МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Бакалаврська кваліфікаційна робота

на тему: Розробка програмної системи для створення 3D турів

Виконав студент 4 курсу групи К-25
Спеціальність 122 комп'ютерні науки
Дерменжі Микита Анатолійович

Керівник к.геогр.н., доцент
Кузніченко Світлана Дмитрівна

Консультант _____
_____

Рецензент к.т.н., доцент
Гнатовська Ганна Арнольдівна

Одеса 2020

ЗМІСТ

ВСТУП

Під час розробки сучасних веб-сервісів та додатків у розробників програмного забезпечення частіше виникає необхідність створювати панорамні тури. Вони можуть бути пов’язані через маркетингові цілі, такі як залучення нових партнерів або для користувачів щодо огляду певних якостей території. І найважливіші моменти під час створення цих систем – це простіші способи збору та обробки інформації.

На сьогодні основними прикладами панорамних турів є «Google Panorama’s» і «Facebook 360». Але вони задіяні у великих системах і не можуть використовуватися як конкретна модель. Деякі з проблем під час використання цих систем можуть бути: повільні зворотні запроси та неоптимізовані дані. Було вирішено, що він повинен виконувати три важливі моменти: оптимізована структура даних, найбільш швидкий робочий процес, ніж у середніх та великих системах, зручний інтерфейс.

Для реалізації всього запланованого потрібно визначити, які платформи та технології будуть використовуватися, і було визначено, що основною платформою як основою для веб-додатків буде ASP.NET Core.

Для обрання способу зберігання даних, було віддано перевагу найкоротшому формату та найшвидшій структурі читання. Отже, на сьогоднішній день JSON включає ці два моменти і має ще одну перевагу – він легкий в розумінні. А також цей спосіб дозволяє розробити структуру за менші строки ніж це було би зроблено використовуючи інші аналоги. Відповідно, найбільш високоефективні структури даних будуть залучені до проекту. Актуальна модель збереження даних, найшвидший і найсучасніший JSON, обрані на початку, тому розробка веб-додатка забезпечує потужну платформу для дизайну.

Метою кваліфікаційної роботи є розробка програмної системи для створення 3D турів. При цьому налаштувати необхідний ступінь оптимізації роботи, використовуючи алгоритми об’єднання кадрів зображень. Додатко-

вою ціллю було створення гнучкого інтерфейсу, який буде дозволяти динамічно створювати, видаляти або оновлювати сцени в режимі реального часу.

Для досягнення поставленої мети були сформульовані наступні завдання:

- провести аналіз предметної області для розробки додатка уникаючи можливих помилок при проектуванні;
- провести порівняльний аналіз існуючих програм аналогів та вивести основні недоліки та переваги для вдосконалення проекту;
- обґрунтувати вибір програмних засобів розробки та технологій;
- провести проектування програмної системи з використанням мови UML, впровадити зображення в подальшому проектуванні;
- виконати реалізацію клієнтської та серверної частин веб-додатка;
- підготувати інструкцію користувача;
- виконати тестування веб-додатка.

Структура дипломної роботи складається з вступу, чотирьох розділів, висновків, переліку посилань на 16 найменувань, додатків. Повний обсяг проекту становить 60 сторінок і містить 15 рисунків.

1 СИСТЕМИ ПАНОРАМНИХ ВІДОБРАЖЕНЬ ТА ЇХ СУЧАСНІ АНАЛОГИ

В цьому розділі надаються загальні положення, щодо методів розробки сучасних систем для зображення панорам, а також існуючі реалізації та оцінка цих аналогів.

1.1 Проблеми, які може вирішити система панорам

Віртуальний тур – це рекламний продукт, що має унікальний дизайн, і включає в себе ряд сферичних панорам з можливістю переходів з однієї сцени в іншу. Завдяки такому підходу глядач переміщується в задане панорамою місце, може ходити по вулицях міста, відвідувати кімнати готелю, підбирати квартири, дивитися та ознайомлюватися з новими локаціями.

Найголовніша перевага – це інтерактивність. Вона дозволяє користувачеві не просто пасивно спостерігати, а й активно брати участь. Найчастіше, базова послуга таких турів – це набір фотознімків або панорам. За допомогою віртуальних турів можна наочно продемонструвати глядачеві зовнішній вигляд офісу, магазину, показати йому зсередини і ззовні виставлені на продаж будинки чи автомобілі, ознайомити його з оформленням інтер’єру.

Основні сфери в яких можуть бути використані такі системи:

- готелі, туристичні комплекси, садиби, будинки відпочинку;
- ресторани, нічні клуби, бари, кафе, казино;
- нерухомість, будівництво і архітектура;
- туризм, подорожі, індустрія розваг;
- торгівельні центри, магазини, автосалони;
- оздоровчі центри, спорт-клуби, салони краси, дитячі центри, фітнес-центри, сфера медичних послуг;
- музеї, архітектурні пам’ятки, визначні місця.

1.2 Переваги використання 3D-турів

Ця інтерактивна функція дозволяє створити ефект присутності в тій або іншій точці. В основному 3D-тури створюються з допомогою панорам, спроектованих на прості 3D-об’єкти типу сфери або циліндру з навігацією між ними.

Віртуальні панорами сприяють підвищенню:

- іміджу компанії як високотехнологічної, інноваційної;
- інвестиційної привабливості;
- відвідування сайту, на якому розміщений тур.

А також дають можливість:

- проведення віртуальних екскурсій для нових потенційних клієнтів;
- демонстрації обладнання салонів, центрів, комплексів і т.д.

Величезною перевагою сферичних панорам і 3D-турів, на відміну від інших способів реклами для продавців послуг є:

- залучення інтересу до компанії, а отже, придбання нових клієнтів, так як віртуальні тури сьогодні викликають інтерес у більшості відвідувачів, збільшують число можливих клієнтів та підвищують доходи компанії;
- оригінальність і привабливість – презентувати і рекламувати легше на відміну від більшості конкурентів способом, що викликає більший інтерес, ніж звичайні фотографії або текст;
- скорочення часу між створенням віртуального туру і знайомством з ним покупця. Якщо на створення буклету та розповсюдження його серед потенційних клієнтів йде значна кількість часу, то віртуальний тур стає доступним мільйонам користувачів інтернет-мережі практично відразу після його створення;
- цілодобова доступність – панорами, розміщені на сайтах доступні для перегляду в будь-який час доби;

– можливість різноманітного використання одних і тих же турів, як в Інтернеті, так і у вигляді презентацій, які можна демонструвати в офісі клієнта, на виставці і т.п. Все це дозволяє розширити охоплення аудиторії;
– простота і оперативність розміщення нових, оновлення та заміни старих віртуальних турів, що є гарантією актуальності представленої інформації.

1.3 Платформа Google Street View для створення панорамних зображень

Google Street View надає панорамні 360-градусні види з позначених доріг по всій його території покриття. Покриття API перегляду вулиць таке ж, як і для програми Google Maps. Список поточно підтримуваних міст для перегляду вулиць доступний на веб-сайті Карт Google.

API JavaScript Maps надає послугу Street View для отримання та маніпулювання зображеннями, які використовуються в Google Maps Street View. Ця послуга перегляду вулиць підтримується у веб-переглядачі.

Зразкове зображення перегляду вулиць показано нижче (рис. 1.1).

Рисунок 1.1 – Зразкове зображення панорами Google Street View

1.3.1 Street View для мобільних пристроїв

Для додатків Android, в яких потрібно відображення сферичних зображень, також званих панорамами, в інтерактивному засобі перегляду сферичних зображень служби Google надають підтримку цієї функції.

Google Maps Android API і Google Maps SDK for IOS надають службу Street View для отримання і обробки зображень, що використовуються в Google Street View. Зображення повертаються у вигляді сферичних панорам.

Кожна сферична панорама Street View являє собою зображення або групу зображень, що забезпечують повний круговий огляд з однієї точки. Зображення виконані відповідно до рівнопроміжкової проекціє (проекція Платі – Карре), що містить 360-градусний горизонтальний огляд (повний оборот) і 180-градусний вертикальний огляд (від напрямку строго вгору до напрямку строго вниз). Отримана 360-градусна панорама визначає проекцію на сфері з допомогою перенесення зображення на двомірну поверхню цієї сфери [14][1)].

1.4 Технології створення панорамних турів

Процес створення віртуальних панорам можна розділити на три етапи:

– фотозйомка об’єкта;

– обробка отриманих зображень;

– кінцеве складання віртуального туру.

Перший етап створення віртуальних турів – зйомка об’єкта, що представляє собою дуже трудомісткий і вкрай відповідальний процес, так як від його результатів безпосередньо буде залежати якість панорами. Для отримання високоякісних панорам з мінімальними спотвореннями слід дотримуватися ряду правил:

[1)] [14] Інтернет джерело «Street View Service»: https://developers.google.com/maps/ documentation/javascript/streetview

- камеру потрібно встановити таким чином, щоб при обранні діафрагми всі кадри в серії опинилися у фокусі (найкраще, якщо камера виявиться в центрі окружності для зйомки);
- набір знімків, що зшиваються потрібно знімати таким чином, щоб місця швів майбутньої сферичної панорами знаходилися на досить однотонних місцях (наприклад, на монолітних стінах в разі ріелторських об’єктів);
- головка штатива повинна бути оснащена рівнями, які призначені для суворого позиціонування камери в просторі;
- для всіх трьох знімків обов’язково слід здійснювати синхронізацію камери в горизонтальній і вертикальній площинах – вирівнювання камери здійснюється за допомогою рівнів;
- кути кругового повороту камери повинні бути рівні 120°, що регулюється шкалою поворотника.

Фотопанорами створюються з декількох спеціально підготовлених фотографій, що перекриваються за допомогою спеціальних програм, які зшивають знімки в єдину панораму, видаляючи спотворення, що незмінно виникають. Таких програм-зшивачів сьогодні пропонується досить багато, причому в кожній програмі використовується особлива технологія зшивання зображень і свій формат створюваних панорам, а саме зшивання може проводитися в автоматичному, ручному або змішаному режимі.

Переглядати фотопанорами можна за допомогою спеціальних браузерів, причому вибирати останні потрібно виходячи з формату панорамного файлу. Багато типів фотопанорам можуть переглядатися і в інтернет-браузері, але тільки при наявності підтримки Java-аплета або при установці спеціального плагіна.

Віртуальні тури збираються з попередньо створених фотопанорам в інших спеціалізованих додатках – будівниках турів. У більшості випадків програма для розробки віртуальних турів орієнтована на власний формат панорамних файлів (хоча іноді передбачається імпорт панорам з інших форма-

тів), тому на практиці будівник турів використовується разом із зшивачем від одного виробника.

Окремі фотопанорами зв’язуються між собою плавними переходами за рахунок виділення на них активних зон (спеціальних областей на фотопанорамах), які не тільки відповідають за переміщення від однієї панорами до іншої, а й використовуються для відображення додаткової інформації про об’єкти. Технологія активних зон дозволяє зробити акцент на окремих частинах панорами – на деталях інтер’єру, на нові товари в торгових центрах, на цікавих пам’ятках, на конкретних виставкових стендах, а також на будь-яку іншу інформацію, на яку необхідно звернути увагу віртуального відвідувача.

Крім того, можливе включення в віртуальний тур інтерактивних планів приміщень і навігаторів, що дозволяють користувачам визначати своє «місце розташування». Перегляд туру може здійснюватися за планом, по точкам переходу або автоматично відповідно до плану, при створенні проекту.

1.5 Існуючі аналоги систем створення панорамних зображень

На сьогоднішній день існує достатня кількість компаній, які займаються створенням 3D-турів на замовлення. Також є відомі платформи, які дозволяють створювати панорамні об’єкти та сцени, вони дозволяють переходити між ними, усі вони є комерційними програмними додатками, з великою кількістю опцій, які частіше всього непотрібні користувачеві. Далі по тексту буде розглянуто деякі з таких аналогів, їх переваги та недоліки. Наприкінці цього розділу будуть винесені заключні цілі, які будуть переслідуватися впродовж цієї роботи.

Найвідоміші додатки для створення віртуальних турів:

- 360cities;
- AIRPANO;
- Google Tour Builder;
- Facebook 360.

360Cities був заснований у 2007 році як видавнича платформа для фотографів-панорам, які хотіли показати свої сферичні 360° панорами в інтерактивному середовищі. У той час інтерактивна 360° фотографія була практично невідома поза відносно невеликої групи прихильників фотографії, і не було доступних онлайн-сервісів, які підтримували б інтерактивну 360° фотографію.

З самого початку вважалося, що ображення 360° високої роздільної здатності будуть цінні для компаній у певних галузях. Більшість початкових зусиль було витрачено на створення надійної платформи та залучення учасників, які прагнуть створити унікальну бібліотеку з якісними зображеннями.

Технічні вимоги:

- JPEG або 8-бітний формат TIFF без шарів;
- у назві файлу використовуються лише стандартні символи ASCII (приклади символів не дозволені: ü, á, ě тощо);
- панорама повинна бути 360° – ліва частина панорами повинна відповідати правій частині панорами, щоб глядач мав можливість оглядатися навколо без видимих швів;
- висота панорами повинна бути меншою, ніж ширина (вертикальні панорами не приймаються).

Завжди повинні бути відповідні та правильно написані метадані до всіх панорам, які завантажуються в 360Cities. Додавання метаданих покращує рейтинг пошуку зображень, збільшуючи ймовірність того, що вони будуть знайдені в запиті пошукової системи в Google, інших пошукових системах та на 360cities.net.

Мета надання метаданих – підвищити ймовірність того, що зображення знайдуть глядачі та потенційні клієнти, що надають ліцензії, які здійснюють пошук.

Клієнти, що надають ліцензії на 360 міст, зазвичай здійснюють пошук англійською мовою, тому основною мовою ваших метаданих повинна бути

англійська. Метадані, які ви додаєте до своїх зображень та відео, допомагають клієнтам ліцензувати пошук того, що вони шукають.

Основна вимога щодо мови – необхідно використовувати англійську мову, коли пишуться заголовки та описи, за винятком імен орієнтирів та інших важливих, відомих місць, які пошукові системи розпізнають.

Зображення, які завантажуються в 360Cities, повинні мати мінімальну ширину 6000 пікселів, щоб бути обраними для 360Cities і бути включеними у файл KMZ 360Cities для Google Planet Earth. Винятки іноді надаються для панорам хорошої якості, які менші. Панорами меншого розміру не виглядають добре у повноекранному режимі. Зображення НЕ слід збільшувати, оскільки це знижує якість зображення. З УСІХ камер dSLR можна зробити панораму шириною 6000 або більше пікселів. Не має значення, якщо оригінальне зображення шириною 6000 або 25000 пікселів.

Віртуальна екскурсія Google – це моделювання існуючого місця, яке зазвичай складається із послідовності зображень надійного представника Street View. У цьому випадку зображення створюються та додаються до списку Google як взаємопов’язаний віртуальний пішохідний тур. Новий віртуальний тур тепер видно разом з іншою інформацією про місцеположення, тож шукачі зможуть краще зрозуміти, що таке відвідування.

Переваги використання Google Tour Builder:

- підвищена якість контенту;
- показує миттєво те, що може бути важко описати;
- оновлений перегляд вулиць дозволяє клієнтам легко знаходити локацію після вибору на картах Google;
- можна передивлятися кількість переглядів.

Отже, як бачимо з технічних вимог, є декілька обмежень, щодо завантаження файлів, а також мов які можна використовувати для роботи з метаданими. Крім цього є значна кількість недоліків, зокрема:

- більшість віртуальних турів не можуть бути вбудованими в сайт, а лише можуть бути окремим доменом чи зображеннями в інтернеті;

- швидкість користуванням цими додатками залишається на низькому рівні;
- ціноутворення на такі рішення, як правило дуже високе;
- адміністрування віртуальними турами викликає багато питань з точки зору інтерфейса;
- для використання того чи іншого рішення необхідна спеціальні пристрої для зйомки таких знімків.

2 ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ РОЗРОБКИ ПРОЕКТУ

Значна частина успішно реалізованого проекту залежить від правильного проектування та вибору технологій на яких він буде створюватися. Саме тому загальна увага повинна бути спрямована на платформи та мови програмування, які будуть використовуватися протягом розробки.

Сьогодні є багато платформ-конкурентів у веб-розробці, які змагаються та постійно розвиваються. Потрібно визначити, що таке веб-додаток, задля того щоб визначитися з поданими технологіями. Веб-додатки – клієнт-серверні додатки, в яких клієнтом виступають браузери, а сервером – веб-сервер на різних платформах.

Логіка веб-додатків розподілена між сервером і клієнтом, зберігання даних здійснюється, переважно, на сервері, обмін інформацією відбувається по мережі. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є кросплатформеними сервісами.

Для створення веб-додатків використовуються різноманітні технології і мови програмування, наприклад:

– PHP;
– ASP та ASP.NET;
– JSP;
– Java;
– CGI;
– Perl;
– Python;
– Ruby on Rails.

Окрім того, що потрібно визначити технології щодо створення клієнт-серверного додатку, також необхідно зазначити супроводжуючі технології, для зберігання даних, побудови графічної інформації та деяких фреймворків.

2.1 Платформа ASP.NET Core 2.1

ASP.NET Core 2.1 – це одне з останніх оновлень з відкритим кодом Microsoft та міжплатформна веб-фреймворк [1][1)]. Потрібно зазначити, які функції були взяті з цього фреймворка.

Core 2.1 включає в себе безліч нових функціональних можливостей, які також варто зазначити:

- покращена продуктивність збірки;
- швидкий HttpClient;
- створення та обмін глобальними інструментами .NET Core;
- нові типи виконання для ефективного використання пам’яті;
- підтримка Alpine Linux;
- підтримка ARM32;
- підтримка стиснення Brotli;
- нові API та криптовалюти;
- можливості створення сумісних пакетів SourceLink;
- багаторівнева компіляція JIT.

Захист веб-додатків за допомогою HTTPS важливіше, ніж будь-коли раніше. Застосування HTTPS у браузері стає все більш розповсюдженим. Сайти, які не використовують HTTPS, зазначають як небезпечні. Браузери також починають впроваджувати, що нові та існуючі веб-функції повинні використовуватися лише з безпечного контексту. Нові вимоги конфіденційності, як глобальний регламент про захист даних (GDPR), вимагають використання HTTPS для захисту даних користувачів. Використовуючи HTTPS під час розробки, ви краще підготуєтеся до запуску програми під HTTPS у виробництві. ASP.NET Core 2.1 також включає функції для застосування HTTPS під час виконання. Ви можете перенаправити весь HTTP-трафік на HTTPS та

[1)] [1] Freeman A.: Pro ASP.NET Core MVC 2 2017.

прямі браузери, щоб забезпечити доступ до вашого сайту через HTTPS за допомогою протоколу HTTP Strict Transport Security (HSTS) (рис. 2.1).

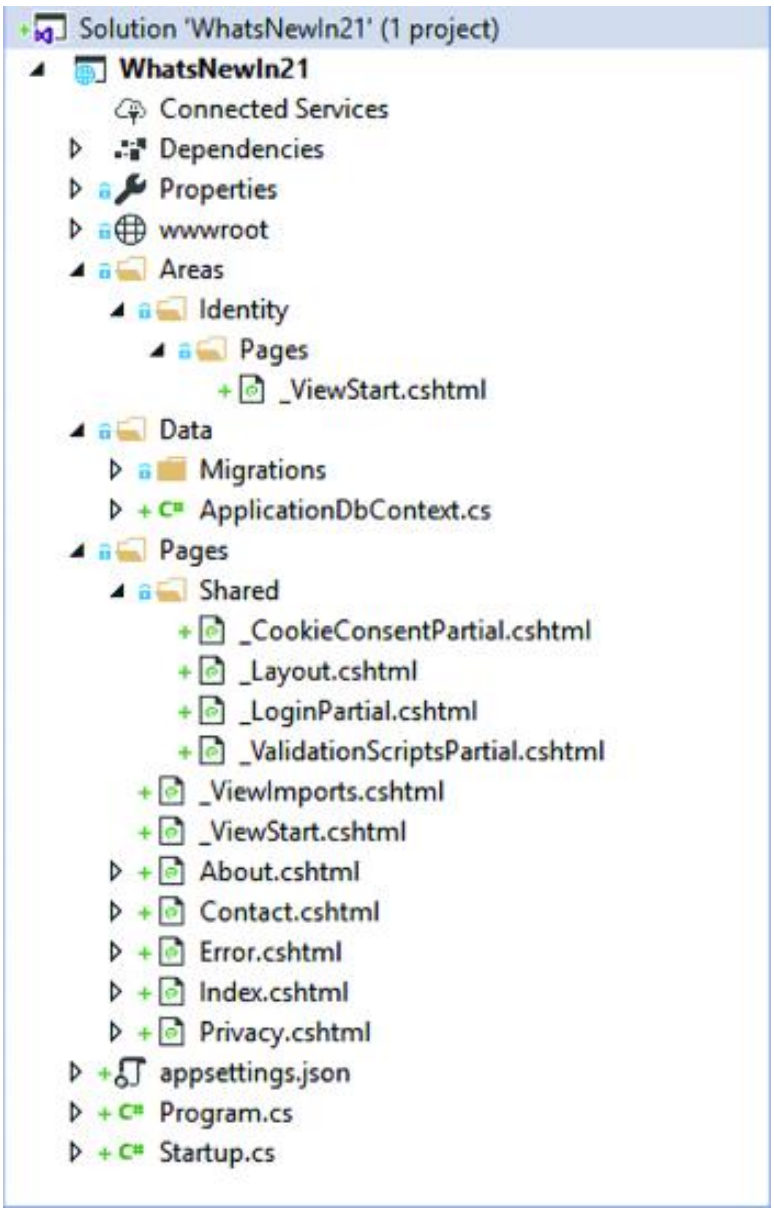

Рисунок 2.1 – Нова структура проекту веб-додатків в ASP.NET Core 2.1

Шаблони проектів ASP.NET Core дозволяють вам використовувати ці функції HTTPS [2][1]. ASP.NET Core допоможе вам написати динамічну логіку візуалізації інтерфейсу, використовуючи суміш HTML та C# під назвою Razor (cshtml). Бібліотеки класів Razor – це нова функція в ядрі ASP.NET, яка

[1] [2] Troelsen A., Japikse P.:Pro C # 7: .NET і .NET Core 8th Edition, Kindle Edition

дозволяє компілювати перегляди та сторінки Razor у бібліотеки класів багаторазового використання, які можна упакувати та ділитись [3][1)].

2.1.1 Покращення Razor Pages

Для перегляду загальних покращень потрібно почати про вдосконалення сторінок Razor, нової функції, введеної в ASP.NET Core 2.0, про яку стало відомо у релізі в вересені 2017 року. Версія 2.1 додає кілька функцій, які не ввійшли до версії 2.0, наприклад, підтримка папок для певних сторінок для пошуку спільних ресурсів.

Найпоширеніші спільні ресурси – це файли компонування та партії. За замовчуванням вони знаходились у папці Pages в ASP.NET Core 2.0, хоча ASP.NET Core MVC виявив би їх, якби вони були розміщені в папці Shared. У версії 2.1, сторінки Razor зараз здійснюють пошук цих спільних файлів, шукаючи їх у таких місцях (по порядку).

Це дозволяє легко змінювати спільні об’єкти, де потрібно, але якщо користувач додає сторінки Razor у існуючий додаток на базі MVC, то він буде мати можливість використовувати будь-які наявні спільні ресурси у своїй папці Shared [7][2)].

Ще одна особливість, яка відсутня у перших версіях сторінок Razor, – це підтримка областей. Завдяки ASP.NET Core 2.1 можна додати папку Pages із сторінками Razor у будь-яку область, що знаходиться в папці Areas програми ASP.NET Core. Корпорація Майкрософт також оновила стандартний шаблон веб-програми Visual Studio для використання областей для функціональності ідентичності.

Разом ці функції значно полегшують організацію сторінок Razor у проектній системі.

[1)] [3] Інтернет-сервіс Microsoft Docs, вступ до сторінок Razor у ядрі ASP.NET

[2)] [7] Galloway J., Wilson B., Scott Allen K.: Professional ASP.NET MVC 5.

2.1.2 Спільні бібліотеки Razor

Ще одна нова особливість 2.1 – це підтримка для завантаження ресурсів Razor з окремих бібліотек або пакетів. Окремі додатки ASP.NET Core часто діляться спільними ресурсами, такими як особливості ідентичності (логін, реєстрація, забутий пароль тощо). Як правило, ці загальні риси призводять до того, що в окремих проектах багато дублікатів коду, що призводить до збільшення технічної заборгованості.

Нова функція бібліотеки класів Razor (RCL) підтримує створення файлів Razor та їх розгортання як асоційовані проекти або пакети NuGet, які може споживати будь-яка кількість програм ASP.NET Core.

Додавши цю функцію, компілятор буде збирати ресурси Razor, автоматично шукаючи пов’язані ресурси у бібліотеках та пакетах, на які посилаються. Раніше ресурси Razor не розроблялися до моменту їх розгортання та запиту. ASP.NET Core 2.1 інтегрує компіляцію Razor в процес збирання, що також призводить до більш швидкого запуску програми.

Ресурси Razor в RCL можуть бути замінені, тому якщо їх використовують для використання спільних активів між проектами, не втрачається можливості налаштувати певні аспекти на основі проекту [12][1)].

RCL сяють для наскрізних проблем додатків, таких як макет, навігація та автентифікація. Фактично, вбудована функція ASP.NET Core Identity здатна використовувати цю підтримку, щоб також більше використовуватись між проектами.

2.1.3 Ідентифікація як надбудова

Щоб скористатися функцією ASP.NET Core Identity у додатку до версії 2.1, як правило, потрібно було прийняти рішення, щоб додати підтримку для

[1)] [12] Мартин Р. С., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C#; Символ-Плюс, 2011.

нього під час створення проекту веб-додатків. А для того, щоб додати підтримку Identity пізніше, зазвичай процес полягає в створенні абсолютно нового проекту веб-додатків з відповідною підтримкою. Потім необхідно було б копіювати файли з нового додатка у існуючу реалізацію. Це не було зручним рішенням.

Завжди важким технічним завданням для ASP.NET Core (і ASP.NET теж) була можливість додавання підтримки для ідентифікування до наявного додатку, частково тому, що ця підтримка включала передові ресурси Razor, які не можна було упакувати окремо, і розгортання нових ресурсів у наявному додатку може не вдатися з безлічі причин.

З додаванням підтримки RCL додавання ідентифікування до вже наявного додатку стає набагато простіше, оскільки процес більше не потребує додавання нових файлів Razor до наявного додатку.

Можуть бути використані готові рішення у Visual Studio, аби додати ідентифікацію до існуючих додатків. Нові програми, побудовані за допомогою останніх шаблонів, будуть використовувати спільні ресурси Razor, а не включати сторінки або представлення, пов’язані з ідентифікацією у сам проект.

Перевага такого підходу полягає в меншій кількості файлів у нових проектах, але без втрати функціональності та можливості гнучкого налаштування поведінки.

За замовчуванням підтримка Identity додає лише область із одним файлом _ViewStart.cshtml. Але це можна змінити, налаштувавши поведінку певних файлів, клацнувши правою кнопкою миші на проект, натиснувши «Додати Новий елемент», а потім вибрати «Додання ідентифікації».

2.1.4 Додаткові функції та оптимізація

У цьому релізі також вдосконалюються деякі інші частини ASP.NET Core 2.1, включаючи вбудований сервер Kestrel. Kestrel тепер використовує керовані сокети для свого транспортного шару за замовчуванням [11][1)].

Ще одне нове доповнення до хостинг-компонента ASP.NET Core – це тип HostBuilder, який буде використан для налаштування частин хоста. HostBuilder дуже схожий на існуючий WebHostBuilder, але не дозволяє вказати клас запуску з веб-проекту. Він розроблений, щоб дозволити налаштування та вирішити деякі проблеми, такі як введення залежностей, конфігурація та ведення журналу для сценаріїв, що не належать до Інтернету, наприклад, розміщені сервіси або консольні програми.

Нарешті, якщо пишемо кінцеві точки API в додатку ASP.NET Core, можемо скористатися деякими вдосконаленнями, доданими в 2.1. По-перше, з'являється можливість додати атрибут [ApiController] до будь-якого з контролерів, які відкривають API. Це додає низку функцій кінцевим точкам, визначеним на цих контролерах, таких як:

- помилки перевірки моделі автоматично повернуть BadRequest (ModelState);
- джерела прив'язки (наприклад, [FromBody]) автоматично робляться для параметрів дії;
- завантаження файлів за допомогою [FromForm] автоматично підводить тип вмісту даних із кількома частинами або формами даних;
- потрібна маршрутизація атрибутів.

Ще одне доповнення – це новий тип повернення, ActionResult <T>. Цей тип повернення використовується замість IActionResult і дозволяє включити інформацію про тип у підпис методу. Такі інструменти, як Swashbuckle, можуть використовувати цю інформацію для створення документації OpenAPI

[1)] [11] Головин, И.Г. Языки и методы программирования: Учебник для студентов учреждений высшего профессионального образования / И.Г. Головин, И.А. Волкова. – М.: ИЦ Академия, 2012.

або Swagger. Без цього типу повернення потрібно буде анотувати методи, які просто повернули IActionResult з атрибутом [ProducesResponseType], щоб викрити цю інформацію [10][1)].

2.1.5 Підвищення продуктивності роботи .NET Core 2.1

У внутрішніх лабораторних роботах Microsoft, продуктивність .NET Core 2.1 на 10% більша, ніж .NET Core 2.0 для сценаріїв простого тексту та JSON, і приголомшливих на 123% більше для більш реалістичного сценарію доступу до даних, затверджуючи позицію .NET Core як один з найшвидших доступних фреймворків (рис. 2.2).

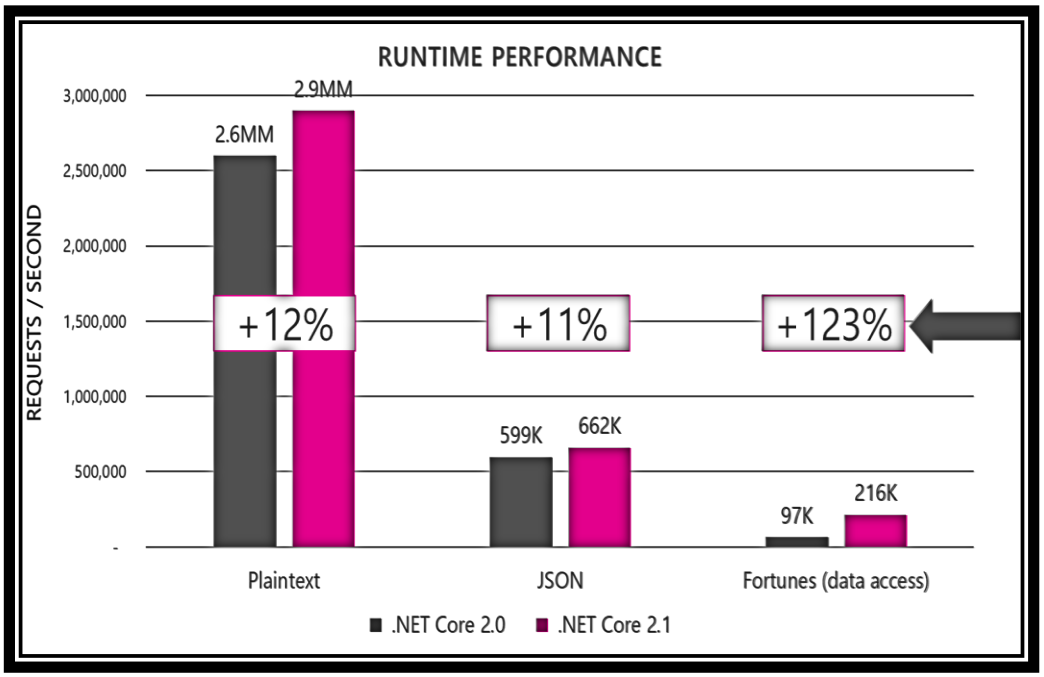

Рисунок 2.2 – Продуктивність роботи NET Core 2.1 у показниках TechEmpower

Отже, було обрано платформу ASP.NET Core 2.1 виходячи з таких показників як:

[1)] [10] Голицына, О.Л. Языки программирования: Учебное пособие / О.Л. Голицына, Т.Л. Партыка, И.И. Попов. – М.: Форум, НИЦ ИНФРА-М, 2013.

- швидкість роботи;
- легкість налаштування;
- кількість якісних функцій;
- безпечність платформи.

Сучасні проблеми потребують сучасних рішень, для можливості розширення системи в майбутньому та підтримки протягом роботи.

2.2 Використання та визначення JSON для оптимізації внутрішніх процесів в системі

JSON (JavaScript Object Notation) – це легкий формат обміну даними (рис. 2.3). Людям легко читати і писати, а машинам легко розбирати та генерувати. Він заснований на підмножині стандарту мови програмування JavaScript 3-е видання ECMA-262 – грудень 1999 р.

JSON – це текстовий формат, повністю незалежний від мови, але використовує конвенції, знайомі програмістам сімейства C мови, включаючи C, C#, Java, JavaScript, Perl та багато інших. Ці властивості роблять JSON ідеальною мовою обміну даними [4][1)].

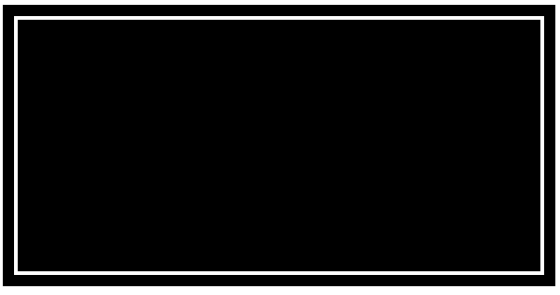

Рисунок 2.3 – Об’єкт в JSON

JSON побудований на двох структурах: колекції пар імен-значень. У різних мовах це реалізується як об’єкт, запис, структура, словник, хеш-

[1)] [4] Інтернет-сервіс ECMA-404 Стандарт обміну даними JSON.

таблиця, список клавіш або асоціативний масив. Впорядкований список значень у більшості мов реалізується як масив (рис. 2.4), вектор, список або послідовність.

Рисунок 2.4 – Масив у JSON

2.3 Ресурс Pannellum

Pannellum – це легкий, безкоштовний та відкритий переглядач панорами для Інтернету. Побудований за допомогою HTML5, CSS3, JavaScript [13][1)] та WebGL, також він підключається безкоштовно. Його можна легко розгорнути як один файл, його розмір всього 21 кбіт, а потім вставити на сторінки у вигляді фрейму. Утиліта конфігурації включена для створення необхідного коду для вбудовування. API включений для більш прогресивних інтеграцій.

Оскільки Pannellum побудований за веб-стандартами, для його роботи потрібен сучасний браузер. Повна підтримка (із відповідними графічними драйверами):

- Firefox 23 та вище;
- Chrome 24 та вище;
- Safari 8 та вище;
- Internet Explorer 11;
- Edge.

[1)] [13] Фридман, А.Л. Основы объектно-ориентированного программиро-вания/ А.Л. Фридман. – М.: Гор. линия-Телеком, 2012.

Список базується на підтримці функцій. Оскільки тестовані лише останні веб-переглядачі, у старих браузерах можуть бути регресії.

Автономний метод, який використовується для вбудовування фреймів є найпростішим у використанні, але JavaScript API є більш потужним і забезпечує більш тісну інтеграцію.

Внутрішній переглядач самостійного аналізує параметри URL-адреси, щоб створити конфігурацію на основі JSON, а потім інстанціює переглядання за допомогою API JavaScript.

Автономний переглядач приймає підмножину параметрів конфігурації як параметри URL. Решта параметрів можна встановити, використовуючи файл конфігурації JSON, вказаний за допомогою спеціального параметра конфігураційного URL. Оскільки Pannellum є статичним вмістом, його можна розмістити практично на будь-якому веб-сервері. Для зручності також надається CDN.

Панорамні зображення можуть бути надані у прямокутній, кубовій карті, або у форматах мультирезолюції [12][1)]. Рівнокутний – найпростіший у використанні, оскільки потрібне лише одне зображення; Метадані XMP Google Photo Sphere автоматично читаються та використовуються, якщо вони надані. Однак, щоб забезпечити підтримку всіх пристроїв, що підтримують WebGL.

Карти кубів потребують шести зображень, але підтримують панорами дещо вищої роздільної здатності, оскільки переважна більшість пристроїв підтримують куби до 4096 пікселів.

Крім того, куб-карти підтримуються резервним рендерінгом на основі трансформації CSS 3D Pannellum, тому вони працюватимуть на старих мобільних пристроях, які не підтримують WebGL [16][2)]. Хоча великі зображення підтримуються, розміри завантажень слід враховувати. Остаточний формат

[1)] [12] Мартин Р. С., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C#; Символ-Плюс, 2011.

[2)] [16] Інтернет джерело «A Lightweight Panorama Viewer for the Web: https://pannellum.org/documentation/overview/

введення – це формат мультирезолюції Pannellum, який генерується з прямокутного зображення за допомогою сценарію Pyneon Gene.py Pannellum.

Цей формат – формат на основі куба на карті, за винятком того, що кожна грань куба – це піраміда зображення плитки замість одного зображення. Підтримуються довільно великі зображення, підтримується рендерінг на основі перетворення CSS 3D, а піраміди зображень швидко завантажуються. Мінусом цього формату є те, що потрібна додаткова робота для перетворення зображення на нього та великої кількості файлів, які необхідно розмістити.

До додаткових особливостей Pannellum належать гарячі точки для відображення інформації, підключення декількох панорам до віртуальних турів та підтримка відео. Найкращий спосіб дізнатися про функції Pannellum та як їх використовувати – це вивчити приклади та посилання на конфігурацію [10][1)].

Оскільки це не очевидно з прикладів, доцільно зазначити, що параметр hotSpotDebug існує для допомоги у розміщенні гарячих точок. Це можна ввімкнути під час налаштування гарячих точок.

2.4 Бібліотеки та фреймворки в системі

Для того щоб бути об’єктивним у розповіді про наявність та використання сторонніх ресурсів, потрібно визначити саме залежності, які були використані під час розробки проекту.

Але, по-перше, слід зазначити які сервера для завантаження були обрані та чому. Мережа доставки вмісту або мережа розповсюдження контенту (CDN) – це географічно розподілена мережа проксі-серверів та їх центрів обробки даних. Мета – забезпечити високу доступність та продуктивність, розподіляючи послугу просторово відносно кінцевих користувачів.

[1)] [10] Голицына, О.Л. Языки программирования: Учебное пособие / О.Л. Голицына, Т.Л. Партыка, И.И. Попов. – М.: Форум, НИЦ ИНФРА-М, 2013.

CDN з’явилися в кінці 1990-х років як засіб для зменшення вузьких місць в Інтернеті, навіть коли Інтернет починає ставати критичним середовищем для людей і підприємств.

З цього часу CDN стали обслуговувати значну частину Інтернет-контенту, включаючи веб-об’єкти (текст, графіку та сценарії), об’єкти для завантаження (медіафайли, програмне забезпечення, документи), програми (електронна комерція, портали), пряму трансляцію ЗМІ, потокові ЗМІ за запитом та сайти соціальних медіа.

Існує основний набір ресурсів Javascript, які використовуються в Інтернеті. Такі пакети, як jQuery, Bootstrap, Backbone.js і YUI, лежать в основі багатьох веб-сторінок.

Для завантаження цих сторінок потрібно завантажити ресурси Javascript. Як результат, є сенс, щоб ресурси знаходилися на максимально швидких з’єднаннях. Однак це лише частина проблеми.

Інша перевага передбачає кешування браузера. Якщо два сайти використовують jQuery, в ідеалі в браузері потрібно завантажити лише один раз, а потім використовувати один і той же код на обох сайтах. Для того, щоб скористатися кешуванням цього браузера, обидва сайти повинні посилатися на один і той же код через одну і ту ж URL-адресу.

Як результат, має сенс не тільки посилатися на CDN для коду Javascript, але і для того, щоб була можливість використовувати той самий CDN, який використовують і інші сайти.

Сьогодні все частіше набагато ефективніше використовувати інші ресурси задля того, щоб проект розроблявся більш продуктивно та швидко. Крім того готові рішення, як правило, надають змогу користуватися перевіреними, протестованими та грамотно оформленими фреймворками з документацією.

Саме тому було вирішено використовувати наступні бібліотеки в проекті, такі як:

– jQuery;

– Bootstrap 4;

– Animate.css;

– Sweet Alert.

Нижче представлені короткі відомості про залежності в проекті та версії бібіотек в ньому:

```
  “libraries”: [
{
    “library”: “jquery@3.3.1”,
    “destination”: “wwwroot/js/jquery/”,
    “files”: [
      “jquery.js”
    ]
  },
  {
    “library”: “twitter-bootstrap@4.1.3”,
    “destination”: “wwwroot/css/bootstrap”,
    “files”: [
      “css/bootstrap.css”,
    ]
},
  {
    “library”: “animate.css@3.7.0”,
    “destination”: “wwwroot/css/animate”,
    “files”: [
      “animate.css”
    ]
},
  {
    “library”: “sweetalert@2.1.2”,
    “destination”: “wwwroot/sweetalert/”
  }
]
```

2.4.1 Фреймворк jQuery

jQuery – бібліотека JavaScript, призначена для спрощення обходу дерев HTML DOM та маніпуляцій, а також обробки подій, анімації CSS та Ajax. Це

безкоштовне програмне забезпечення з відкритим кодом, що використовує дозволену ліцензію MIT. Станом на травень 2019 року jQuery використовується 73% з 10 мільйонів найпопулярніших веб-сайтів. Веб-аналіз показує, що це найбільш широко розгорнена бібліотека JavaScript з великим запасом, яка має в 4 рази більше використання, ніж будь-яка інша бібліотека JavaScript.

Окрім розуміння того, що таке jQuery, корисно зрозуміти, чому розробники використовують саме його:

- jQuery робить програмування JavaScript більш швидким та більш ефективним;
- jQuery є відкритим кодом (тобто кожен може внести свій внесок або змінити) та має велике співтовариство користувачів, тобто його постійно підтримують та оновлюють;
- jQuery має велику документацію;
- jQuery може бути пов’язаний з будь-якими іншими бібліотеками JavaScript, якими ви можете користуватися;
- jQuery має багато плагінів, які дозволяють розширити функціональність jQuery за потреби.

Бібліотека jQuery є JavaScript файлом, яка включає всю його DOM, події (events), ефекти (effects), і Ajax функції. Вона може бути додана до web-сторінки посиланням на локальну копію, або на одну з копій доступних на публічному сервері.

<script type=“text/javascript” src=“jquery.js”></script>

Для прикладу, візьмемо просту функції обробник і продемонструємо написання на цьому фреймворку:

```
function panoramaClick(viewer, event) {
    var coords = viewer.mouseEventToCoords(event);
    console.log(coords[0], coords[1]);
    if ($(‘#option-add’).parent().hasClass(‘active’)) {
        if (coords[0] != 0 && coords[1] != 0) {
```

```
                $('#hotspot-scene').val($('#list-tab > a.list-
group-item.active').first().data('scene-name'));
                  $('#hotspot-yaw').val(coords[1]);
                $('#hotspot-pitch').val(coords[0]);
                $('#add-modal-hotspot').modal('show');
            }
        }
    }
```

Як бачимо, для пошуку будь-якого елемента, можна використовувати звичайні селектори CSS, а також є багато інших функцій пошуку таких, як: parent, find, child, siblinghs та інші.

Саме ці функції спрощують інтерфейс взаємодії. Крім того, варто наголосити на тому, що зміна значень в цьому фреймворку досить легка, наприклад, для того щоб змінити значення будь-якого елемента введення, потрібно лише викликати функцію val та передати до неї аргумент, яка означає – «має значення».

Важливими функціями jQuery є:

- вибір DOM: jQuery надає селекторам для отримання елемента DOM на основі різних критеріїв, таких як назва тегу, id, ім'я класу css, ім'я атрибута, значення, n-та дитина в ієрархії тощо;
- маніпуляція з DOM: дає можливість керувати елементами DOM за допомогою різних вбудованих функцій jQuery, наприклад, додавання або видалення елементів, зміна вмісту html, клас css і т.д;
- спеціальні ефекти: дозволяють застосовувати спеціальні ефекти до елементів DOM, наприклад, показати або приховати елементи, зменшити видимість або зменшити видимість, ефект ковзання, анімацію тощо;
- події: бібліотека jQuery включає функції, еквівалентні подіям DOM, як клік, dblclick, mouseenter, mouseleave, blur, keyup, keydown тощо. Дані функції автоматично вирішують проблеми між переглядачами;
- ajax: jQuery також включає прості у використанні функції ajax для завантаження даних із серверів, не завантажуючи всю сторінку;

– підтримка крос-браузера: бібліотека jQuery автоматично обробляє проблеми між веб-браузером, тому користувачеві не потрібно турбуватися про це. jQuery підтримує IE 6.0+, FF 2.0+, Safari 3.0+, Chrome і Opera 9.0+.

2.4.2 Фреймворк Bootstrap

В проекті було використано фреймворк Bootstrap версії 4, через те що прогрес цього фреймворку дуже великий, саме через це необхідно використовувати найсучасніші версії, аби в майбутньому вилучити проблеми із сумісністю залежностей різних версій бібліотек в проекті.

Bootstrap – це вільний і відкритий вихідний код CSS, спрямований на адаптивну мобільну розробку інтернету. Він містить шаблони дизайну на основі CSS та JavaScript для типографії, форм, кнопок, навігації та інших компонентів інтерфейсу [9][1)].

Марк Отто оголосив Bootstrap 4, 29 жовтня 2014 року. Перша альфа-версія Bootstrap 4 була випущена 19 серпня 2015 року. Перша бета-версія була випущена 10 серпня 2017 року. Помітити призупинену роботу на Bootstrap 3, 6 вересня 2016 року, щоб звільнити час для роботи над Bootstrap 4. Bootstrap 4 було завершено 18 січня 2018 року [8][2)].

До суттєвих змін належать:

– основне перезапис коду;
– заміна менше на Сасс;
– додавання Reboot, набір змін, характерних для елемента CSS, в одному файлі на основі Normalize;
– відмова від підтримки для IE8, IE9 та iOS 6;
– підтримка гнучких ящиків CSS;

[1)] [9] Mastering Bootstrap 4: Master the latest version of Bootstrap 4, 2nd Edition, Kindle Edition – Benjamin Jakobus, Jason Marah.
[2)] [8] Jake Spurlock. Bootstrap. Responsive Web-Development. – O'Reilly, 2013.

– додавання параметрів налаштування навігації.

Bootstrap 4 підтримує останні версії Google Chrome, Firefox, Internet Explorer, Opera та Safari (крім Windows). Він також підтримує підтримку IE9 та останнього Firefox.

Популярність Bootstrap пов’язана з тим, що він дозволяє верстати сайти в кілька разів швидше, ніж це можна виконати на «чистому» CSS і JavaScript. А в нашому світі час – це найдорожчий ресурс. Також його популярність пов’язана з доступністю. Вона полягає в тому, що на ньому навіть початківець розробник може верстати досить якісні макети, які важко було б виконати без глибоких знань веб-технологій і достатньої практики.

Фреймворк Bootstrap являє собою набір CSS і JavaScript файлів. Щоб його використовувати ці файли необхідно просто підключити до сторінки. Після підключення стануть доступні інструменти даного фреймворка: колоночная система (сітка Bootstrap), класи і компоненти.

Переваги Bootstrap:

– зменшення кількості часу, що витрачається на розробку;

Традиційно використання фреймворків і бібліотек значно полегшує роботу розробникам і дозволяє розробляти проекти швидше.

– адаптивність;

Bootstrap дозволяє створювати адаптивні сайти. Дизайн сайту буде коректно відображатися на екранах пристроїв різних розмірів незалежно від їх діагоналі.

– крос-браузерність;

Сайти, зроблені з використанням Bootstrap, будуть однаково відображатися у всіх сучасних браузерах.

– легкість у використанні і швидкість в освоєнні;

– зрозумілий код;

Bootstrap дозволяє писати якісний і зрозумілий код, який легко зрозуміє інший розробник. Це значно спрощує розробку в команді.

– єдність стилів.

Елементи Bootstrap виглядають гармонійно між собою і дозволяють створювати сторінки і сайти в єдиному стилі.

Недоліки Bootstrap:

– шаблонність;

Сайти, розроблені за допомогою Bootstrap, схожі один на одного: однакова структура, навігація, кнопки. Кожен новий сайт схожий на безліч вже створених і це не дуже добре. Вирішити проблему можна – треба відмовитися від використання готових рішень і максимально змінювати шаблон в залежності від побажань замовника і ідей дизайнерів.

– відсутність гнучкості;

Незважаючи на всі переваги, Bootstrap – інструмент, який має свої обмеження (повертаємося до попереднього пункту про те, що всі сайти на ньому схожі один на одного). Тому Bootstrap може не підійти для реалізації якихось проектів.

– старі браузери;

Так як Bootstrap намагається йти в ногу з часом і постійно оновлюється, в старих браузерах сайти на Bootstrap можуть відображатися некоректно.

Отже, Bootstrap – це інструмент, що дозволяє швидко створити сайт з стандартних блоків. У цьому полягає і його перевага, і його недолік – можна швидко отримати якісний сайт, при цьому втративши в оригінальності. Втім, якщо розробник добре знає CSS і Bootstrap, то він зможе модифікувати стандартні блоки і зробити дизайн, що відрізняється від інших сайтів [15][1)].

Bootstrap відмінно підходить для маленьких проектів, де потрібно швидко зібрати верстку, але в цілому він використовується на сайтах різного масштабу: на вересень 2018 року Bootstrap використовують 17,8% сайтів у мережі (і їх кількість постійно зростає).

[1)] [15] Front-end Development with ASP.NET Core, Angular and Bootstrap / Simone Chiaretta: John Wiley & Sons, Inc.

3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Структура та приклади використання технологій проекту

3.1.1 Призначення JsonSerializer

В інформатиці в контексті зберігання даних, серіалізація – це процес перекладу структур даних або стану об’єкта у формат, який може зберігатися (наприклад, у файлі або буфері пам’яті) або передаватися (наприклад, через мережеве з’єднання) та бути реконструйованим пізніше.

Коли отримана серія бітів перечитується відповідно до формату серіалізації, її можна використовувати для створення семантично ідентичного клону вихідного об’єкта. Для багатьох складних об’єктів, таких як ті, що широко використовують посилання, цей процес не є простим. Серіалізація об’єктно-орієнтованих об’єктів не включає жоден з пов’язаних з ними методів, з якими вони були раніше пов’язані.

Цей процес серіалізації об’єкта також називають маршалінгом об’єкта. Протилежна операція, витягуючи структуру даних із ряду байтів, – це десеріалізація [5][1)].

Найшвидший метод перетворення між текстом JSON та об’єктом .NET – це використання JsonSerializer. JsonSerializer перетворює .NET-об’єкти в їх еквівалент JSON і навпаки, зіставляючи імена властивостей .NET-об’єктів у імена властивостей JSON та копіюючи значення.

Для простих сценаріїв, де потрібно перетворити на рядок JSON і з нього, методи SerializeObject і DeserializeObject на JsonConvert забезпечують просту у використанні обгортку через JsonSerializer.

```
Scene scene = new Scene();
scene.Name = “Scene 1”;
scene.ExpiryDate = new DateTime(2008, 12, 28);
scene.Sizes = new string[] { “Small”, “Medium”, “Large” };
```

[1)] [5] Документація Newtonsoft Json.NET.

```
    string output = JsonConvert.SerializeObject(scene);
    //{
    //  “Name”: “Scene 1”,
    //  “ExpiryDate”: “2008-12-28T00:00:00”,
    //  “Sizes”: [
    //    “Small”,
    //    “Medium”,
    //    “Large”
    //  ]
    //}
    Scene deserialized = JsonCon-
vert.DeserializeObject<Scene>(output);
```

Обидва SerializeObject і DeserializeObject мають перевантаження, які приймають об’єкт JsonSerializerSettings. JsonSerializerSettings дозволяє використовувати багато параметрів JsonSerializer, перелічених нижче, використовуючи прості методи серіалізації.

Для більшого контролю над тим, як об’єкт серіалізується, JsonSerializer можна використовувати безпосередньо. JsonSerializer здатний читати та записувати текст JSON безпосередньо в потік через JsonTextReader та JsonTextWriter [6][1)].

Також можуть використовуватися інші типи JsonWriters, такі як JTokenReader / JTokenWriter для перетворення вашого об’єкта в та з LINQ в JSON об’єкти або BsonReader / BsonWriter для перетворення в BSON і з нього.

```
    Scene scene = new Scene();
    scene.ExpiryDate = new DateTime(2008, 12, 28);
    JsonSerializer serializer = new JsonSerializer();
    serializer.Converters.Add(new
JavaScriptDateTimeConverter());
    serializer.NullValueHandling = NullValueHandling.Ignore;
    using (StreamWriter sw = new StreamWriter(@”c:\json.txt”))
    using (JsonWriter writer = new JsonTextWriter(sw))
    {
        serializer.Serialize(writer, scene);
    }
```

[1)] [6] Інтернет-сервіс Newtonsoft. Популярні високопродуктивні рамки JSON.

JsonSerializer має ряд властивостей, щоб налаштувати спосіб серіалізації JSON. Вони також можуть бути використані з методами на JsonConvert через перевантаження JsonSerializerSettings.

Було вирішено виключити базу даних із складу, оскільки надсилання запитів та очікування їх відповіді тягне за собою кілька проблем, таких як неоптимізована програма, менша розуміння структури, проблематичність зміни та додавання іншої логіки.

Але коли використовується лише серіалізація, це рішення дозволяє змінювати дані в режимі реального часу (рис. 3.1).

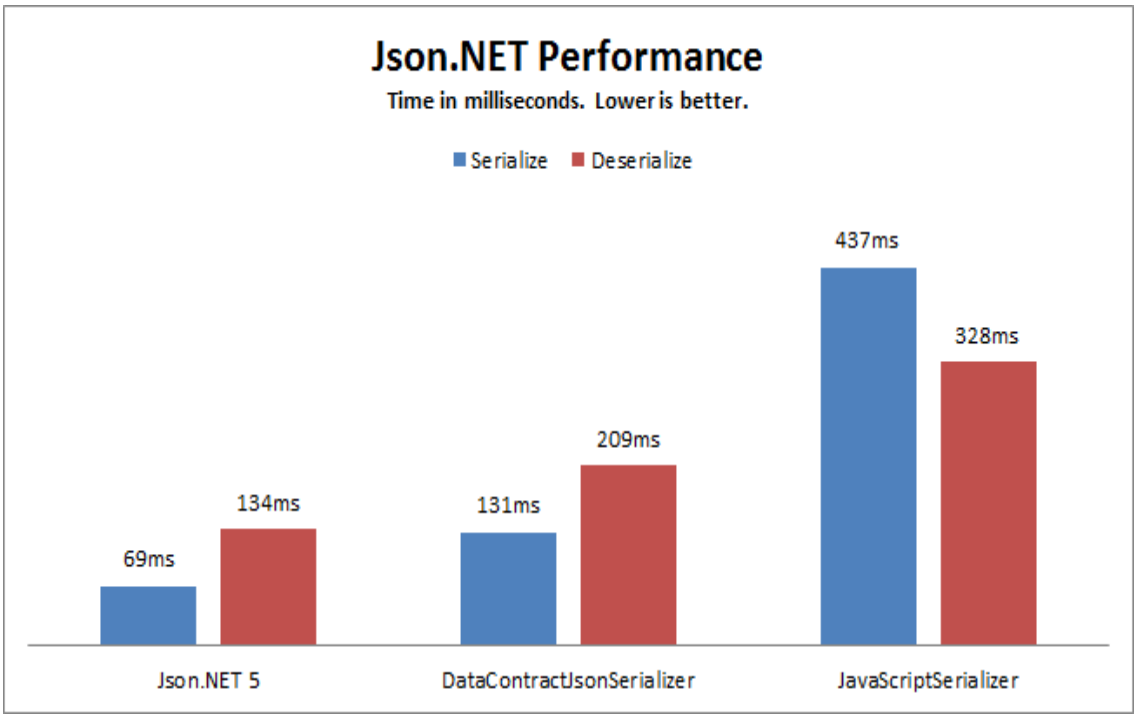

Рисунок 3.1 – Переваги використання серіалізації Newtonsoft

3.1.2 Компонент Loader

Loader (або «завантажувач») – це головна структура проекту для ініціалізації та подальшої роботи системи. Цей компонент дозволяє постійно тримати масив даних в поточному стані та допомагає при отриманні панорамних сцен.

```
public static class Loader
{
   public static IEnumerable<Scene> Scenes { get; private
                                                    set; }
    public static string Main =>
Scenes.FirstOrDefault(scene => scene.IsMain)?.Name;
    public static IEnumerable<Scene> Units =>
Scenes.Where(scene => scene.IsUnit && !scene.IsMain);
    static Loader()
    {
        Scenes = SettingProvider.Deserialize();
        if (Scenes == null)
        {
            Scenes = new List<Scene>();
        }
    }
```

В наданому лістингу можна побачити серіалізацію та зворотний до неї процес – десеріалізацію. Ці процеси дозволяють швидко та без зайвих пристроїв, такі як бази даних чи інші сервіси, підключити здатність зберігання, читання, а також видалення та зміни поточних даних.

Варто зазначити, що серіалізація – це процес перетворення будь-якої структури даних у послідовність бітів. Зворотною до операції серіалізації є операція десеріалізації – відновлення початкового стану структури даних із бітової послідовності.

```
SettingMessage UpdateScenes()
{
    return SettingProvider.Serialize(Scenes);
}
SettingMessage UpdateScenes(IEnumerable<Scene> scenes)
{
    var message = SettingProvider.Serialize(scenes);
    if (!message.IsError)
    {
        Scenes = scenes;
    }
    return message;
}
```

Серіалізація використовується для передавання об’єктів мережею й для збереження їх у файлах. Наприклад, потрібно створити розподілений застосунок, різні частини якого мають обмінюватися даними зі складною структурою. У такому випадку для типів даних, які передбачається передавати, пишеться код, який здійснює серіалізацію і десеріалізацію.

Об’єкт заповнюється необхідними даними, потім викликається код серіалізації, в результаті виходить, наприклад, документ із строками JSON.

Серіалізація надає декілька корисних можливостей:

– метод реалізації зберігання об’єктів, який зручніший, ніж запис їх властивостей в текстовий файл на диск і повторна збірка об’єктів читанням файлів;
– метод здійснення віддалених викликів процедур, як, наприклад, у SOAP;
– метод розповсюдження об’єктів, особливо в технологіях компонентно-орієнтованого програмування, таких як COM і CORBA;
– метод виявлення змін у даних, що змінюються з часом.

```
public static SettingMessage AddScene(Scene scene){
    var newScenes = new List<Scene>(Scenes);
    newScenes.Add(scene);
    var message = SettingProvider.Serialize(newScenes);
    if (!message.IsError)
    {
        Scenes = newScenes;
    }
    return message;}
```

3.1.3 Використання технології LINQ

Абревіатура LINQ позначає цілий набір технологій, що створюють і використовують можливості інтеграції запитів безпосередньо в мову C#. Традиційно запити до даних виражаються у вигляді простих рядків без перевірки типів при компіляції або підтримки IntelliSense.

Крім того, розробнику доводиться вивчати різні мови запитів для кожного типу джерел даних: баз даних SQL, XML-документів, різних веб-служб. Технології LINQ перетворюють запити в зручну мовну конструкцію, яка застосовується аналогічно класам, методам і подіям.

Для розробника, який створює запити, найбільш очевидною частиною LINQ є інтегроване вираження запиту. Вирази запиту використовують декларативний синтаксис запиту.

За допомогою синтаксису запиту можна виконувати фільтрацію, впорядкування і групування даних з джерела даних, обходячись мінімальним обсягом програмного коду. Одні і ті ж базові вирази запиту дозволяють однаково легко отримувати і перетворювати дані з баз даних SQL, наборів даних ADO .NET, XML-документів, XML-потоків і колекцій .NET.

У наступному коді показано повний приклад використання запиту.

```
    public static SettingMessage UpdateScene(Scene scene)
    {
        // принцип використання LINQ, щоб отримати першу сцену
зі списку
        var sceneToUpdate = Scenes.FirstOrDefault(x => x.Name
== scene.Name);

        sceneToUpdate.AutoLoad = scene.AutoLoad;
        sceneToUpdate.Hfov = scene.Hfov;
        sceneToUpdate.Hotspots = scene.Hotspots;
        sceneToUpdate.Image = scene.Image;
        sceneToUpdate.IsMain = scene.IsMain;
        sceneToUpdate.IsUnit = scene.IsUnit;
        sceneToUpdate.Name = scene.Name;
        sceneToUpdate.ShowFullscreenCtrl = sce-
ne.ShowFullscreenCtrl;

        var message = SettingProvider.Serialize(Scenes);
        return message;
    }
```

3.1.4 Структура серіалізованого файла сцен

Як у загальній частині цієї роботи було сказано, що головна структура має вигляд списку сцен для основної роботи програми. Отже, щоб розуміти яким чином проводиться маніпуляція панорамними зображеннями необхідно розібратися із древовидною структурою самого серіалізованого файла. Цей файл є звичайною строкою з переносами, але у форматі JSON.

```
{
  “$values”: [
    {
      “$type”: “OdekuTour.Settings.Scene, OdekuTour”,
      “Name”: “Scene #1”,
      “Panorama”: “images/Scene1.jpg”,
      “Image”: “Scene1.jpg”,
      “ShowFullscreenCtrl”: false,
      “AutoLoad”: true,
      “Hfov”: 180,
      “IsMain”: true,
      “IsUnit”: true,
      “Hotspots”: {
        “$values”:[]
      }
    }
  ]
}
```

Він складається з масиву сцен, які мають такі основні параметри, як:

– тип сцени;

– назва сцени;

– посилання до файлу;

– перемикач на завантаження в повноформатному режимі;

– перемикач на автозавантаження;

– радіус обзору цієї сцени;

– позначка на головну сцену;

– позначка на розділ;

– масив гарячих точок або покажчиків.

В той же час, як видно із серіалізованої строки вище, є також масив із гарячих точок (Hotspots). В ньому, як правило, зберігається список точок, які дозволяють переходити до інших сцен та розділів.

Завдяки тому, що кожна гаряча точка має єдиний інтерфейс, ця структура має безкінечний потенціал до розширення. Це значить, що записуючи кожну точку до масиву, позначається окремий тип, який стосується саме її, що допомагає при ініціалізації програми завантажувати кожну точку окремо та отримувати дані, які стосуються саме її типу. З боку ж додатку, є єдиний інтерфейс, який дозволяє працювати з ним як з єдиною структурою.

```
{
  “$type”: “OdekuTour.Settings.Hotspots.SceneHotspot”,
  “Pitch”: 3.3632171673033673,
  “Yaw”: 1.419129560576073,
  “Type”: “scene”,
  “SceneId”: “Scene #2”,
  “Title”: “Pointer #1”,
  “Subtitle”: “Sub title”,
  “Description”: “Description”,
  “Footer”: “Quote”
}
```

З зазначенного коду винесемо характеристику однієї з точок:

– Pitch – в даному випадку цей параметр відзначає нахил точки відносно панорманого зображення;

– Yaw – це параметр, що задає обертання навколо інерціальної рамки;

– тип сцени;

– ідентифікатор;

– заголовок;

– підзаголовок;

– опис цієї сцени;

– цитата до опису чи самої назви сцени.

Кути Ейлера – це три кути, введені Леонардом Ейлером для опису орієнтації жорсткого тіла щодо нерухомої системи координат. Вони також можуть представляти орієнтацію рухомої системи відліку у фізиці або орієнтацію загальної основи в тривимірній лінійній алгебрі.

Кути Ейлера можна визначити за елементарною геометрією або за складом обертів. Геометричне визначення демонструє, що три складені елементарні обертання (обертання навколо осей системи координат) завжди є достатніми для досягнення будь-якого цільового кадру.

Три елементарні обертання можуть бути зовнішніми (обертання навколо осей x, y, z вихідної системи координат, які, як передбачається, залишаються нерухомими), або внутрішніми (обертання навколо осей обертової системи координат XYZ, солідарні з рухомим тілом, яке змінює своє орієнтація після кожного елементарного обертання).

3.1.5 Функції-допоміжники для будування сцен

Для будування сцен необхідно не тільки завантажувати інформацію, щодо типу зображення, чи навіть його контенту, а також ще відображати деякі метадані, які зберігаються для подання користувачеві необхідної для розуміння інформації.

Саме для таких цілей було створено функції-допоміжники, які займаються побудуванням та відображенням інформації, відстеженням різних подій, а також виконанням функцій-обробників під час настання такої події.

```
function hotspot(hotSpotDiv, args) {
    $(hotSpotDiv).mouseenter(function (e)
{
        $(‘#description.card>.card-body>.card-
title’).text(“”).append(args.title);
        $(‘#description.card>.card-body>.card-
subtitle’).text(“”).append(args.subtitle);
        $(‘#description.card>.card-body>.card-
text’).text(“”).append(args.description);
```

```
            $(‘#description.card>.card-footer>.card-
text’).append(“”);
            $(‘#description.card>.card-footer’).remove();
            if (args.footer && args.footer.trim() != ‘‘) {
                var footer = ‘<div class=\”card-footer\”><p
class=\”card-text text-right text-warning\”>‘
                    + args.footer + ‘</p></div>‘;
                $(‘#description.card’).append(footer);
            }
            $(‘#description’).removeClass(‘fadeIn animated
fadeOut invisible’).addClass(‘animated fadeIn’);
        });
        $(hotSpotDiv).mouseout(function () {
            $(‘#description’).addClass(‘animated fadeOut’);
        });
    }
```

Вище зазначена функція-допоміжник займається якраз відображенням інформації о відомих даних про сцену. В даному випадку при потраплянні курсора миші на гарячу точку – відображається модальне вікно, яке показує користувачам цю службову інформацію. Нижче представлені також функції-допоміжники, але вони займаються іншими подіями. При клацанні мишкою по гарячій точці, вона запускає перемикання між сценами, що дає можливість користувачеві трансферуватися між панорамічними зображеннями.

Крім того, запускається тригер, який починає процес відображення активного розділу та вимикає попередній. Таким чином відбувається плавний перехід між частинами додатку:

```
    function hotspotClickHandler(obj, args) {
        $(‘#description’).addClass(‘animated fadeOut’);
        $(‘.nav-link’).removeClass(‘active’);
        $(‘#’ + args.id).addClass(‘active’);
    }

    $(‘#@main.GetHashCode()’).addClass(‘active’).click(() => {
        viewer.loadScene(‘@main.GetHashCode()’);
        $(‘.nav-link’).removeClass(‘active’);
        $(this).addClass(‘active’);
    });
```

3.2 Реалізації алгоритму

Структура проекту поєднується з декількох аспектів. Фактично проект розділяє три основні частини: сайт адміністратора, сайт користувача та алгоритм зберігання та створення панорамних турів.

Переглядаючи частину для користувачів, можна побачити, що вона є не тільки зрозумілою для користувачів, але й простою та зрозумілою для подальшої розробки програмістами та підтримки програмного забезпечення в цілому.

Структура виглядає так: контролери, середній рівень, налаштування, перегляди, конфігурації, завантажувач сцен і саме сцени. У свою чергу налаштування можна розширити за допомогою точок доступу, помічника сцени та SettingProvider.

Основна частина, яка будує повний тур панорами, розроблена сторінками «Razor» в розширенні cshtml. Розділ сцен генерується цим кодом:

```
‘scenes’: {
    @foreach (var scene in scenes)
    {
    @: ‘@scene.Name.GetHashCode()’: {
    @:   ‘panorama’: ‘@scene.Panorama’,
    @:   ‘showFullscreenCtrl’:
    @scene.ShowFullscreenCtrl.ToString().ToLower(),
    @:   ‘autoLoad’: @scene.AutoLoad.ToString().ToLower(),
    @:   ‘hfov’: @scene.Hfov.ToString().Replace(“,”, «.”),
    @:   ‘hotSpots’:[
    @:    // цей блок кода створює гарячі точки
    @: ]},
}
```

Для генерації точок переходу між сценами генерується масив цих точок, як зазначено в прикладі вище. Цей масив створюється циклом послідовного перебору точок, які збережені на стороні сервера. Така структура дозволяє показувати нові сцени миттєво, а сцени які були змінені або видалені, будуть моментально змінюватися.

Крім того така структура дозволяє створювати будь-які типи гарячих точок, тобто система є розширюваною. На етапі розробки, було створено два основні типи гарячих точок:

– точки інформації;

– точки переходу між сценами.

Така генерація дозволяє дуже швидко завантажувати зображення для їх обробки, а також маніпулювати їх розташовуванням на екрані.

```
     foreach (var hotspot in scene.Hotspots)
     {
              @:{
              @: ‘pitch’: @hotspot.Pitch.ToString().Replace(“,”,
«.”),
              @: ‘yaw’: @hotspot.Yaw.ToString().Replace(“,”,
«.”),

             @: ‘type’: ‘@hotspot.Type’,
              @: ‘createTooltipFunc’: hotspot,
              @: ‘createTooltipArgs’: {
              @:      ‘title’: ‘@hotspot.Title’,
              @:      ‘subtitle’: ‘@hotspot.Subtitle’,
              @:      ‘description’: ‘@hotspot.Description’,
              @:      ‘footer’: ‘@hotspot.Footer’,
              @: },
              if(hotspot is InfoHotspot infoHotspot)
              {
                  @: ‘URL’: ‘@infoHotspot.URL’,
              }
              else if (hotspot is SceneHotspot sceneHotspot)
              {
                  @: ‘clickHandlerFunc’: hotspotClickHandler,
                  @: ‘clickHandlerArgs’:
                  @: { ‘id’:
‘@sceneHotspot.SceneId.GetHashCode()’ },
                  @: ‘sceneId’:
‘@sceneHotspot.SceneId.GetHashCode()’
              }
              @:},
       }
```

Конструктор покажчиків був розроблений для створення інформації про карту кожного разу, коли ми вибираємо її.

У створеному додатку є кілька областей. Насправді він розділений на адміністраторів, користувачів та основні частини. Як показано на скріншоті нижче, ви можете побачити всю структуру, як у папці від Visual Studio 2017 IDE (рис. 3.2).

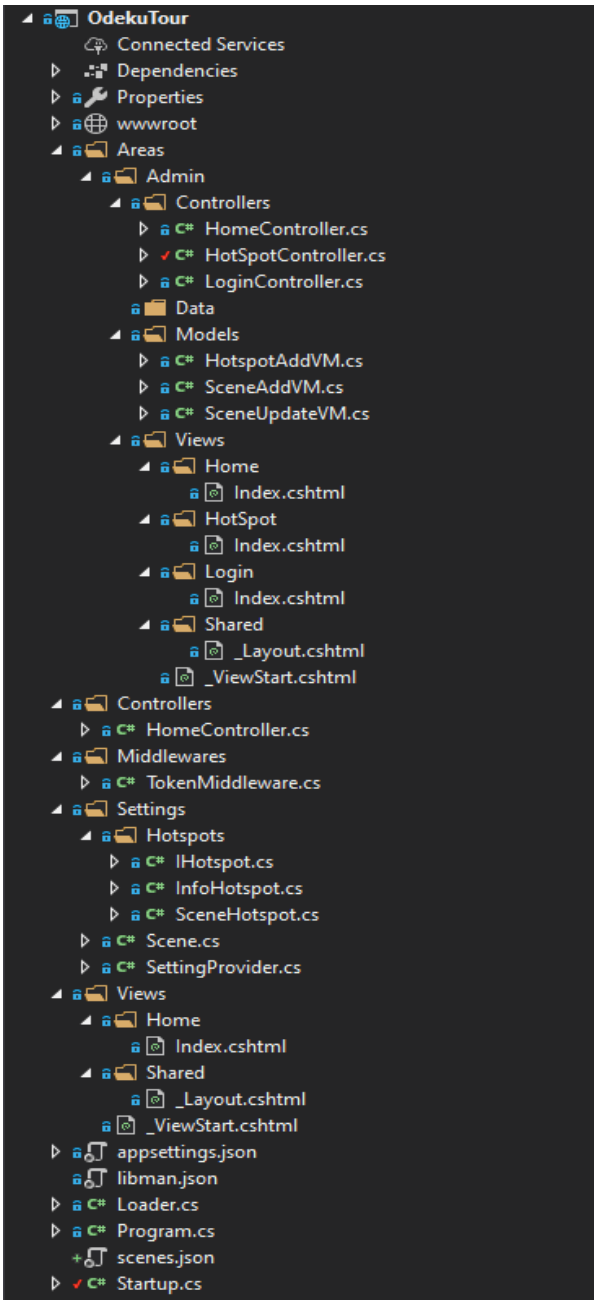

Рисунок 3.2 – Структура проекту

Обробник кліків Hotspot викликається для зміни поточного розташування в панорамному турі.

Існують також й інші помічники, які дозволяють користувачу перемикатися між розділами та змінювати активні статуси для них.

Тіло класу завантажувача також важливо, оскільки воно серіалізує та десеріалізує всі дані для панорамного туру. Він має статичний модифікатор та кілька методів додавання, оновлення та читання сцен в системі.

Також слід описати сутності SettingProvider та SettingMessage. Класи SettingProvider та SettingMessage використовуються для збереження даних у файлах у файловій системі завдяки потокам. Вони потрібні всій системі для її конфігурації та первинної ініціалізації.

3.3 Шаблон розробки MVC

Вся структура веб-додатків представляє Model-View-Controller. MVC – це модель дизайну, яка використовується для роз'єднання інтерфейсу користувача (вид), даних (модель) та логіки програми (контролер). Ця закономірність допомагає досягти відокремлення проблем (рис. 3.3).

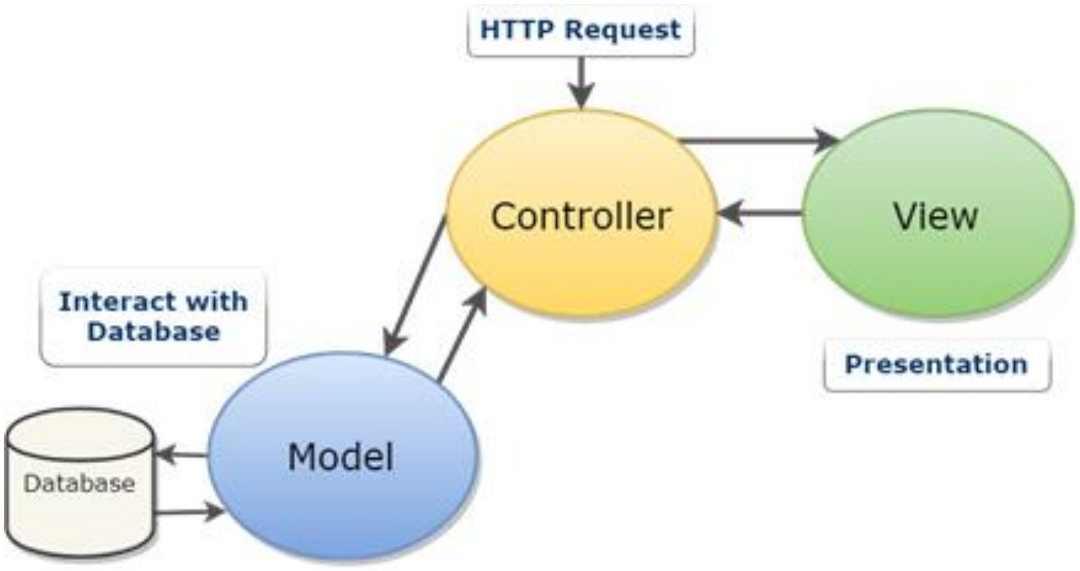

Рисунок 3.3 – MVC шаблон

Використовуючи шаблон MVC для веб-сайтів, запити направляються до «Контролера», який відповідає за роботу з «Моделлю» для виконання дій або отримання даних.

«Контролер» вибирає «Вид» для відображення та надає його «Моделі». Перегляд відображає остаточну сторінку на основі даних у «Моделі» [7][1)].

Модель містить лише дані програми, вона не містить логіки, що описує, як представити дані користувачеві. View представляє користувачеві дані моделі. Представлення знає, як отримати доступ до даних моделі, але не знає, що ці дані означають або що може зробити користувач, щоб маніпулювати ними.

Контролер існує між видом і моделлю. Він «слухає» події, викликані поданням (або іншим зовнішнім джерелом), і виконує відповідну реакцію на ці події. У більшості випадків реакція викликає метод на моделі. Оскільки подання та модель пов’язані через механізм сповіщення, результат цієї дії автоматично відображається у поданні. Часто існує маса рішень, коли MVC не використовує модель як базу даних і цей проект не є винятком.

Переваги:

– кілька розробників можуть працювати одночасно над моделлю, контролером та видами;

– MVC дозволяє логічно групувати пов’язані дії на контролері разом. Погляди на конкретну модель також групуються;

– моделі можуть мати кілька переглядів.

Недоліки:

– рамкова навігація може бути складною, оскільки вона вводить нові шари абстракції і вимагає від користувачів адаптації до критеріїв декомпозиції MVC;

– знання про декілька технологій стає нормою. Розробники, що використовують MVC, мають бути кваліфікованими у кількох технологіях.

[1)] [7] Galloway J., Wilson B., Scott Allen K.: Professional ASP.NET MVC 5.

MVC є скоріше архітектурним зразком, але не для повного застосування. MVC здебільшого відноситься до інтерфейсу користувача, шару взаємодії програми. Але все одно буде потрібен рівень бізнес-логіки, можливо, якийсь рівень обслуговування та рівень доступу до даних.

4 ОПИС ПРОГРАМНОГО ПРОДУКТУ ТА ІНСТРУКЦІЯ КОРИСТУВАЧА

Врешті був реалізований проект, який забезпечує функціональність створення нових панорамних турів та безліч функцій для налаштування системи. Цей проект слід розглядати як окремий модуль, хоча він може бути незалежним веб-додатком. Слід зазначити, що додаток забезпечує базові інтерфейси для підключення до іншої системи або частин цієї системи.

Для того, щоб увійти, потрібно вписати доменне ім’я в адресний рядок. Потім ввести ім’я користувача та пароль, вони можуть бути змінені у файлі appsettings.json (рис. 4.1).

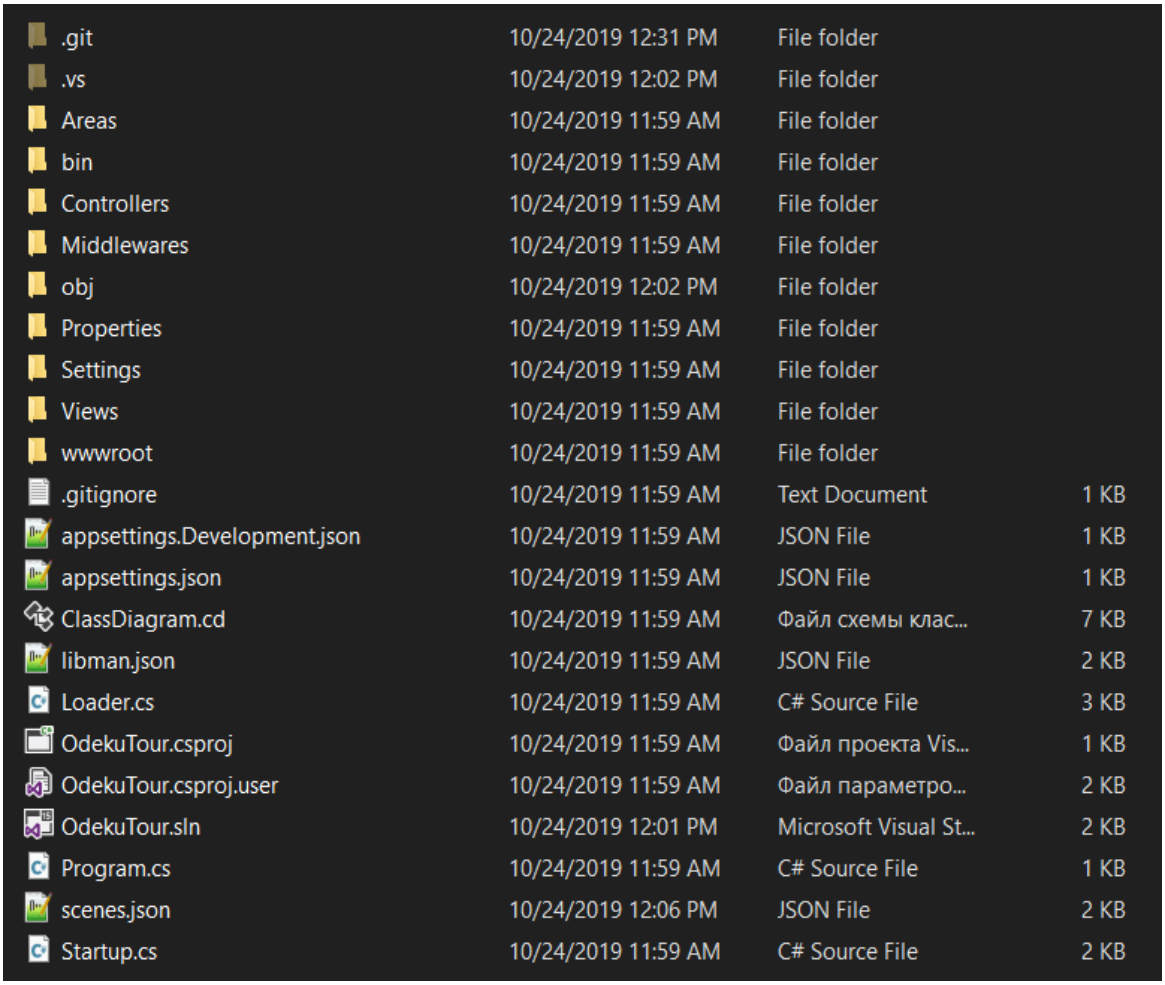

Рисунок 4.1 – Компоненти проекту

У таблиці «Головне» розміщені секції (панорами), за допомогою кнопки «додавання» можна створити нові панорами та встановити настройки для кожної з них. Модальне вікно (рис. 4.2) для додавання сцени виглядає наступним чином:

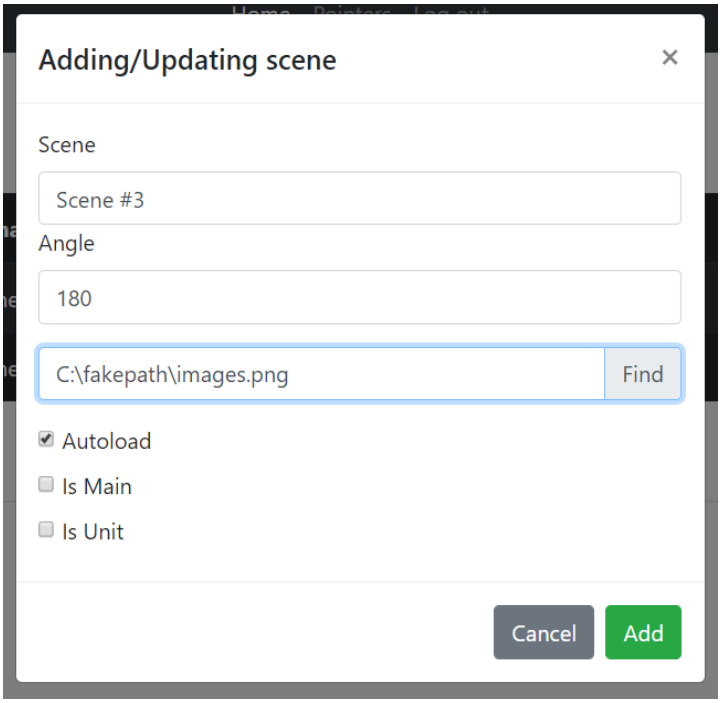

Рисунок 4.2 – Додавання сцени

Поле «Сцена» необхідне для запису назви самої панорами, яка відображатиметься на сайті для користувачів. Поле «Кут огляду» – дозволяє налаштувати кут огляду в градусах для користувача. У третьому рядку адміністратор вибирає файл, який буде панорамою. Опція «Запуск» – визначає, чи буде сцена завантажуватися автоматично або натисканням кнопки.

«Основна сцена» – визначає, чи буде завантажена панорама при першому завантаженні сайту для користувача. «Розділ» – визначає, чи буде панорама в заголовках сайту. Головний екран завантаження панорами виглядає наступним чином (рис. 4.3).

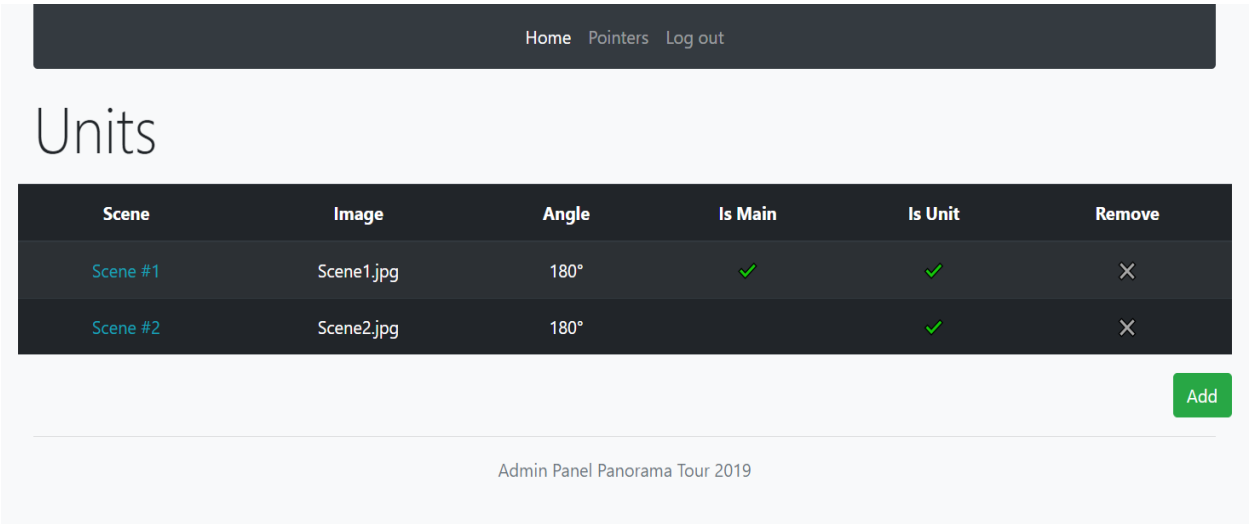

Рисунок 4.3 – Вигляд списку сцен

Користувальницький інтерфейс створений таким чином, що коли користувач наводить назву завантаженого зображення, підказка з'явиться та відобразить стиснене зображення. Якщо ви натиснете на ім'я, з'явиться модальне вікно, на якому зображення буде відображатися в повному розмірі. Сцена оновлюється за тим же сценарієм, що і додаток. Адміністратор має право змінити всі параметри, а потім натиснути кнопку зміни. Збереження будуть застосовані після перезавантаження сторінки.

Для того, щоб видалити сцену, достатньо клацнути лівою кнопкою миші на хресті, який знаходиться в лінії панорами. Натисніть кнопку «Видалити», і панорама буде видалена зі списку.

Наступний розділ – «Покажчики» (рис. 4.4). Цей розділ дозволяє користувачеві налаштувати гарячі точки, які будуть використані для навігаційної екскурсії в майбутньому. А саме:

Кнопки «Додати» та «Видалити» перемикають режими взаємодії з покажчиками. Вказівник – точка, яка встановлена на сцені для вказівки додаткової інформації або для переходу на інші сцени:

– виберіть режим «Переміщення», щоб прокрутити сцену;
– виберіть режим «Додати» та натисніть на область сцени, куди потрібно додати вказівник;

– виберіть режим «Видалити», а потім клацніть лівою кнопкою миші на покажчику, який потрібно видалити.

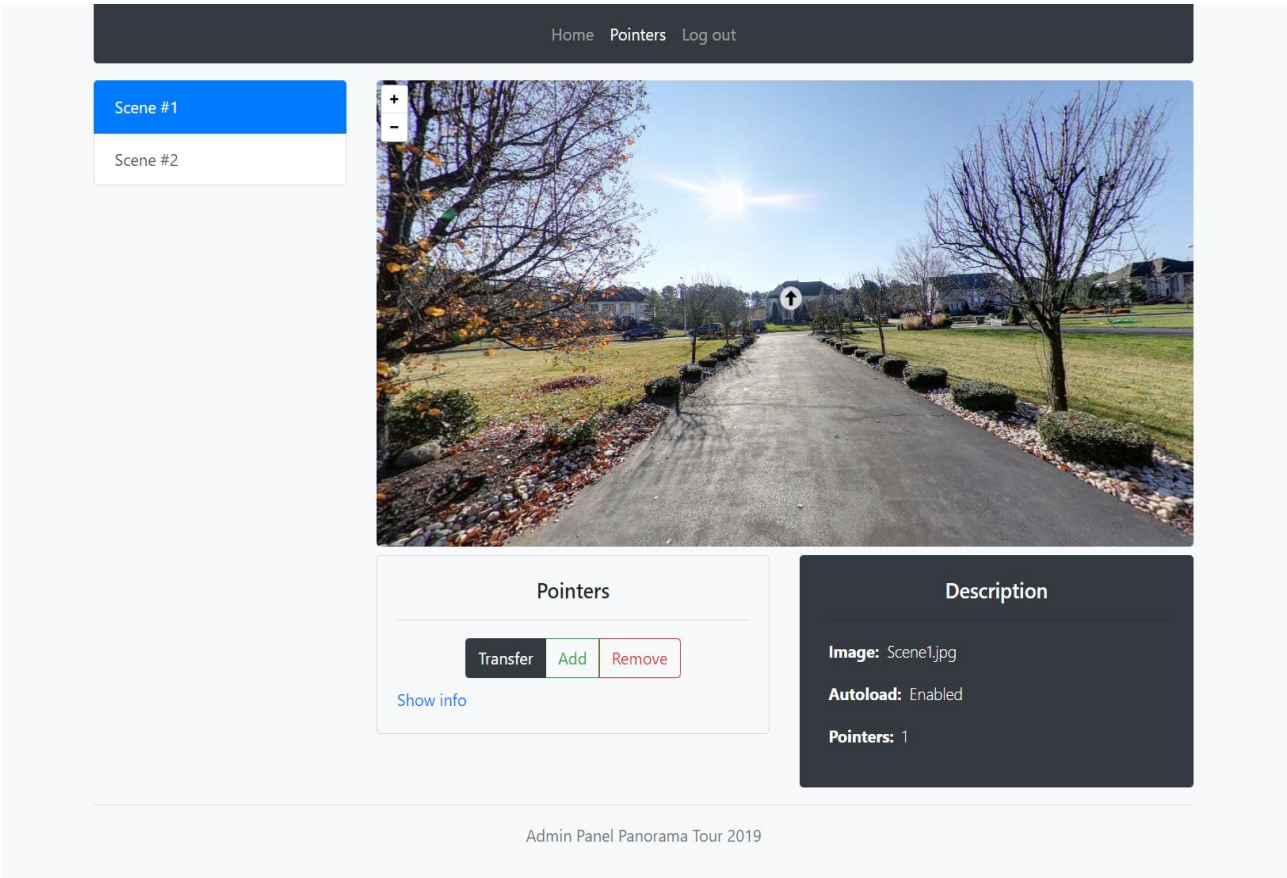

Рисунок 4.4 – Меню покажчиків

Покажчик додається за допомогою додаткового вікна (рис. 4.5), яке дозволяє вводити деякі параметри:

– назва;

– підзаголовок;

– опис;

– підпис;

– можливість переходу на інший сайт;

– можливість переходу на іншу сцену.

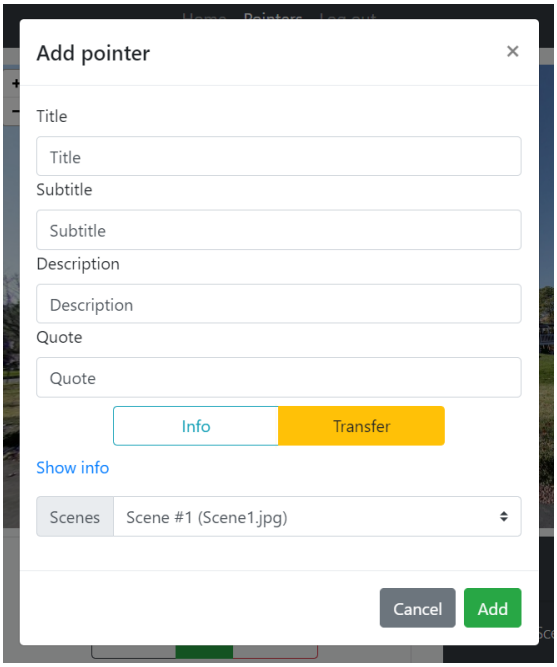

Рисунок 4.5 – Додавання покажчиків

Є варіант із додаванням нового індексу, який перемістить користувача на сторінку в браузері. Нижче – друга версія вказівника – «Перехід до нової сцени». Головна сторінка виглядатиме так (рис. 4.6):

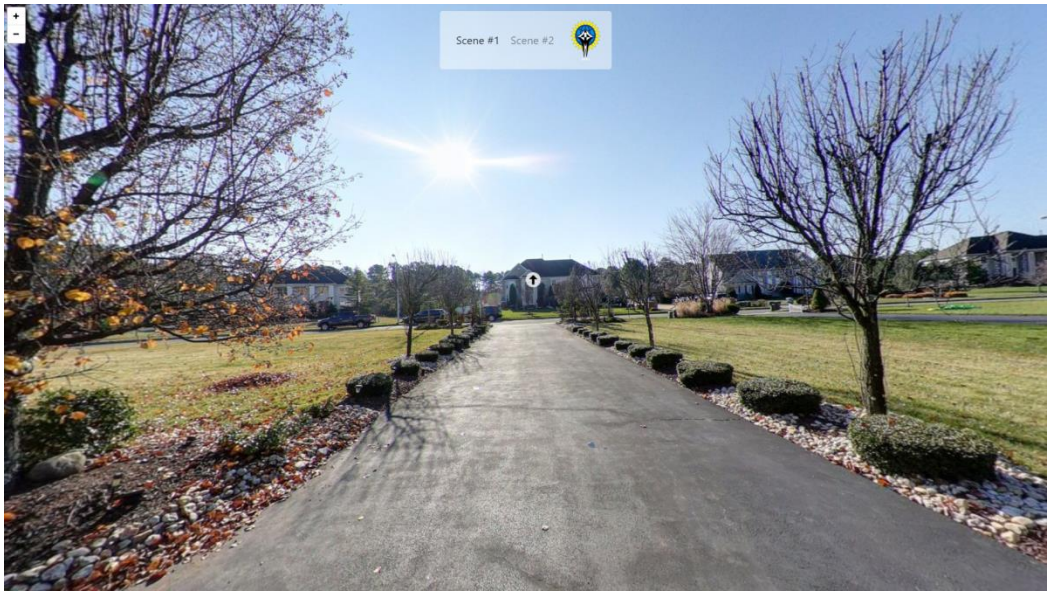

Рисунок 4.6 – Головна сторінка панорами

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було проведено розробку алгоритмічних зв’язок та аналіз існуючих аналогів. На початку роботи було спроектовано диаграму класів для більш швидкої розробки системи та пошуку оптимальних рішень. Підготовлений звіт про проведену працю з посиланнями на використаний матеріал, в тому числі підручники та інтернет джерела. Були розроблені системи адміністрування даними, а також система споживання контента.

На етапі проектування було обрано систему контроля версій – Git. Такий захід дозволив працювати гнучко та контролювати кожний процес розробки. Імплементація проекту була впроваджена завдяки платформі ASP.Net Core, використовуючи шаблон проектування MVC. Для швидкої розробки та легкої підтримки було обрано процес серіалізації як приорітетний, натомість виключена система баз даних. Це призвело до підвищення ефективності програмного продукту, а також більш простої системи його налаштовування.

Проект може бути запущений на будь-якій версії та ОС, завдяки тому, що він є інтернет-сервісом. Отже, даний додаток можна вважати кросплатформеним, тому що застосований фреймворк дозволяє розгортувати його на сервері з різними операційними системами.

Наприкінці розробки для тестування та реалізації проекту, було використано хостинг, на якому було розгорнута збірка. Для додавання сцен були створені панорамні зображення високої резолюції.

Отже, був розроблений окремий модуль для створення та візуалізації панорамних турів з оптимізованим зберіганням даних та винайденим алгоритмом візуалізації. Це дозволяє в режимі реального часу змінювати дані зображення, завантажувати або видаляти їх, створювати нові панорами. Також ця версія програми дозволяє користувачам економити свій час на розробку та оптимізацію ресурсів, необхідних для підтримки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Freeman A.: Pro ASP.NET Core MVC 2 2017.
2. Troelsen A., Japikse P.:Pro C # 7: .NET і .NET Core 8th Edition, Kindle Edition
3. Інтернет-сервіс Microsoft Docs, вступ до сторінок Razor у ядрі ASP.NET [Чинний від 10.06.2019р.]: https://docs.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-3.0&tabs=visual-studio (дата звернення 06.04.2020).
4. Інтернет-сервіс ECMA-404 Стандарт обміну даними JSON [Чинний від 12.12.2017р.]: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf (дата звернення 06.04.2020).
5. Документація Newtonsoft Json.NET [Чинний від 01.09.2013р.]: https://www.newtonsoft.com/json (дата звернення 06.04.2020).
6. Інтернет-сервіс Newtonsoft. Популярні високопродуктивні рамки JSON [Чинний від 27.11.2018 р.]: https://www.newtonsoft.com/json (дата звернення 06.04.2020).
7. Galloway J., Wilson B., Scott Allen K.: Professional ASP.NET MVC 5.
8. Jake Spurlock. Bootstrap. Responsive Web-Development. – O'Reilly, 2013.
9. Mastering Bootstrap 4: Master the latest version of Bootstrap 4, 2nd Edition, Kindle Edition – Benjamin Jakobus, Jason Marah.
10. Голицына, О.Л. Языки программирования: Учебное пособие / О.Л. Голицына, Т.Л. Партыка, И.И. Попов. – М.: Форум, НИЦ ИНФРА-М, 2013.
11. Головин, И.Г. Языки и методы программирования: Учебник для студентов учреждений высшего профессионального образования / И.Г. Головин, И.А. Волкова. – М.: ИЦ Академия, 2012.
12. Мартин Р. С., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C#; Символ-Плюс, 2011.

13. Фридман, А.Л. Основы объектно-ориентированного программирования/ А.Л. Фридман. – М.: Гор. линия-Телеком, 2012.
14. Інтернет джерело «Street View Service» [Чинний від 16.04.2020р.]: https://developers.google.com/maps/documentation/javascript/streetview (дата звернення 24.05.2020).
15. Front-end Development with ASP.NET Core, Angular and Bootstrap / Simone Chiaretta: John Wiley & Sons, Inc.
16. Інтернет джерело «A Lightweight Panorama Viewer for the Web» [Чинний від 23.02.2019р.]: https://pannellum.org/documentation/overview/ (дата звернення 25.05.2020).

ДОДАТОК А
ДІАГРАМА КЛАСІВ

Рисунок А.1 – Діаграма класів

ДОДАТОК Б

АЛГОРИТМ ГЕНЕРАЦІЇ ПАНОРМАНИХ ЗОБРАЖЕНЬ

```
$(document).ready(function () {
     var viewer = pannellum.viewer('panorama', {
          'default': {
               'firstScene': '@main.GetHashCode()',
               "sceneFadeDuration": 1000
          },
          'scenes': {
               @foreach (var scene in scenes)
               {
                  @: '@scene.Name.GetHashCode()': {
                  @:   'panorama': '@scene.Panorama',
                  @:   'showFullscreenCtrl':
@scene.ShowFullscreenCtrl.ToString().ToLower(),
                  @:   'autoLoad':
@scene.AutoLoad.ToString().ToLower(),
                  @:   'hfov':
@scene.Hfov.ToString().Replace(",", "."),
                  @:   'hotSpots':[
                         foreach (var hotspot in scene.Hotspots)
                         {
                              @:{
                              @: 'pitch':
@hotspot.Pitch.ToString().Replace(",", "."),
                              @: 'yaw':
@hotspot.Yaw.ToString().Replace(",", "."),
                              @: 'type': '@hotspot.Type',
                              @: 'createTooltipFunc': hotspot,
                              @: 'createTooltipArgs': {
                              @:      'title': '@hotspot.Title',
                              @:      'subtitle':
'@hotspot.Subtitle',
                              @:      'description':
'@hotspot.Description',
                              @:      'footer':
'@hotspot.Footer',
                              @: },
```

```
                                if(hotspot is InfoHotspot
infoHotspot)
                                {
                                @: ‘URL’: ‘@infoHotspot.URL’,
                                }
                                else if (hotspot is SceneHotspot
sceneHotspot)
                                {
                                @: ‘clickHandlerFunc’:
hotspotClickHandler,
                                @: ‘clickHandlerArgs’: { ‘id’:
‘@sceneHotspot.SceneId.GetHashCode()’ },
                                @: ‘sceneId’:
‘@sceneHotspot.SceneId.GetHashCode()’
                                }
                                @:},
                           }
                    @: ]},
               }
          }
     });
     viewer.on(‘scenechange’, function () {
          setTimeout(function () {
               $(fakeLoaderEl).fadeOut();
          }, 300);
     });
     setTimeout(function () {
          viewer.loadScene(‘@main.GetHashCode()’);
     }, 300);
     viewer.on(‘mousedown’, function (event) {
          console.log(viewer.getPitch(), viewer.getYaw());
          console.log(viewer.mouseEventToCoords(event));
     });
     function hotspot(hotSpotDiv, args) {
          $(hotSpotDiv).mouseenter(function (e) {
               $(‘#description.card>.card-body>.card-
title’).text(“”).append(args.title);
               $(‘#description.card>.card-body>.card-
subtitle’).text(“”).append(args.subtitle);
```

```
                $(‘#description.card>.card-body>.card-
text’).text(“”).append(args.description);
                $(‘#description.card>.card-footer>.card-
text’).append(“”);
                $(‘#description.card>.card-footer’).remove();
                if (args.footer && args.footer.trim() != ‘’) {
                     var footer = ‘<div class=\”card-footer\”><p
class=\”card-text text-right text-warning\”>‘
                          + args.footer + ‘</p></div>‘;
                     $(‘#description.card’).append(footer);
                }
                $(‘#description’).removeClass(‘fadeIn animated
fadeOut invisible’).addClass(‘animated fadeIn’);
          });
          $(hotSpotDiv).mouseout(function () {
                $(‘#description’).addClass(‘animated fadeOut’);
          });
     }
     function hotspotClickHandler(obj, args) {
          $(‘#description’).addClass(‘animated fadeOut’);
          $(‘.nav-link’).removeClass(‘active’);
          $(‘#’ + args.id).addClass(‘active’);
     }
     $(‘#@main.GetHashCode()’).addClass(‘active’).click(() => {
          viewer.loadScene(‘@main.GetHashCode()’);
          $(‘.nav-link’).removeClass(‘active’);
          $(this).addClass(‘active’);
     });
     @foreach (var unit in units)
     {
          @: $(‘#@unit.Name.GetHashCode()’).click(function() {
          @:     viewer.loadScene(‘@unit.Name.GetHashCode()’);
          @:     $(‘.nav-link’).removeClass(‘active’);
          @:     $(this).addClass(‘active’);
          @: });
     }
});
```