювання форматування, ніж у простому HTML. Також оскільки стилі можна зберігати у зовнішніх файлах, що підключаються, які в більшості випадків кешуються браузерами, їх використання дозволить прискорити завантаження вашого сайту на користувальницькій машині.

У HTML є деякі інструменти та функції, які дублюють можливості CSS. Але все одно каскадні таблиці більш зручні та швидкі. Тому фронтендерам та іншим веб-фахівцям так важливо CSS і мову HTML вивчити хоча б мінімально [5].

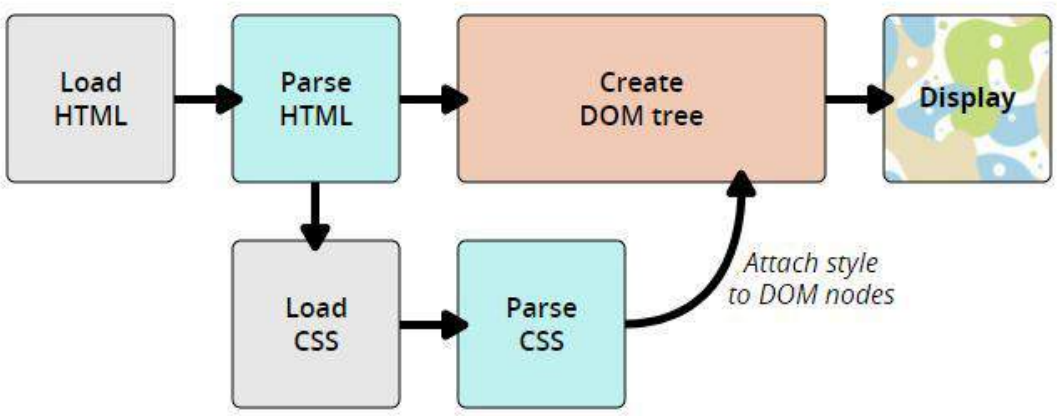


Рисунок 1.6 – Схема відображення сторінки за допомогою HTML та CSS

HTML і CSS є критичними компонентами веб-розробки, які допомагають перетворювати ідеї на візуально привабливі та функціональні веб-сторінки. HTML забезпечує структуру, а CSS – стиль. Розуміння цих мов допомагає розробникам створювати веб-додатки, що задовольняють потреби сучасного інтернет-суспільства.

1.2.2. TypeScript

TypeScript – об’єктно-орієнтовна мова програмування з відкритим кодом, розроблена і підтримувана Microsoft, що являє собою надбудову JavaScript. Це означає, що будь-який код JavaScript є валідним TypeScript кодом. Натомість TypeScript привносить деякі нові функції та синтаксис до JavaScript, що робить код більш надійними, зручним в обслуговуванні та масштабуванні.

Однією з головних особливостей TypeScript є те що це компільована мова програмування, на відміну від інтерпретованої мови JavaScript. Це означає, що перед запуском вашої програми, код TypeScript буде скомпільований не в що інше як в чистий JavaScript код, який вже і буде запущенний браузером. Хоч на перший погляд це й здається як зайвий крок, але це дає перевагу для виявлення помилок на етапі компіляції, що у випадку великих додатків і команд, які ці додатки розробляють, впливає на швидкість розробки. Більш того, компіляція дозволяє TypeScript розширити фінкціонал, який потім “перетвориться” в чистий JavaScript.

Інша відмінність від JavaScript полягає в тому, що TypeScript є строго типізованою мовою програмування. Це означає, що змінні, функції, аргументи та інші конструкції мають бути визначені за допомогою конкретного типу даних. Якщо ви не знайомі з концепцією строго-типізованих мов програмування, не хвилюйтеся, в наступній главі я на простому прикладі поясню даний концепт.

І останнє що хотілось би зазначити, TypeScript підтримує об’єктно-орієнтовані принципи, такі як класи (classes), інтерфейси (interfaces), перерахування (enumerations) та узагальнена типізація (generic types) які у повній мірі розкривають потенціал та силу цієї мови програмування.

Таблиця 1.2 – Порівняльна характеристика TypeScript і JavaScript

	TypeScript	JavaScript
Типізація	Підтримка статичної типізації	Динамічна типізація
Помилки	Помилки виявляються на етапі компіляції	Помилки виявляються під час виконання
Розширення	Надбудова JS, що забезпечує зворотну сумісність	Базова мова програмування
Інструменти	Хороша підтримка IDE та інших інструментів розробки	Великий вибір інструментів і бібліотек
Сумісність	Можна використовувати JS-код	Можливе використання TS у JS-проєктах
Розробка	Дає змогу створювати більш надійний і підтримуваний код	Швидке розроблення та прототипування
Сучасні можливості	Підтримка сучасних стандартів JSt	Необмежений доступ до можливостей JS
Прототипування	Доступна функція прототипування	JS не підтримує створення прототипів
Компіляція	TypeScript потрібно скомпілювати	JavaScript не потребує компіляції

Головна перевага TypeScript – це його строга типізація. У великих проєктах це дуже важливо і дозволяє запобігти проблемам, які складно відстежити при використанні простого JavaScript.

Найважливіший недолік полягає в тому, що браузер не розуміє TypeScript, потрібен компілятор або транспайлер який перетворить цей код на простий JavaScript браузером [6].

1.2.3. Фреймворки Bootstrap та Angular

Bootstrap – це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-застосунків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-застосунків.

Основна мета Bootstrap — створювати адаптивні веб-сайти, орієнтовані на мобільні пристрої. Він забезпечує оптимальну роботу всіх елементів інтерфейсу веб-сайту на будь-якому розмірі екрана. Також, даний фреймворк розроблено для сумісності з основними веб-переглядачами, забезпечуючи послідовне відтворення та функціональність у різних веб-браузерах.

Переваги використання Bootstrap:

- швидке кодування - Bootstrap надає набір попередньо розроблених компонентів, стилів і утиліт, які можна легко налаштувати, що прискорює процес розробки, дозволяючи розробникам швидко створювати адаптивні та візуально привабливі інтерфейси;

- адаптивний дизайн – фреймворк створено з адаптивною сітковою системою, яка гарантує, що веб-додатки доступні та добре відформатовані на різних пристроях і розмірах екрана, що сприяє узгодженій взаємодії з користувачем;

- послідовний вигляд і відчуття - елементи дизайну Bootstrap узгоджуються між різними компонентами, забезпечуючи єдиний і професійний вигляд, ця узгодженість може покращити загальну естетику веб-додатку;

- кросбраузерність - Bootstrap розроблено для бездоганної роботи в різних веб-переглядачах, зменшуючи ймовірність проблем із сумісністю, це забезпечує стабільний досвід для користувачів незалежно від вибору браузера;

- розширена документація - Bootstrap постачається з вичерпною документацією, яка включає приклади, фрагменти коду та рекомендації, документація полегшує процес навчання та є цінним ресурсом для розробників;

- адаптивність параметрів налаштування - хоча Bootstrap надає стилі та компоненти за замовчуванням, його можна налаштувати, легко змінювати стилі, використовувати змінні SASS або створювати власні теми відповідно до брендингу та вимог до дизайну своїх програм.

Недоліки використання Bootstrap:

- подібний вигляд сторінок - оскільки Bootstrap широко використовується, веб-сайти та програми, розроблені з його допомогою, можуть мати подібний вигляд, що потенційно може призвести до дещо загального або впізнаваного вигляду;

- збільшення розміру файлу - включення всієї бібліотеки Bootstrap може призвести до збільшення розміру файлу, ніж необхідно, щоможе вплинути на час завантаження сторінки, особливо якщо використовується лише піднабір функцій, але можна докласти зусиль та оптимізувати продуктивність;

- обмежена дизайнерська творчість - значне використання стилів Bootstrap за замовчуванням може обмежити творчу свободу дизайнерів, що призведе до веб-сайтів, які мають схожий зовнішній вигляд.

Підсумовуючи, Bootstrap — це потужний і широко використовуваний інтерфейсний фреймворк, який пропонує низку переваг для веб-розробки. Однак розробникам слід ретельно розглянути вимоги до своїх проєктів і потреби в налаштуванні, щоб визначити, чи підходить Bootstrap [7].

Angular — це фреймворк з відкритим кодом від компанії Google, написаний на TypeScript, і його основна мета - створення ефективних і складних односторінкових веб-додатків. Цей фреймворк допомагає створювати інтерактивні дизайни додатків, які покращують якість взаємодії з користувачем. І причинами покращення взаємодії з користувачем є відсутність внутрішніх посилань, відсутність перезавантаження сторінки, а керування всім вмістом видно на одному екрані за допомогою інтерактивних елементів [8].

Angular працює на основі принципу "data-binding" (зв'язування даних), який дозволяє автоматично оновлювати дані на сторінці під час зміни моделі даних.

Це робить додаток швидкодіючим і дозволяє уникнути ручного оновлення даних.

Плюси Angular:

- зрілий фреймворк, який має хорошу підтримку з боку учасників і є full package;

- використовується разом з TypeScript та має виняткову підтримку для цього (перевірка статичних типів може бути дуже корисна для великих застосунків, а також додати продуктивності розробникам, які працюють на Java і C#);

- angular-language-service — забезпечує інтелектуальні можливості та автозаповнення шаблону HTML-компонента;

- одностороння прив’язка даних, яка забезпечує виняткову поведінку застосунку, що зводить до мінімуму ризик помилок;

- структура та архітектура спеціально створені для великої масштабованості проєкту.

Мінуси Angular:

- різноманітність різних структур (Injectables, Components, Pipes, Modules тощо), що ускладнює вивчення в порівнянні з Vue, у якого є тільки Component;

-відносно повільна продуктивність, враховуючи різні показники. Проте це можна легко виправити, використовуючи Change detection strategy, що допомагає вручну контролювати процес рендерингу компонентів.

Найголовнішою сутністю в Angular є модулі. Мета модуля – об'єднати між собою компоненти, послуги та інші сутності, а також зв'язати їх семантично.

Single Page Application (SPA чи односторінковий застосунок) реалізує зручні для користувача сервіси, що наповнені інтерактивом. Найпростішим прикладом є Gmail.

SPA — веб-застосунок, розташований на одній фізичній HTML сторінці. Така сторінка одноразово завантажує усі необхідні ресурси (JavaScript, CSS, images тощо.) і більше не перезавантажується. Переходи за посиланнями не призводять до реального перезавантаження сторінки, а її вміст змінюється «на льоту», тобто динамічно. За необхідністю виконується запит на сервер для

отримання даних, і після їх отримання формується контент «нової» сторінки. Кожна окрема віртуальна сторінка прописується у маршрутизаторі (router). У нашому застосунку існує два маршрути: головна та чат.

Angular — один з фреймворків, що дозволяє реалізовувати описані Single Page Application.

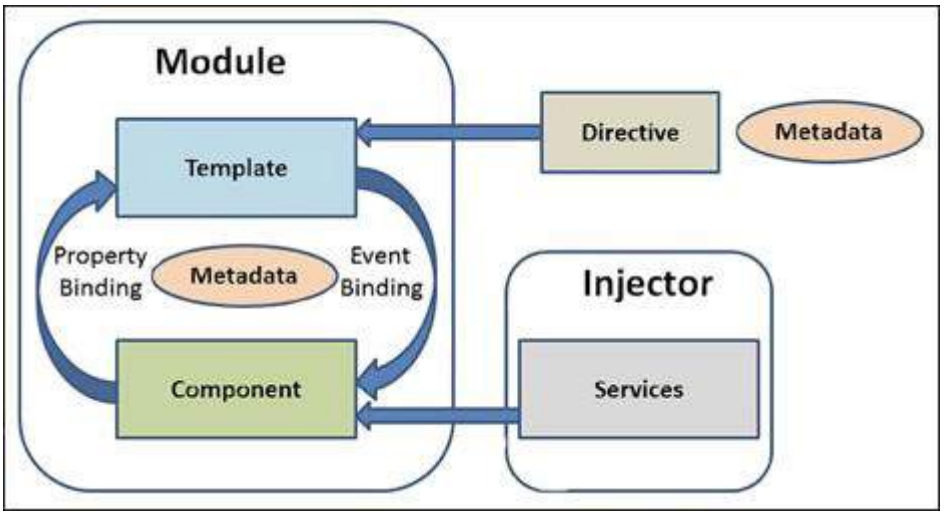


Рисунок 1.7 – Архітектура модульного додатку на Angular

Варто зазначити, що Angular застосунки пишуться на TypeScript, а не на чистому JavaScript. Версія синтаксису для JavaScript не отримала широкого розповсюдження, тому на даний момент у документації усі синтаксичні конструкції описані з використанням синтаксису TypeScript.

Коротко оглянемо основні пункти. Архітектура Angular складається з:

- модулів (Module);
- компонент (Component);
- шаблонів (Template);
- сервісів (Service);
- роутеру (Router);
- Pipe;
- Directives.

Модулі (Module) – структурні одиниці застосунку, які інкапсулюють певну логіку. В Angular це структури, які зберігають певні компоненти, директиви та сервіси, об'єднані певною логікою. Прикладом може слугувати профіль користувача, модуль для написання листа, огляд списку листів тощо.

Компоненти (Component) – TypeScript клас, який зберігає дані та логіку відображення цих даних у шаблоні (представленні). Шаблон тісно пов'язаний з компонентом. Дані з компонента можна з легкістю відображати у шаблоні, використовуючи спеціальний синтаксис. Компонент також може «знімати» дані з шаблона та отримувати їх безпосередньо у скрипті.

Шаблон (Template) – фрагмент html-коду з додаванням спеціального синтаксису. Він дозволяє впроваджувати в шаблон дані з компонента без використання innerHTML та подібних методів. Шаблон прописується у компоненті та є частиною його конфігурації.

Сервіс (Service) в Angular являє собою typescript класи, які виконують задачі, пов'язані з отриманням, зберіганням та обробкою даних. Наприклад, логування, перетворення даних для подальшої передачі у компонент, звернення до backend та ін. На відміну від компонентів та директив сервіси не працюють з представленнями (шаблонами) напряму.

Роутер (Router) – маршрутизатор, який призначений для переходу між екранами з метою відображення різного контенту. Іншими словами, коли в адресному рядку браузера у вас змінюється фрагмент URL, маршрутизатор відстежує ці зміни та завантажує ту або іншу частину застосунку.

Завершимо теоретичну частину та перейдемо до написання коду.

Директиви – це класи, які додають додаткову поведінку до елементів у ваших додатках Angular. Використовуйте вбудовані директиви Angular для керування формами, списками, стилями та тим, що бачать користувачі.

Пайпи – більш специфічна конструкція, декоратор, який позначає клас як канал і надає метадані конфігурації.

1.2.4. Аналіз засобів відображення 3D у веб-застосунках

Тривимірна графіка в Інтернеті розвивалася протягом останніх дев’яти років на основі бібліотеки веб-графіки, WebGL, API, який з’єднує браузери та комп’ютерне обладнання. WebGL робить можливим рендеринг інтерактивної 2D і 3D графіки на низькому рівні. За цей час з’явилося кілька фреймворків рівня абстракції, що працюють через WebGL. Вони полегшують і прискорюють створення інтерактивної 3D-графіки. Існує два способи реалізації тривимірної графіки у веб-додатках: за допомогою використання комбінованих програм та безпосередньо написання коду - використання бібліотек та фреймворків. Найпопулярнішими бібліотеками для відображення 3D веб-графіки є Three.js, Babylon.js та AFrame.

Three.js – це бібліотека JavaScript з відкритим кодом, яка використовується для відображення графіки, 3D і 2D об’єктів у веб-браузері. Вона надає потужні інструменти та функції для створення реалістичних 3D об'єктів, анімації та взаємодії на веб-сторінках. Three.js базується на WebGL, що дозволяє вам використовувати потужну графічну обладнання користувачів для створення іммерсивних досвідів[9].

Велика частина будь-якого візуалізації, тривимірної чи іншої, — це математика. Як правило, у вас є окрема математична бібліотека, яка обробляє всю дивовижну лінійну алгебру та обчислення, які можуть використовуватися для відтворення складної графіки, але Three.js містить усе це у своїй величезній бібліотеці. Вона містить класи, спеціально розроблені для 3D-математики, і оскільки все це міститься в одній бібліотеці, розробники можуть бути впевнені, що все працюватиме разом, звичайно, це буде краще, ніж 20 бібліотек, які потенційно не можуть поєднуватися одна з одною, тож мати все під одним дахом.

Основними перевагами Three.js є:

1. Легке підключення та доступне використання.

2. Можливість відображення і маніпулювання тривимірною графікою на вебсторінках за допомогою мови JavaScript.

3. Можливість додавання та видалення 3D об'єктів в режимі реального часу.

4. Кросплатформеність. Користувачі можуть побачити 3D модель на будь18 яких платформах, адже при перегляді їх об’єднує браузер.

5. Можливість працювати з інтерфейсним елементом canvas, завдяки чому 3D модель може відображатись і на багатьох мобільних пристроях.

До недоліків можна віднести:

1. Для відображення 3D об’єктів у браузері необхідно створити сцену, камеру та візуалізатор, що є мало зрозумілим для звичайного розробника.

2. Підтримка лише у сучасних браузерах [9].

Хоча Three.js широко використовується, існують інші бібліотеки та фреймворки, які пропонують подібні функції для 3D-графіки та візуалізації.

Babylon.js — це потужний 3D-дизайнер із відкритим кодом для Інтернету. Він підтримує WebGL і WebXR і розроблений, щоб бути простим у використанні, масштабованим і багатофункціональним. Babylon.js підходить як для початківців, так і для досвідчених розробників і має велику та активну спільноту.

A-Frame — це веб-фреймворк для створення досвіду віртуальної реальності (VR) за допомогою HTML і сутностей. Він створений на основі Three.js і забезпечує декларативний синтаксис для створення 3D і VR сцен. A-Frame особливо добре підходить для веб-додатків VR.

У таблиці 1.3 наведено стислий огляд ключових аспектів для кожного фреймворку, але важливо зазначити, що вибір між Three.js, Babylon.js і A-Frame залежить від конкретних вимог проєкту та вподобань розробника. Кожен фреймворк має свої сильні сторони та добре підходить для різних випадків використання.

Таблиця 1.3 – Порівняльна характеристика засобів відображення 3D у веб-застосунках

Особливість	Three.js	Babylon.js	A-Frame
Простота використання	Більш крутий процес навчання завдяки широкому API.	Відомий своїм дружнім API, що робить його доступним.	Надзвичайно зручний для початківців із синтаксисом, схожим на розмітку.
Особливості та можливості	Комплексний і багатофункціональний. Висока можливість налаштування.	Багатофункціональний з акцентом на простоті використання. Надає інструменти для фізики, зіткнень і систем частинок.	Спрощує розробку VR за допомогою синтаксису, схожого на розмітку. Швидке створення прототипів для VR.
Спільнота та документація	Велика та активна спільнота. Велика документація та приклади.	Активна спільнота з хорошою документацією.	Зростаюча спільнота з хорошою документацією. Простота приваблює розробників, зацікавлених у VR.

Продовження таблиці 1.3 – Порівняльна характеристика засобів відображення 3D у веб-застосунках

Продуктивність	Пропонує баланс між продуктивністю та гнучкістю. Можна оптимізувати для різних випадків використання.	Відомий високою продуктивністю, особливо для розробки ігор. Вбудовані інструменти для оптимізації продуктивності.	Створено для досвіду віртуальної реальності з оптимізацією продуктивності. Добре підходить для простіших 3D-сцен.
Випадки використання	Підходить для проєктів, які вимагають високого рівня налаштування та гнучкості. Підходить для складних 3D програм і візуалізацій.	Підходить для розробки ігор, візуалізації архітектури та проєктів, де важлива простота використання. Хороший баланс між функціями та простотою.	Найкраще підходить для досвіду VR, простих 3D-сцен і швидкого прототипування. Пріоритет віддає простоті та швидкому розвитку.

Якщо порівнювати ці бібліотеки за популярністю, то можна помітити що Three.js займає лідируючі позиції на ринку за останні 5 років (рис. 1.7).

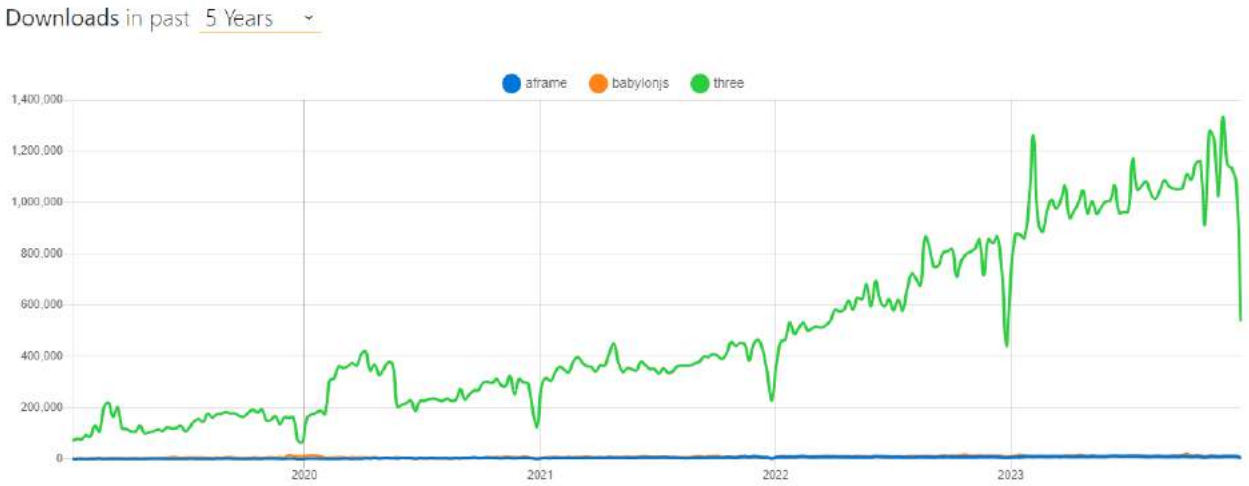


Рисунок 1.8 – Графік популярності бібліотек (Three.js, Babylon.js та AFrame) для відображення 3D веб-графіки

Підсумовуючи, Three.js — це надійна та універсальна 3D-бібліотека, яка чудово підходить для створення складних і реалістичних 3D-сцен. Його повний набір функцій, гнучкість, потужна підтримка спільноти та перевірений досвід роботи роблять його переконливим вибором для розробки веб-додатку з реалізацією 3D-сцени забудови, надаючи необхідні інструменти та ресурси для наочного представлення моделі в Інтернеті.

1.3. Веб-програмування та ГІС

Поява Інтернету змінила все навколо нас, і ГІС не є винятком. Веб-ГІС – це шаблон або архітектурний підхід до реалізації сучасної ГІС, що дозволяє розповсюджувати карти та інструменти обробки без обмежень за часом і місцем для користувачів.

1.3.1. WEB-геоінформаційні системи

Геоінформаційний веб-додаток – це комплекс програмних компонентів, базових карт, а також тематичних даних, що взаємодіють між собою та

створюють можливості візуалізації, передачі та управління даними використовуючи в основі веб-додаток.

Веб-ГІС дозволяє візуально взаємодіяти з геопросторовими даними в Інтернеті (на наших мобільних телефонах, настільних програмах, планшетах і практично будь-де за допомогою веб-браузера).

Успадкувавши потужність Інтернету та мережі, веб-ГІС стає багатофункціональною та надає низку переваг, що зумовлюють зростання попиту та актуальності. Головною перевагою є глобальне охоплення, що дозволяє зручно ділитися геоданими в межах власної організації та взаємодіяти з користувачами у різних куточках світу. Ця своєрідна універсальність відкриває нові можливості для глобального обміну інформацією та співпраці. Далі, важливим аспектом є низька вартість розробки веб-ГІС-програм. В порівнянні з традиційними настільними рішеннями, створення та впровадження веб-додатків виявляється економічно вигіднішим. Це стимулює раціональне використання фінансових ресурсів. Третій аспект полягає у масштабованій природі веб-ГІС, що дозволяє ефективно обслуговувати значну кількість користувачів завдяки хмарним технологіям. Це забезпечує гнучкість та доступність програмного забезпечення для великої аудиторії.

Крім того, веб-ГІС відзначається високою крос-платформенною сумісністю, забезпечуючи працездатність на різних операційних системах і пристроях. Це визначає високий ступінь універсальності та доступності для користувачів з різних технічних областей. Зручність використання та інтуїтивний інтерфейс веб-ГІС-додатків дозволяють широкому колу користувачів використовувати їх без значного попереднього досвіду в області геоінформатики.

Нарешті, простота обслуговування визначається автоматичним оновленням веб-клієнтів при доступі до програми. Це не лише забезпечує користувачам доступ до найновіших версій програм та даних, але і спрощує адміністрування системи, не вимагаючи вручну оновлювати кожного окремого користувача [10].

Ці характеристики вказують як на переваги, так і на серйозні завдання, які стоять перед веб-ГІС. Наприклад, простота в користуванні веб-ГІС стимулює

інтерес широкої публіки, але при цьому необхідно пам'ятати про користувачів Інтернету, які не мають досвіду роботи з ГІС. І назад, для підтримки великої кількості користувачів потрібно, щоб веб-ГІС була масштабованою.

Є численні комерційні приклади веб-картографічних програм, у тому числі ті, які призначені для споживачів, див. Google Maps, Yahoo Maps і MapQuest, а також ті, що відповідають обчислювальним потребам професіоналів із геоінформації. Обидві групи додатків, хоча це найважливіше для першої групи, прагнуть постійно покращувати досвід користувача та взаємодію з геопросторовою інформацією, надаючи високодетальні геопросторові дані та зображення високої роздільної здатності великих смуг земної поверхні. Користувальницькі інтерфейси в цих додатках розроблені таким чином, щоб бути високоінтерактивними та швидко реагувати, і, як правило, до них легко отримати доступ за допомогою мобільних додатків, які надають користувачеві послуги на основі визначення місцезнаходження в режимі реального часу. Для розробки та впровадження цих веб-ГІС-додатків, а також для надсилання та отримання даних, які підтримують ці додатки, веб-технології використовуються як для споживачів, так і для професіоналів у цій галузі. В першу чергу клієнти та сервери Веб-ГІС зазвичай спілкуються через протокол HTTP, де найпростішою формою веб-ГІС може бути один сервер і один або більше клієнтів. Однак загалом архітектури веб-ГІС розроблені у вигляді трирівневої системи, яка включає рівень даних і може бути повністю розподілена в Інтернеті та взаємодіяти одна з одною за допомогою веб-служб. Ці веб-сервіси включають, але не обмежуються ними, службу веб-функцій (WFS) і службу веб-карт (WMS) [10-11].

1.3.2. Тривимірні веб-геоінформаційні системи

3D веб-геоінформаційні системи (3D веб-ГІС) представляють собою передовий напрямок в області візуалізації та аналізу географічної інформації в онлайн-середовищі. Однією з основних характеристик цих систем є їхня

здатність відображати геодані у тривимірному форматі, що дозволяє користувачам отримати глибше розуміння просторових взаємозв'язків та структури географічних об'єктів.

Однією з ключових можливостей 3D веб-ГІС є глобальна доступність. Це означає, що користувачі можуть легко ділитися геоданими в своїй організації та взаємодіяти з іншими за кордоном, сприяючи глобальному обміну інформацією та співпраці.

Переваги використання 3D веб-ГІС включають покращену візуалізацію, що полегшує аналіз геоданих та прийняття рішень. Вони також дозволяють створювати інтерактивні віртуальні моделі, які можуть бути використані в різних галузях, включаючи урбаністику, транспорт, екологію та геологію.

Незважаючи на великі можливості, існують певні недоліки використання 3D веб-ГІС. Одним із них є технічні вимоги, які можуть бути важкими для деяких користувачів, особливо тих, які не мають доступу до потужних обчислювальних ресурсів чи швидкого Інтернет-з'єднання. Крім того, не завжди просто підготувати та обробити тривимірні дані для використання в системі.

Застосування 3D веб-ГІС широке і розмаїте. Вони використовуються в містобудуванні для планування та візуалізації міського простору, в транспортній галузі для аналізу та оптимізації транспортних мереж, в екологічних дослідженнях для вивчення впливу людської діяльності на природні екосистеми, а також в геологічних дослідженнях для аналізу та моделювання геологічних утворень.

Загалом, 3D веб-ГІС є потужним інструментом для геопросторового аналізу, що забезпечує нові можливості для вивчення та розуміння географічної інформації в інтерактивному та тривимірному форматі.

1.4. Аналіз ПЗ для веб-програмування

Для створення веб-застосунків використовуються спеціальні програми, які називаються редакторами вхідного коду. Вони можуть бути окремим додатком, або інтегрованим в інтегроване середовище розробки (IDE). Редактори програмного коду мають деякі можливості, які спрощують та прискорюють написання та редагування коду, такі як підсвітка синтаксису, автодоповнення, перевірка правильності розстановки дужок, структурний видрук, контекстна допомога по коду та багато інших. Такі редактори надають зручний спосіб для запуску компілятора, інтерпретатора, налагоджувача або інших програм необхідних у процесі розробки програмного забезпечення. Незважаючи на те, що багато текстових редакторів можуть бути використані для редагування тексту програм, якщо вони не мають розширених можливостей, що автоматизують або спрощують введення та модифікацію коду, то вони не можуть називатися «редакторами початкового коду», а просто є «текстовими редакторами», які також можуть бути використані для редагування програм.

Вибір редактору вхідного коду для веб-програмування є практичним рішенням, яке суттєво впливає на повсякденну роботу розробника та результати проєкту. Багато факторів має бути враховано у цьому виборі, і вони часто залежать від особистих уподобань, вимог проєкту та зручності використання.

Для вибору оптимального програмного забезпечення нижче проаналізовано програмні продукти, які можуть вирішити поставлені завдання.

1.4.1. Notepad++

Notepad++ – безкоштовний текстовий редактор для Windows із підсвічуванням синтаксису великої кількості мов програмування та розмітки. Є чудовою заміною стандартного блокнота Windows. Він базується на компоненті Scintilla, написаний на C++ з використанням STL та поширюється під ліцензією

GPL. Базова функціональність програми може бути розширена як за рахунок плагінів, так і сторонніх модулів, таких як компілятори та препроцесори.

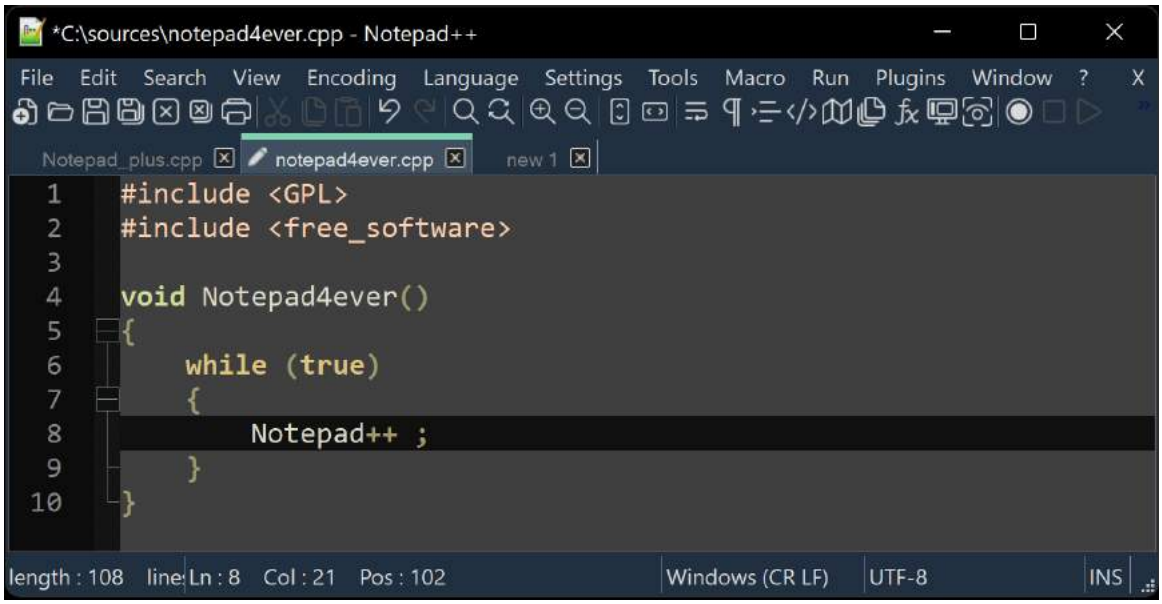


Рисунок 1.8 – Вікно редактора коду Notepad++

Основні особливості Notepad++:

- підсвічування синтаксису і згортання блоків, згідно з мов програмування;
- вбудований редактор тексту, з допомогою якого можна побачити надрукований текст таким, яким він був на екрані монітора;
- слова при наборі в редакторі автоматично додруковуються;
- можливість одночасної роботи з декількома документами і їх перегляду;
- можливість підтримки регулярних виразів Пошуку/Заміни;
- можливість динамічної зміни вікна перегляду;
- підтримує велику кількість мов;
- можливість залишати замітки;
- автоматичне виділення дужок під час редагування тексту;
- програма може записувати і запускати макроси.

Недоліки Notepad++

- редактор не платформний і працює тільки з операційними системами Windows;
- не розпізнає декілька мов у одному документі [12].

1.4.2. Atom

Atom – це текстовий редактор з відкритим вихідним кодом, доступний для кількох платформ, створений компанією GitHub. Він відомий як “хакований до основи” редактор коду, оскільки його можна легко налаштувати за допомогою CSS, HTML, JavaScript і інших популярних веб-технологій. Atom надає гнучкі можливості налаштування, а також підтримує велику кількість мов програмування за замовчуванням.

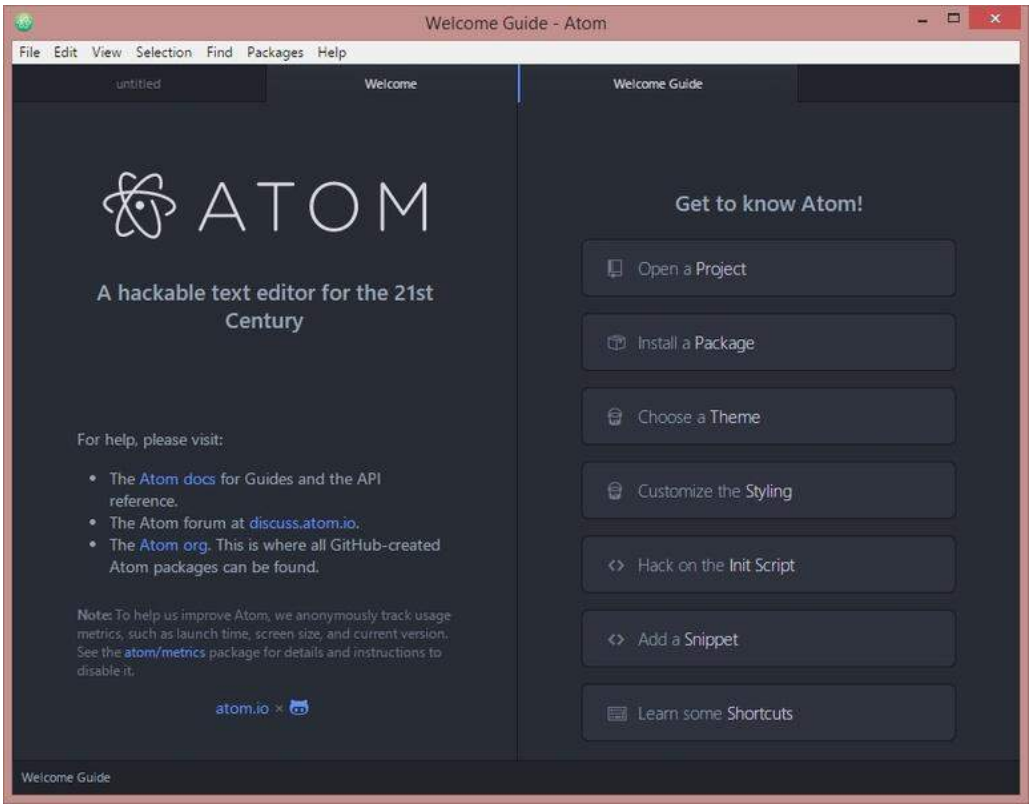


Рисунок 1.9 – Вікно редактора коду Atom

Незважаючи на те, що Atom є відносно новим програмним забезпеченням у світі текстових редакторів, воно набуло неймовірної популярності з 2014 року завдяки простоті використання та налаштування.

Основними перевагами Atom є те, що він повністю безкоштовний, оснащений функцією інтелектуального автозавершення, яка допомагає розробникам швидше вводити коди, попередньо встановленою темою інтерфейсу користувача та синтаксису, а також функціями налаштування, які дозволяють користувачам додати індивідуальний підхід до інтерфейсу користувача. Однак його продуктивність і високе використання ресурсів значно знижують його попит для тих, хто працює над великими проєктами або з менш потужним обладнанням.

Особливості та переваги:

- модульність: Atom пропонує багато пакетів, які можна встановити для додавання нових функцій або зміни поведінки редактора;
- програмне забезпечення з відкритим кодом, що означає, що розробники можуть вільно доповнювати його функціональність, внести внески в кодову базу та створювати свої власні плагіни та теми;
- співпраця в реальному часі: дозволяє розробникам ділитися своїми робочими просторами з іншими, співпрацюючи над проєктами у реальному часі;
- має вбудовану підтримку Git і GitHub, що робить його незамінним інструментом для розробників, які активно використовують ці системи контролю версій;
- наявність функції автоматичного завершення (Smart Autocompletion), яка забезпечує можливість швидко написати код, пропонуючи варіанти в залежності від того, що було введено [13].

1.4.3. Visual Studio Code

Visual Studio Code – швидкий, функціональний і, що найголовніше, абсолютно безкоштовний редактор коду від компанії Microsoft, який пропонує користувачеві всі необхідні інструменти для розробки.

Є підсвічування синтаксису для різних мов програмування, автозавершення коду з використанням технології IntelliSense, можливість налагодження коду прямо в редакторі, повноцінна підтримка Git та інших SCM, наявність повноцінного терміналу та багато іншого.

Крім цього функціонал редактора можна значно розширити за рахунок величезної кількості різноманітних плагінів та доповнень, які не тільки підвищують комфорт роботи з кодом, але й дають можливість персоналізувати додаток, змінюючи теми оформлення, додаючи різні іконки тощо.

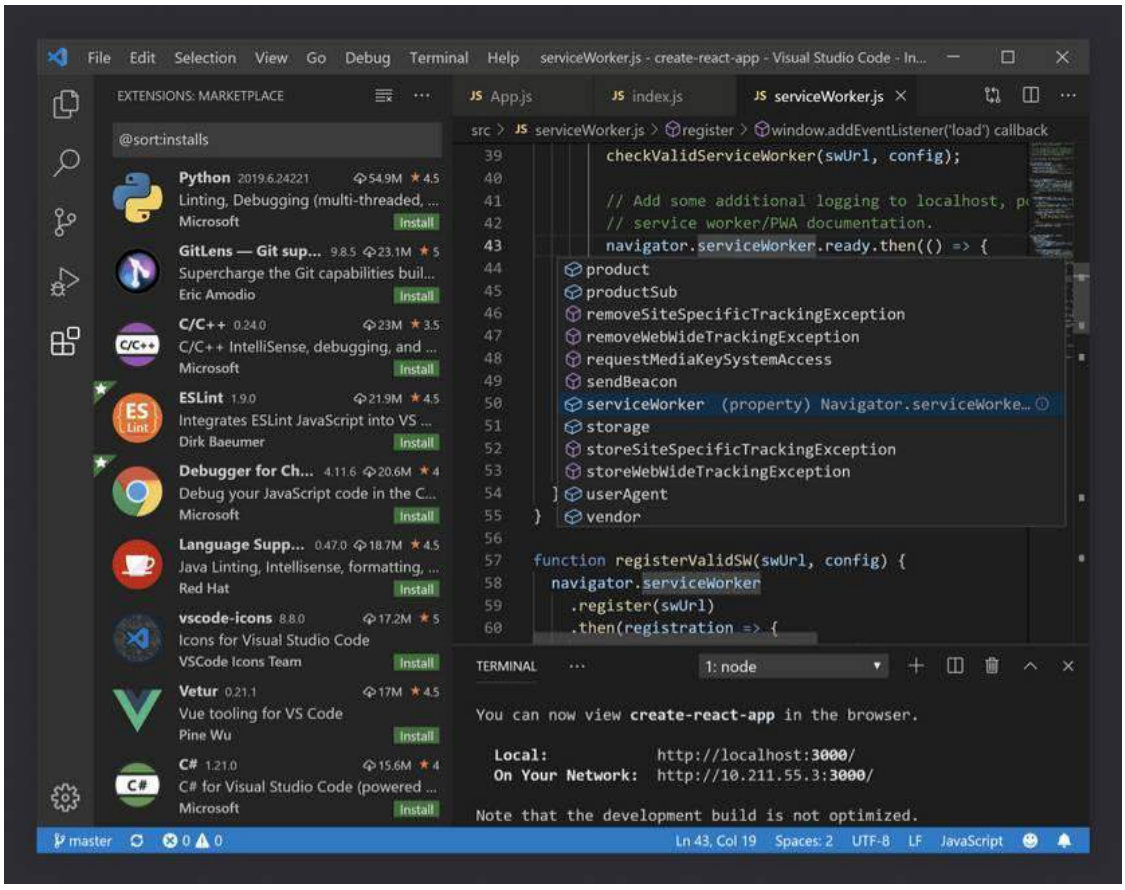


Рисунок 1.10 – Вікно редактора коду Visual Studio Code

Ключові особливості програми:

-висока швидкість роботи;

-величезна кількість підтримуваних мов програмування "з коробки" (наприклад, JavaScript, TypeScript, CSS, HTML тощо), також є можливість додавати будь-які інші мови за допомогою вбудованого репозиторію VS Code Marketplace;

-кросплатформність та безкоштовність;

-підсвічування синтаксису, автозавершення коду з використанням технології IntelliSense та ін;

-вбудований налагоджувач коду;

-доступна співпраця над проєктами в режимі реального часу(Live Share);

-підтримка Git та інших систем керування версіями, що дозволяє робити комміти прямо з редактора;

-величезна кількість різних плагінів і додатків, наприклад, додаток Live Server, що дозволяє постити на локальний сервер власні веб-застосунки[14].

1.4.4. Порівняльна характеристика редакторів коду

Для вибору програми реалізації методики розробки веб-додатку було виділено певні критерії: кросплатформність, не великі вимоги до апаратного забезпечення, можливість розширення функціональності, наявність відлагоджувача коду та підтримка широкого вибору мов програмування. Порівняльну характеристику редакторів коду за цими критеріями відображено у табл. 1.4.

Таблиця 1.4 – Порівняльна характеристика редакторів коду

Особливість	Visual Studio Code	Notepad++	Atom
Ліцензія	Відкритий код	Відкритий код	Відкритий код
Підтримувані платформи	Windows, macOS, Linux	Windows	Windows, macOS, Linux
Багатомовна підтримка	Підтримує численні мови	Обмежений вибір мов	Підтримує багато мов
Можливість розширення	Величезний ринок розширень	Багато плагінів	Багато плагінів
Керування версіями (Git)	Вбудована підтримка Git	Доступний плагін	Доступний плагін
Підтримка налагодження	Вбудований відлагоджувач	Відсутній	Налагодження доступне через плагіни
Співпраця онлайн (Live Share)	Підтримує	Відсутня	Підтримує
Інтелектуальне завершення коду (IntelliSense)	Так	Так	Так
Автоматизація завдань	Вбудована автоматизація	Відсутня	Підтримує через плагін
Використання ресурсів	Помірний	Легкий	Помірний
Простота використання	Зручний, простий у навігації	Просто і зрозуміло	Зручний для початківців
Підтримка спільноти	Велика та активна спільнота	Активна спільнота	Активна спільнота

Продовження таблиці 1.4 – Порівняльна характеристика редакторів коду

Оновлення та технічне обслуговування	Регулярні оновлення, потужна підтримка	Регулярні оновлення	Відносно регулярні оновлення
Вартість	безкоштовно	безкоштовно	безкоштовно
Кросплатформна сумісність	Так	Немає	Так

В результаті аналізу програмних продуктів для редагування вхідного коду, було обрано утиліту Visual Studio Code, як основний засіб розробки веб-додатку. Перевагою у бік даної програми стала можливість реалізації всіх частин додатку без використання сторонніх програмних продуктів. Також обраний програмний продукт легкий у використанні, має вбудований репозиторій розширень VS Code Marketplace, що розширюють можливості до використання ПО у ГІС.

РОЗДІЛ 2 РОЗРОБКА МЕТОДИКИ СТВОРЕННЯ 3D-МОДЕЛІ БУДІВЛІ НА ІНТЕРАКТИВНІЙ КАРТОГРАФІЧНІЙ ОСНОВІ В СТРУКТУРІ WEB-ДОДАТКУ

Методики побудови тривимірної моделі будівлі на картографічній основі в структурі WEB-додатку складається з трьох головних етапів: розробка WEB-додатку, отримання 3D-моделі забудови та інтеграція 3D-моделі в структуру WEB-додатку. Для зручності дані етапи було поділено на менші частини й застосовано щодо обраного ПЗ. Для планування об’ємів робіт було створено схему процесу відтворення моделі в структурі WEB-додатку за допомогою бібліотеки Three.js (рис. 2.1).

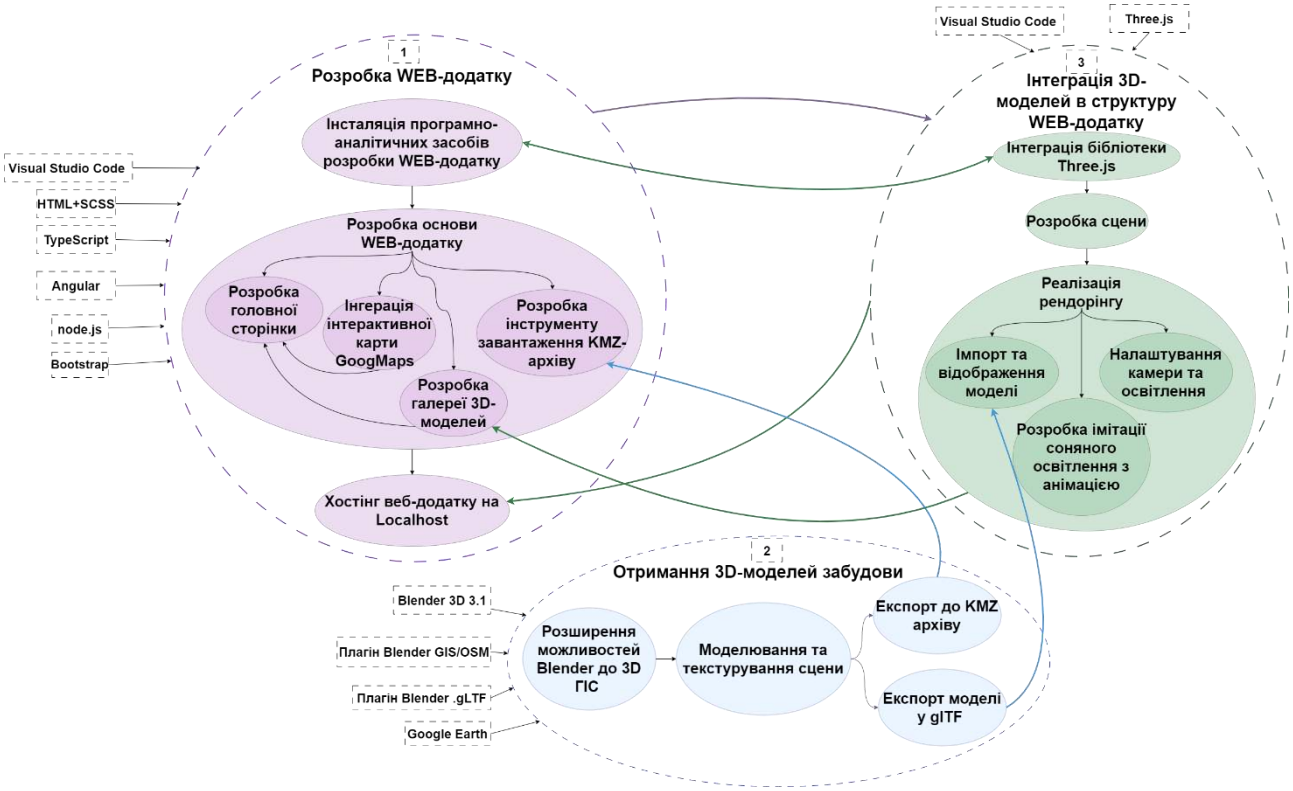


Рисунок 2.1 – Схема методики розробки веб-додатку з інтегрованою тривимірною моделлю

2.1 Розробка WEB-додатку

Першим кроком методики було інсталяція Visual Studio Code та створення основи веб-додатку. Для розробки веб-додатку було створено папку на локальному диску комп'ютера, яка зберігає всі файл розширенням .html та інші допоміжні файли з яких складається веб-додаток.

За основний інструмент розробки веб-додатку було обрано фреймворк Angular, який спрощує створення односторінкових багатокомпонентних додатків. Керування та налаштування Angular виконується за допомогою Angular CLI – командного рядка консолі. Щоб установити та використовувати інтерфейс командного рядка, а також запустити сервер додатків Angular, було інстальовано середовище виконання JavaScript Node.js і npm (менеджер пакетів Node.js). Середовище виконання Node.js було отримано з офіційного сайту https://nodejs.org/en та інстальовано на ПК (рис. 2.2).

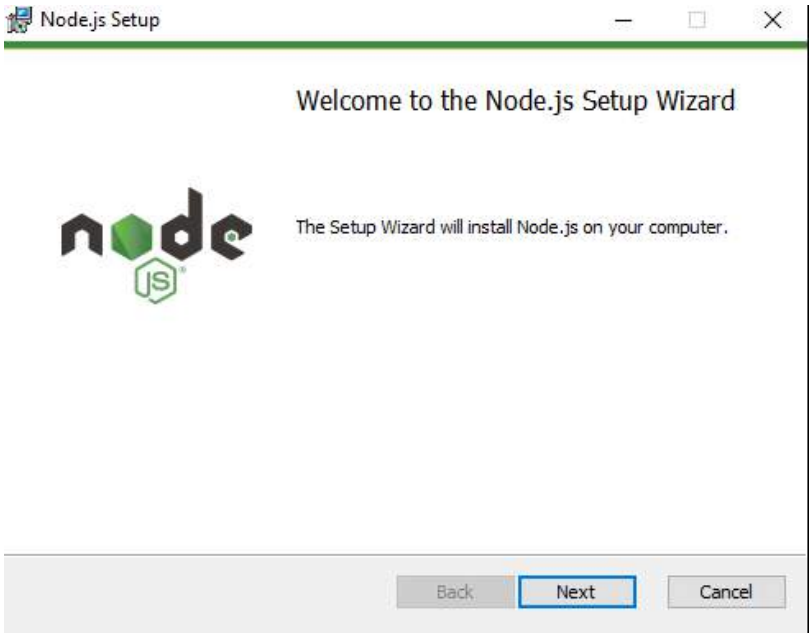


Рисунок 2.2 – Вікно інсталяції Node.js

Для того аби перевірити коректність встановлення середовища виконання JavaScript (Node.js) та менеджеру пакетів Node.js було введено команду «npm –version» у терміналі консолі Visual Studio Code. Якщо пакет встановлено вірно та найактуальнішу версію то вікно консолі видає номер версії (рис. 2.3).

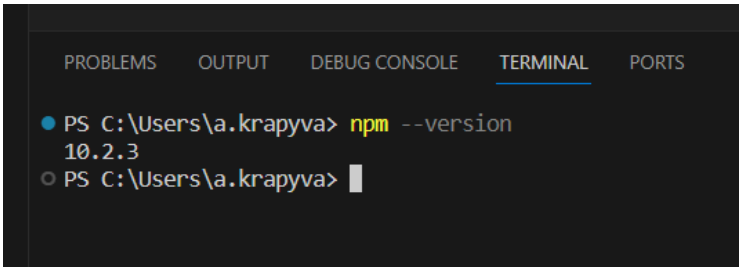


Рисунок 2.3 – Перевірка версії Node.js у терміналі консолі

Також, версію Node.js можна перевірити за допомогою розширюваної оболонки з інтерфейсом командного рядка, що поєднує в собі мову сценаріїв і інструментарій управління конфігурацією і автоматизації робіт для Windows – PowerShell, ввівши запит «nvm ls» або «nvm install <version>».

Наступним корком було активовано Angular CLI, через термінам за допомогою запиту «npm install -g @angular/cli» (рис. 2.4).

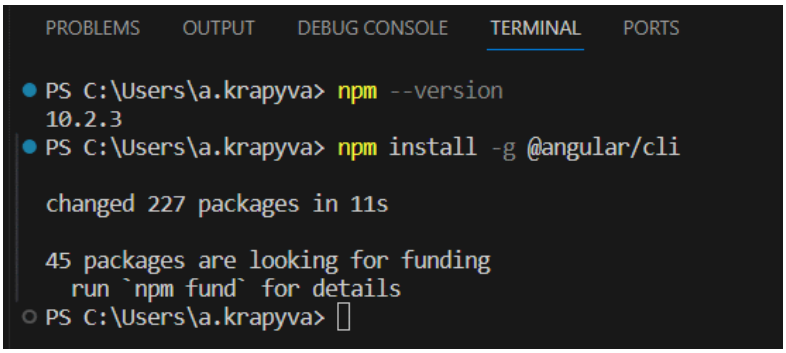


Рисунок 2.4 – Підключення Angular CLI

Після підключення інтерфейсу командного рядка від Angular відкривається можливість легко створювати та редагувати веб-додатки, використовуючи легкі команди Angular CLI.

Наступним кроком було створено новий проєкт веб-додатку, за допомогою команди «ng new [name]». Команда ng new створює папку робочої області Angular і генерує новий скелет програми. Робоча область може містити кілька програм і бібліотек. Початкова програма, створена командою ng new , знаходиться на верхньому рівні робочої області. Створюючи нову додаткову програму або модуль в робочій області, вона переміщується до projects/підпапки.

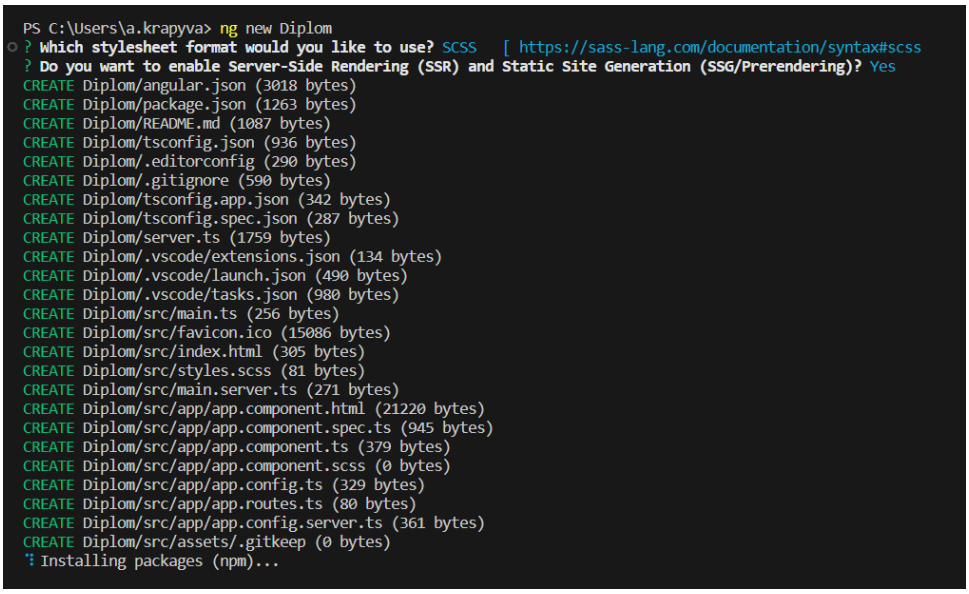


Рисунок 2.5 – Створення проєкту веб-додатку

В результаті генерації базового проєкту Angular утворюється додаток з усіма необхідними файлами для кореневого модуля з кореневим компонентом і шаблоном. Кожна програма має src папку, яка містить логіку, дані та ресурси (рис. 2.6).

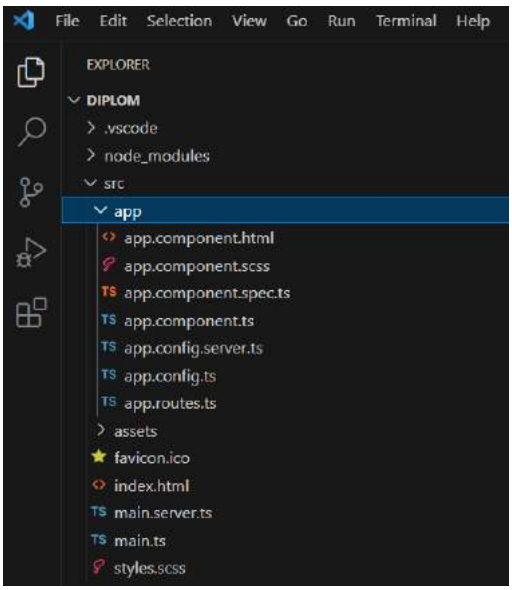


Рисунок 2.6 – Створений проєкт веб-додатку

Створений файл app-component.html відповідає за основу нашого додатку. Наступний файл з такою ж назвою і розширенням .scss відповідає за стилі у нашому проєкті, а файл .ts – визначає інтерактивність сайту та можливість виконання дій. Повні лістинги файлів створеного веб-додатку представлено додатками А-Ж.

Структурно файл html складається з тегів розмітки <head>, <body> та <footer>, які відповідають за основний зміст сторінки. Зазвичай, head містить заголовок сайту, кодування сайту та теги підключення стилів, body – основне наповнення сторінки, footer – посилання на автора або іншу загальну інформацію (часто однаковий для всіх сторінок веб-додатку).

Для того щоб наповнити основну сторінку веб-додатку компонентами було встановлено Angular Material через запит командного рядку «npm install --save @angular/material». Далі з Angular Material було імпортовано контейнер для заголовку додатку, контейнери вмісту та кнопки до app-component.ts (лістинг 2.1). Додавання необхідних компонентів відбувається за рахунок методу import та вказування шляху до компонент у попередньо встановленому Angular Material.

Лістинг 2.1 – Підключення необхідних компонент з Angular Material до app-component.ts

```
import { MatButtonModule } from '@angular/material/button';
import { MatCardModule } from '@angular/material/card';
import { MatIconModule } from '@angular/material/icon';
import { MatListModule } from '@angular/material/list';
import { MatToolbarModule } from '@angular/material/toolbar';
```

Наступним кроком було додано контейнер заголовку до основи додатку - app-component.html. В середині контейнеру було прописано назву додатку за допомогою тегу та додано логотип за допомогою <img>.

Розміщення тесту було виконано за допомого тегу <p>, який є блоковим елементом, завжди починається з нового рядка, у якому абзаци тексту, що йдуть один за одним, розділяються між собою відступом. Жирне обрамлення деяких частин тесту було виконано за допомогою тегу <b>. Також було використано теги <ul> та <ol>, які відображають списки – маркований або нумерований відповідно. Заголовки текстів було оголошено за допомогою тегів <h1> – <h6>, які відповідають за шість рівнів заголовків секцій, де <h1> – це найбільший заголовок і <h6> – найменший.

Оголошення зображень наповнення блоків головного меню було виконано за допомогою тегу <img>, призначеного для відображення на веб-сторінці зображень у графічному форматі GIF, JPEG або PNG. Адреса файлу з зображенням задається через атрибут src. Також тег <img> має свої атрибути стилів для розміщення та коректного відображення на сторінці.

На лістингу 2.2 представлено оголошення заголовку веб-додатку, використання кодування символів UTF-8 та посилання на використання стилів тесту, рисунків та обрамлення сайту.

Лістинг 2.2 – Створення контейнеру заголовку додатку

```
<mat-toolbar color="primary" class="h-auto justify-content-center">
  <img src="https://www.khai-gis.info/static/blog/img/theme-3/logo-act.png"
class="logo">
  <span class="h4 p-2 mb-0">Віртуальна мадрівка територією "ХАІ"</span>
</mat-toolbar>
```

Далі було перейдено до наповнення головної сторінки. Для цього було використано тег , який являє собою окремий блок вмісту та призначений для поділу веб-сторінки на фрагменти. Далі було додано дві компоненти <mat-card>, які являють собою відокремлені контейнери з деяким вмістом. Перший контейнер <mat-card> відповідає за головну інформацію, кнопку завантаження архіву та містить фрейм для стилізованої карти Google Maps (лістинг 2.3). Для впорядкування наповнення картки також було використано теги . Відображення кнопки здійснюється за допомогою тегу <button>.

Лістинг 2.3 – Наповнення контейнеру <mat-card> з головною інформацією

```
<mat-card class="p-3 w-100 d-flex">
      <div class="w-100 d-flex h-100">
        <div class="col-8">
          <mat-card-header>
            <mat-card-title>
              <h5></h5>
            </mat-card-title>
          </mat-card-header>
          <div class="ms-3 me-3 general-info">

            <p style="color: rgb(46, 60, 102);"><img style="width: 55px; "

src='https://upload.wikimedia.org/wikipedia/commons/thumb/7/79/Logo_Luftfahrtins
titut_Charkiw.svg/1200px-Logo_Luftfahrtinstitut_Charkiw.svg.png'>
              Національний аерокосмічний університет імені М. Є. Жуковського
"Харківський авіаційний інститут" -
              провідний український вищий навчальний заклад, який пропонує
якісну освіту в галузі авіації та
              аерокосмічної
              техніки. Студентське містечко нараховує 8 навчальних корпусів, 2
науково-дослідні інститути, спортивний
              комплекс, 9 студентських
              гуртожитків, 6 житлових будинків та інші адміністративні споруди,
що розміщені на території площею близько
              50 гектарів.</p>

            <div class="text-center mt-3">
              <button>Завантажити KMZ-архів</button>
            </div>
          </div>
        </div>
        <div class="col-4 overflow-x-hidden">

        </div>
      </div>
    </mat-card>
```

Другий контейнер <mat-card> відповідає за галерею відображення 3D-моделей у форматі каруселі. Де за допомогою викликання класів дій відбувається

відображення моделей, переміщення між слайдами, анімація переміщення слайдів, смуга прогресу завантаження моделі та блоки підписів. Реалізація оголошених класів інтерактивних дій відбувалася за допомогою мови TypeScript, лістинг якої представлено додатком В.

Лістинг 2.4 – Наповнення контейнеру <mat-card> галереї 3D-моделей

```
    <mat-card class="col-12 p-3 " [ngClass]="{'full-screen': isFullscreen}">
      <mat-card-header class="justify-content-between">
        <mat-card-title>
          <h5 style="color: rgb(46, 60, 102);"> Галерея 3D-моделей</h5>
        </mat-card-title>
        <button [disableRipple]="true" class="full-screen-icon" mat-icon-button
(click)="toggleFullscreen()">
          <mat-icon>{{ isFullscreen ? 'fullscreen_exit' : 'fullscreen' }}</mat-
icon>
        </button>
      </mat-card-header>

      <div class="progress" *ngIf="totalLoaded != files.length">
        <div class="progress-bar" role="progressbar"
[style.width.%]="getLoadedPercent()"
          [attr.aria-valuenow]="totalLoaded" [attr.aria-valuemin]="0"
[attr.aria-valuemax]="files.length"> rendering
          3d models</div>
      </div>
      <div id="3d-models-carousel" class="carousel carousel-dark slide w-100 h-
100"
        [ngClass]="{'invisible': totalLoaded != files.length}" data-bs-
ride="carousel">
        <div class="carousel-indicators">
          <ng-container *ngFor="let file of files; let i = index">
            <button type="button" data-bs-target="#carouselExampleCaptions"
[attr.data-slide-to]="i"
              [ngClass]="{'active': i == currentIndex }" aria-current="true"
aria-label="Slide 1"></button>

          </ng-container>
        </div>
        <div class="carousel-inner w-100 h-100">
          <ng-container *ngFor="let file of files; let i = index" class="w-100
h-100">
            <div class="carousel-item w-100 h-100" [class.active]="i ==
currentIndex || totalLoaded != files.length">
              <app-three-model-view #3dView [pathToModel]="'assets/3d-models/' +
file.name" (loaded)="onModelLoaded(i)"
                class="w-100 h-100"></app-three-model-view>
              <div class="carousel-caption d-none d-md-block">
                <h2>{{file.title}}</h2>
                <h4 class="text-white-shadow">{{file.subTitle}}</h4>
              </div>
            </div>
          </ng-container>
        </div>
        <button class="carousel-control-prev" type="button" data-bs-
target="#carouselExampleCaptions"
```

```
          (click)="prevSlide()" data-bs-slide="prev">
          <span class="carousel-control-prev-icon" aria-hidden="true"></span>
          <span class="visually-hidden">Previous</span>
        </button>
        <button class="carousel-control-next" type="button" data-bs-
target="#carouselExampleCaptions"
          (click)="nextSlide()" data-bs-slide="next">
          <span class="carousel-control-next-icon" aria-hidden="true"></span>
          <span class="visually-hidden">Next</span>
        </button>
      </div>
    </mat-card>
```

Для реалізації стилів веб-додатку було використано безкоштовний набір інструментів Bootstrap. Інтеграцію даного фреймворку представлено лістингом 2.5. Додавання набору інструментів Bootstrap до проєктів Angular відбувається у файлі конфігурації робочого простору angular.json (додаток Д).

Лістинг 2.5 – Інтеграція фреймворку Bootstrap

```
"styles": [
  "node_modules/bootstrap/scss/bootstrap.scss",
  "src/styles.scss"
],
```

Задання стилів, застосованих для налаштовувань відображення всіх компонент веб-додатку виконано на основі інструментів фреймворку Bootstrap та представлено додатком Б.

Наступним кроком було перейдено до інтеграції інтерактивної стилізованої карти Google Maps до блоку основної інформації. Для цього було перейдено до онлайн сервісу Google My Maps та створено новий проєкт карти (рис. 2.7).

Далі було додано перейдено до області інтересу та налаштовано відображення базової карти. Наступним кроком було додано шар мітки, що позначає об’єкт обраний для моделювання (рис. 2.8).

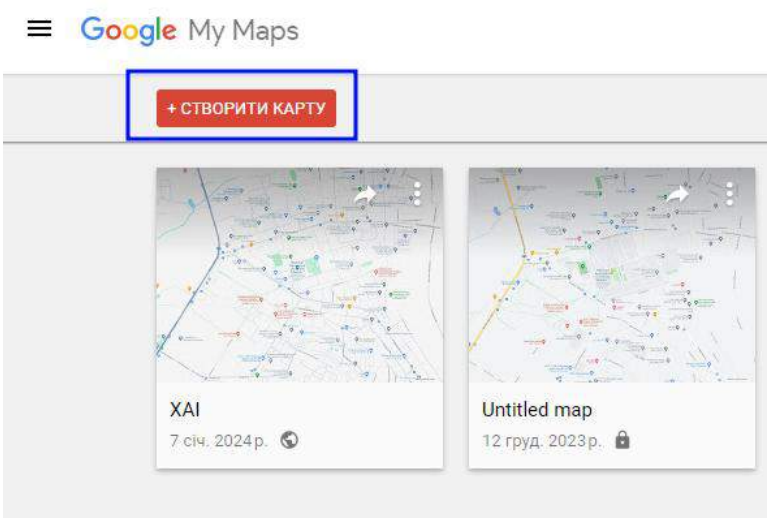


Рисунок 2.7 – Створення проєкту карти у Google My Maps

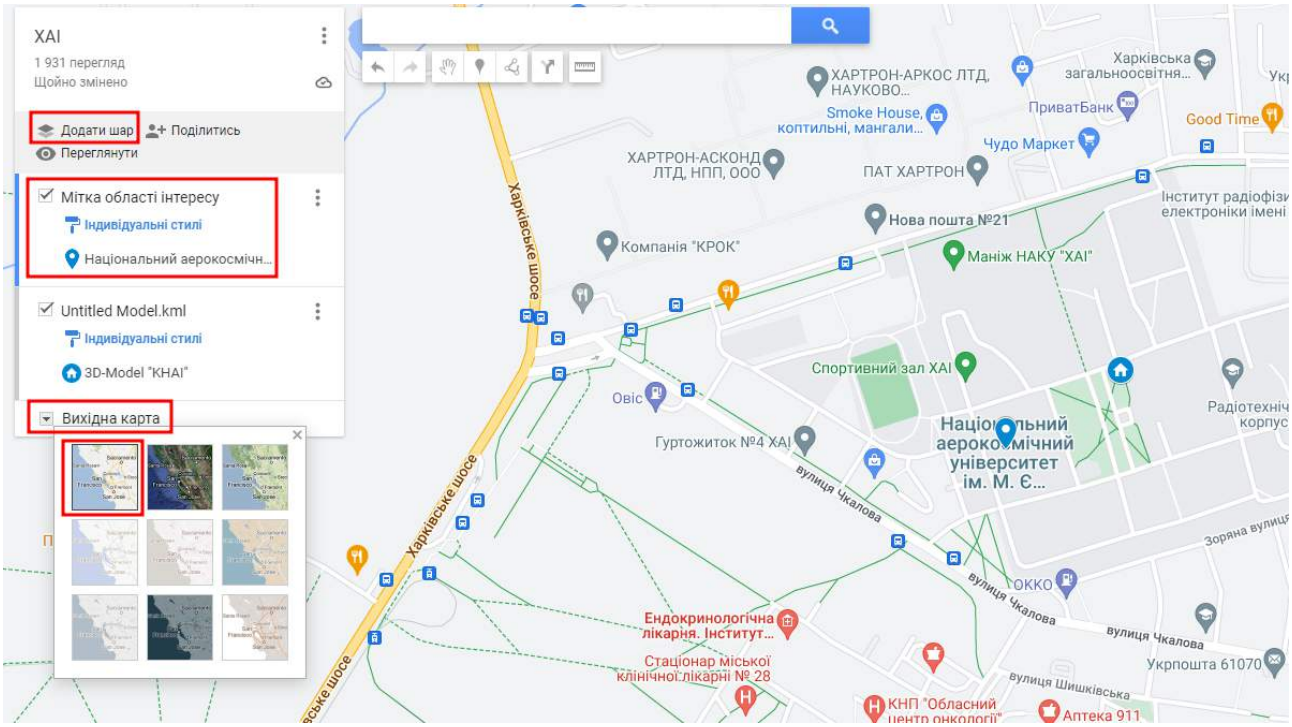


Рисунок 2.8 – Налаштування карти у Google My Maps

Після всіх налаштувань проєкту було перейдено до властивостей створеної карти та обрано функцію «Встановити на мій сайт». Дана функція генерує код тегу <iframe> HTML, котрий дозволяє інтегрувати віджети сторонніх веб-

ресурсів або будь-які інші незалежні документи. Отриманий код додавання створеної карти представлено лістингом 2.6.

Лістинг 2.6 – Наповнення тегу <iframe> для інтеграції проєкту Google My Maps

```
<iframe id="map" class="map"

src="https://www.google.com/maps/d/u/0/embed?mid=1i7mMVPbLOlpyiF8SCdiJ6kZT5d2gw6ua&ehbc=2E312F"></iframe>
```

Для стилізації та відображення карти було використано клас «map» детальний лістинг коду якого представлено у додатку Б.

Отриманий код віджету Google My Maps було інтегровано до раніше створеного тегу у контейнері <mat-card> головної інформації.

Для перегляду та перевірки компіляції розробленого коду було використано можливості Angular CLI, а саме запит «ng serve».

«ng serve» - це команда в Angular, яка використовується для збірки проєкту та запуску локального сервера розробки. Цей сервер зчитує зміни у вихідних файлах вашого Angular проєкту та автоматично оновлює веб-сторінку у браузері, щоб відобразити зміни. Використовуючи ng serve, можна легко розробляти та тестувати проєкт без необхідності вручну перезапускати сервер при зміні коду.

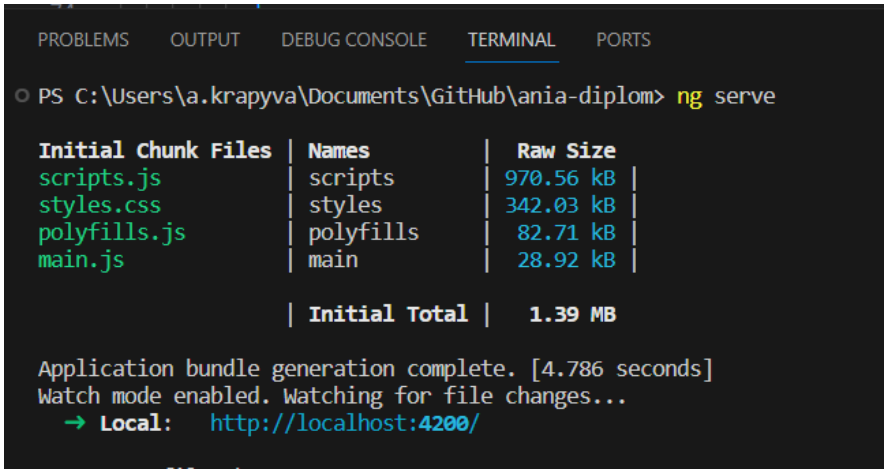


Рисунок 2.9 – Реалізація запуску запиту «ng serve»

В результаті реалізації запиту збірки проєкту та запуску локального сервера розробки, утворюється посилання на розроблений веб-додаток, який можна відкрити у браузері та відстежувати зміни в проєкті.

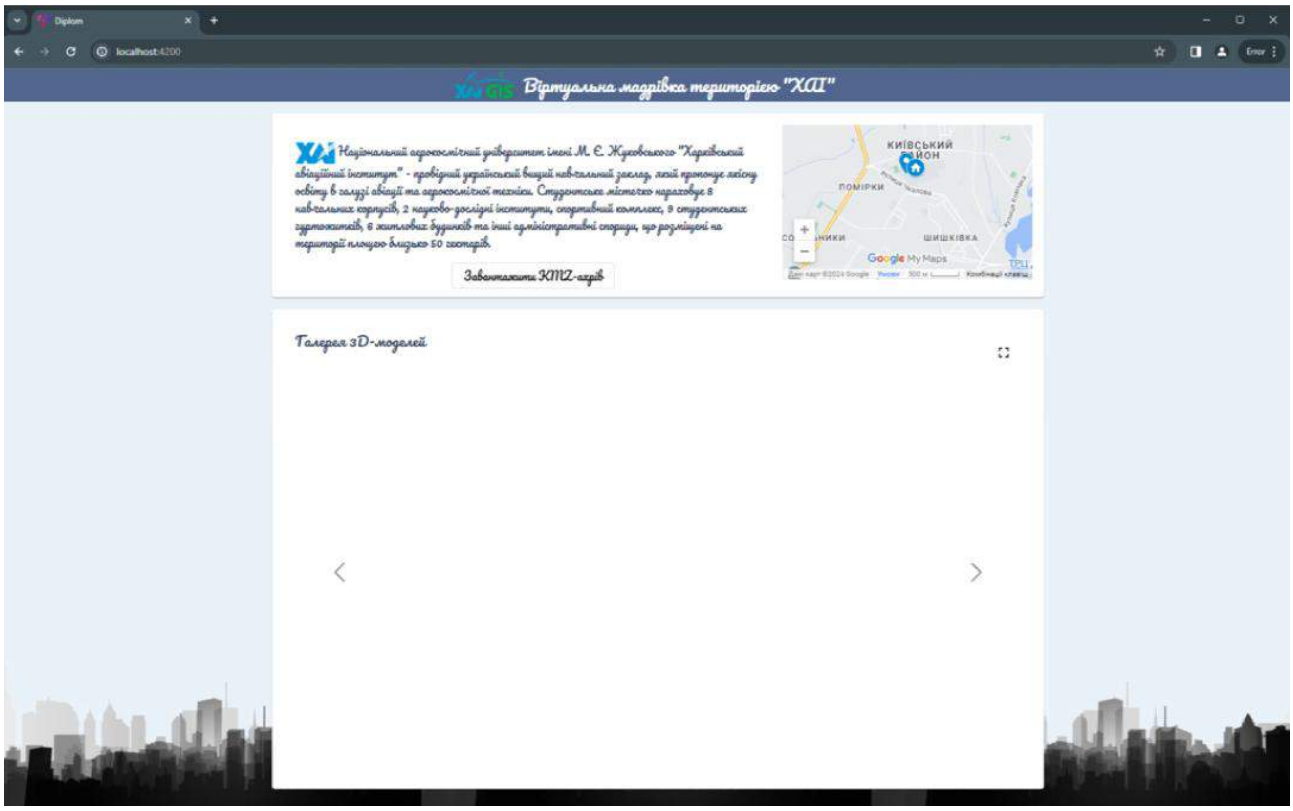


Рисунок 2.10 – Розроблений веб-додаток

В результаті зборки так запуску створеного вище коду на локальному сервері було отримано основу веб-додатку (рис. 2.10).

2.2 Отримання 3D-моделей місцевості

Для розробки тривимірних зображень картографічної спрямованості було обрано програмний продукт Blender 3.1 [18]. Можливості застосування розширень геоінформаційного характеру та реалізації всіх частин моделювання без використання сторонніх програмних продуктів є основними перевагами застосування обраної програми для реалізації методики.

В якості об’єкту моделювання було обрано територію студентського містечка Харківського Національного аерокосмічного університету ім. М.Є. Жуковського "ХАІ".

Першим кроком було розширення програмних можливостей Blender до 3D ГІС. Для цього було перейдено на відкритий вебсервіс для спільної розробки програмного забезпечення – GitHub та завантажено проєкт «Blender-OSM» [19] для доступу до ГІС технологій через програмний продукт Blender 3.1. Далі було інстальовано отримані доповнення через налаштування програми (рис. 2.11).

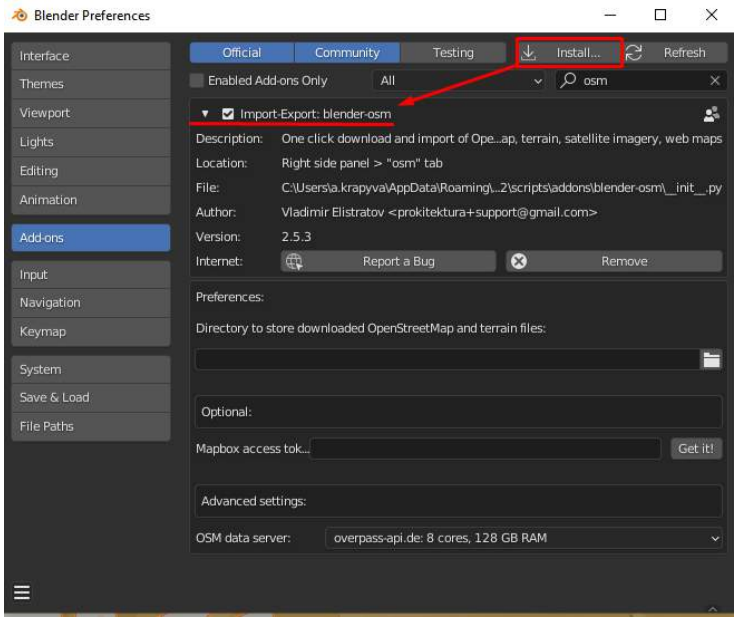


Рисунок 2.11 – Завантаження ГІС доповнення програмного продукту

Потім на панелі інструментів було обрано вкладку OSM та перейдено до інструменту виділення зони інтересу для підвантаження її до робочої області Blender. Інструмент Виділення перенаправляє до веб-сторінки з базовою мапою OpenStreetMap (OSM), де дуже легко можна виділити бажану область та скопіювати координати меж зони для завантаження даних до робочої області програми (рис. 2.12).

Рисунок 2.12 – Виділення зони інтересу

Отримані координати меж обраної області було вставлено до панелі інструментів OSM та обрано функцію імпортувати ландшафт (рис. 2.13). Дана функція будує у тривимірному просторі шар рельєфу місцевості, використовуючи базу даних OpenStreetMap (рис. 2.14).

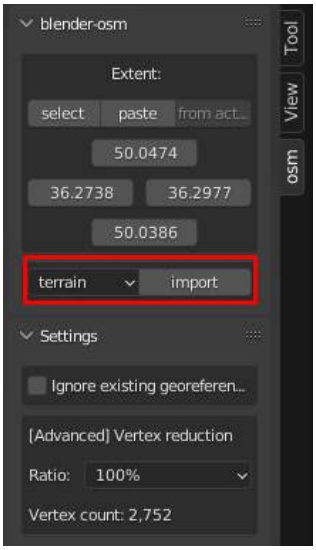


Рисунок 2.13 – Застосування функції імпорту рельєфу

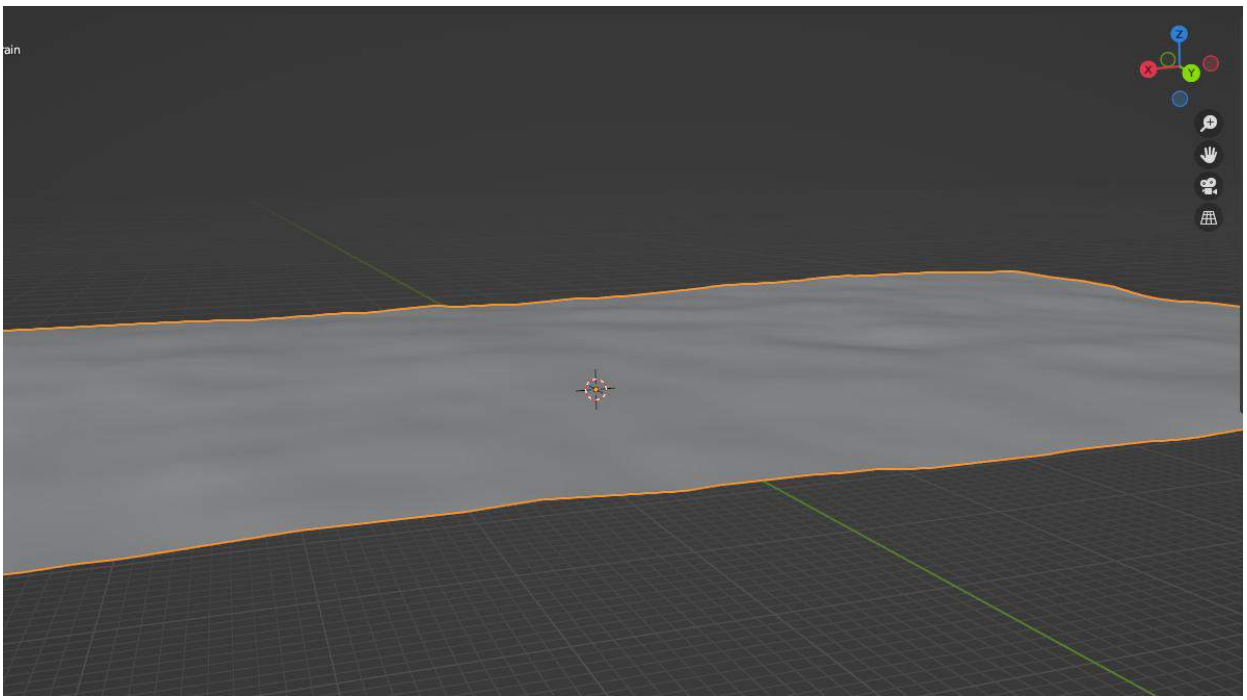


Рисунок 2.14 – Отриманий шар рельєфу місцевості

Для візуалізації місцевості та подальшого використання було завантажено зображення оверлею. В якості оверлею було обрано супутникові знімки з умовно безкоштовного сервісу Mapbox та додано через панель інструментів OSM (рис. 2.15-2.16). Для відображення рельєфу з застосуванням оверлею було перейдено на вікно текстурного відображення об’єктів.

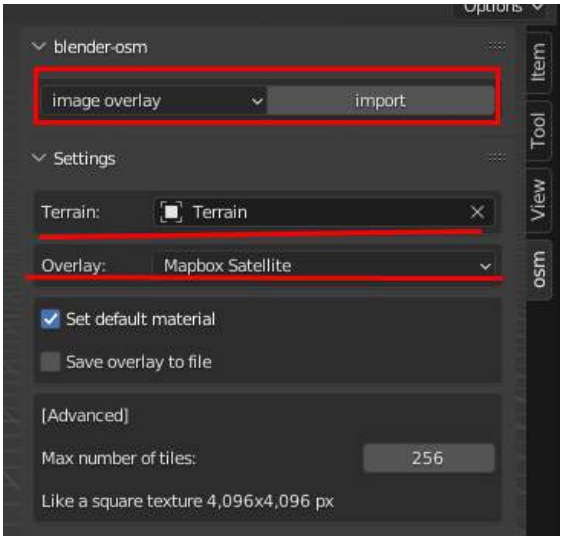


Рисунок 2.15 – Налаштування завантаження оверлею

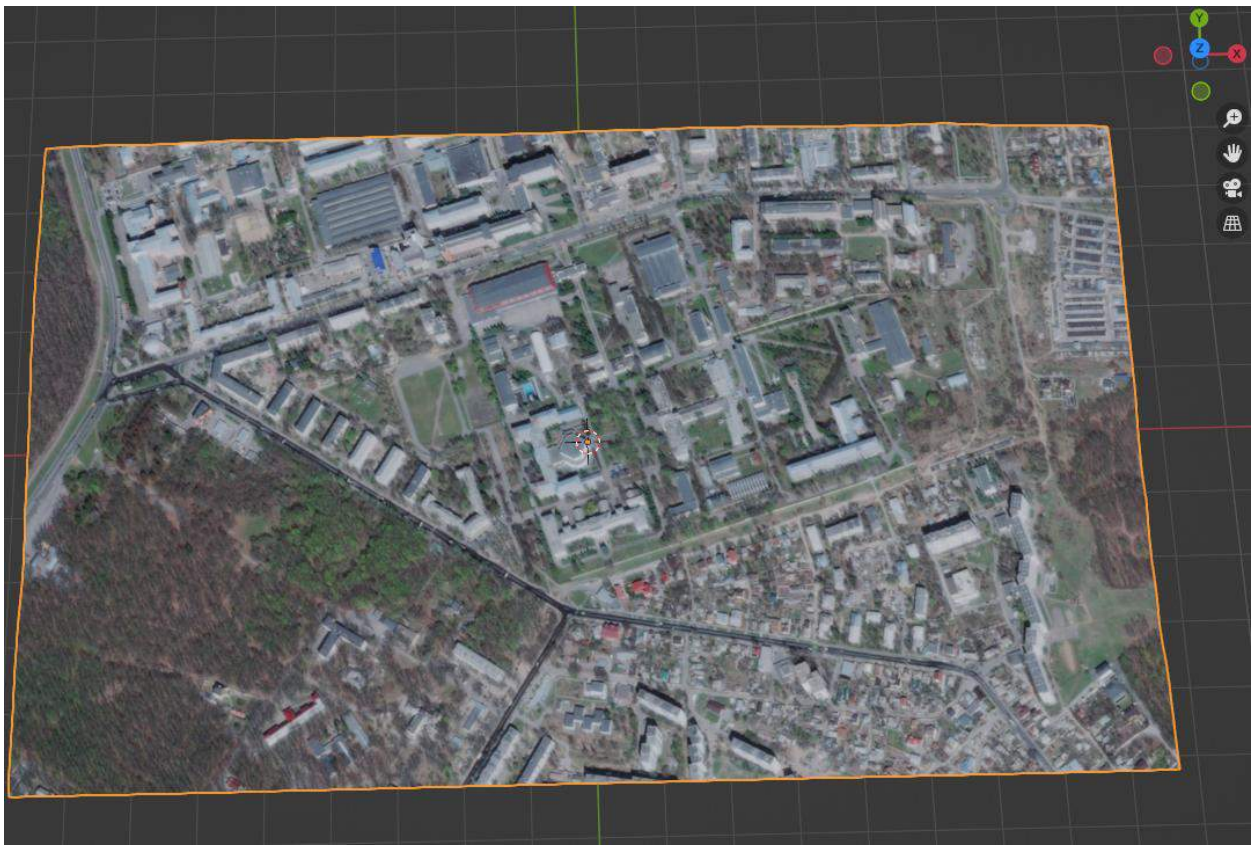


Рисунок 2.16 – Відображення текстурованого рельєфу

Далі було перейдено до отримання тривимірного зображення забудови обраної місцевості за допомогою розширення OSM. Даний модуль, використовуючи базу даних та мапу OSM, підіймає двовимірну мапу за зазначеною висотою, отримуючи об’ємні фігури будівель зазначених на мапі. Налаштування даного модулю (рис. 2.17) дозволяють отримати геодані шару будівель, рослинності, водних об’єктів, доріг та залізничних шляхів.

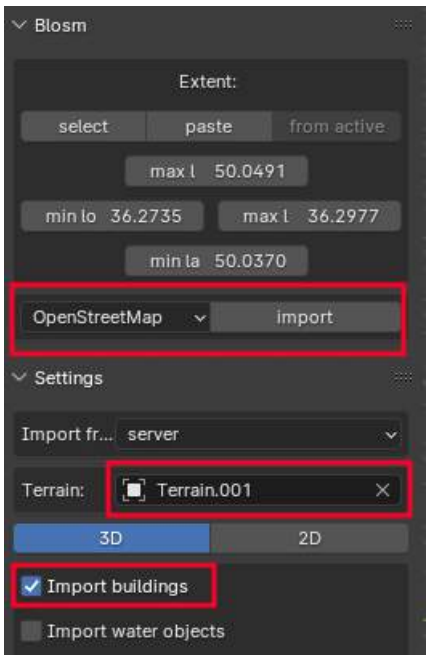


Рисунок 2.17 – Налаштування модулю отримання геоданих з OSM

Робота даного модулю потребує часу в залежності від кількості об’єктів та розмірів обраної області. Модуль використовує базу геоданих OSM, на основі яких і підіймає картографічну основу за висотними показниками. Після застосування модулю було отримано тривимірну модель будівель (рис. 2.18).

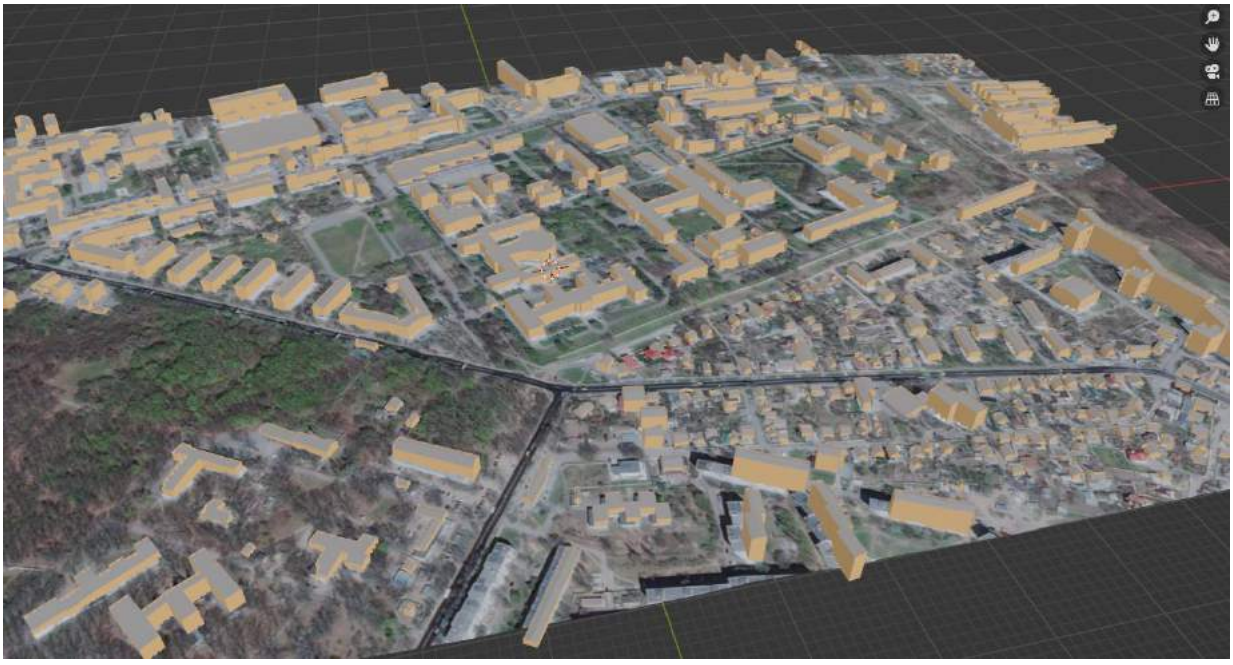


Рисунок 2.18 – Тривимірна модель забудови

Оскільки форма отриманих моделей будівель не зовсім відповідає реальній конфігурації, через створення мапи OSM методом прямої векторизації космічних знімків, було прийнято рішення дещо модифікувати отриману модель.

Спочатку було проаналізовано та порівняно отримані моделі з реальними будівлями за допомогою фото та Google Street View, для виявлення фронту роботи. Далі було перейдено до режиму редагування та допомогою інструментів редактору налаштовано вигляд дахів, етажність та екстер'єр отриманої забудови.

Оскільки отримана модель все ще не дуже відповідає реальній конфігурації, то наступним кроком було налаштування відображення текстур поверхонь отриманої 3D-моделі. Спочатку було перейдено до налаштування відображення дахів будівель моделі. Для цього було відокремлено поверхню дахів від іншої частини будівель та накладено на них текстуру оверлею.

Далі було перейдено до текстурування площини стін будівель моделі. Для цього було завантажено фотознімки екстер'єру будівель з відкритих інтернет ресурсів та особистих архівів. Отримані зображення було поділено на матеріали та застосовано окремо до кожної будівлі. Також, в якості текстур для деяких площин було використано заливку кольором. Отриману текстуровану зображено на рис.2.19.

Рисунок 2.19 – Отримана текстурована модель

Для збільшення деталізації та наочності, було прийнято рішення відокремити одну будівлю від створеної моделі на попередньому етапі та наблизити її відображення до реального. Для цього, у програмному продукті Blender, було відкрито експортовану модель та виділено будівлю навчально-лабораторного корпусу «ХАІ». Далі цю будівлю було сепаровано від основної моделі та експортовано, також було експортовано модель рельєфу. Потім було створено новий проєкт Blender, у який імпортовано ці моделі.

Рисунок 2.20 – Отриманий проєкт моделі НЛК «ХАІ»

Наступним кроком, за допомогою інструментів редагування моделі, було деталізовано зовнішній вигляд будівлі. А саме, було модернізовано фасади,

різнорівневу конструкцію частин корпусу та деталізовано інші частини екстер’єру будівлі. Також було обрізано модель рельєфу до масштабів будівлі.

Потім було налаштовано вигляд моделі рельєфу під корпусом. Для цього моделі було надано об’єм, задано межі накладення газону та пішохідних доріжок (рис. 2.21).

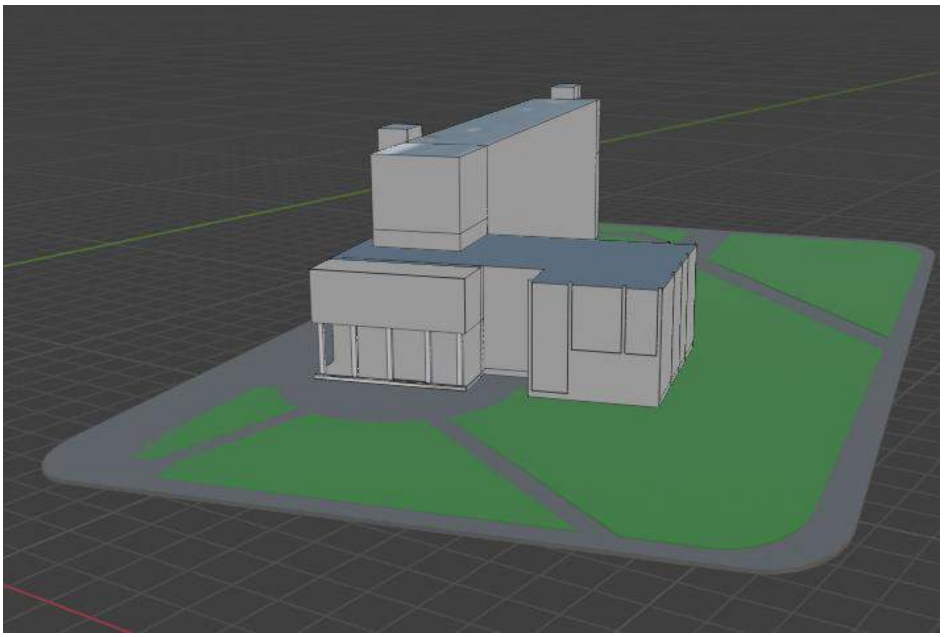


Рисунок 2.21 – Отриманий модернізований вигляд моделі НЛК «ХАІ»

Отримана модель вже більш наочно демонструє вигляд навчального корпусу, але для наближення до реальності було прийнято рішення додати деталі навколо будівлі – лавочки, пам’ятники та дерева. Для спрощення реалізації даного вигляду моделі дерев, лавочок та літака було отримано з безкоштовного онлайн ресурсу обміну тривимірної графіки – Sketchfab [20]. Завантажені моделі було дещо модернізовано та налаштовано для об’єднання з моделлю корпусу. Також, за допомогою найпростіших фігур було реалізовано модель пам’ятнику «Шарі», що знаходиться бля входу до корпусу.

Далі отриманими моделям було задано текстури і матеріали для більш реального вигляду.

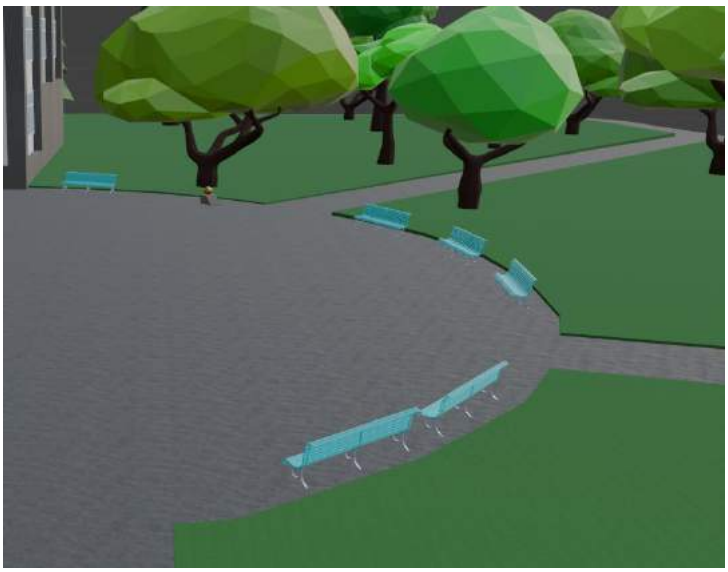


Рисунок 2.22 – Деталізація вигляду місцевості додаванням моделей ситуації

Далі було об’єднано отриману модель корпусу з моделями деталізації і таким чином було отриману детальну, наближену до реальності модель місцевості навколо навчально-лабораторного корпусу «ХАІ» (рис. 2.23).

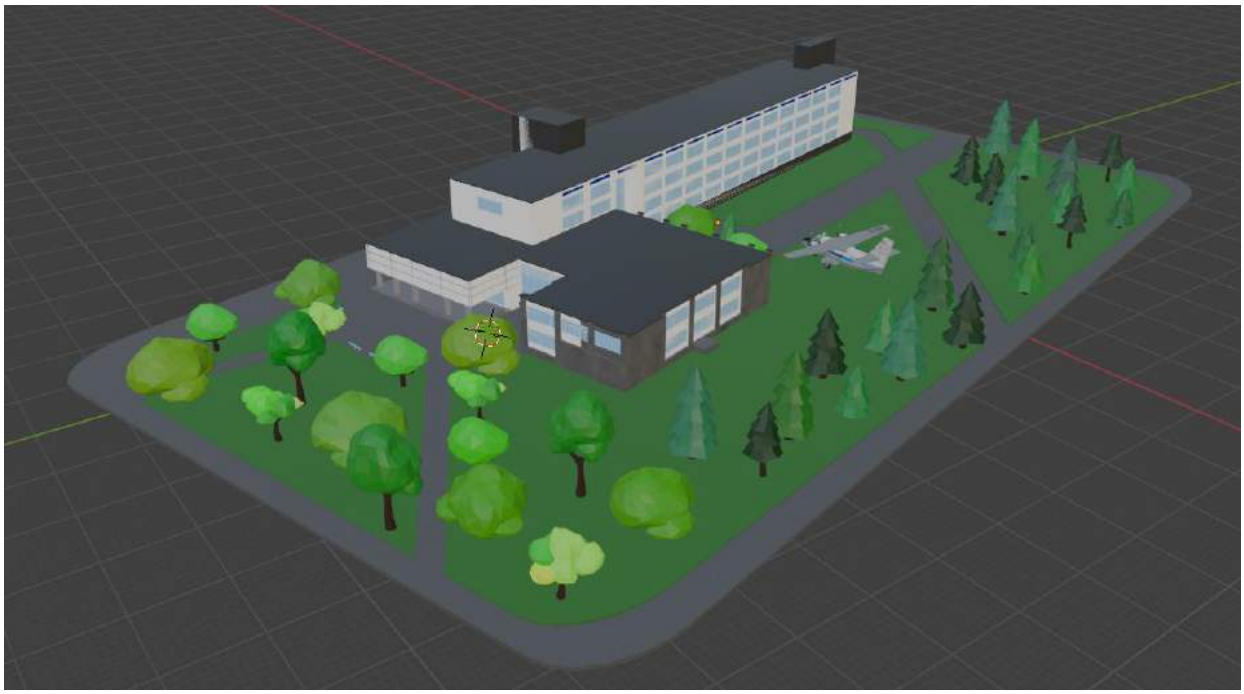


Рисунок 2.23 – Отримана модель навчально-лабораторного корпусу ХАІ

Наступним етапом є отримання файлу побудованої моделі у розширенні візуалізації тривимірних мап компанії Google – KMZ. Для цього було експортовано отриману модель у форматі збереження 3D-графіки – COLLADA файлом розширення .dae. Потім було запущено програмний продукт Google Earth та завантажено, раніше експортовану, 3D модель студмістечка ХАІ та налаштовано її відносно поверхні Землі за координатою Z. Після чого було збережено тривимірну модель як KMZ файл. Формат KMZ використовується для географічного опису та візуалізації двовимірних та тривимірних мап компанії Google.

Файли KMZ здатні відображати інформацію у програмах Here Maps, Google Earth Maps та інших геопросторових додатках. Широта та довгота використовуються у поєднанні зі значеннями ухилу та підйому місцевості, висоти, причому всі ці значення дозволяють формувати тривимірне зображення місцевості. Файли KMZ являють собою файли KML, а також охоплюють будь-яку кількість підтримуваних файлів, заархівованих і стислих за допомогою методу Zip.

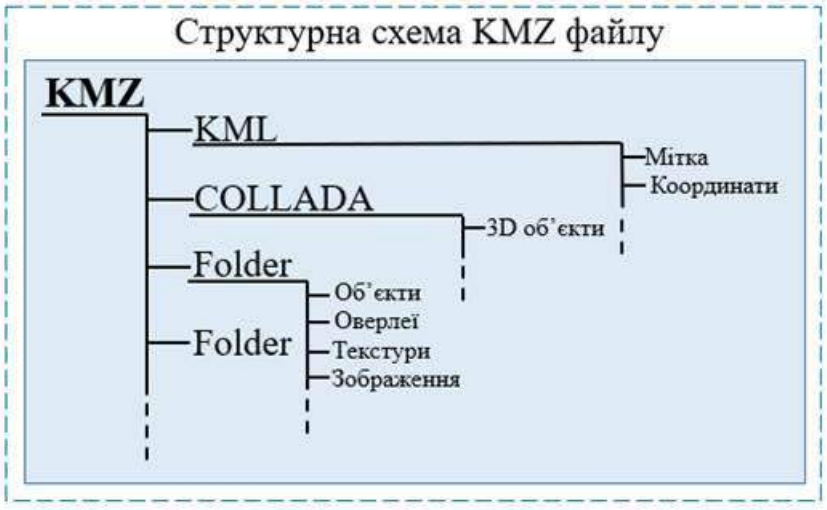


Рисунок 2.24 – Структурна схема файлу 3D-моделі у форматі KMZ

Файли KMZ можуть містити велику кількість різноманітних файлів, що містять зовнішні шари, зображення, тривимірні моделі та ін на рівні кореневого

документа. Файли KML та KMZ є предметом вивчення у рамках розробки міжнародного стандарту, що здійснюється консорціумом Open Geospatial Consortium [21].

Останнім етапом створення тривимірних моделей є отримання файлів сцени у форматі передачі та завантаження 3D моделей у веб та нативні застосунки – glTF.

glTF (GL Transmission Format) є відкритим стандартом для передачі та обміну тривимірних моделей у веб-розробці. Створений Консорціумом Khronos Group, glTF виходить за межі простої форми обміну файлами і включає в себе набір стислених та оптимізованих даних, що робить його ідеальним для використання у веб-проєктах, AR (розширеної реальності) та VR (віртуальної реальності) застосуваннях. glTF зберігає інформацію про 3D-модель у форматі JSON. Використання JSON мінімізує як розмір 3D-ресурсів, так і обробку під час виконання, необхідну для розпакування та використання цих ресурсів [22].

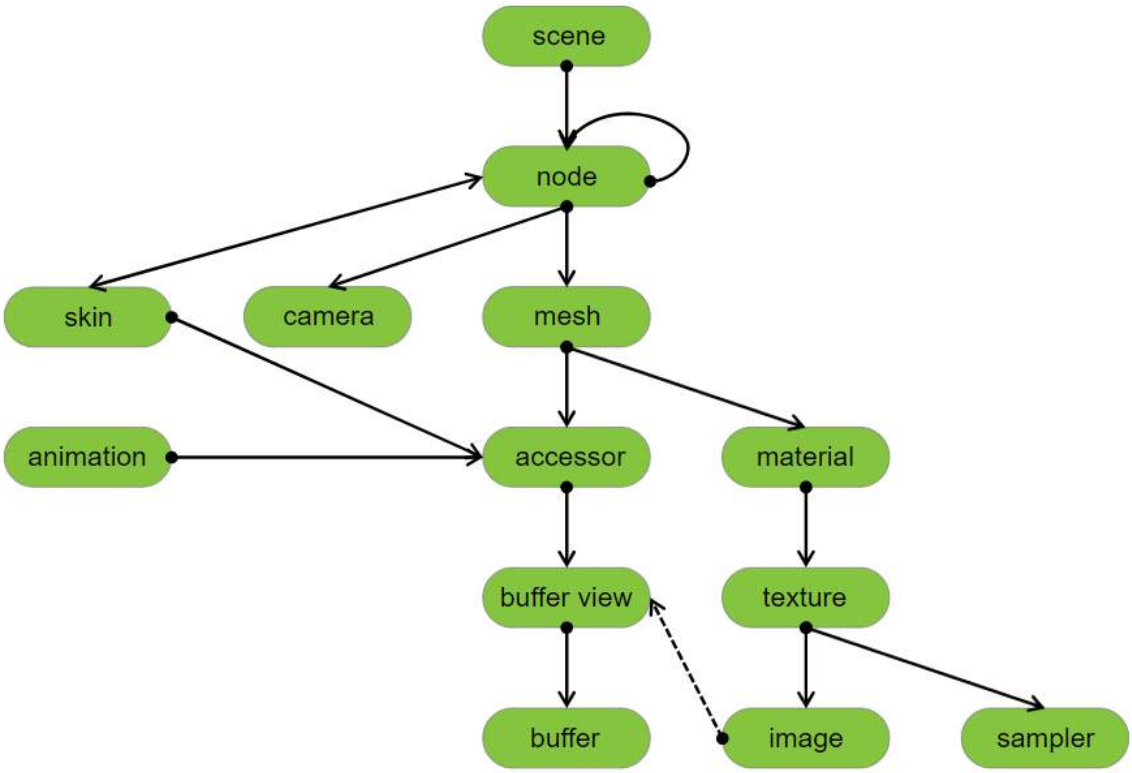


Рисунок 2.25 – Структурна схема файлу 3D-моделі у форматі glTF

Для отримання файлів моделей формату glTF, було встановлено розширення до середовища розробки моделей – Blender. Розширення «Blender glTF 2.0 Importer and Exporter» було отримано з відкритого вебсервіс для спільної розробки програмного забезпечення – GitHub [23]. Після встановлення розширення у прогрманому продукті з’являється можливіть експорту моделі у форматі передачі та обміну для веб-розробки.

Далі було перейдено до меню файлів та обрано функцію «Експорт у glTF».

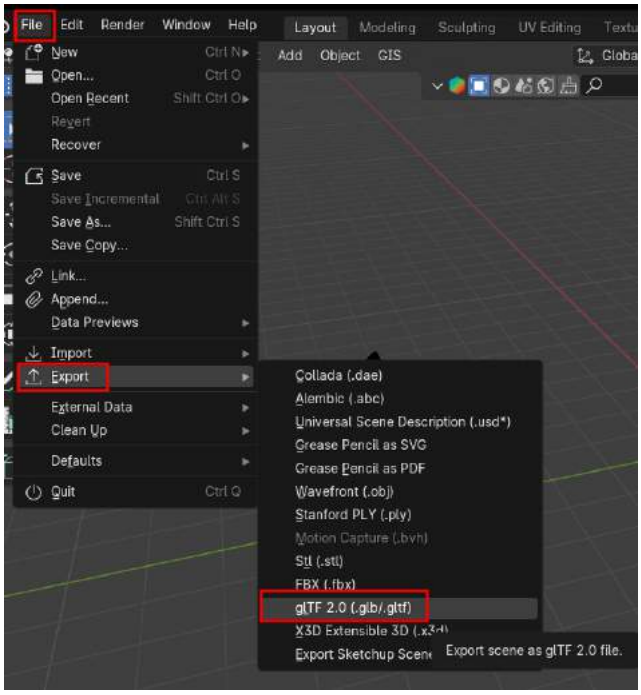


Рисунок 2.26 – Отримання тривимірних моделей у форматі glTF

2.3 Інтеграція 3D-моделі в структуру Web-додатку

Останнім етапом методики є інтеграція отриманих тривимірних моделей в структуру розробленого веб-додатку. Для подальшого використання та впровадження отриманих 3D-моделей у веб-додатку було завантажено файли створених моделей до теки статичних активів проєкту Angular – теки assets.

Першим кроком було інтегровано отриманий KMZ файл моделі студмістечка ХАІ до раніше створеного класу кнопки «Завантажити KMZ-файл»

у структурі контейнера <mat-card> головної інформації (лістинг 2.7). Також було прописано логіку виконання завантаження файлу з веб-додатку при натисканні кнопки (лістинг 2.8). Натискання кнопки завантаження архіву викликає клас "onDownloadArchiveClick()", який посилається на додану модель у assets та використовує можливості браузера для збереження файлу.

Лістинг 2.7 – Реалізація HTML кнопки «Завантажити KMZ-файл»

```
<div class="text-center mt-3">
              <button (click)="onDownloadArchiveClick()" mat-stroked-
button>Завантажити KMZ-архів</button>
            </div>
```

Лістинг 2.8 – Реалізація логіки (TS) класу натискання кнопки «Завантажити KMZ-файл»

```
onDownloadArchiveClick() {
    fetch('assets/archives/KHAI.kmz')
      .then((response) => response.blob())
      .then((blob) => {
        saveAs(blob, 'KHAI.kmz');
      })
      .catch((error) => {
        console.error('Error fetching the file:', error);
      });
  }
```

Наступним кроком було створено нову компоненту Angular, що відповідає за завантаження та візуалізацію отриманих тривимірних моделей у «Галереї 3D-моделей». Було прийнято рішення використати саме компоненту для реалізації цієї цілі, бо вона надає можливість уникнути повторювання циклів відображення для декількох моделей. Також, застосування компоненти відображення моделей надає перевагу у зменшенні коду при майбутньому розширенні веб-додатку та відображенні тривимірних сцен на інших сторінках.

Створення компоненти для налаштувань відображення тривимірних сцен було виконано Angular CLI командою «ng generate component three-model-view». Ця команда створює новий каталог з заданою назвою (three-model-view) та всіма

необхідними файлами виконання компоненти (.html, .css, .ts) у кореневій системі веб-додатку.

Для відтворення тривимірних зображень у структурі веб-додатку було використано бібліотеку відображення графіки Three.js. Встановлення бібліотеки виконується через термінал командного рядку за допомогою команди «npm install --save three». Імпортування самої бібліотеки та її інструментів у проєкт веб-додатку було виконано за допомогою інструменту «import{}» у файлі виконання подій створеної компоненти відображення 3D-моделей – three-model-view.component.ts (лістинг 2.9).

Лістинг 2.9 – Імпортування бібліотеки та інструментів Three.js

```
import * as THREE from 'three';
import { GLTFLoader } from 'three/examples/jsm/loaders/GLTFLoader.js';
import { OrbitControls } from 'three/examples/jsm/controls/OrbitControls.js';
```

Роботу з 3D графікою WebGL за допомогою Three.js можна умовно розбити на такі етапи: налаштування сцени, завантаження моделі, додавання камери, налаштування світла (освітлення), рендеринг (візуалізація) та анімація.

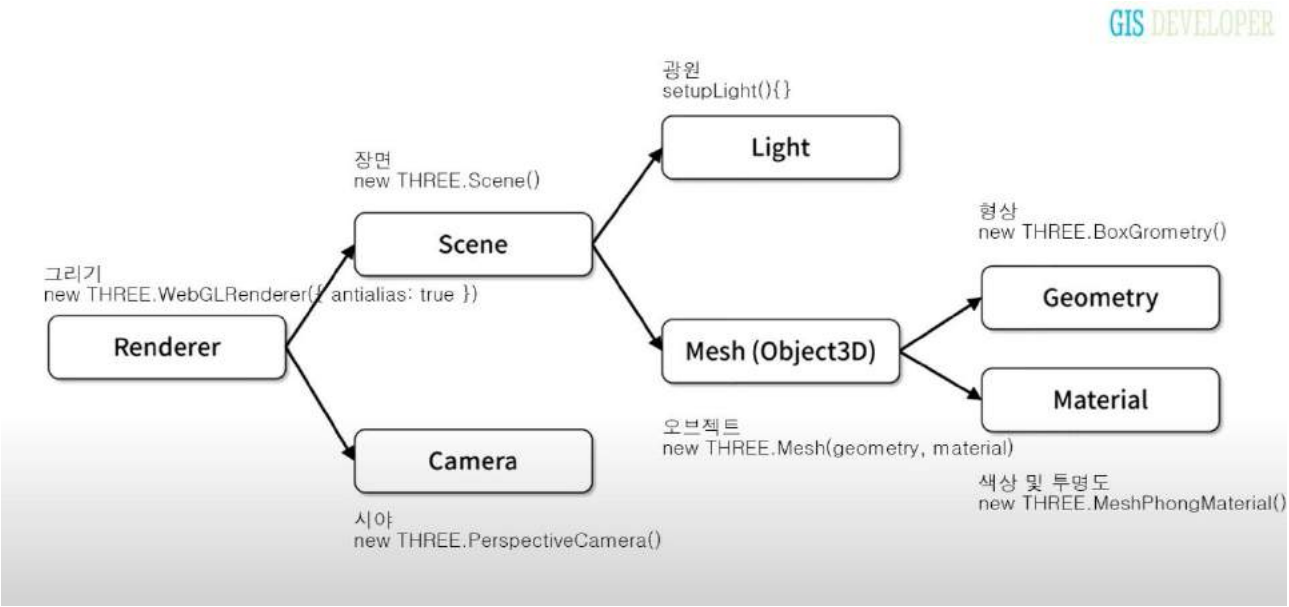


Рисунок 2.27 – Структурна схема візуалізації 3D графіки за допомогою Three.js

Першим кроком було оголошено основні елементи візуалізації тривимірної графіки: Scene – площина для відображення, PerspectiveCamera – інструмент захоплення об’єктів у площині відображення, Renderer – візуалізатор, який дозволяє показувати сцену захоплену камерою та Loader – завантажувач моделей.

Лістинг 2.10 – Оголошення основних змінних візуалізації 3D графіки

```
scene = new THREE.Scene();
camera!: THREE.PerspectiveCamera;
renderer = new THREE.WebGLRenderer();
loader = new GLTFLoader();
```

Далі потрібно задати характеристики для оголошених елементів, тобто створити їх з указаними параметрами відображення. Для зручності і читабельності коду всі аспекти створення сцени було розбито по методам.

Перший метод setSceneProperties було створено для задання параметрів сцені (лістинг 2.11). За замовчуванням сцена додається з чорним кольором фону, але для кращого відображення моделей цей колір було змінено на білий. Встановлення параметру кольору може бути важливим для задання загальної атмосфери сцени та забезпечення чіткості відображення об'єктів.

Лістинг 2.11 – Метод створення сцени – setSceneProperties

```
setSceneProperties() {
    this.scene.background = new THREE.Color(0xffffff);
  }
```

Наступним кроком було створено метод addCamera(), що дозволяє задавати параметри створюваної камери (лістинг 2.12). В середині цього методу створюється новий об'єкт камери за допомогою конструктора THREE.PerspectiveCamera. Параметри цього конструктора вказують на різні властивості створюваної камери, і на те що вона буде захоплювати. Перший параметр вказує на поле зору камери у градусах. Другий параметр вказує на відношення ширини до висоти камери. Зазвичай це відношення відповідає

аспектному відношенню вікна чи контейнера рендерера. Відповідне використання ширини та висоти допомагає уникнути деформацій об'єктів на відображенні. Третій параметр вказує на ближню площину відсічення (near clipping plane). Все, що знаходиться ближче до камери за цією площиною, буде відсічено і не буде відображено. Четвертий параметр вказує на дальню площину відсічення (far clipping plane). Все, що знаходиться далі від камери за цією площиною, також буде відсічено. Так, як ми будемо відображати доволі великі об’єкти (модель студмістечка) вище описані параметри було задано з високими значеннями, щоб створити широкий кут огляду та відобразити максимум об’єктів.

Лістинг 2.12 – Метод створення камери – addCamera()

```
addCamera() {
    this.camera = new THREE.PerspectiveCamera(75,
this.rendererContainer.nativeElement.clientWidth /
this.rendererContainer.nativeElement.clientHeight, 0.2, 2000);
  }
```

Так, як можливі завантажуванні моделі можуть мати різний масштаб, то при рендеренгу сцени застосовувати однакові параметри камери було б не доречно. Тому було прийнято рішення розробити метод dinamicCameraPositionOnLoad (лістинг 2.13), який динамічно встановлює позицію камери на основі розмірів моделі, так щоб камера охоплювала всю модель.

Лістинг 2.13 – Метод динамічного встановлення камери, щодо розмірів завантажуваної моделі

```
dinamicCameraPositionOnLoad(gltf: any) {
    const boundingBox = new THREE.Box3().setFromObject(gltf.scene);
    // Get the dimensions (width, height, depth) of the bounding box
    const width = (boundingBox.max.x - boundingBox.min.x) / 3;
    const height = (boundingBox.max.y - boundingBox.min.y) + (boundingBox.max.y
- boundingBox.min.y) * 0.2;
    const depth = (boundingBox.max.z - boundingBox.min.z) / 1.5
    this.camera.position.set(width, height, depth)
    this.isCameraPositionApplied = true;
  }
```

Далі було задано метод loadModel(), що відповідає за завантаження тривимірної моделі у сцену (лістинг 2.14). Так, як реалізація відображення моделей створюється не для одного об’єкту, було створено функцію для асинхронного завантаження моделі з шляху this.pathToModel (він знаходиться у теці статичних активів проєкту assets). Для коректного відображення моделей було створено функцію перевірки застосовної позиції камери до сцени. Якщо камера не застосована, то викликається метод this.dinamicCameraPositionOnLoad(gltf). Це може вказувати на те, що позицію камери слід адаптувати до завантаженої моделі. Після перевірки встановленої камери відбувається саме завантаження моделі методом this.scene.add(gltf.scene) і викликання методу відображення декількох моделей в одній сцені traverseModel(node: THREE.Object3D).

Лістинг 2.14 – Метод завантаження моделей до сцени loadModel()

```
loadModel() {
    this.loader.load(this.pathToModel, (gltf) => {
      if (!this.isCameraPositionApplied) {
        this.dinamicCameraPositionOnLoad(gltf);
      }
      this.traverseModel(gltf.scene)
      this.scene.add(gltf.scene)
    });
  }
```

Наступним кором було створено метод рендерінгу addRenderer(), що візуалізує те що відбувається у сцені і захоплено камерою (лістинг 2.15). Спочатку, за допомогою функції setSize було задано розмір створюваного відображення, з параметрами контейнеру де будуть відображатися моделі, тобто розмірами контейнеру галереї. Далі було створено саме площину відображення, яку потім було активовано у створеній галереї.

Лістинг 2.15 – Метод реалізації рендеренгу сцени

```
addRenderer() {
    this.renderer.setSize(this.rendererContainer.nativeElement.clientWidth,
this.rendererContainer.nativeElement.clientHeight);
    this.renderer.shadowMap.enabled = true;
    this.rendererContainer.nativeElement.appendChild(this.renderer.domElement);
  }
```

Таким чином було додано ключові елементи візуалізації тривимірних моделей, які вже зараз відображаються на сторінці веб-додатку. Але вказані характеристики сцени не дають можливості відобразити налаштовані текстури та шейдери моделей – моделі відображаються темними об’єктами у сцені (рис. 2.8). Тому для коректного відображення моделей було прийнято рішення додавання джерела освітлення сцени.

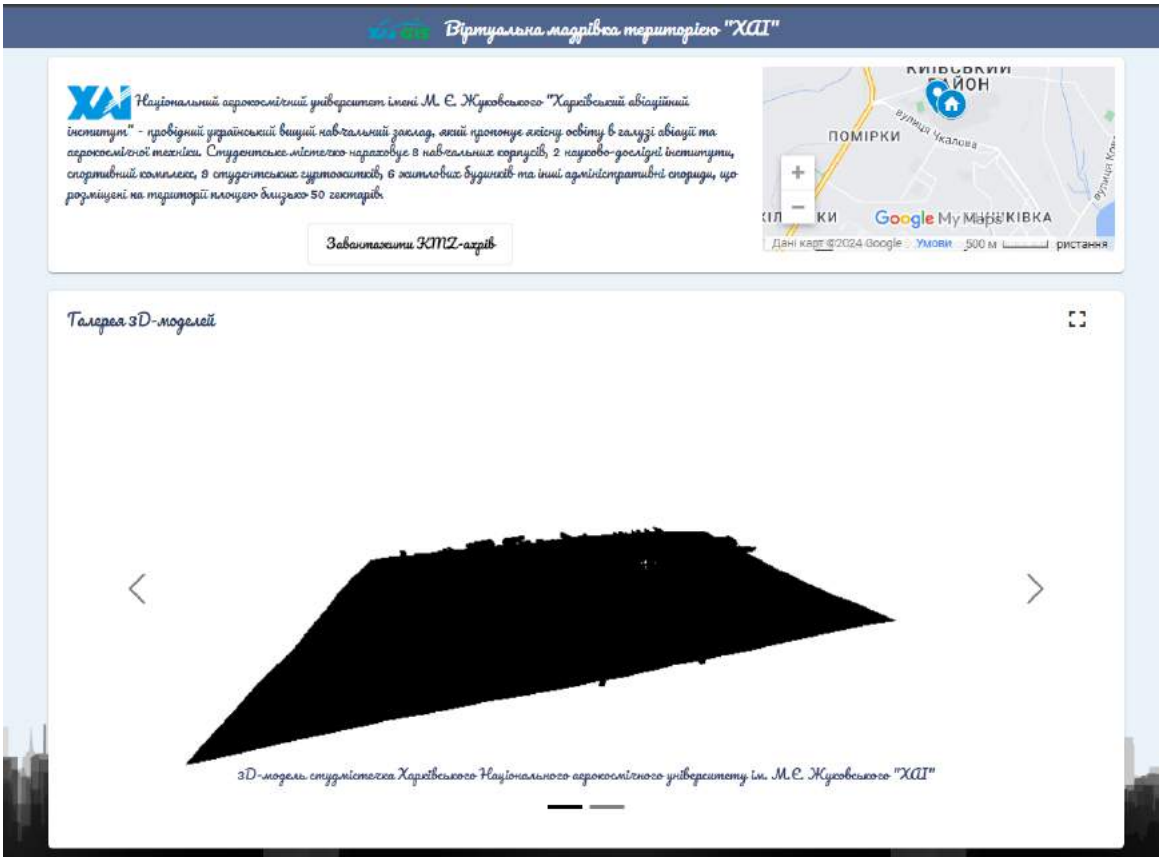


Рисунок 2.27 – Розроблений початковий вигляд відображення моделей у веб-додатку

Для реалізації освітлення сцени було створено метод addGlobalLight (лістинг 2.16). В середині методу було створено об'єкт HemisphereLight, який представляє собою світло, розподілене навколо всього простору сцени. Параметри даного об’єкту вказують на кольори світла на небі та на землі, а також на інтенсивність світла. Також, дане освітлення має параметр тіней hemiLight.castShadow, який дозволяє відображати тіні об’єктів у сцені. Цей параметр важливий для створення реалістичного візуального ефекту в тривимірних сценах, де об'єкти викидають тіні на інші об'єкти або поверхні.

Лістинг 2.16 – Налаштування відображення світла методом addGlobalLight

```
addGlobalLight() {
    const hemiLight = new THREE.HemisphereLight(0x0000ff, 0x00ff00, 0.6);
    hemiLight.color.setHSL(0.6, 1, 0.6);
    hemiLight.groundColor.setHSL(0.095, 1, 0.75);
    hemiLight.position.set(0, 50, 0);
    hemiLight.castShadow = true;
    this.scene.add(hemiLight)
  }
```

Після реалізації джерела світла можна побачити відображення моделі з текстурами (рис. 2.28), але отриманий варіант візуалізації не передає реалістичність створених моделей. Тому для додавання більшої реалістичності було прийнято рішення створити ще одне джерело світла, яке б імітувало сонячне освітлення та додавало б створеним моделям натуральний вигляд.

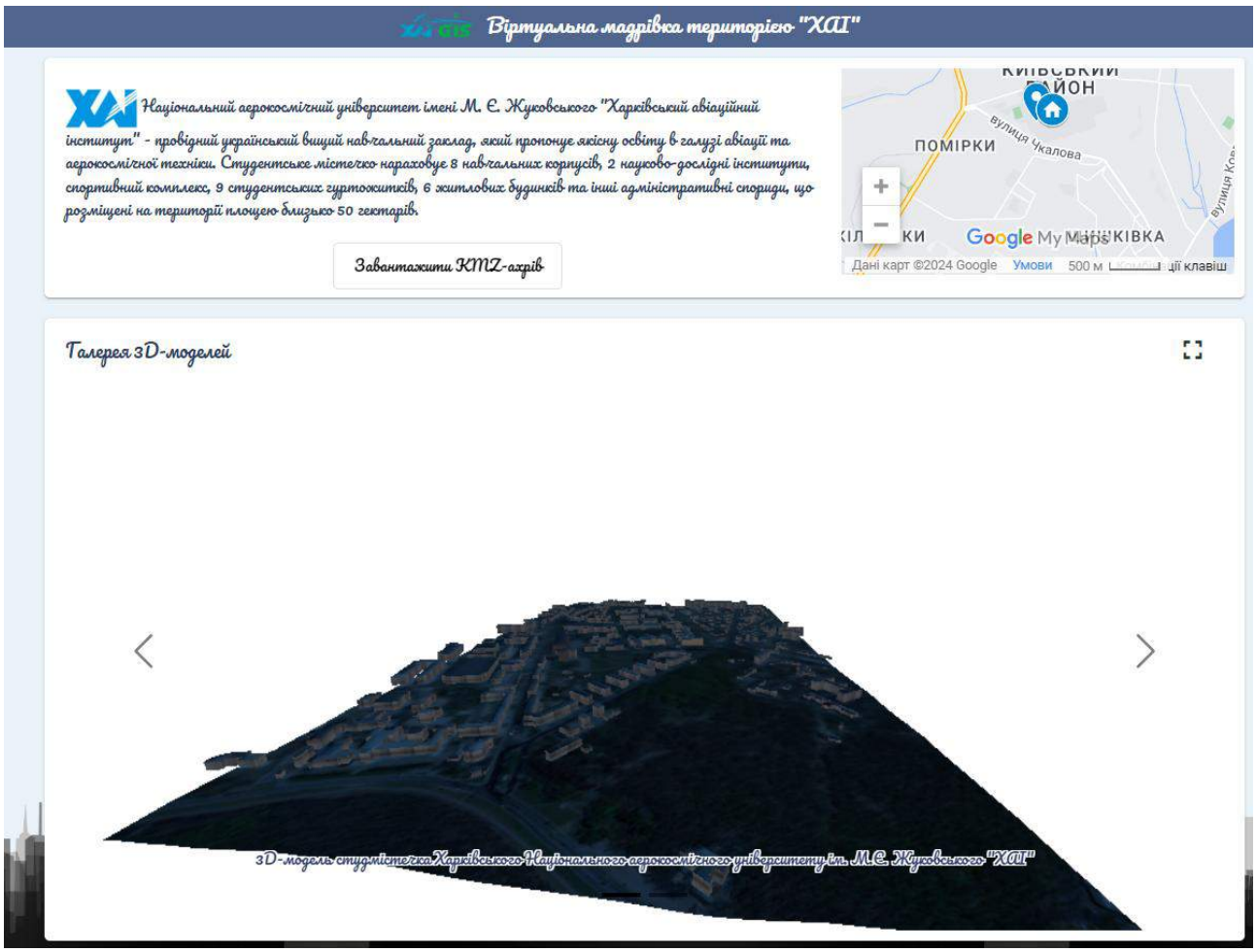


Рисунок 2.28 – Візуалізація сцени моделей з одним джерелом освітлення

Наступним етапом є додавання управління орбітою до сцени – надання можливості інтерактивної взаємодії з тривимірною сценою. Для цього було створено метод addOrbitControls (лістинг 2.17), який відповідає за створення та додавання об'єкта управління орбітою до сцени.

OrbitControls - це частина бібліотеки Three.js, яка надає інтерактивне управління камерою. Зазвичай використовується для можливості обертання, масштабування та переміщення камери в тривимірній сцені за допомогою миші або тачпаду.

Лістинг 2.17 – Задання методу інтерактивного управління моделями – addOrbitControls

```
addOrbitControls() {
    this.controls = new OrbitControls(this.camera, this.renderer.domElement);
  }
```

Далі для того щоб наблизити моделі до реальності було створено два методи відображення сонячного світла: джерело світла - метод addSunLight (лістинг 2.18) та об’єкт імітації сонця - метод addSun (лістинг 2.19).

Метод addSunLight() відповідає за додавання напрямного світла (світла від сонця) до сцени. Спочатку створюється об'єкт напрямного світла (DirectionalLight), Параметри якого включають в себе колір світла у форматі HEX та інтенсивність світла. Далі встановлюється позиція світла у тривимірному просторі, у даному випадку це визначає напрямок, з якого світить сонце. Створене джерело світла також має властивість створення тіней об’єктів, яку було активовано за допомогою функції this.sunLight.castShadow = true. Така властивість особливо важлива для створення реалістичного освітлення та тіней у тривимірних сценах. Відображення тіней має свої параметри: розмір, радіус, область генерування, зсув від об’єкту та генерування біля джерела, які відповідають за те як і куди будуть відкидатися тіні від об’єктів. Збільшення показників відображення карт тіней може призвести до більшої деталізації тіней, але може також вимагати більше ресурсів.

Лістинг 2.18 – Реалізація методу сонячного освітлення addSunLight

```
addSunLight() {
    this.sunLight = new THREE.DirectionalLight('#f6e12473', 10); // Color,
Intensity
    this.sunLight.position.set(100, 120, 100); // Set position (represents sun
direction)
    this.sunLight.castShadow = true; // Enable shadow casting by the sun
    this.sunLight.shadow.mapSize.width = 8096;
    this.sunLight.shadow.mapSize.height = 8096;
    // Enable soft shadows
    this.sunLight.shadow.radius = 10;
    this.sunLight.shadow.camera.top = 270;
    this.sunLight.shadow.camera.bottom = -170;
```

```
    this.sunLight.shadow.camera.left = -200;
    this.sunLight.shadow.camera.right = 200;
    this.sunLight.shadow.camera.near = 25;
    this.sunLight.shadow.camera.far = 350;
    this.sunLight.shadow.bias = -0.005;
    this.scene.add(this.sunLight)
}
```

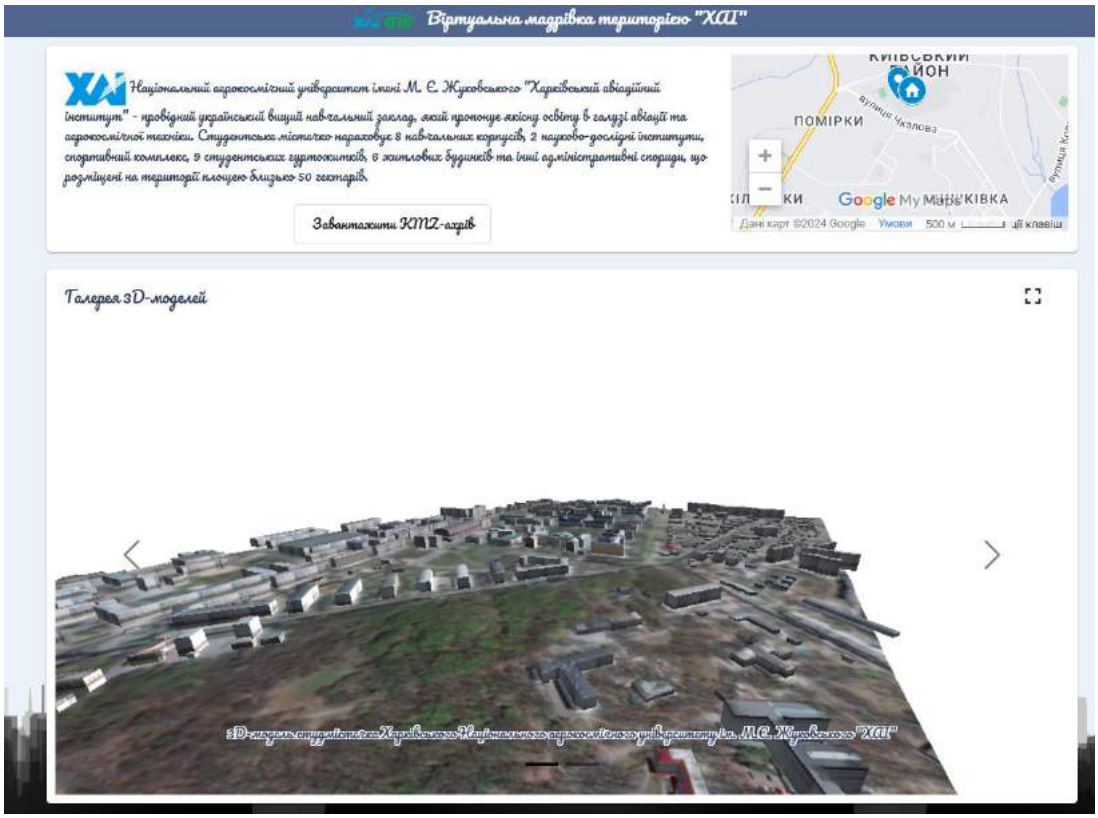


Рисунок 2.29 – Відображення сцени моделей з доданим сонячним освітленням

Наступним кроком було реалізовано метод addSun, що відповідає за додавання об'єкта сонця до тривимірної сцени. Для реалізації імітація об’єкта сонця було використано можливості бібліотеки Three.js, що дозволяють створювати та додавати нові тривимірні об’єкти на сцену. Спочатку було створено геометрію майбутнього сонця за допомогою конструктора конструктора THREE.SphereGeometry, що створює об’єкт сферичної форми з заданими параметрами. Параметри конструктора вказують на радіус сфери (у даному випадку 10), а також кількість сегментів по ширині та висоті (32, 32). Це визначає рівень деталізації сфери. Далі за допомогою THREE.MeshBasicMaterial було задано матеріал створеної сфери. THREE.MeshBasicMaterial є простим

матеріалом, що не використовує тінь чи відбиття, і має один кольоровий параметр. Далі було об’єднано створену геометрію та матеріал у єдиний меш – модель, та задано параметри позиції відображення отриманого мешу.

Лістинг 2.19 – Реалізація створення об’єкту сонця методом addSun

```
addSun() {
    const sunGeometry = new THREE.SphereGeometry(10, 32, 32); // Adjust radius
and segments as needed
    // Create a material for the sun (yellowish color)
    const sunMaterial = new THREE.MeshBasicMaterial({ color: 0xffff00 }); //
Adjust color as needed
    // Create a mesh using the geometry and material
    this.sun = new THREE.Mesh(sunGeometry, sunMaterial);
    this.sun.position.set(100, 120, 100); // Adjust position as needed
    // Add the sun to the scene
    this.scene.add(this.sun);
  }
```

На даному етапі створений меш сонця є статичним та не імітує сонячне освітлення. Для того щоб задати імітацію сонячного освітлення від нашого мешу було створено метод амімації.

Метод animate (лістинг 2.20) відповідає за анімацію сцени, зокрема рух сонця у круговій орбіті навколо сцени, використовуючи requestAnimationFrame для забезпечення анімації в циклі, оновлюючи кадри при максимальній частоті оновлення екрану браузера. В середині циклу було задано об'єкт THREE.Sphere, який використовується для обчислення Bounding Sphere сцени, що дозволяє отримати сферу, яка описує об'єм всієї сцени. Для розрахування позиції сонця було отримано радіус сфери навколо сцени та за допомогою тригонометричних функцій cos та sin було отримано константи позицій сонця. Далі, використовуючи отримані константи позицій сонця, було задано позиціювання для створеного джерела сонячного світла та мешу сонця. Потім через кутові можливості було задано параметр відображення світла, що дозволяє скривати сонце якщо воно опускається нижче моделі. Далі через функцію this.sunLight.lookAt було задано направлення сонячного світла, таким чином щоб воно завжди світило у центр сцени.

Лістинг 2.20 – Реалізації анімації сонячного освітлення сцени

```
animate(): void {
    requestAnimationFrame(() => this.animate());
    const boundingSphere = new THREE.Sphere();
    new
THREE.Box3().setFromObject(this.scene).getBoundingSphere(boundingSphere);

    const radius = boundingSphere.radius; // Set the radius of the circular path
    // Calculate new position based on the circular path
    const x = Math.cos(this.angle) * radius;
    const y = Math.sin(this.angle) * radius;

    this.sunLight.position.set(x, y, 0);
    this.sun.position.set(x, y, 0);

    if (y < 0 && y > -180) {
      this.angle = 0;
    } else {
      this.angle += 0.001;
    }
    this.sunLight.lookAt(0, 0, 0)
    this.controls.update();
    this.renderer.render(this.scene, this.camera);
  }
```

Рисунок 2.30 – Відображення сцени моделей зі створеним сонячним світлом від рухомого сонця

В результаті застосування всіх методів було утворено компоненту проєкту, яка реалістично візуалізує додані моделі у галереї. Повні лістинги компоненти three-model-view пердставлено у додатках Е-И.

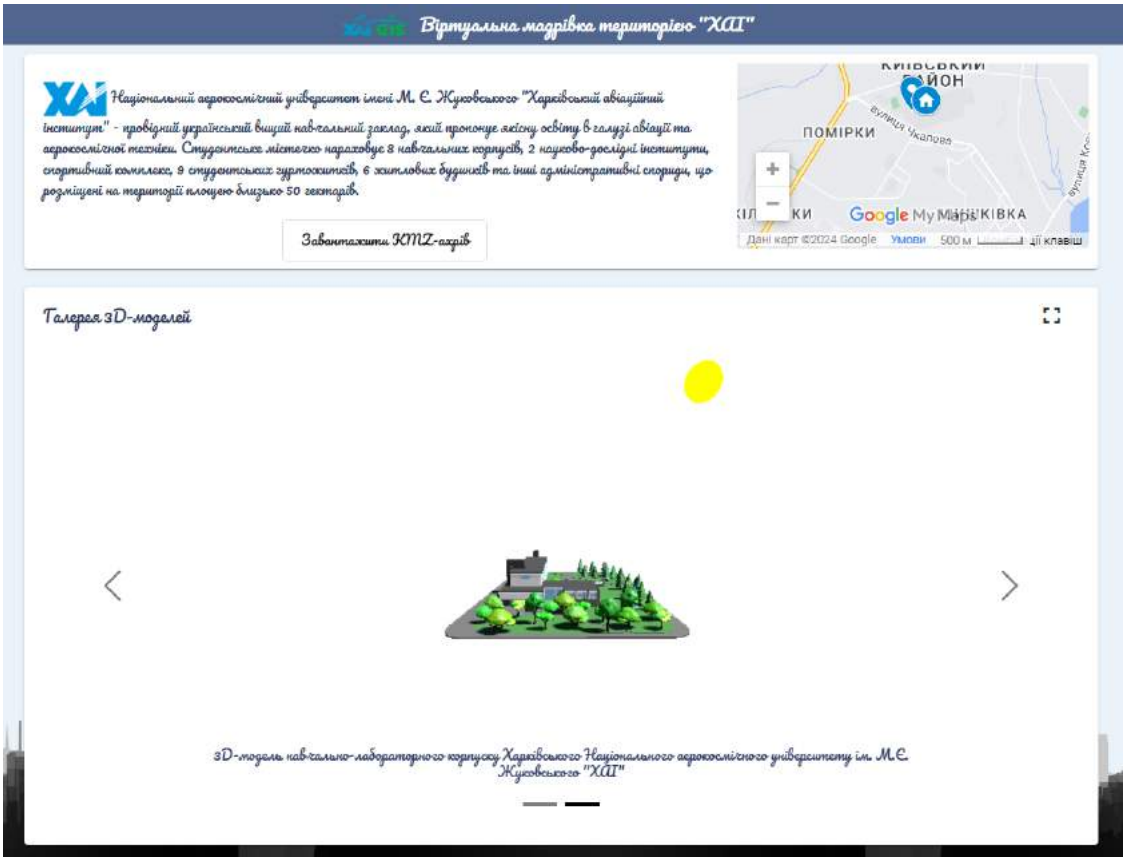


Рисунок 2.31 – Отриманий веб-додаток з інтегрованими 3D-моделями

ВИСНОВКИ

У результаті даної роботи було створено загальнодоступну методику створення тривимірної моделі забудови на інтерактивній картографічній основі в структурі WEB-додатку за допомогою бібліотеки Three.js.

В ході виконання кваліфікаційної роботи було проведено аналіз методів відображення тривимірної моделей у структурі веб-додатку. В якості технології вирішення поставлених задач було обрано бібліотеку Three.js. Для реалізації веб-додатку було обрано: мову розмітки HTML, мову стилів CSS, мову подій TypeScript, бібліотеку стилів Bootstrap.

Отримана методика вирішує проблеми візуалізації та аналізу об’єктів забудови на веб-платформах, використовуючи новітній підхід за допомогою об’єднання 3D-моделювання, ГІС та інтернет технологій .

Успішна реалізація методики демонструє, що використання бібліотеки Three.js є дієвим та інноваційним інструментом для відтворення 3D-графіки геоінформаційного характеру в браузері, дозволяючи створювати інтерактивні Web-ГІС різної направленості.

Отримана методика сприяє впровадженню інтерактивної візуалізації в геоінформаційну сферу, що забезпечує більш ефективне управління ресурсами та розвиток різних галузей економіки. Враховуючи динамічний розвиток цифрових технологій та попит на інтерактивні рішення, ця методика може знайти широке застосування в Україні та світі, сприяючи розвитку інновацій в геоінформаційних технологіях.

Розроблена методика надає зручний інструмент для віртуального моделювання та візуалізації міських просторів, що дозволяє вирішувати завдання з урбаністичного планування, визначення оптимального розміщення нових будівель та інфраструктури, а також аналізу впливу нових об'єктів на оточуючий середовища.

ПЕРЕЛІК ПОСИЛАНЬ

1. Технології розробки WEB-ресурсів [Текст] : навчальний посібник / В. П. Молчанов, О. К. Пандорін. – Харків : ХНЕУ ім. С. Кузнеця, 2019. – 130 с.

2. Що таке веб-програмування: особливості та огляд технологій [Електронний ресурс] / Режим доступу: https://vynnyky-visnyk.com.ua/2023/05/17/shcho-take-veb-prohramuvannia-osoblyvosti-ta-ohliad-tekhnolohii// – 23.10.2023

3. Як обрати архітектуру для веб додатку [Електронний ресурс] / Режим доступу: https://blog.ithillel.ua/articles/web-application-architecture – 24.10.2023

4. Фреймворки у мовах програмування: навіщо служать та як їх вибрати [Електронний ресурс] / Режим доступу: https://cloud.itstep.org/blog_3/frameworks-in-programming-languages-what-are-they-for-and-how-to-choose-them – 26.10.2023

5. HTML and CSS: Design and Build Websites [Текст] Author: Jon Duckett /J ohn Wiley & Sons; First Edition *ISBN: 978-1118008188 – 2011 - 490c

6. Learning TypeScript: Enhance Your Web Development Skills Using Type-Safe JavaScript 1st Edition [Текст] Author: Josh Goldberg, / O'Reilly Media; *ISBN: 978-1098110338 – 2022 – 318c

7. Bootstrap [Електронний ресурс] / Режим доступу: https://getbootstrap.com/– 26.10.2023

8. Angular [Електронний ресурс] / Режим доступу: https://angular.io/docs/ – 26.10.2023

9. Three.js Essentials [Текст] Author: Jos Dirksen / Packt Publishing, Birmingham, England, *ISBN: 978-1783980864 – 2014 – 198c

10. Getting to Know Web GIS, fourth edition [Текст] ESRI / Incorporated, *ISBN: 978-1589485921 – 2020 – 490 c

11. Towards Self-Service GIS–Combining the Best of the Semantic Web and Web GIS. MDPI. [Електронний ресурс] / Режим доступу: https://www.mdpi.com/2220-9964/9/12/753 – 05.11.2023

12. Notepad++ [Електронний ресурс] / Режим доступу: https://notepad-plus-plus.org/– 06.11.2023

13. Atom – a hackable text editor for the 21st Century [Електронний ресурс] / Режим доступу: https://atom-editor.cc/– 06.11.2023

14. Microsoft. Visual Studio Code - Code Editing. [Електронний ресурс] / Режим доступу: https://code.visualstudio.com/– 06.11.2023

15. 3D Modeling For Beginners: Learn everything you need to know about 3D Modeling! [Текст] Thilakanathan D./ CreateSpace Independent Publishing Platform *ISBN: 978-1530799626– 2016 – 240c

16. 3D GIS and 3D Modeling. SATPALDA : Satellite Imagery and Geospatial Services [Електронний ресурс] / Режим доступу: https://satpalda.com/blogs/3d-gis-and-3d-modeling/ – 06.11.2023

17. Базові поняття і терміни веб-технологій [Текст] – Кільченко А.В., Поповський О.І., Тебенко О.В. – Київ –ІІТЗН НАПН України – 2014 – 49 с.

18. Beginning Blender: open source 3D modeling, animation, and game design [Текст] Flavell L. / New York: Apress *ISBN: 9781430231264– 2010 – 420 с.

19. Blender-OSM: OpenStreetMap and terrain for Blender / Github [Електронний ресурс] – Режим доступу: https://github.com/vvoovv/blender-osm – 15.12.2023

20. Download Free 3D Models - Royalty Free. Sketchfab. [Електронний ресурс] – Режим доступу: https://sketchfab.com/features/free-3d-models?ref=footer– 20.12.2023

21. Розширення файлу .KMZ [Електронний ресурс] – Режим доступу: https://fileinfo.com/extension/kmz – 21.12.2023.

22. Що таке файл GLTF? [Електронний ресурс] – Режим доступу: https://docs.fileformat.com/uk/3d/gltf/ – 21.12.2023

23. Blender glTF 2.0 Importer and Exporter / Github [Електронний ресурс] – Режим доступу: https://github.com/KhronosGroup/glTF-Blender-IO

ДОДАТОК А – Лістинг HTML основи веб-додатку

```
<mat-toolbar color="primary" class="h-auto justify-content-center">
  <img src="https://www.khai-gis.info/static/blog/img/theme-3/logo-act.png"
class="logo">
  <span class="h4 p-2 mb-0">Віртуальна мадрівка територією "ХАІ"</span>
</mat-toolbar>
<section class="w-100 m-20 d-flex flex-column">
  <div class="d-flex hvh-25 m-3 w-100">
    <mat-card class="p-3 w-100 d-flex">
      <div class="w-100 d-flex h-100">
        <div class="col-8">
          <mat-card-header>
            <mat-card-title>
              <h5></h5>
            </mat-card-title>
          </mat-card-header>
          <div class="ms-3 me-3 general-info">

            <p style="color: rgb(46, 60, 102);"><img style="width: 55px; "

src='https://upload.wikimedia.org/wikipedia/commons/thumb/7/79/Logo_Luftfahrtins
titut_Charkiw.svg/1200px-Logo_Luftfahrtinstitut_Charkiw.svg.png'>
              Національний аерокосмічний університет імені М. Є. Жуковського
"Харківський авіаційний інститут" -
              провідний український вищий навчальний заклад, який пропонує
якісну освіту в галузі авіації та
              аерокосмічної
              техніки. Студентське містечко нараховує 8 навчальних корпусів, 2
науково-дослідні інститути, спортивний
              комплекс, 9 студентських
              гуртожитків, 6 житлових будинків та інші адміністративні споруди,
що розміщені на території площею близько
              50 гектарів.</p>

            <div class="text-center mt-3">
              <button (click)="onDownloadArchiveClick()" mat-stroked-
button>Завантажити KMZ-архів</button>
            </div>
          </div>
        </div>
        <div class="col-4 overflow-x-hidden">
          <iframe id="map" class="map"

src="https://www.google.com/maps/d/u/0/embed?mid=1i7mMVPbLOlpyiF8SCdiJ6kZT5d2gw6
ua&ehbc=2E312F"></iframe>

        </div>
      </div>
    </mat-card>
  </div>
  <div class="d-flex hvh-65 m-3 mt-0  w-100">
    <mat-card class="col-12 p-3 " [ngClass]="{'full-screen': isFullscreen}">
      <mat-card-header class="justify-content-between">
        <mat-card-title>
          <h5 style="color: rgb(46, 60, 102);"> Галерея 3D-моделей</h5>
        </mat-card-title>
        <button [disableRipple]="true" class="full-screen-icon" mat-icon-button
(click)="toggleFullscreen()">
          <mat-icon>{{ isFullscreen ? 'fullscreen_exit' : 'fullscreen' }}</mat-
icon>
        </button>
```

```
      </mat-card-header>

      <div class="progress" *ngIf="totalLoaded != files.length">
        <div class="progress-bar" role="progressbar"
[style.width.%]="getLoadedPercent()"
          [attr.aria-valuenow]="totalLoaded" [attr.aria-valuemin]="0"
[attr.aria-valuemax]="files.length"> rendering
          3d models</div>
      </div>
      <div id="3d-models-carousel" class="carousel carousel-dark slide w-100 h-
100"
        [ngClass]="{'invisible': totalLoaded != files.length}" data-bs-
ride="carousel">
        <div class="carousel-indicators">
          <ng-container *ngFor="let file of files; let i = index">
            <button type="button" data-bs-target="#carouselExampleCaptions"
[attr.data-slide-to]="i"
              [ngClass]="{'active': i == currentIndex }" aria-current="true"
aria-label="Slide 1"></button>

          </ng-container>
        </div>
        <div class="carousel-inner w-100 h-100">
          <ng-container *ngFor="let file of files; let i = index" class="w-100
h-100">
            <div class="carousel-item w-100 h-100" [class.active]="i ==
currentIndex || totalLoaded != files.length">
              <app-three-model-view #3dView [pathToModel]="'assets/3d-models/' +
file.name" (loaded)="onModelLoaded(i)"
                class="w-100 h-100"></app-three-model-view>
              <div class="carousel-caption d-none d-md-block">
                <h2>{{file.title}}</h2>
                <h4 class="text-white-shadow">{{file.subTitle}}</h4>
              </div>
            </div>
          </ng-container>
        </div>
        <button class="carousel-control-prev" type="button" data-bs-
target="#carouselExampleCaptions"
          (click)="prevSlide()" data-bs-slide="prev">
          <span class="carousel-control-prev-icon" aria-hidden="true"></span>
          <span class="visually-hidden">Previous</span>
        </button>
        <button class="carousel-control-next" type="button" data-bs-
target="#carouselExampleCaptions"
          (click)="nextSlide()" data-bs-slide="next">
          <span class="carousel-control-next-icon" aria-hidden="true"></span>
          <span class="visually-hidden">Next</span>
        </button>
      </div>
    </mat-card>
  </div>

</section>
```

ДОДАТОК Б – Лістинг CSS основи веб-додатку

```
mat-list-item:hover {
    background: rgb(209, 209, 209);
}

.hvh-65 {
    height: 65vh;
}

.full-screen {
    position: fixed;
    top: 0;
    left: 0;
    width: 96vw;
    margin: 2vw !important;
    height: 90vh;
    z-index: 1000;
    overflow: hidden;
    border-radius: 0;
    box-shadow: 0px 0px 0px 180px rgba(87, 87, 87, 0.631);
    .model-container {
        height: 100% !important;
        width: 100% !important;
    }
}

::ng-deep .full-screen {
    app-three-model-view {
        height: 100% !important;
        width: 100% !important;
        .model-container {
            height: 100% !important;
            width: 100% !important;
        }
    }
}

.model-container {
    min-height: 100% !important;
    min-width: 100& !important;
}

.m-20 {
    padding: 0 20vw;
}

.hvh-25 {
    height: 25vh !important;
}
.logo {
    width: 90px;
    height: 40px;
}
::ng-deep .mat-mdc-icon-button:hover .mat-mdc-button-persistent-ripple::before {
    opacity: 0 !important;
  }

  .text-white-shadow {
    color: rgb(46, 60, 102);
    text-shadow: -1px -1px 0 white,
    1px -1px 0 white,
```

```
    -1px 1px 0 white,
    1px 1px 0 white;
  }

  .mat-toolbar.mat-primary {
    background-color: rgb(80, 102, 143) !important;
  }

@media screen and (max-height: 1000px) {
    .general-info {
        font-size: 12px;
    }
    h4 {
        font-size: 12px;
    }
    h5 {
        font-size: 14px !important;
    }

    .h4 {
        font-size: 16px;
    }
    .logo {
        width: 60px;
        height: 20px;
    }

    .full-screen-icon {
        padding: 0px;
        height: 24px;
    }
    .mat-mdc-card-header {
        padding: 8px 8px 0px;
    }
    .full-screen-icon {
        margin-top: -8px;
    }
    .m-3 {
        margin: 8px !important;
    }
    .p-3 {
        padding: 8px !important;
    }
    .ms-3 {
        margin-left: 8px !important;
    }
    .ms-3 {
        margin-right: 8px !important;
    }
}

.map {
    width: 100% !important;
    height: calc(100% + 60px);
    margin-top: -70px;
}
```

ДОДАТОК В – Лістинг TS основи веб-додатку

```
import { CommonModule } from '@angular/common';
import { AfterViewInit, Component, ElementRef, QueryList, ViewChildren } from
'@angular/core';
import { MatButtonModule } from '@angular/material/button';
import { MatCardModule } from '@angular/material/card';
import { MatIconModule } from '@angular/material/icon';
import { MatToolbarModule } from '@angular/material/toolbar';
import { RouterOutlet } from '@angular/router';
import { ThreeModelViewComponent } from './three-model-view/three-model-
view.component';
import { saveAs } from 'file-saver';
@Component({
  selector: 'app-root',
  standalone: true,
  imports: [
    CommonModule,
    RouterOutlet,
    MatToolbarModule,
    MatIconModule,
    MatButtonModule,
    MatListModule,
    MatCardModule,
    ThreeModelViewComponent
  ],
  templateUrl: './app.component.html',
  styleUrl: './app.component.scss'
})
export class AppComponent implements AfterViewInit {
  @ViewChildren('3dView') elementsToObserve!:
QueryList<ThreeModelViewComponent>;
  isFullscreen: boolean = false;
  showCard: boolean = true;
  files = [{
    name: 'KHAI.glb',
    title: '',
    subTitle: '3D-модель студмістечка Харківського Національного аерокосмічного
університету ім. М.Є. Жуковського "ХАІ"'
  }, {
    name: 'ULK_KHAI.glb',
    title: '',
    subTitle: '3D-модель навчально-лабораторного корпуску Харківського
Національного аерокосмічного університету ім. М.Є. Жуковського "ХАІ"'
  }]
  currentIndex = 0;
  totalLoaded = 0;
  loadedFiles: number[] = []
  constructor() {

  }

  ngAfterViewInit(): void {
    const carousel = document.getElementById('3d-models-carousel');
    if (carousel) {
      carousel.addEventListener('slid.bs.carousel', (event: any) => {
        this.currentIndex = event.to;
      });
    }
  }
```

```
  onModelLoaded(index: number) {
    if (!this.loadedFiles.includes(index)) {
      this.totalLoaded++;
      this.loadedFiles.push(index);
    }
  }

  nextSlide() {
    this.currentIndex = (this.currentIndex + 1) % this.files.length;
  }

  prevSlide() {
    this.currentIndex = this.currentIndex > 0 ? this.currentIndex - 1 :
this.files.length - 1;
  }

  toggleFullscreen(): void {
    this.showCard = false; // Hide the card momentarily to expand it
    this.isFullscreen = !this.isFullscreen;
    setTimeout(() => {
      this.showCard = true;
    })
  }

  getLoadedPercent(): number {
    return this.totalLoaded / this.files.length * 100;
  }

  onDownloadArchiveClick() {
    fetch('assets/archives/KHAI.kmz')
      .then((response) => response.blob())
      .then((blob) => {
        saveAs(blob, 'KHAI.kmz');
      })
      .catch((error) => {
        console.error('Error fetching the file:', error);
      });
  }
}



export interface Tile {
  color: string;
  cols: number;
  rows: number;
  text: string;
}
```

ДОДАТОК Г – Лістинг TS конфігурацій основи веб-додатку

```ts
import { ApplicationConfig } from '@angular/core';
import { provideRouter } from '@angular/router';

import { routes } from './app.routes';
import { provideAnimations } from '@angular/platform-browser/animations';

export const appConfig: ApplicationConfig = {
  providers: [provideRouter(routes), provideAnimations()]
};
```

ДОДАТОК Д – Лістинг JSON файлу конфігурації Angular проєкту

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "ania-diplom": {
      "projectType": "application",
      "schematics": {
        "@schematics/angular:component": {
          "style": "scss"
        }
      },
      "root": "",
      "sourceRoot": "src",
      "prefix": "app",
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:application",
          "options": {
            "outputPath": "dist/diplom",
            "index": "src/index.html",
            "browser": "src/main.ts",
            "polyfills": [
              "zone.js"
            ],
            "tsConfig": "tsconfig.app.json",
            "inlineStyleLanguage": "scss",
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [
              "node_modules/bootstrap/scss/bootstrap.scss",
              "src/styles.scss"
            ],
            "scripts": [
              "node_modules/bootstrap/dist/js/bootstrap.bundle.min.js",
              "node_modules/three/build/three.min.js"
            ]
          },
          "configurations": {
            "production": {
              "budgets": [
                {
                  "type": "initial",
                  "maximumWarning": "500kb",
                  "maximumError": "1mb"
                },
                {
                  "type": "anyComponentStyle",
                  "maximumWarning": "2kb",
                  "maximumError": "4kb"
                }
              ],
              "outputHashing": "all"
            },
            "development": {
              "optimization": false,
              "extractLicenses": false,
              "sourceMap": true
```

```
            }
          },
          "defaultConfiguration": "production"
        },
        "serve": {
          "builder": "@angular-devkit/build-angular:dev-server",
          "configurations": {
            "production": {
              "buildTarget": "diplom:build:production"
            },
            "development": {
              "buildTarget": "diplom:build:development"
            }
          },
          "defaultConfiguration": "development"
        },
        "extract-i18n": {
          "builder": "@angular-devkit/build-angular:extract-i18n",
          "options": {
            "buildTarget": "ania-diplom:build"
          }
        },
        "test": {
          "builder": "@angular-devkit/build-angular:karma",
          "options": {
            "polyfills": [
              "zone.js",
              "zone.js/testing"
            ],
            "tsConfig": "tsconfig.spec.json",
            "inlineStyleLanguage": "scss",
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [
              "src/styles.scss"
            ],
            "scripts": []
          }
        }
      }
    }
  },
  "cli": {
    "analytics": false
  }
}
```

ДОДАТОК Е – Лістинг HTML компоненти відображення 3D-моделей

```
<div class='model-container w-100 h-100' #rendererContainer>
</div>
```

ДОДАТОК Ж – Лістинг CSS компоненти відображення 3D-моделей

```
.model-container {
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
}
```

ДОДАТОК И – Лістинг TS компоненти відображення 3D-моделей

```
import { AfterViewInit, ChangeDetectorRef, Component, ElementRef, EventEmitter,
Input, OnDestroy, Output, Renderer2, ViewChild } from '@angular/core';
import * as THREE from 'three';
import { GLTFLoader } from 'three/examples/jsm/loaders/GLTFLoader.js';
import { OrbitControls } from 'three/examples/jsm/controls/OrbitControls.js';

@Component({
  selector: 'app-three-model-view',
  standalone: true,
  imports: [],
  templateUrl: './three-model-view.component.html',
  styleUrl: './three-model-view.component.scss'
})
export class ThreeModelViewComponent implements AfterViewInit, OnDestroy {
  @ViewChild('rendererContainer') rendererContainer!: ElementRef;
  @Input() pathToModel = 'assets/3d-models/KHAI.glb'
  @Output() loaded = new EventEmitter<boolean>();
  scene = new THREE.Scene();
  camera!: THREE.PerspectiveCamera;
  renderer = new THREE.WebGLRenderer();
  loader = new GLTFLoader();
  controls!: OrbitControls;
  sunLight!: THREE.DirectionalLight
  sun!: THREE.Mesh
  angle = 0
  sunHelper!: THREE.DirectionalLightHelper
  rendering = true;
  isCameraPositionApplied = false;

  constructor() { }

  ngAfterViewInit(): void {
    this.render();
    this.detectContainerSizeChange();
  }

  render() {
    this.setSceneProperties();
    this.addRenderer();
    this.addCamera();
    this.addGlobalLight();
    this.addSun();
    this.addSunLight();
    this.addOrbitControls();
    this.loadModel()
  }

  detectContainerSizeChange() {
    const observer = new ResizeObserver(entries => {
      entries.forEach(entry => {
        this.renderer.setSize(this.rendererContainer.nativeElement.clientWidth,
this.rendererContainer.nativeElement.clientHeight);
      });
    });
    observer.observe(this.rendererContainer.nativeElement);
  }

  setSceneProperties() {
    this.scene.background = new THREE.Color(0xffffff);
  }
```

```
  addRenderer() {
    this.renderer.setSize(this.rendererContainer.nativeElement.clientWidth,
this.rendererContainer.nativeElement.clientHeight);
    this.renderer.shadowMap.enabled = true;
    this.rendererContainer.nativeElement.appendChild(this.renderer.domElement);
  }

  addCamera() {
    this.camera = new THREE.PerspectiveCamera(75,
this.rendererContainer.nativeElement.clientWidth /
this.rendererContainer.nativeElement.clientHeight, 0.2, 2000);
  }

  addGlobalLight() {
    const hemiLight = new THREE.HemisphereLight(0x0000ff, 0x00ff00, 0.6);
    hemiLight.color.setHSL(0.6, 1, 0.6);
    hemiLight.groundColor.setHSL(0.095, 1, 0.75);
    hemiLight.position.set(0, 50, 0);
    hemiLight.castShadow = true;
    this.scene.add(hemiLight)
  }

  addSun() {
    const sunGeometry = new THREE.SphereGeometry(10, 32, 32); // Adjust radius
and segments as needed
    // Create a material for the sun (yellowish color)
    const sunMaterial = new THREE.MeshBasicMaterial({ color: 0xffff00 }); //
Adjust color as needed
    // Create a mesh using the geometry and material
    this.sun = new THREE.Mesh(sunGeometry, sunMaterial);
    this.sun.position.set(100, 120, 100); // Adjust position as needed
    // Add the sun to the scene
    this.scene.add(this.sun);
  }

  addSunLight() {
    this.sunLight = new THREE.DirectionalLight('#f6e12473', 10); // Color,
Intensity
    this.sunLight.position.set(100, 120, 100); // Set position (represents sun
direction)
    this.sunLight.castShadow = true; // Enable shadow casting by the sun
    this.sunLight.shadow.mapSize.width = 8096;
    this.sunLight.shadow.mapSize.height = 8096;
    // Enable soft shadows
    this.sunLight.shadow.radius = 10;
    this.sunLight.shadow.camera.top = 270;
    this.sunLight.shadow.camera.bottom = -170;
    this.sunLight.shadow.camera.left = -200;
    this.sunLight.shadow.camera.right = 200;
    this.sunLight.shadow.camera.near = 25;
    this.sunLight.shadow.camera.far = 350;
    this.sunLight.shadow.bias = -0.005;
    this.scene.add(this.sunLight)
  }

  addOrbitControls() {
    this.controls = new OrbitControls(this.camera, this.renderer.domElement);
  }

  loadModel() {
    this.loader.load(this.pathToModel, (gltf) => {
      if (!this.isCameraPositionApplied) {
        this.dinamicCameraPositionOnLoad(gltf);
```

```
      }
      this.traverseModel(gltf.scene)
      this.scene.add(gltf.scene);
      this.animate();
    });

    this.scene.onAfterRender = () => {
      this.rendering = false;
      this.loaded.emit(true)
    };
  }

  dinamicCameraPositionOnLoad(gltf: any) {
    const boundingBox = new THREE.Box3().setFromObject(gltf.scene);
    // Get the dimensions (width, height, depth) of the bounding box
    const width = (boundingBox.max.x - boundingBox.min.x) / 3;
    const height = (boundingBox.max.y - boundingBox.min.y) + (boundingBox.max.y
- boundingBox.min.y) * 0.2;
    const depth = (boundingBox.max.z - boundingBox.min.z) / 1.5
    this.camera.position.set(width, height, depth)
    this.isCameraPositionApplied = true;
  }

  traverseModel(node: THREE.Object3D): void {
    const visitedNodes = new Set<THREE.Object3D>();

    const traverse = (child: THREE.Object3D) => {
      if (visitedNodes.has(child)) {
        return;
      }

      visitedNodes.add(child);
      // Ensure the child is a Mesh to access Mesh-specific properties
      if (child instanceof THREE.Mesh) {
        child.castShadow = true;
        child.receiveShadow = true;
      }

      child.children.forEach(traverse)
    };

    traverse(node);
  }

  animate(): void {
    requestAnimationFrame(() => this.animate());
    const boundingSphere = new THREE.Sphere();
    new
THREE.Box3().setFromObject(this.scene).getBoundingSphere(boundingSphere);

    const radius = boundingSphere.radius; // Set the radius of the circular path
    // Calculate new position based on the circular path
    const x = Math.cos(this.angle) * radius;
    const y = Math.sin(this.angle) * radius;

    this.sunLight.position.set(x, y, 0);
    this.sun.position.set(x, y, 0);

    if (y < 0 && y > -180) {
      this.angle = 0;
    } else {
      this.angle += 0.001;
    }
    this.sunLight.lookAt(0, 0, 0)
```

```
    this.controls.update();
    this.renderer.render(this.scene, this.camera);
  }

  ngOnDestroy(): void {
    this.dispose()
  }

  public dispose() {
    this.renderer.dispose()
  }
}
```

ДОДАТОК К – Плакат за темою кваліфікаційної роботи

МЕТОДИКА СТВОРЕННЯ 3D МОДЕЛІ БУДІВЛІ НА ІНТЕРАКТИВНІЙ КАРТОГРАФІЧНІЙ ОСНОВІ В СТРУКТУРІ WEB-ДОДАТКУ ЗА ДОПОМОГОЮ БІБЛІОТЕКИ THREE.JS

Спеціальність: 193 - Геодезія та землеустрій
Освітня програма: Геоінформаційні системи і технології

Виконавець:
Студентка гр. №462м, Крапива А.А.

Керівник:
к.т.н., доцент каф. 407, Нечаусов А.С.

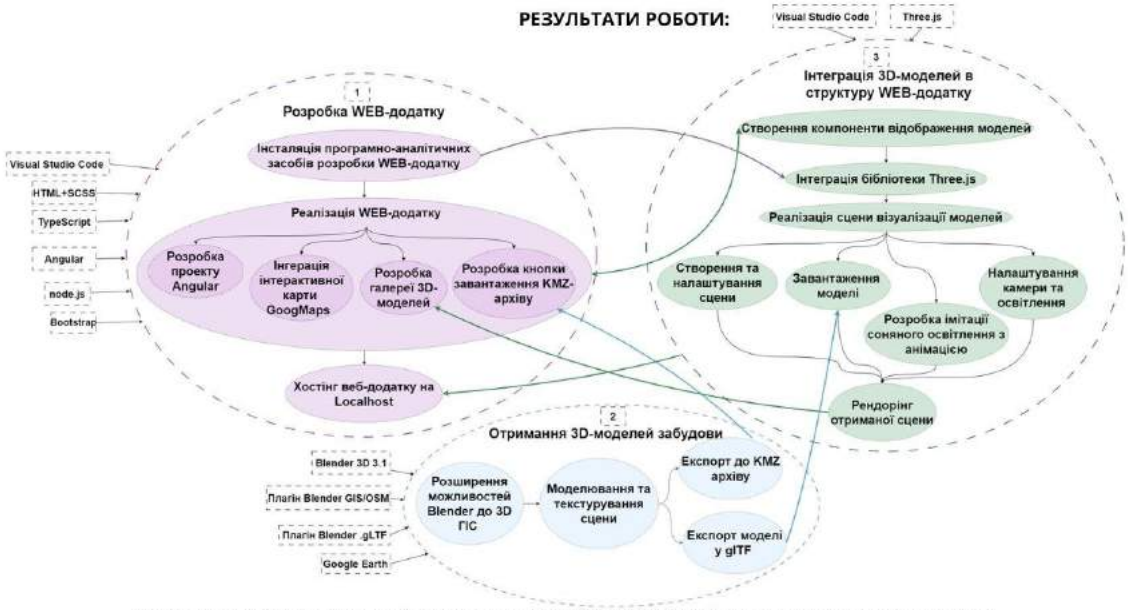
Мета роботи - підвищення ефективності процесу створення інтерактивних веб-додатків для демонстрації тривимірних сцен із картографічним змістом, зокрема місцевості та забудови за рахунок поєднання сучасних засобів тривимірного моделювання, засобів картографування, обробки геоінформаційних даних та сучасних засобів веб-розробки у єдину універсальну методику.

Об'єкт дослідження - процеси створення та відтворення 3D моделей та сцен картографічної спрямованості на базі інтерактивних веб-додатків.

Предмет дослідження - методи створення 3D моделей будівель на інтерактивній картографічній основі в структурі WEB-додатку за допомогою бібліотеки Three.js.

Окремі завдання для досягнення поставленої мети: 1)Аналіз програмно-аналітичних засобів відображення 3D-моделей у веб-додатках; 2) Аналіз програмних продуктів для веб-розробки; 3) Розробка веб-додатку; 4) моделювання тривимірного зображення на картографічній основі; 5)Моделювання сцени відображення тривимірного зображення в структурі веб-додатку.

РЕЗУЛЬТАТИ РОБОТИ:



Методика створення 3D моделі будівлі на інтерактивній картографічній основі в структурі WEB-додатку за допомогою бібліотеки Three.js

Отриманий веб-додаток з інтегрованою 3D-моделлю студмістечка "ХАІ"

Отриманий веб-додаток з інтегрованою 3D-моделлю НЛК "ХАІ"

ДОДАТОК Л – Презентація за темою кваліфікаційної роботи

Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»
Факультет ракетно-космічної техніки
Кафедра геоінформаційних технологій та космічного моніторингу Землі

кваліфікаційна робота магістра

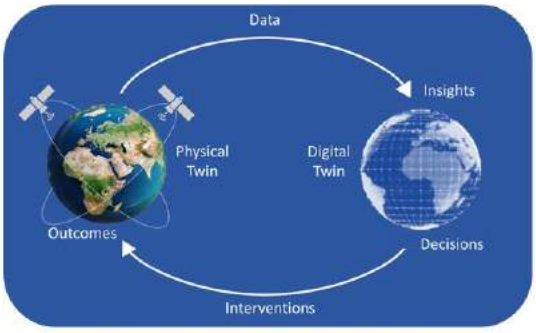
спеціальність 193 «Геодезія та землеустрій»
освітня програма «Геоінформаційні системи та технології»

Методика створення 3D моделі будівлі на інтерактивній картографічній основі в структурі WEB-додатку за допомогою бібліотеки Three.js

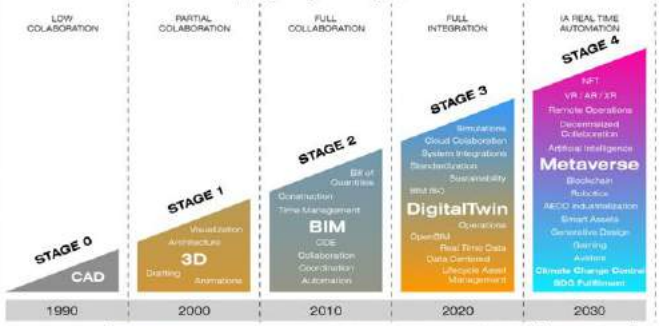
Виконала: студентка групи 462-м
Крапива А. А.
Керівник: к.т.н., доцент каф. 407
Нечаусов А. С.

Харків - 2024

Актуальність роботи



Діаграма розвитку ЦММ



Цифровий двійник Землі є основною ініціативою Європейської Комісії, запропонований програмою DestinE (Destination Earth), що має об'єднати дані спостереження Землі та наземні вимірювання зі складним соціальним, економічним і екологічним моделюванням, для розуміння стійкість нашої планети та покращення всіх систем управління.

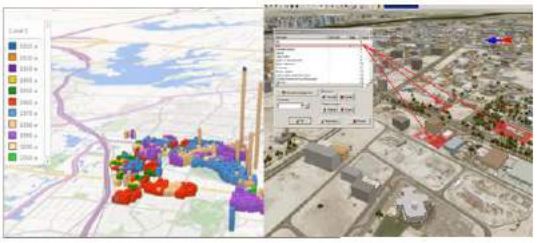
Моделювання + Інтернет технології = Цифрові двійники

Переваги використання тривимірних WEB-ГІС

Можливість реалізації та застосування геоданих у реальному часі

Покращення візуалізації , що полегшує аналіз геоданих та прийняття рішень

Наочність ситуаційної та аналітичної інформації



Глобальна доступність тривимірних геоданих

3

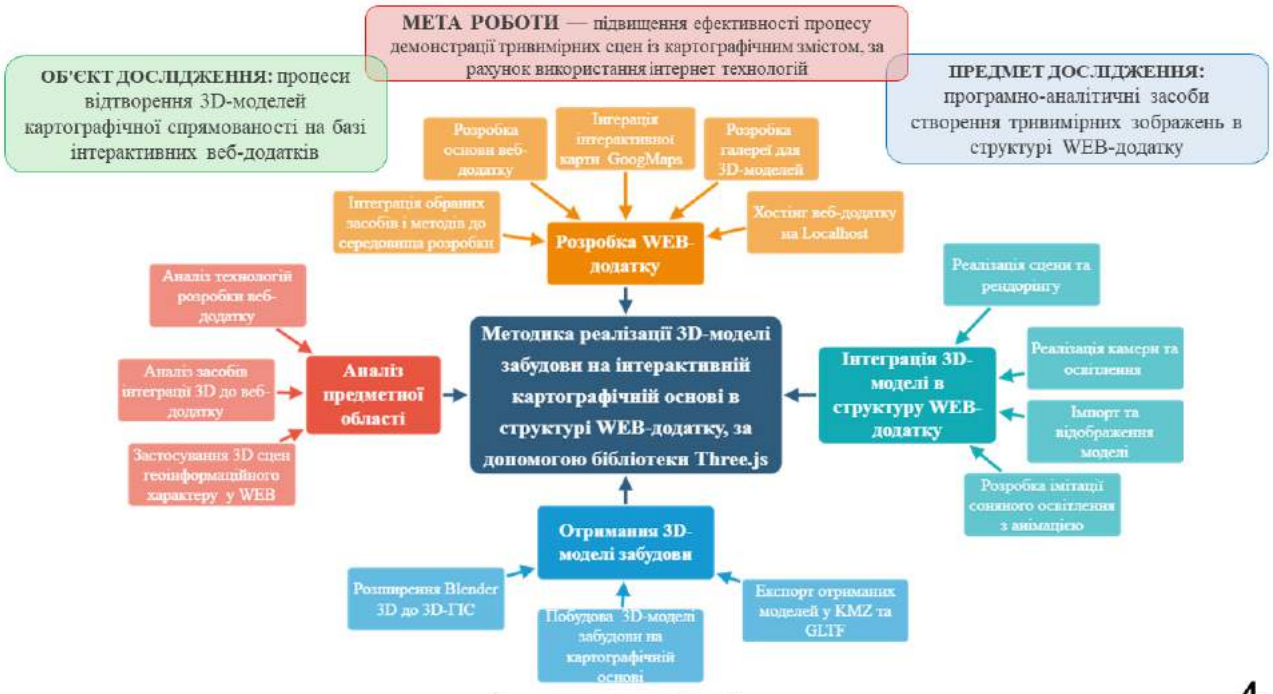


Схема основних етапів роботи

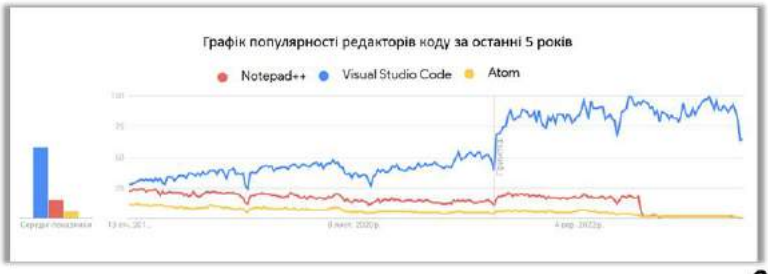
4

Програмно-технологічні засоби створення 3D-графіки в структурі веб-додатку

5

Аналіз редакторів коду для WEB-розробки

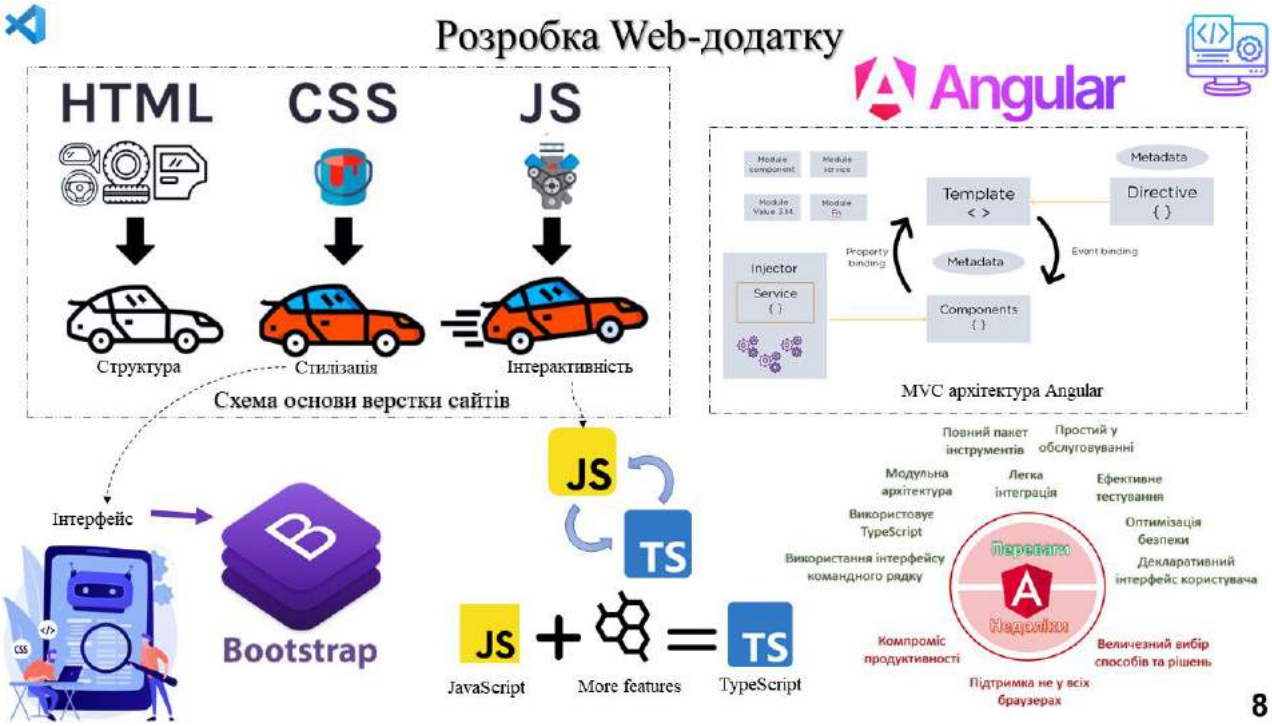
Програми / Критерії	Notepad++	Atom	Visual Studio Code
Загальнодоступність	⊕	⊕	⊕
Кросплатформність	⊖	⊕	⊕
Помірне використання ресурсів	⊕	⊕	⊕
Можливість розширення функціональності	±⊖	±⊖	⊕
Багатомовна підтримка	⊖	±⊖	⊕
Інтелектуальне завершення коду (IntelliSense)	⊕	⊕	⊕
Підтримка налагодження	⊖	±⊖	⊕
Автоматизація завдань	⊖	±⊖	⊕
Простота та зручність використання	⊕	±⊖	⊕
Співпраця онлайн (Live Share)	⊖	⊖	⊕
Керування версіями (Git)	⊖	±⊖	⊕



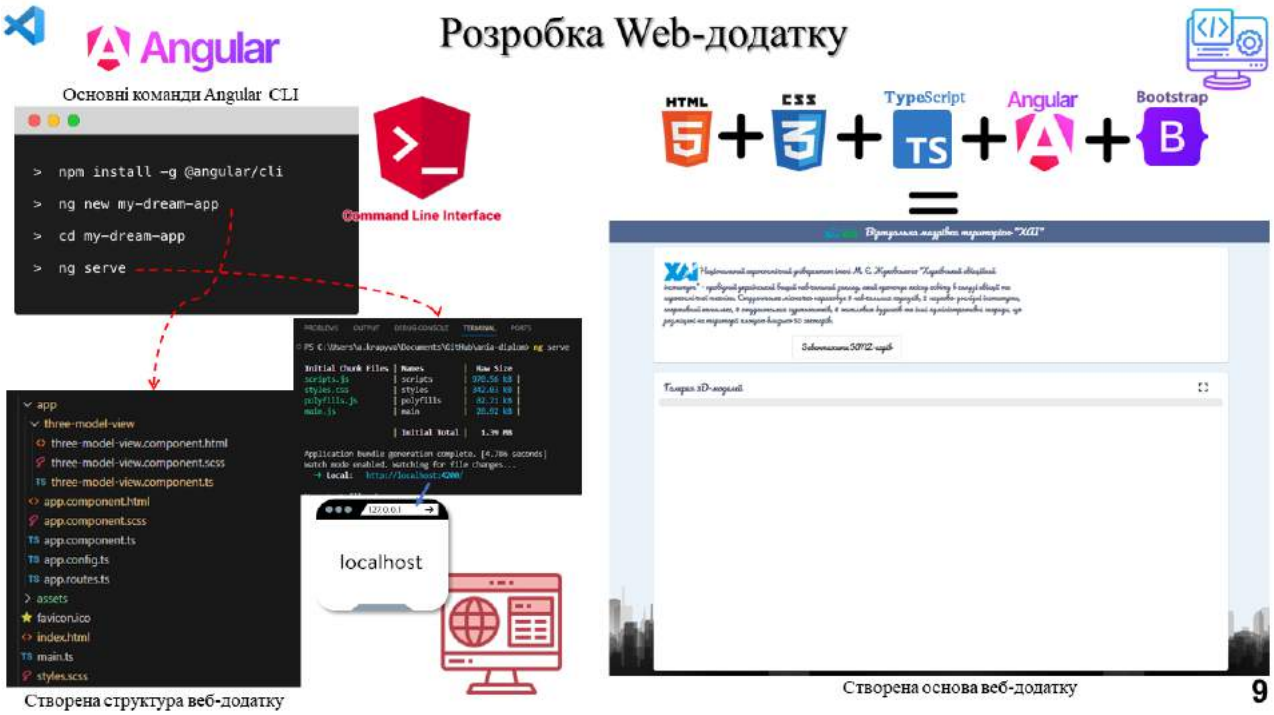
6

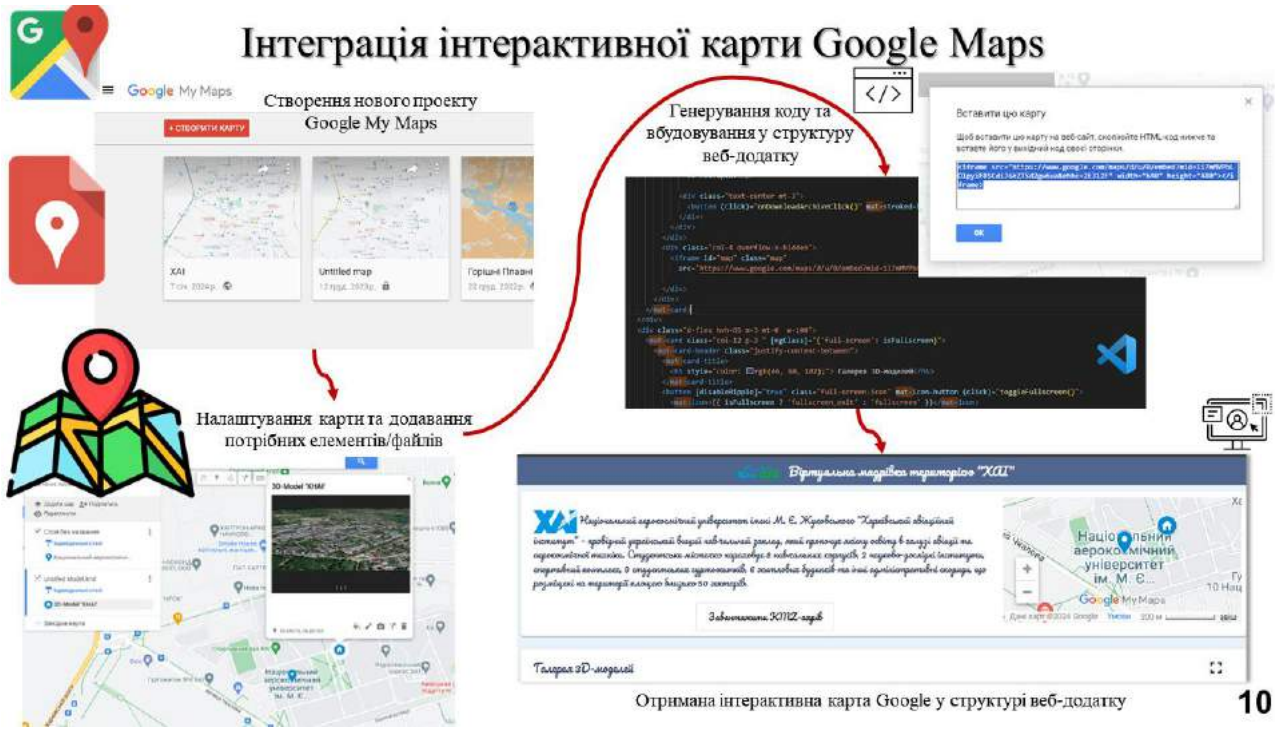

СТРУКТУРНА СХЕМА МЕТОДИКИ СТВОРЕННЯ 3D-МОДЕЛІ БУДІВЛІ НА ІНТЕРАКТИВНІЙ КАРТОГРАФІЧНІЙ ОСНОВІ В СТРУКТУРІ WEB-ДОДАТКУ
Visual Studio Code
Three.js
1
Розробка WEB-додатку
Інсталяція програмно-аналітичних засобів розробки WEB-додатку
Visual Studio Code
HTML+SCSS
TypeScript
Angular
node.js
Bootstrap
Реалізація WEB-додатку
Розробка проекту Angular
Інтеграція інтерактивної карти GoogMaps
Розробка галереї 3D-моделей
Розробка кнопки завантаження KMZ-архіву
Хостінг веб-додатку на Localhost
3
Інтеграція 3D-моделей в структуру WEB-додатку
Створення компоненти відображення моделей
Інтеграція бібліотеки Three.js
Реалізація сцени візуалізації моделей
Створення та налаштування сцени
Завантаження моделі
Налаштування камери та освітлення
Розробка імітації сонячного освітлення з анімацією
Рендорінг отриманої сцени
2
Отримання 3D-моделей забудови
Blender 3D 3.1
Плагін Blender GIS/OSM
Плагін Blender .gLTF
Google Earth
Розширення можливостей Blender до 3D ГІС
Моделювання та текстурування сцени
Експорт до KMZ архіву
Експорт моделі у glTF

7


Розробка Web-додатку
HTML
CSS
JS
Структура
Стилізація
Інтерактивність
Схема основи верстки сайтів
Angular
Module component
Module service
Module Value 3.14
Module Fn
Metadata
Template < >
Directive { }
Injector
Service { }
Property binding
Metadata
Event binding
Components { }
MVC архітектура Angular
Інтерфейс
Bootstrap
JS
TS
JS
JavaScript
More features
TS
TypeScript
Повний пакет інструментів
Простий у обслуговуванні
Модульна архітектура
Легка інтеграція
Ефективне тестування
Використовує TypeScript
Оптимізація безпеки
Використання інтерфейсу командного рядка
Переваги
Недоліки
Декларативний інтерфейс користувача
Компроміс продуктивності
Величезний вибір способів та рішень
Підтримка не у всіх браузерах

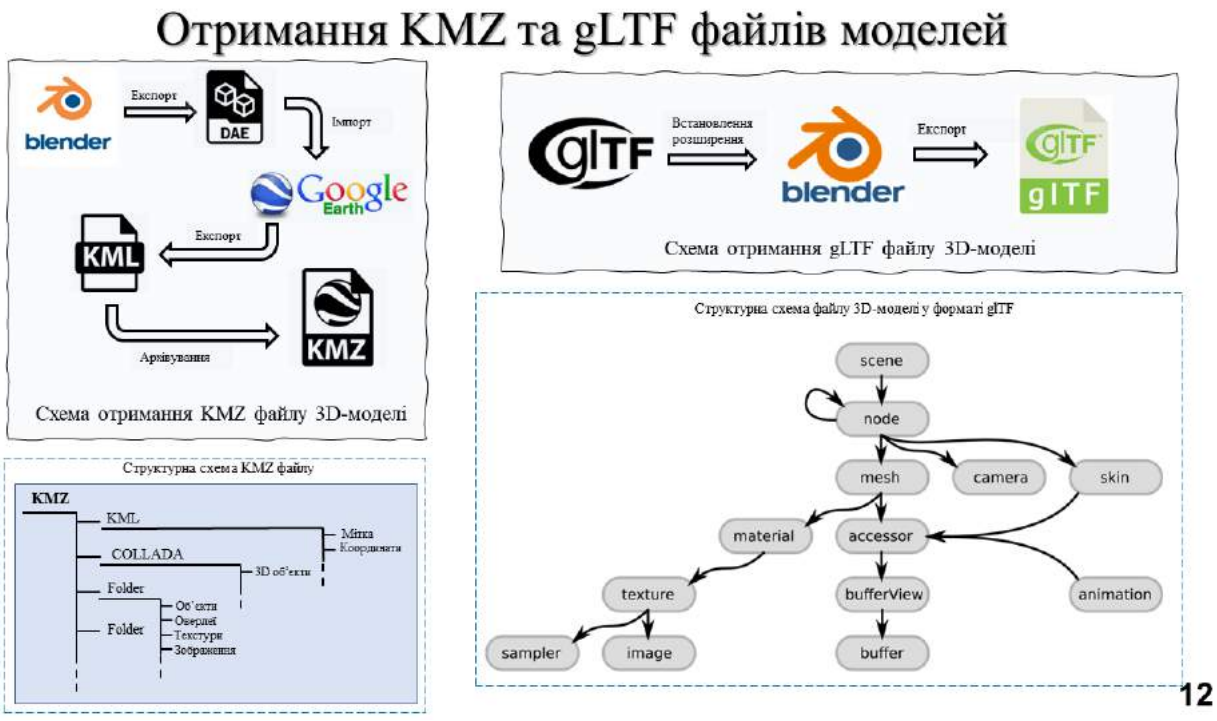
8


Розробка Web-додатку
Angular
Основні команди Angular CLI
> npm install -g @angular/cli
> ng new my-dream-app
> cd my-dream-app
> ng serve
Command Line Interface
HTML
CSS
TypeScript
Angular
Bootstrap
app
three-model-view
three-model-view.component.html
three-model-view.component.scss
three-model-view.component.ts
app.component.html
app.component.scss
app.component.ts
app.config.ts
app.routes.ts
assets
favicon.ico
index.html
main.ts
styles.scss
localhost
Створена структура веб-додатку
Створена основа веб-додатку
9


Інтеграція інтерактивної карти Google Maps
Google My Maps
Створення нового проекту Google My Maps
Генерування коду та вбудовування у структуру веб-додатку
Вставити цю карту
Налаштування карти та додавання потрібних елементів/файлів
Отримана інтерактивна карта Google у структурі веб-додатку
10

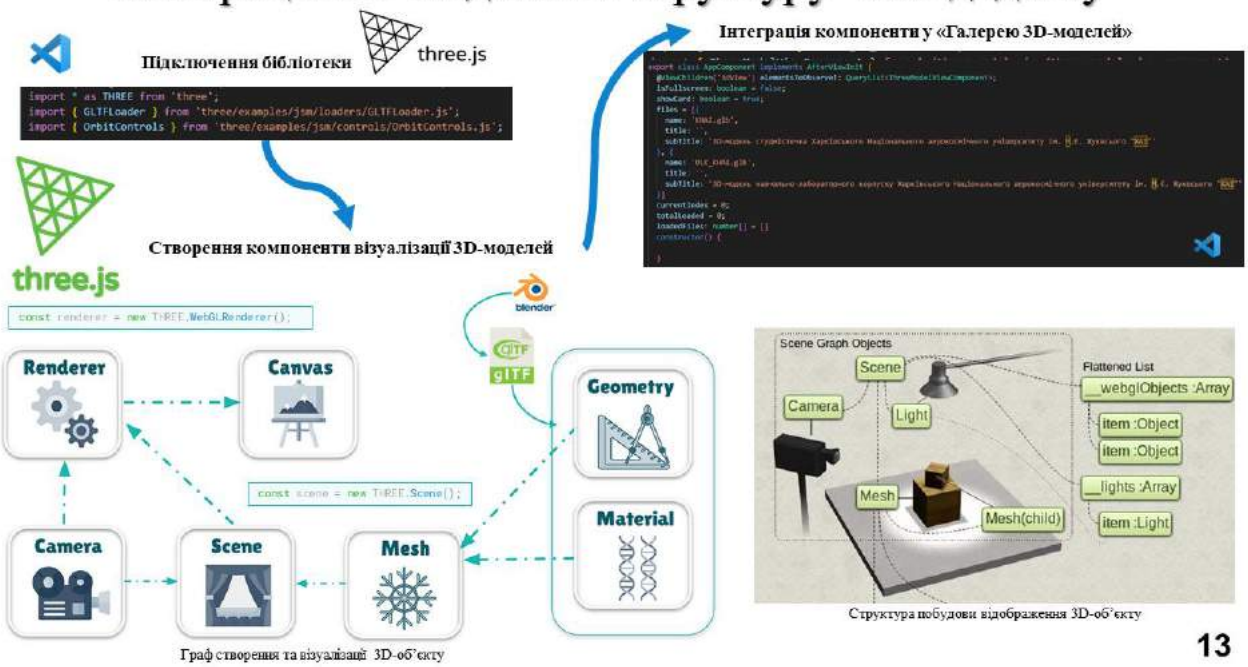

Отримання 3D-моделей забудови
blender
GIS
Plugins
3D-GIS
2D карта рельєфу
3D карта рельєфу
2D карта міста
3D карта міста
OpenStreetMap
Модернізація та текстурування
Отримана модель студмістечка «ХАІ»
Відокремлення та деталізація
Отримана модель навчально-лабораторного корпусу ХАІ

11


Отримання KMZ та gLTF файлів моделей
blender
Експорт
DAE
Імпорт
Google Earth
KML
Експорт
Архівування
KMZ
Схема отримання KMZ файлу 3D-моделі
Структурна схема KMZ файлу
KMZ
KML
Мітка
Координати
COLLADA
3D об'єкти
Folder
Об'єкти
Folder
Оверлеї
Текстури
Зображення
gITF
Встановлення розширення
blender
Експорт
gITF
Схема отримання gLTF файлу 3D-моделі
Структурна схема файлу 3D-моделі у форматі gITF
scene
node
mesh
camera
skin
material
accessor
texture
bufferView
animation
sampler
image
buffer

12

Інтеграція 3D-моделей в структуру Web-додатку

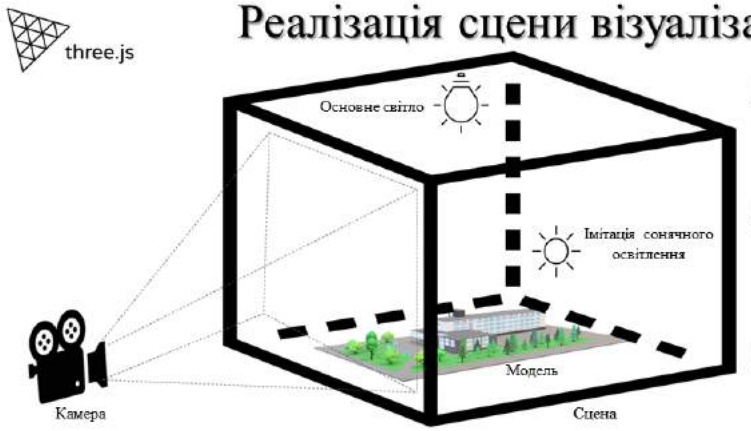


Граф створення та візуалізації 3D-об'єкту

Структура побудови відображення 3D-об'єкту

13

Реалізація сцени візуалізації моделей

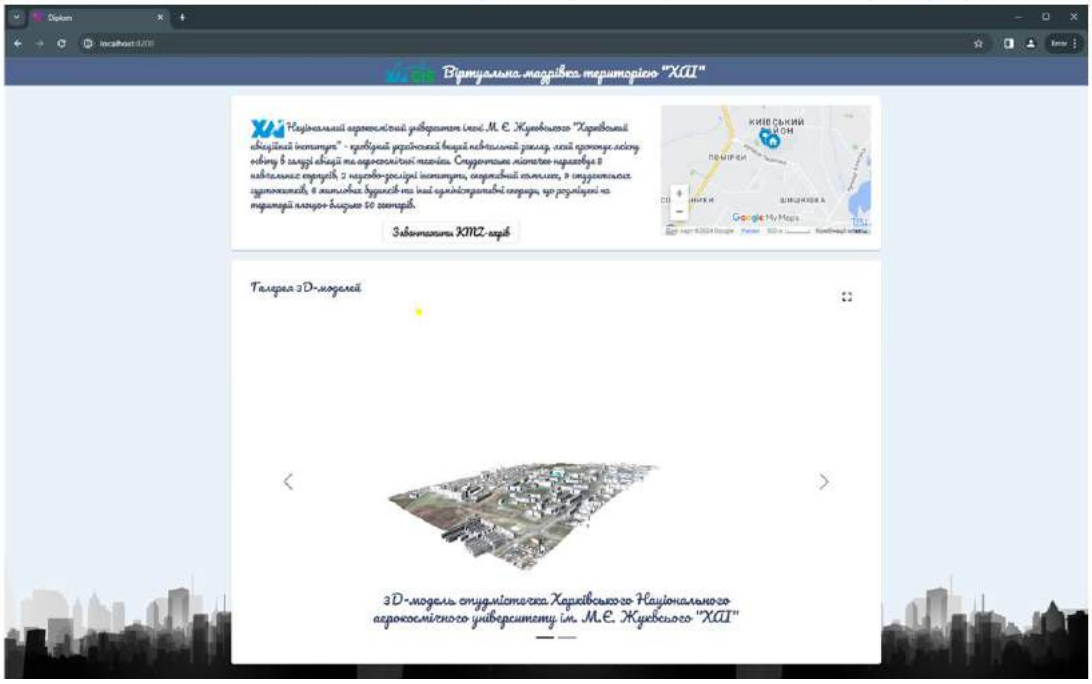


1. Створення сцени, додавання камери та моделі
2. Налаштування основного освітлення та інтерактивного контролю
3. Створення імітації сонячного освітлення з анімацією

Відображення моделі на різних етапах створення візуалізації

14

Розроблений web-додаток для демонстрації тривимірних сцен із картографічним змістом



15

Результати та висновки:

1) Отримана методика вирішує проблеми візуалізації та аналізу об’єктів забудови на веб-платформах, використовуючи новітній підхід за допомогою об’єднання 3D-моделювання, ГІС та інтернет технологій .

2) Успішна реалізація методики демонструє, що використання бібліотеки Three.js є дієвим та інноваційним інструментом для відтворення 3D-графіки геоінформаційного характеру в браузері, дозволяючи створювати інтерактивні Web-ГІС різної направленості.

3) Отримана методика сприяє впровадженню інтерактивної візуалізації в геоінформаційну сферу, що забезпечує більш ефективне управління ресурсами та розвиток різних галузей економіки. Враховуючи динамічний розвиток цифрових технологій та попит на інтерактивні рішення, ця методика може знайти широке застосування в Україні та світі, сприяючи розвитку інновацій в геоінформаційних технологіях.

4) Розроблена методика надає зручний інструмент для віртуального моделювання та візуалізації міських просторів, що дозволяє вирішувати завдання з урбаністичного планування, визначення оптимального розміщення нових будівель та інфраструктури, а також аналізу впливу нових об'єктів на оточуючий середовища.

16

Апробації

XIX Науково-технічна конференція факультету ракетно-космічної техніки «Сучасні проблеми ракетно-космічної техніки і технології» - Моделювання наслідків можливої техногенної аварії на прикладі Кременчуцького водосховища з використанням геоінформаційних технологій - Харків, 2023р. - с.59

14-а Міжнародна студентська науково-технічна конференція "ПерСиК 2023" «Перспективні мережі та комп'ютерні технології» - Використання TIN-моделі рельєфу для прогнозування наслідків прориву дамби на Кременчуцькому водосховищі - Харків, 2023р. – с.46